



Prescription Processing System for A Large Scale Pharmaceutical Business

**A dissertation submitted for the Degree of Master of
Information Technology**

**A. K. Kulasinghe
University of Colombo School of Computing
2017**



Declaration

The thesis is my original work and has not been submitted previously for a degree at this or any other university/institute.

To the best of my knowledge it does not contain any material published or written by another person, except as acknowledged in the text.

Student Name:

Registration Number:

Index Number:

Signature:

Date:

This is to certify that this thesis is based on the work of
Mr./Ms.

under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Certified by:

Supervisor Name:

Signature:

Date:

Abstract

Prescription processing in a pharmaceutical business involves a set of stages that are crucial to a consumer's health. These stages span from reading prescriptions, checking drug availability based on required quantities, collecting drugs from store, verifying the collected drugs against the prescription before issuing them to the customer, billings and issuing. In large scale pharmacies multiple prescriptions are processed simultaneously which leads to a single employee processing several prescriptions together or several employees being involved with different stages of a single prescription. This could result in human errors due to manual handling, leading to high time consumption, issuance of wrong drugs, dosages and incorrect instructions that can be fatal in certain situations.

An application that enables modeling the prescription life cycle to isolate priorities at each life cycle stage, facilitates minimization and elimination of these errors. The tools and technologies today enable separation and loose coupling of UI, Business logic and Data layer, to emphasize clear focus on responsibilities of each layer there by increasing the viability of extending, enriching and changing applications as time and requirements demand. Microservices that enable building small, light weight, independent services and Business Process Modeling which enables modeling work flow logic, together enable creating Business Process Management applications.

The project is an implementation of a 'Prescription processing system for a large scale pharmaceutical business', built using Microservices and Business process management technologies to address the issues involved with manual handling in the pharmaceutical businesses domain. The solution consists of a distributed system, accessible online, by pharmaceutical staff and consumers which also facilitates pharmacy-consumer interaction. The solution implementation is based on the incremental development methodology, where design and implementation details were gradually expanded as the project evolved, which ultimately lead to a significant separation of UI, Business services/processes and Data layers, creating opportunity for easy extension and incorporation of features.

Acknowledgement

I do hereby like to take this opportunity to acknowledge and express my sincere gratitude to my project supervisor Dr. M. K. Silva, for the guidance, corporation and feedback offered to me through out this project.

I would also like to thank numerous WSO2 staff members of who supported and shared their product and platform knowledge concerning WSO2 products and Angular2 implementations.

I also take this opportunity to thank pharmaceutical professionals who co-orporated with me through out the literature survey and evaluation by sharing their knowledge, experience and opinion on the pharmaceutical domain and the developed application.

Last but not the least, I would like to extend my appreciation towards my parents and my husband for the support and courage given to me throughout the entire project period in many ways.

Table of Contents

Declaration.....	i
Abstract.....	ii
Acknowledgement.....	iii
List of Tables.....	vi
List of Figures.....	vii
List of Abbreviations.....	viii
Chapter 1 : Introduction.....	1
1.1 Problem Statement.....	2
1.2 Motivation.....	2
1.3 Author's Approach to address the Problem.....	3
1.4 Objectives.....	3
1.5 Area of Study.....	4
1.6 Scope.....	4
1.7 Related Projects.....	5
1.8 Structure of the dissertation.....	5
Chapter 2 : Literature Review.....	7
2.1 Introduction.....	8
2.2 Prescription processing & Computerization of the process.....	8
2.2.1 Prescription processing in the pharmaceutical Business Domain.....	8
2.2.2 Issues and drawbacks of the manual process.....	9
2.2.3 Why computerize prescription processing?.....	12
2.3 Business process management concepts and technologies.....	13
2.3.1 Business process management.....	13
2.3.2 Business process modeling for Business Process Management.....	14
2.3.3 Business process management Applications.....	15
2.4 Use of Microservices in Business Applications.....	16
2.5. Solution.....	17
2.5.1 Critical review of existing solutions.....	17
2.5.2 Process requirements to address the identified issues.....	18
2.5.3 Technology requirements addressing technology issues of existing solutions.....	19
2.5.4 Critical review of technologies to be used.....	19
2.5.5 Finalized Technology Requirements.....	23
2.6. Summary.....	23
Chapter 3 : Analysis and Design.....	24
3.1. Introduction.....	25
3.2. Analysis.....	25
3.2.1 Requirement Specification.....	25
3.2.2 Approach to the system design.....	26
3.3. Design.....	27

3.3.1.High level Solution Architecture.....	27
3.3.2.1.Consumer Interfaces.....	29
3.3.2.2.Business process.....	32
3.3.2.3.Business Services.....	35
3.3.2.4.Operational Systems.....	37
3.4.Summary.....	39
Chapter 4 : Implementation.....	40
4.1 Introduction.....	41
4.2.UI Layer Implementation.....	41
4.2.1Angular 2.....	41
4.2.2 Use of Angular 2 in the UI Layer.....	43
4.3 Services Implementation.....	51
4.3.1Microservices implementation with WSO2 MSF4J.....	51
4.3.2Entity Class Implementation.....	52
4.3.3Service Class Implementation.....	54
4.3.4Use of Java Persistence API (JPA) for data manipulation.....	56
4.3.5. JPA implementation within a Microservice.....	59
4.3.6.Service Start up.....	63
4.3.7.Sample service.....	64
4.3.8.Exposed resources.....	65
4.4 Business Process Implementation.....	66
4.4.1Prescription processing business process.....	66
4.4.2Business Process Illustration using Activiti Eclipse 2.0 Designer.....	67
4.5 Hosting the application.....	77
4.5.1Use of Nginx.....	77
4.5.2Deployment summary.....	79
4.6 User Interfaces and User Manual.....	79
4.7 Summary.....	79
Chapter 5 : Evaluation & Testing.....	80
5.1 Introduction.....	81
5.2 Testing.....	81
5.2.1Test Approach.....	81
5.2.2Test Results.....	82
5.3 Evaluation.....	85
5.3.1Evaluation approach.....	85
5.3.2Analysis of evaluation feedback by pharmaceutical professionals.....	85
5.3.3Analysis of evaluation feedback by potential consumers.....	87
5.4Summary.....	89

Chapter 6 : Conclusion & Future Work.....	90
6.1 Introduction.....	91
6.2 Work carried out.....	91
6.3 Revelations.....	91
6.4 Lessons Learnt.....	92
6.5 Future Work.....	93
6.6 Summary.....	94
References.....	95
Appendices.....	99
Appendix A – User Manual.....	100
Appendix B– Test Plan.....	115
Appendix C – Test Cases.....	117
Appendix D – Apache JMeter 2.13 Script.....	123

List of Tables

Table 2.1 Product comparison.....	17
Table 2.2 UI Framework comparison.....	20
Table 2.3 Business process management application comparison.....	21
Table 2.4 UI Framework comparison.....	22
Table 4.1 Exposed resources.....	65
Table 4.2 Business Tasks.....	70
Table 4.3 Deployment summary.....	79

List of Figures

Figure 2.1 Prescription processing process and issues.....	11
Figure 2.2 Business Process Management.....	14
Figure 3.1 Usecase Diagram.....	26
Figure 3.2 High level solution architecture.....	27
Figure 3.3 Detailed solution architecture.....	29
Figure 3.4 Prescription Component Class Diagram.....	31
Figure 3.5 Prescription State transition.....	34
Figure 3.6 Prescription Service Class Diagram.....	36
Figure 3.7 Database model diagram.....	38
Figure 4.1 High level view of UI layer.....	43
Figure 4.2 Microservices.....	56
Figure 4.3 JPA basic flow.....	57
Figure 4.4 JPA Query control flow.....	59
Figure 4.5 Microservices Structure.....	64
Figure 4.6 Business process involvement in the solution.....	66
Figure 4.7 Business Process Engine.....	67
Figure 4.8 Prescription handling business process in BPMN2.0 notation.....	69
Figure 4.9 Process Start – General.....	70
Figure 4.10 Process Start - Form.....	71
Figure 4.11 User task - General.....	71
Figure 4.12 User task - Form.....	71
Figure 4.13 Service task – General.....	72
Figure 4.14 Service task – Main config.....	73
Figure 4.15 Exclusive gateway – General.....	74
Figure 4.16 Message Flow – General.....	74
Figure 4.17 Message Flow – Main config.....	75
Figure 4.18 End event – General.....	75
Figure 4.19 Business process running on WSO2 Business process server.....	76
Figure 5.1 JMeter summary report.....	84
Figure 5.2 Customer preference to upload prescriptions online.....	88
Figure 5.3 Customer preference to use online prescription submission feature.....	88
Figure 5.4 Customer preference to track status online.....	89
Appendix A : Figure 1 Home Page.....	100
Appendix A : Figure 2 Login.....	100
Appendix A : Figure 3 Consumer Home page.....	101
Appendix A : Figure 4 Staff Home page.....	101
Appendix A : Figure 5 Sign up.....	102
Appendix A : Figure 6 Online prescription list.....	103
Appendix A : Figure 7 Online prescription.....	104
Appendix A : Figure 8 Prescription List.....	105
Appendix A : Figure 10 Add prescription item.....	106
Appendix A : Figure 11 Prescription items.....	107
Appendix A : Figure 12 Prescription in New state.....	107

Appendix A : Figure 13 Promote prescription to collect state.....	108
Appendix A : Figure 14 Prescription in Collect state.....	108
Appendix A : Figure 15 Promote prescription to verify state.....	109
Appendix A : Figure 16 Prescription in Verify state.....	109
Appendix A : Figure 17 Promote prescription to pay state.....	110
Appendix A : Figure 18 Prescription in Pay state.....	110
Appendix A : Figure 19 Promote prescription to pay state.....	111
Appendix A : Figure 20 Prescription in Issue state.....	111
Appendix A : Figure 21 Promote prescription to complete state.....	112
Appendix A : Figure 22 Completed prescription.....	112
Appendix A : Figure 23 Completed prescription history.....	112
Appendix A : Figure 25 Prescription demoted to collect state.....	113
Appendix A : Figure 26 Prescription history of demoted prescription.....	114
Appendix D : Figure 1 JMeter script.....	123

List of Abbreviations

BPM	Business Process Management
BPMN	Business Process Management Notation
MSF4J	Microservices Framework for JavaScript
JPA	Java Persistence API

Chapter 1 : Introduction

1.1 Problem Statement

Processing prescriptions in a pharmaceutical business involves, reading prescriptions, checking availability of drugs, calculating drug quantities based on dosage, collecting drugs from the store, verifying drugs against the prescription before issuing them to the customer and billing.

In a large scale business, a large number of prescriptions is processed simultaneously and these tasks are handled manually by different parties. This creates opportunity for various discrepancies.

Miscommunications that occur among different parties handling a single prescription and human errors made during simultaneous handling of several prescriptions are commonly occurrences in large pharmacies. One fatal result that might arise from such dependencies is the issuance of wrong drugs, incorrect drug quantities and dosage instructions.

Rectifying human errors identified before issuance of prescribed items and billing mishaps lead to high time consumption in processing a prescription.

When customer perspective is considered, they are required to be physically present at a pharmacy to hand over a prescription and wait in line for prescriptions to be processed and prepared. Ability to get in touch with a pharmacy remotely is preferred by consumers as it will save a lot of time for them when such pharmacies with a high work load is considered.

1.2 Motivation

The motivation of the author is to derive a software solution, to address the above problems exploiting the advantages of open source applications handling business process management and re-usable software components.

1.3 Author's Approach to address the Problem

In the event a distributed system is available, parties at a pharmacy, handling different tasks of the process can collaborate online during various stages of prescription processing, there by reducing the time spent, avoiding re-work and discrepancies. Customers will have the facility to submit prescriptions to the pharmacy online and be notified once prescription processing is completed and drugs are ready to be collected. Existing systems which handle such processes mostly do not take into account how the quality of service provided for customers can be improved. Most of them are commercial, built from scratch and have their data layer and business logic tightly coupled with the application layer making it extremely complicated or impractical to incorporate changes.

1.4 Objectives

- To create a web based system, modeling the prescription processing life cycle of a large scale pharmaceutical business.
- To develop a prescription work flow that enhances the quality of service provided to customers considering customer aspects and requirements in prescription processing.
- To develop a system using open source applications handling business process management and re-usable software components, to exhibit easy application maintenance and adaptability.
- To demonstrate the efficiency and practicality of using off-the-shelf components to build a system, as opposed to building one from scratch, in order to minimize the 'time to market' factor.

1.5 Area of Study

The area of study of the project is pharmaceutical business process management using open source applications and off the shelf software components.

The business process followed by large scale pharmaceutical businesses for prescription processing is studied under this project in order to identify and analyze the process flow, involved parties, issues and draw-backs of the manual process.

The project also studies utilization of off-the-shelf software components such as business process management applications to model work flow logic and service management applications to implement independent services there by eliminating the need to repetitively implement common functionality in applications.

1.6 Scope

1. Prescription processing life cycle of a pharmacy, from receiving a prescription to processing to issuing the prescribed items to the consumer, will be handled by the system. Stock and payment handling is not considered for the implementation.
2. The process life cycle is modeled using a business process modeling application, loosely coupling it with the application and data layers to facilitate ease of changing.
3. Data layer is modeled using, a database exposing data to the service layer which in turn exposes its services to the Application layer in the form of RESTful APIs , thereby eliminating business logic hard coding into the application layer.
4. The application layer is accessible by employees and consumers online.
5. The features include the following.

Allow employees to

- Add / view prescriptions and manage prescription work flow

Consumers should be able to

- Submit prescriptions online until the actual prescription is presented

during collection of prescribed items.

- Be aware of prescription status in the prescription processing life cycle

1.7 Related Projects

Given below are already existing applications that cover aspects related to the project.

- **Meditech EHR solution:** A comprehensive health-care information system facilitating interaction among hospital staff and handling prescription processing.
- **Siemens Pharmacy:** Provides comprehensive support for pharmacy work flow, pharmacy resource handling, clinical decision support and communication.
- **Horizon Meds Manager:** A pharmacy information system that helps improving pharmacy performance, provider work flows, and patient safety.

All these are commercial applications, built from scratch which have their business logic tightly coupled with the application and data layers.

1.8 Structure of the dissertation

The dissertation is composed of the following structure.

- **Introduction:** Introduction to the project and problem domain of prescription processing.
- **Literature Review:** Research revelations on the pharmaceutical business domain, its issues, addressing the identified issues, a critical review of existing solutions / technology requirements and derivation of process and technology requirements.

- Analysis and Design: The software requirement specification and software design of the prescription processing application.
- Implementation: Implementation details of the application.
- Evaluation and Testing: Information related to the application evaluation and testing tasks carried out.
- Conclusion and Future Work: Project conclusion and prospects for future work related to the application.
- References : A list of referred web and other publications refereed
- Appendices: Additional information related to the project

Chapter 2 : Literature Review

2.1 Introduction

This chapter consists of literature and research revelations based on the pharmaceutical business domain with relevance to prescription processing, the manual process and its issues, with the intention of deriving technology based solutions to the issues in the manual process and existing software solutions.

2.2 Prescription processing & Computerization of the process

2.2.1 Prescription processing in the pharmaceutical Business Domain

The pharmaceutical industry is responsible for producing and marketing pharmaceuticals used for medication. Companies operating within this domain, deal with generic or brand medications and medical devices. Pharmacies can either be outlets of such manufacturing companies, or may operate independently as buyers of pharmaceutical manufacturing companies from which consumers buy products.

Prescription processing is the main and the most critical purpose of the existence of a pharmacy. It involves the sequence of tasks carried out from the moment a prescription is received by the pharmacy to the point where the medications, medical equipment or medical accessories are handed over to the customer.

In a typical scenario, prescriptions are handed over to the patient or a guardian who will then, present the prescription at a retail pharmacy or a pharmacy at the relevant hospital/clinic and will wait for the prescription processing to complete. The pharmacist, then reads the prescription, checks availability of the items in the prescription, calculates required drug quantities based on specified dosage and duration, fetches drugs in required quantities from store, packages them, carries out billing, verifies the items once again and hands over the items to the customer. These tasks are performed by one or many employees of the pharmacy depending on the pharmacy's daily work load. i.e. In a large scale pharmacy, fetching drugs from the store, packaging them for the customer and billing may be performed by 3 different

employees where as the same employee many perform all 3 activities in a small pharmacy.

2.2.2 Issues and drawbacks of the manual process

Typically, prescription processing is handled manually where as billing is the only task that is computerized. This process is time consuming and utilizes more resources that could be minimized with the correct use of technology.

Issues associated with the manual process and their causes are as follows.

- The customer should be physically present at the pharmacy to handover prescriptions.

This is required to assure validity of the prescription and also as there is no means for customers to connect with the pharmacy and present the prescription online.

- Wrong drugs, quantities and drug dosages being issued

Miscommunications among several parties dealing with the same prescription lead to this, which could be fatal in certain cases.

- Errors when specifying dosage instructions for drugs

This is a common issue experienced, where the dosage written on the drug label differs from what the medical practitioner has prescribed.

- High consumption of time to process a prescription

This is a common problem with large scale pharmacies that deal with many prescriptions simultaneously. Prescriptions may be stuck at collection, billing etc, there by holding the processing of subsequent prescriptions. Customers wait in line, for their prescriptions to be accepted for processing and completed.

- The customer needs to be physically present at the pharmacy to be notified when the items are ready to be collected.

The customer may need to step out while the prescription is being processed, in case the processing of prescription takes a considerable amount of time. They do not have any means to be notified when it is ready to be collected.

- Billing mishaps

Receipts may be issued for incorrect items, quantities and even incorrect prescriptions due to manual handling and lack of integrity with the initially received prescription and quantities.

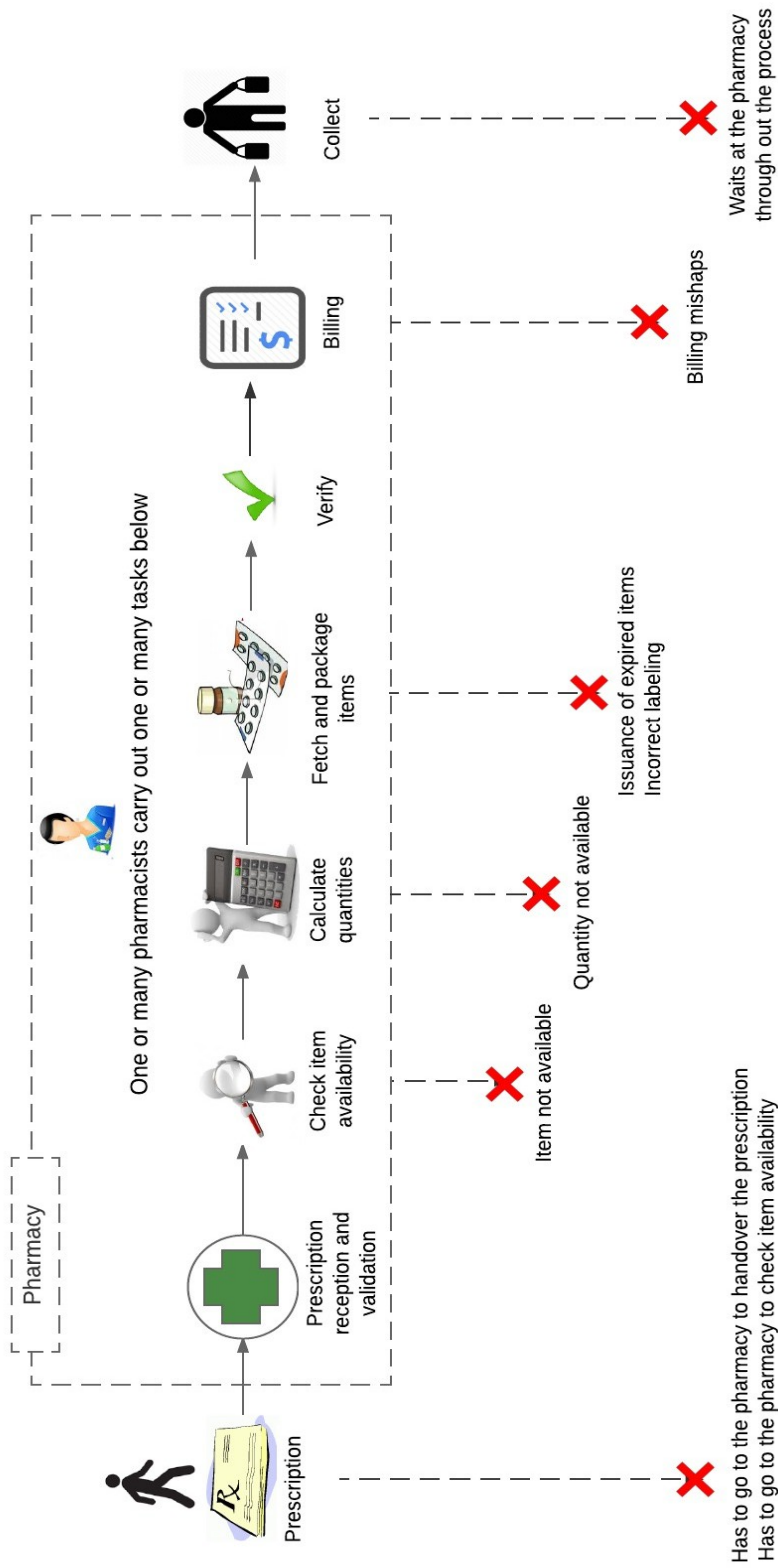


Figure 2.1 Prescription processing process and issues

2.2.3 Why computerize prescription processing?

Today's businesses demand quick decisions based on facts and analysis. This requires fast access to information with less tolerance for errors. Many businesses with routine processes do not possess information maintained in an organized manner and they lack analytical abilities and resources required to serve information demands. Proper organization and maintenance of data allows business stakeholders to derive answers and decisions rapidly. When the issues in the prescription processing domain are considered it is evident that these issues are caused by the lack of information availability, accessibility, organization and collaboration among stakeholders. A computerized system accessible over the Internet enables collaboration among different stakeholders of the business and make information available to required parties.

A good customer experience increases the potential for business growth as the businesses today are customer centric. Top customer service trends for the year 2016 include aspects related to ease, effectiveness and emotion according to the Forrester article "Trends 2016: The Future Of Customer Service" published by Kate Leggett on 5th January 2016. [1]. Customer service technology integration should take these aspects into consideration in order to create a good experience for customers.

When the customer perspective of a pharmaceutical business is considered, pharmacies can provide means of submitting a scanned copy of the prescription online, instead of expecting customers to be physically present at the pharmacy to hand over the prescription. The pharmacy can also have means of notifying the customer when the prescription is processed and the prescribed items are ready to be collected. This makes the process efficient and convenient for the customer.

Within the pharmacy, the collaboration among employees possessing different roles related to prescription processing can be made easier by making the prescription available to all involved parties via the system. The employee receiving the prescription can enter prescription details and if the required drugs are available, the

party that is collecting drugs can access the information related to this prescription from his/her work station and collect the available drugs in specified quantities and package them for the customer. The system can also be used to print labels for drugs issued to the customer specifying standard instructions and dosages retrieved from the prescription. Then these items can be verified at the front counter once again for safety purposes, and invoiced. The inventory of the pharmacy can be updated based on the product quantities issued per prescription. All these make the process of prescription processing efficient and accurate, ultimately minimizing and almost eliminating room for human errors.

2.3 Business process management concepts and technologies

2.3.1 Business process management

A business process is a set of interconnected activities that represent a business use case to produce a specific output. A process may have 0 or many inputs and an output. When a business process is executed, the sub-processes are executed in either a synchronous and asynchronous manner to produce the final output, while interacting with humans or equipment.

Business Process Management is an approach which seeks to improve the processes within an organization by increasing process speed, reducing cycle time and costs, with the intension of making the overall process efficient and effective. It is a way to identify and adjust existing processes in order to align them with the business objectives. This involves:

- Focusing on the outcomes instead of tasks
- Improving process before automation takes place
- Establishing processes and assigning ownership
- Standardizing process across the organization
- Enabling room for continuous improvement

Business process management involves several stages as depicted in Figure 2.2.

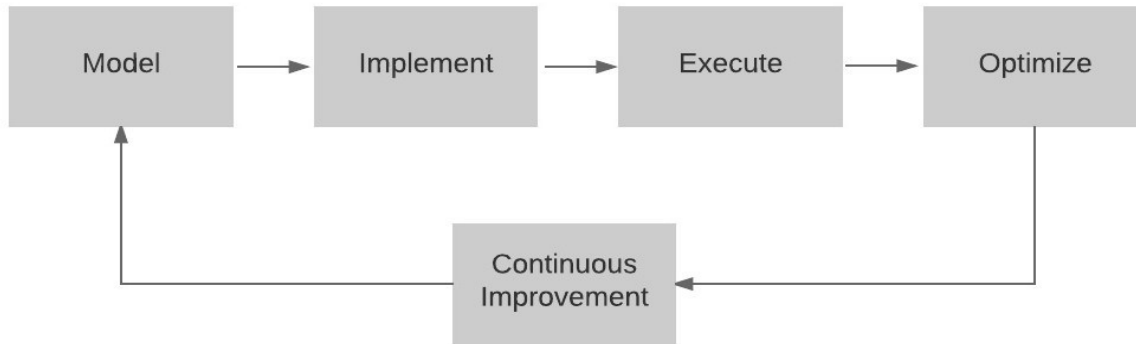


Figure 2.2 Business Process Management

Modeling stage involves studying the process and capturing business process at a high level. It involves subjecting the processes to 'what-if' analysis under real world business conditions.

Implementation of the derived model is carried out next. The process flows are implemented using business process management tools. Execution stage is where the implemented business processes are deployed or launched in order to follow in actual business operations. The process execution and performance is monitored at the next stage. Optimize stage involves introducing process and performance improvements for the bottlenecks, inefficiencies and any other issues identified in the on going process at the monitoring stage. An additional, yet important step to this cycle is 'Continuous Business Process Improvement'. This encourages the cycle to be repeated continuously in order to incorporate changes identified as improvements to the business process.

2.3.2 Business process modeling for Business Process Management

Business process modeling is a technique used to define, outline business processes, data flows, data stores and systems of a business in order to study and analyze them with the intent of restructuring business activities for improvement.

It enables isolating the process life cycle from other layers and making it independent of other components of a software application. This creates opportunity to have a clear focus on the business process life cycle, while incorporating life cycle changes without impacting and changing the entire application adversely.

Business process modeling incorporates different methodologies and tools to model business processes. These are of two types.

1. Modeling and simulation

This is used to visually represent a process. Flows charts, Data flow diagrams, Interaction diagrams, Use case diagrams, Activity diagrams based on UML and Business process modeling notation (BPMN) are some of the examples.

2. Programming

This provides programming interfaces for applications to incorporate BPM engines with them, in order to implement business processes programmatically Eg. Business Process Execution Language (BPEL), XML Process Definition Language (XPDL), Java Process Definition Language (jBPM) .

2.3.3 Business process management Applications

A Business Process Management Application is a tool that allows defining business process work flows by enabling collaboration with a business process engine in order to achieve a particular business goal and provide meaningful metrics to the management. These applications can be integrated with a front end application implementing a user friendly interface in order to hide the complex implementation and back end details from the end user.

There are various commercial and open source business process management applications that can be used to implement business processes depending on user requirements.

When automating the life cycle of an entity, handling of the business process logic can be delegated to a business process management application. The business process management application can be made aware of the life cycle states and the state order. On business process invocation, the business process management application governs updating life cycle states according to the process logic defined. This separates life cycle handling from other layers of a comprehensive application, thereby enforcing clear focus on life cycle states and allowing easy incorporation of life cycle logic changes

2.4 Use of Microservices in Business Applications

A service is a software component that enables a specific functionality of a system. A business applications uses the functionality enabled by one or more underlying services. Typical enterprise applications consist of a client side application , a database and a server side application that handles HTTP requests, execute domain logic and database transactions. Changes to an application usually involve changes to the server side application. When an application is composed of tightly coupled, heavy services that are incapable of functioning independently, it makes it difficult to build, maintain and extend the application.

In contrast, if the application is composed of a set of lightweight, independent services, it enables parallel building of application components, easy maintenance and easy extension by disassembling, re-assembling and adding components as required. Use of microservices is a new and convenient way of implementing such services by following best practices.

A microservice is an independently deployable, modular service, which runs on its own process, which is 'micro' or small in capacity as the name implies. A set of microservices function together by communicating using lightweight mechanisms to serve a specific purpose.

In a nutshell, microservices enable the following.

- Software based on microservices can be separated into multiple component services and these separate services can be tweaked and re-deployed independently.
- Microservices are organized based on business capabilities as opposed to traditional development approaches where different teams have different focuses (UI, logic, data) that sometimes overlap.
- Provides a simple mechanism to receive a request, process and generate a response.
- Enforce decentralized governance as opposed to traditional systems.
- Ideal for evolving systems that may not adhere to a specific technology/device.

There are several Microservices frameworks in market namely, WSO2 Microservices Framework for Java, Spring boot etc.

2.5. Solution

2.5.1 Critical review of existing solutions

Table 2.1 consists of the comparison of a set of existing pharmaceutical solutions.

	Product		
Feature	Meditech EHR solution	Siemens Pharmacy	McKesson Horizon Meds Manager
Proprietaryity	Proprietary	Proprietary	Proprietary
Scope	A comprehensive electronic health record (EHR) system. Consists of e-prescribing functionality and allows secure exchange of data among pharmacy, providers and patients. Does not handle a variety of activities within the pharmacy. SQL Server based	Integrates patient information with the pharmacy. Conducts medical screening based on prescribed drugs and patient health history.	Offers pharmacy management features including allergy screening, drug-disease interactions, drug-dose checking etc Supports Oracle RAC
Stakeholders Collaboration	Collaborates within the hospital among hospital staff	Collaborates with patients	Collaborates with other McKesson software
Product access	Via Desktop client	Via Desktop client	Via Desktop client
Operating system support	Windows based	OpenVMS based	Supports windows and Linux

Table 2.1 Product comparison

When the above comparison, along with revelations based on the author's research is considered there are several conclusions that can be drawn.

- Lack of products addressing the drawbacks of the manual process. i.e., Existing products mostly consider billing, drug information storage and drug screening. They do not consider work flow within the pharmacy or customer interactions.
- Most of the well established products are proprietary.
- Most of the products do not support open source operating systems such as Linux.
- Consist of thick clients , there by making it essential to install the product on each and every access terminal. (i.e. PC)
- The applications are tightly coupled with the database layer and the logic is built into the product, making it impractical to incorporate database changes.
- Business process and work flow is an inbuilt feature of the application it self, which makes it extremely complex to incorporate changes to the process work flow if required.

2.5.2 Process requirements to address the identified issues

Requirements from work flow perspective:

- The system should allow the pharmacy to add and store details of prescriptions brought to the pharmacy by customers or submitted online.
- Employees should be able to view added prescription whenever required.
- Different employees engaged in prescription processing should be able to update prescription status, throughout various stages from start to end based on the employee role.

Requirements from customer perspective

- The system should allow registered customers to submit scanned copies of prescriptions online until the actual prescription is presented during collection.

- It should also be possible for the pharmacy to notify customers on the status of their prescriptions in the process.

2.5.3 Technology requirements addressing technology issues of existing solutions

As revealed by the review of existing solutions above, the technologies selected for the developed solution should support the following.

- The solution should be accessible over the Internet in order to eliminate the requirement to install the software in each and every client machine.
- The solution should use open source technology and be able to run on open source operating systems.
- De-couple data layer from application layer in order to remove data access logic from the application, to eliminate point to point connections among data and application layers, to enable managed and controlled data access, to create room for database changes and enabling data and service component re-use.
- De-couple business process and work flow from the application layer in order to enable easy and quick modeling of process work flows, easy incorporation of work flow changes and to make the application layer independent of the process work flow.

There exists a number of off-the-shelf products dedicated to handling of application specific layers. Business process management applications, Data and Service management applications serve the purpose of separating data layer and work flow logic implementation from the front end application.

2.5.4 Critical review of technologies to be used

In order to facilitate a system that supports the above requirements, the author identified the need for the following technologies.

- A UI framework
- A Business process management application
- A Microservices framework

UI Framework

A UI framework exposes APIs to build UI components. This is required in order to build the front end application. Table 2.2 contains a comparison of widely used UI frameworks.

Feature	UI Framework	
	jquery UI	Angular 2
Description	A lightweight JavaScript library supporting client side scripting for JavaScript based web applications	A Model–view–controller (MVC) based framework by Google for building client-side web applications.
Proprietary	Open source	Open source
RESTful API support	Not supported	Supported
Supported scripting languages	JavaScript	JavaScript
MVC Pattern support	Not supported	Supported
Form Validation	Not supported	Supported
Localization	Not supported	Supported

Table 2.2 UI Framework comparison

As per the above comparison, jquery UI is merely a JavaScript library which supports client side scripting. Angular 2 is a more sophisticated framework for building front end applications which conform to MVC architecture. As a result of being a framework, Angular 2 supports features including templating, data binding, routing (for single page applications) and security.

The project targets separating the business logic layer from the application layer. In order to realize that, it is required to expose business services to the front end application. The ability of Angular 2 to consume RESTful APIs is one of the critical deciding factors when selecting a UI framework for the solution. Therefore it was decided to use Angular 2 to create the front end application.

Business Process Management Applications

A business process management system is required to separate the work flow logic implementation from the application layer.

Table 2.3 consists of a comparison among existing business process management applications.

Features	Application	
	Apache ODE (Orchestration Director Engine)	WSO2 Business Process Server 3.6.0
Description	WS_BPEL based business process server	Activiti and Apache ODE based Business Process Management Server
Proprietary	Open source	Open source
Supported standards	WS-BPEL	WS-BPEL / BPMN 2.0
Management console support	No management console	Consists of a management console
Platform Advantage	Does not consist of a platform with other features required for the project	Consists of a middle ware platform with products with features supporting data services, enterprise integration, messaging and many other solutions.

Table 2.3 Business process management application comparison

BPMN is the emerging standard for business process modeling and it consists of a user friendly graphical notation. It is also an added advantage to have management/UI console support for easy management and monitoring of business processes. Therefore, after considering the above comparison and the support for the project requirements, WSO2 Business Process Server 3.6.0 was selected as the business process management application for the author's solution.

Microservices Framework for service development

Table 2.4 below contains a comparison of microservices development frameworks.

Feature	Microservices Framework	
	Spring Boot	WSO2 MSF4J 2.1.0
Description	Facilitates creating stand-alone spring applications.	A quick development model using simple annotations and Spring Framework capabilities.
Size	13MB	9MB
Memory consumption	Minimum memory requirement is 25MB	Low memory footprint compared to Spring Boot (7MB)
Performance	-	Faster Runtime, Higher throughput and lower latency when compared to Spring Boot
Security	-	Built in security with WSO2 Identity Server
JAXRS Annotation support	No direct support	Yes
Built in Analytics	No	Built in to the framework using WSO2 Data analytics Server

Table 2.4 UI Framework comparison

Considering revelations of the above comparison, WSO2 MSF4j (Microservices Framework for Java) appears to be a lightweight, high performant, secure, annotation based framework additionally supporting Analytics in case needed. Therefore the author is inclined towards building the components of the service layer based on WSO2 MSF4J 2.1.0.

2.5.5 Finalized Technology Requirements

As concluded based on the above reviews and comparisons, the technologies and off the shelf software products are mentioned in Table 2.6 are selected for the author's application.

Technology/Application	Product
UI framework	Angular 2
Business process management application	WSO2 Business Process Server 3.6.0
Microservices Framework	WSO2 Microservices Framework for Java 2.1.0

Table 2.6 Finalized applications and technologies

UI framework in the context of author's solution

- Builds the UI layer
- Consumes services via APIs exposed in underlying business process and service layers and exposes them to application users in a user friendly manner

Business Process Management Application in the context of author's solution,

- Models the work flows, also known as business processes
- Exposes business processes in the form of RESTful web services

Microservices Framework in the context of author's solutions

- Build and host microservices that define application functionality including data access.

2.6. Summary

The literature review conducted in this chapter reveals the process requirements for prescription management process, technology based solutions for identified issues and technologies to be used in the author's software solution. The next chapter consists of the analysis and design of the system.

Chapter 3 : Analysis and Design

3.1. Introduction

This chapter consists of the analysis and design details of the project along with elaborated explanations of how the solution is arrived at. The author initially derives functional and non-functional requirements and then the design requirements of the solution based on the identified functional and non functional requirements.

3.2. Analysis

3.2.1 Requirement Specification

The requirements identified are types functional and non-functional requirements.

Functional Requirements

- The consumers should be able to register themselves.
- Registered consumers should be able to submit scanned copies of prescriptions online.
- Registered consumers who have submitted prescriptions online, should be able to check prescription status online.
- The staff should be able to enter details of prescriptions submitted manually or online.
- The staff should be able to accept or reject prescriptions submitted online.
- The pharmaceutical staff should be able to promote new prescriptions to the following stages of the prescription processing life cycle.
 - New → Collect → Verify → Pay → Issue → Completed

Non-Functional Requirements

- Authorization based on user role when exposing system data and functionality to pharmaceutical staff and pharmaceutical product consumers.

Figure 3.1 consists of the Use case diagram of the application. Note that the Login use case is not included for clarity.

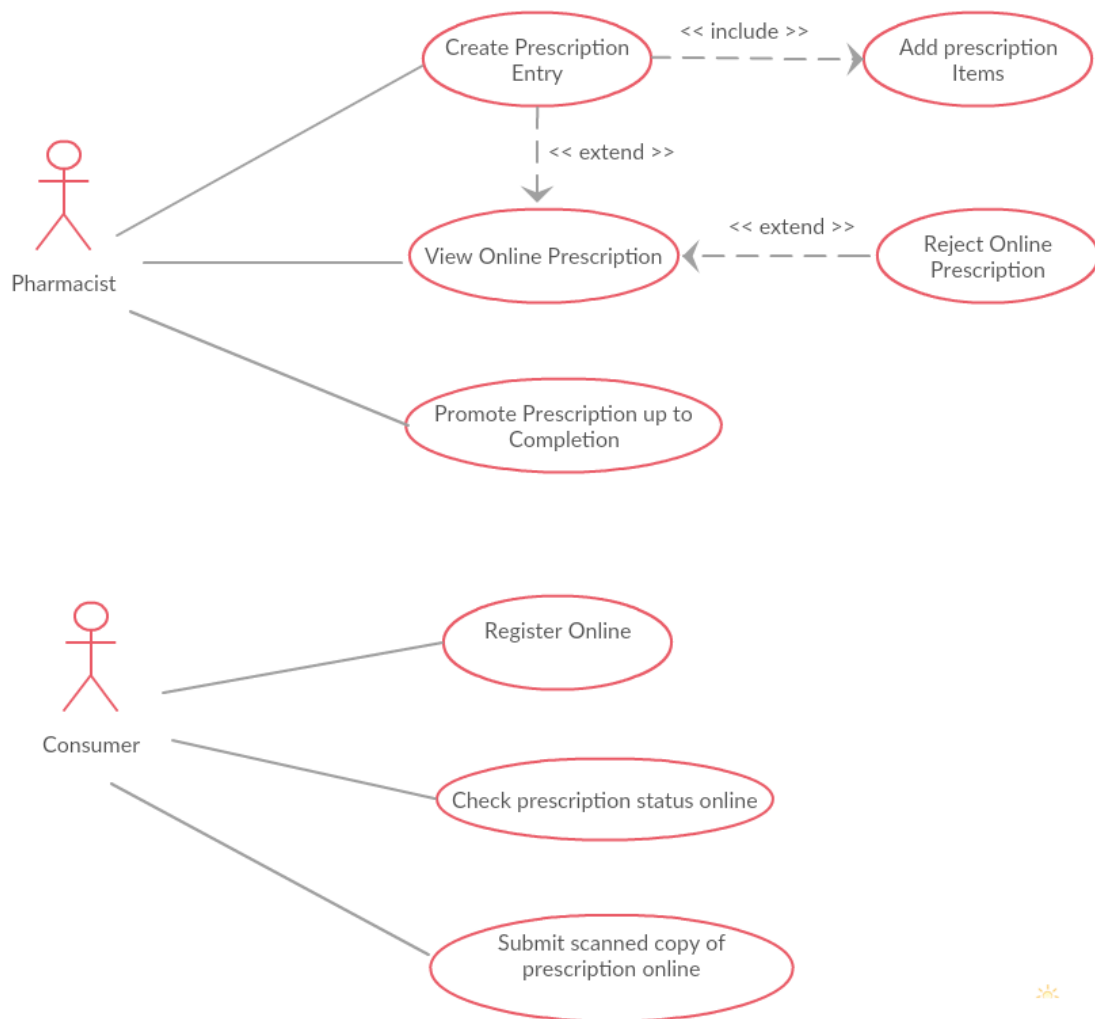


Figure 3.1 Usecase Diagram

3.2.2 Approach to the system design

As revealed through the literature review, the tight coupling of components of a system including tight UI, business logic and data layer coupling hinders and obstructs system improvement and incorporation of changes. Therefore the author intends to approach a solution with a clear separation of concerns based on a multi tier architecture.

The main architectural concern of the solution is to separate Data, Business logic and Presentation layer from each other there by ensuring loose coupling among these layers.

3.3. Design

3.3.1. High level Solution Architecture

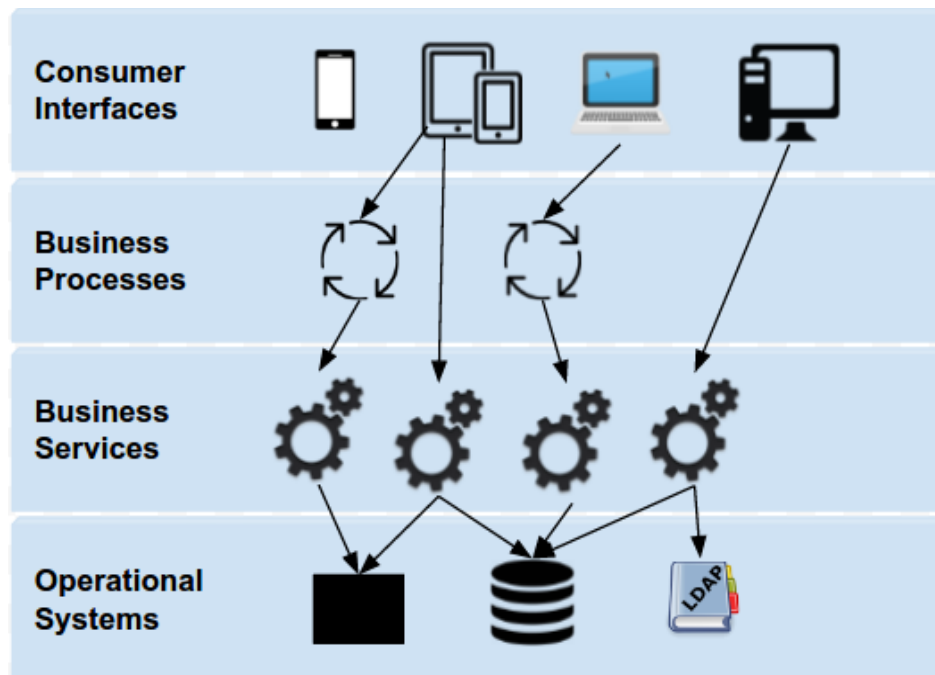


Figure 3.2 High level solution architecture

The Figure 3.2 depicts the high level solution architecture of the proposed system. The solution is to be built based on 4 layers that interact to form the complete solution. These 4 layers together correspond to Presentation, Business Logic and Data layers. The consumer Interface layer corresponds to the presentation layer. Business Process and Business Services layer together form the business logic layer. Operational Systems layer corresponds to the Data layer.

The high level explanation of these layers is as follows.

Consumer Interfaces

This is the User Interface Layer which can be exposed via the web through the computer, mobile devices and various other interfaces. The services exposed by the solution are accessed by consumers via this layer. The user interface of author's solution addresses 2 distinct communities , Pharmaceutical product consumers and Pharmaceutical staff.

Business Processes

Business processes consist of several loosely coupled services that are sequentially aligned to form a process. I.e.: Prescription work flow service that governs the prescription life cycle. Business processes are triggered by human tasks and a triggered business process governs navigation within the work flow.

Business Services

Services that perform database interactions and enable other functionality are located in this layer. Eg; Drug service

Operational Systems

This layer consists of other components and systems providing data required for the solution to function. Eg: Databases, User stores to maintain consumer/patient information etc.

Figure 3.3 below depicts the expanded solution architecture with component mapping.

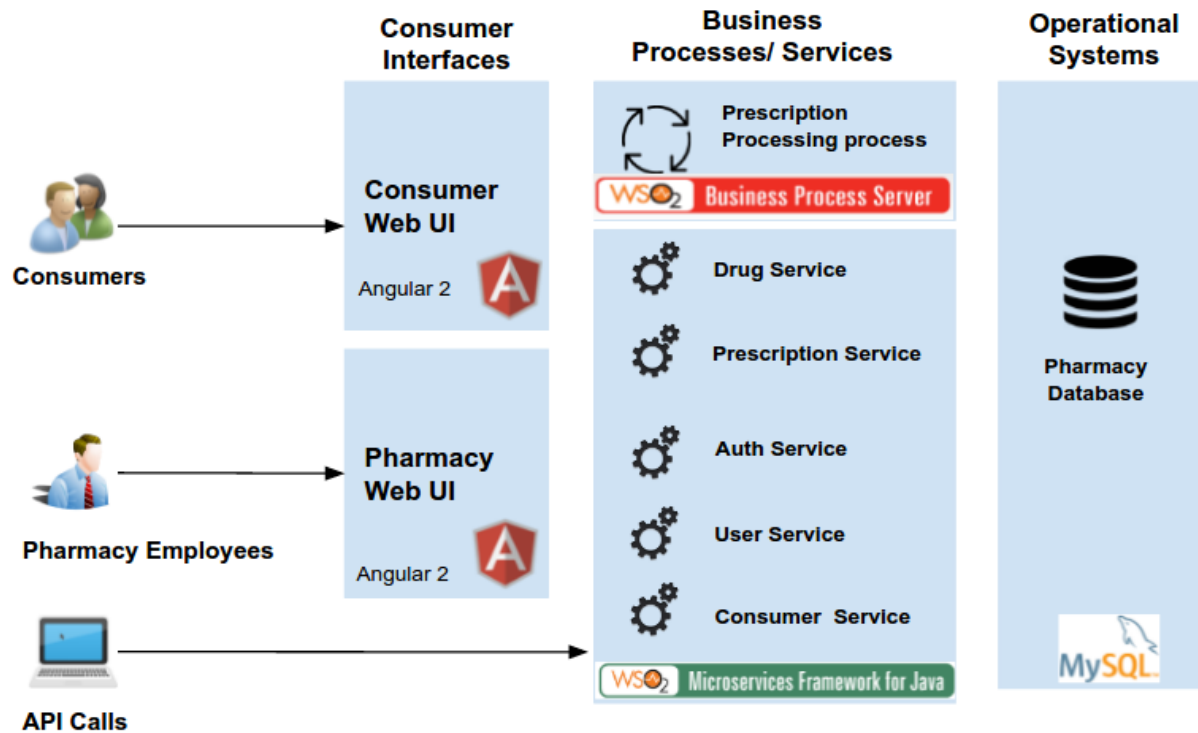


Figure 3.3 Detailed solution architecture

The next 3 sections explain Figure 3.3 in detail.

3.3.2.1. Consumer Interfaces

This is the UI layer which exposes interfaces to the 2 distinct parties using the application namely, pharmaceutical product consumers and pharmaceutical staff.

To facilitate the above functional requirements, the following interfaces are required.

Pharmaceutical product consumer web UI

Requires interfaces to:

- Login to the application
- Register at the pharmacy
- Upload prescriptions
- View prescription details and status

Pharmacy web UI

Requires interfaces to:

- Add and View prescriptions and manage prescription statuses
- View prescriptions submitted online

This layer will be implemented using Angular 2.

Prescription handling UI plays a major role in the UI layer which is represented by the prescription component of the Angular 2 application. The class diagram of the prescription component is given in Figure 3.4. Further implementation details of the prescription component are elaborated in section 4.2.2 of Implementation Chapter.

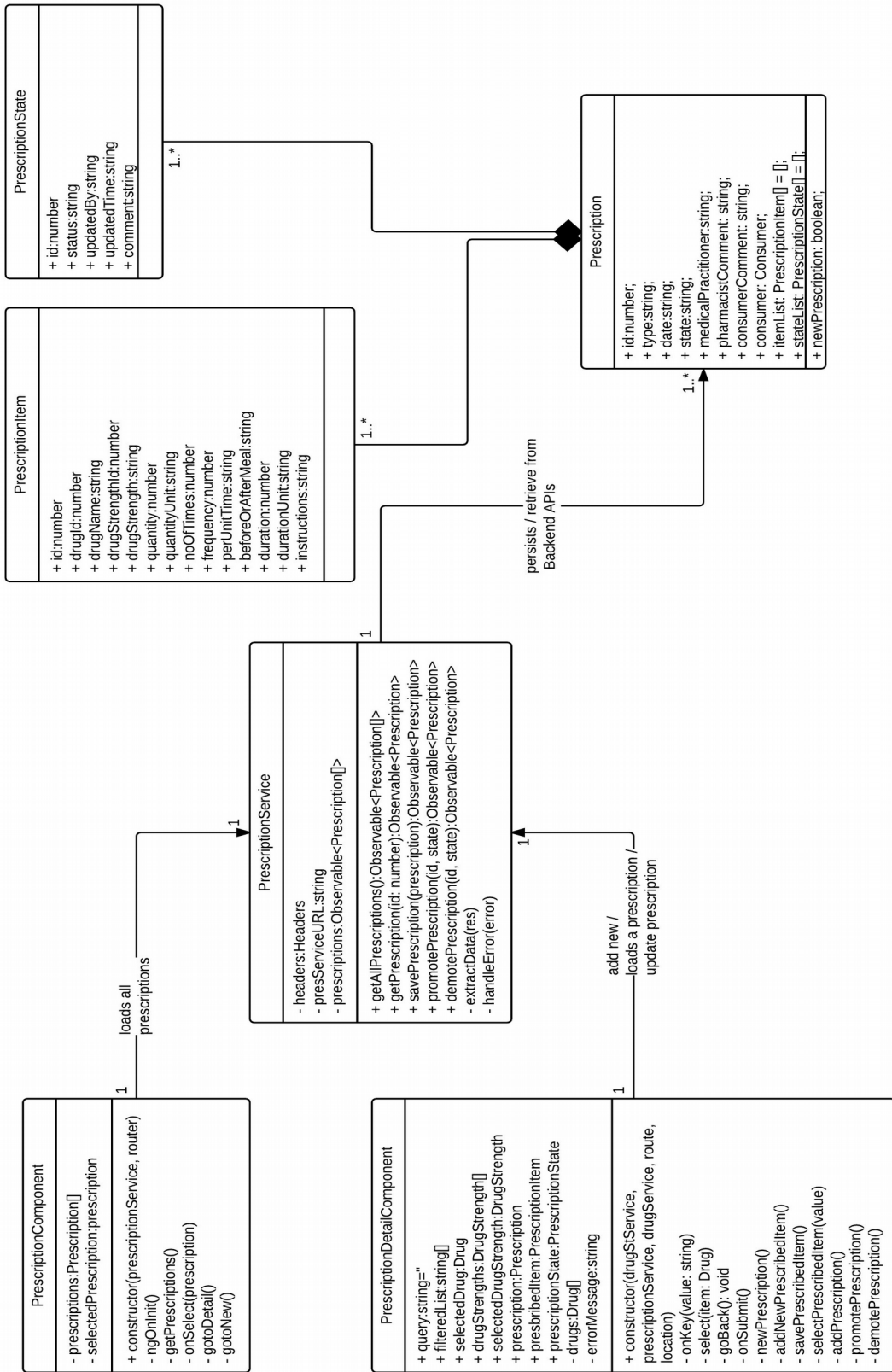


Figure 3.4 Prescription Component Class Diagram

3.3.2.2. Business process

This layer consists of the 'Prescription Processing Process' , which handles the work flow throughout the prescription life cycle within the pharmacy domain. This is implemented using the WSO2 Business Process Server 3.6.0.

Features that define business process requirements

- The Business Process spans over the involvement of 6 distinct roles in the pharmacy namely
 1. Admin : Administrator of the entire application with privileges to perform all actions
 2. Receiver : The user who initially verifies and enters a prescription in the system
 3. Collector : The user who collects prescribed items from storage
 4. Verifier : The user who verifies collected items against the prescription
 5. Cashier : The user who collects payment for a prescription
 6. Issuer : The user who hands over the prescribed items to the consumer
- A user may have one or more of the above roles.
- The business process starts with a user with 'receiver ' role entering details of a prescription manually. When a prescription is added to the system in this manner, its state will be marked as 'New'.
- In order to proceed to collect items specified in the prescription , its state should be promoted to as 'Collect'.
- A user with 'collector ' role then collects the required drugs/items, packages them and promotes the prescription state to 'Verify'.
- A user with 'verifier ' role then verifies the collected items against the prescription and promotes the state to 'Pay' if the verification is successful. If the verification fails the prescription state is demoted to 'Collect.' (The collector will have to re-collect the items and promote the prescription to 'Verify once again)
- A user with ' cashier' role accepts payment for prescriptions in state 'Pay' and

promotes the state to 'Issue' of the payment is successful.

- In case the payment fails, the process will be ended by marking the prescription state as 'Payment failure'.
- A user with 'issuer' role then , hands over the prescribed items to the customer and promotes the prescription state to 'Completed'.

State transition of a prescription within the business process depicted in Figure 3.5.

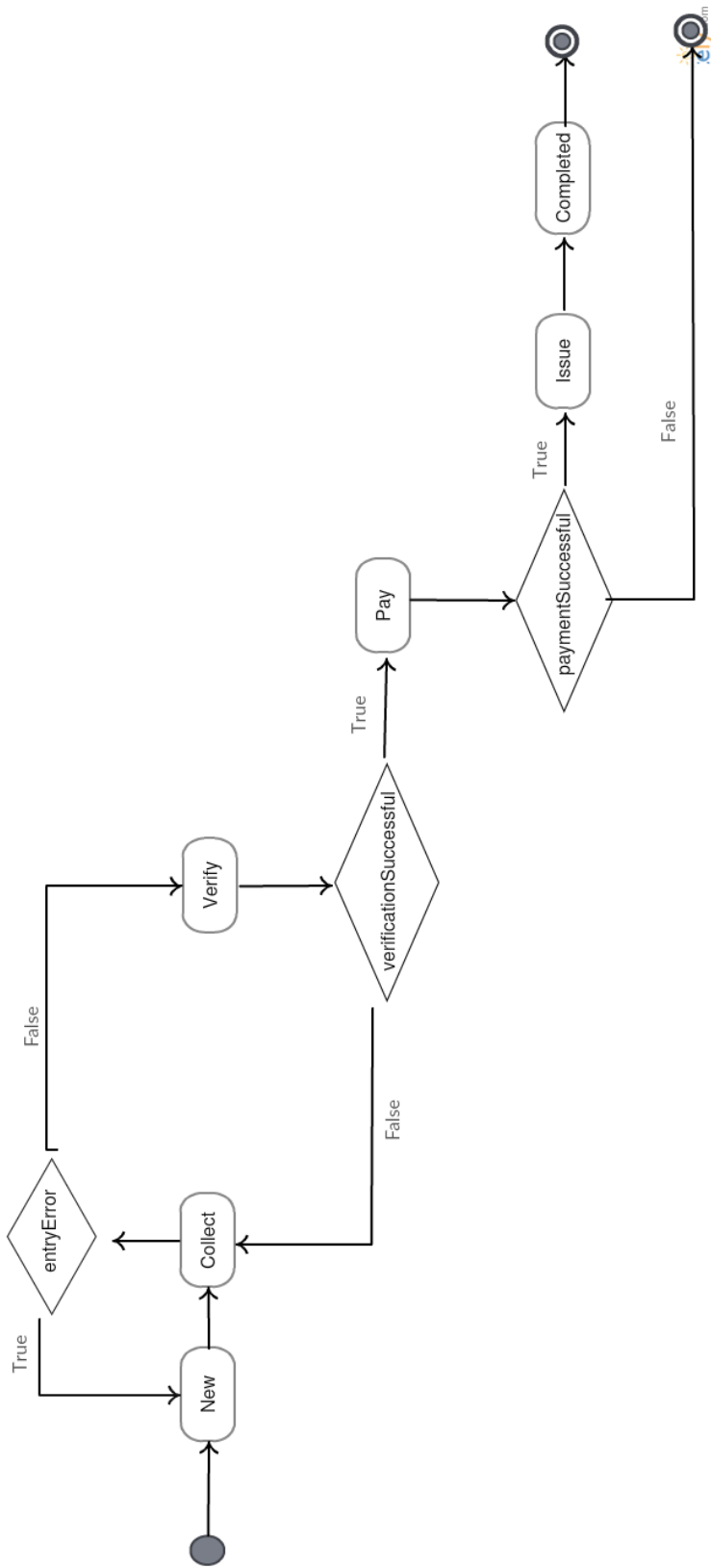


Figure 3.5 Prescription State transition

3.3.2.3. Business Services

The services defined in this layer directly facilitate business functions.

Drug Search Services

Defines back end services required for the drug management functionality.

Prescription Service

This service facilitates prescription handling which is an internal function used to manage prescription processing and it exposes itself to the 'Prescription Processing Process' of the business process layer.

Auth Service

Defines back end services required for login/logout functionality.

User Service

Defines back end services required for pharmacy staff authorization.

Consumer Service

Defines back end services required for the Consumer detail management.

These services will be implemented using WSO2 Microservices Framework for Java.

Prescription service plays a major role in the service layer. The class diagram depicting the Classes and communication among classes in the prescription service is given in Figure 3.6. Further implementation details of the prescription service are elaborated in section 4.3.1 of the Implementation Chapter.

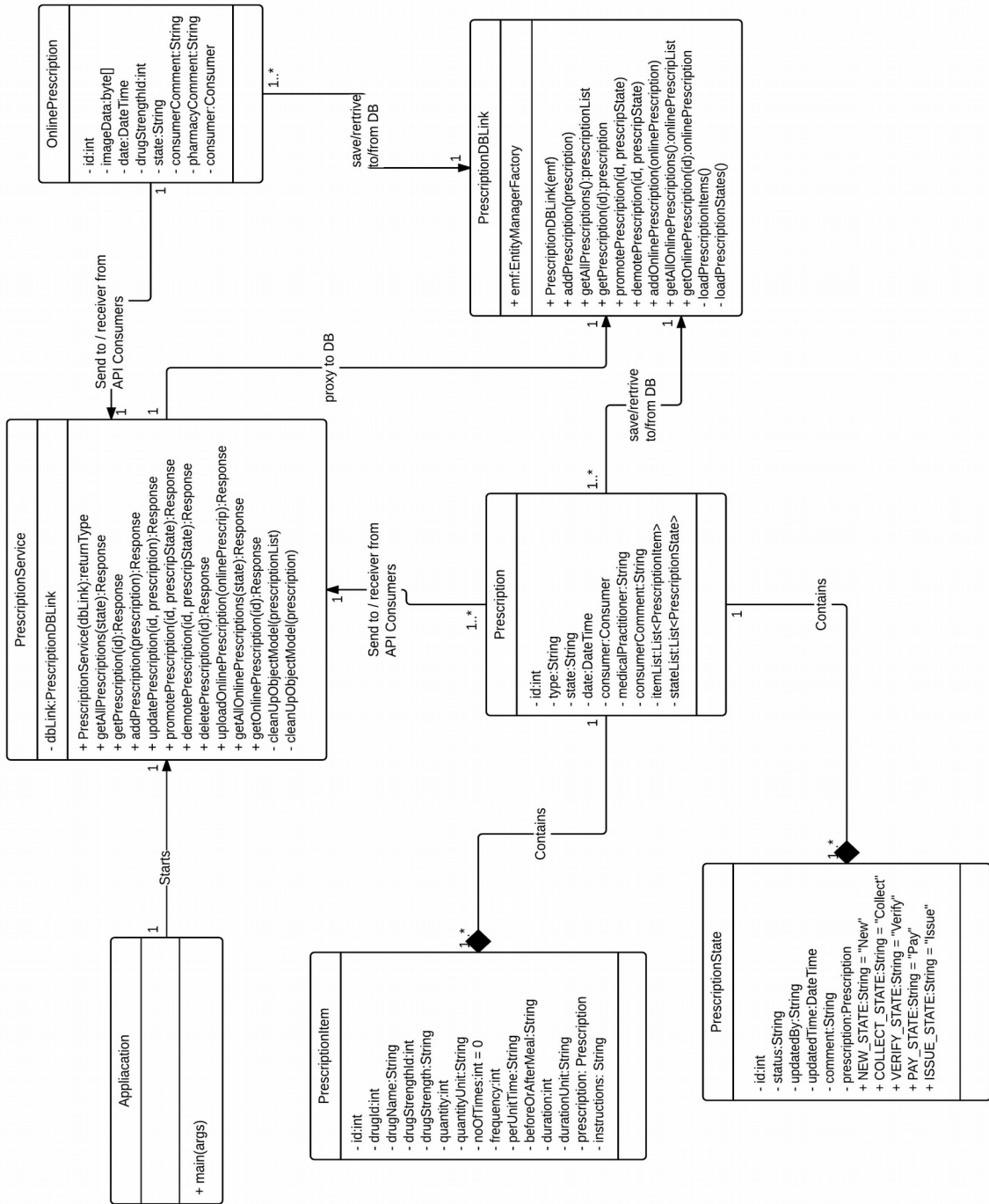


Figure 3.6 Prescription Service Class Diagram

3.3.2.4. Operational Systems

This layer corresponds to the database layer which holds drug, prescription and consumer related data.

The complete database design derived as per the application requirements is depicted in Figure 3.7.

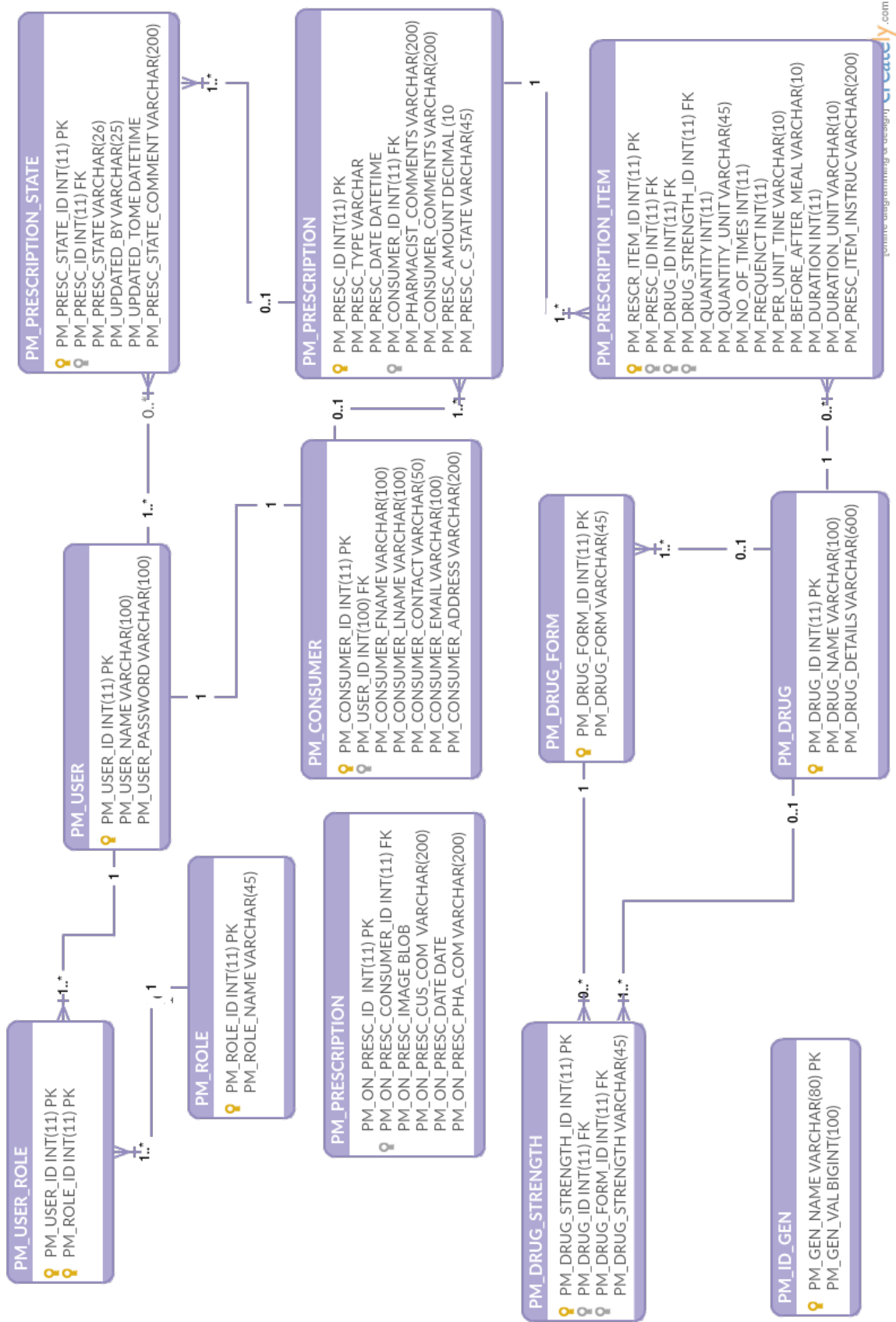


Figure 3.7 Database model diagram

3.4. Summary

The analysis and design tasks carried out in this chapter provide a detailed and a planned approach for implementation of different layers of the solution. Initially the author derived functional and non functional requirements of the solution. Then solution architecture has been derived and expanded from high level to Detailed solution architecture describing different components associated with it. This has further been elaborated with design context explanations of UI, Business process, Business Services and Operational systems layers. The next chapter consists of application implementation related details and illustrations.

Chapter 4 : Implementation

4.1 Introduction

In this chapter, the author attempts to provide an insight to the third party components used and then to the approaches followed to implement the application by explaining implementation details at each layer.

4.2. UI Layer Implementation

4.2.1 Angular 2

Angular is a HTML, JavaScript or TypeScript based framework that enables building client side applications. Angular applications are composed of one or more modules which contain

- HTML templates with Angularized markup
- Component classes to manage the HTML templates
- Services defining application logic

The building blocks of an Angular application are as follows.

Modules

Angular modules are named **NgModules** are identified with **@NgModule** decorator. Each Angular application is composed of at least the root module named AppModule or AppModule and several modules. Each module corresponds to a dedicated feature/domain within the application.

A NgModule consists of following properties, which determine module and its scope.

- declarations : classes that belong to the module
- exports : declarations that should be visible to other modules
- imports : other modules of which the exported classes are required
- providers : services that should be visible to the entire application

- bootstrap : main application view or root component hosting all other views.

Components

A component corresponds to a particular view of the UI. The application logic of a component is defined inside a class which interacts with the view.

Templates

A component view is defined in a HTML based template instructing how to render it.

Metadata

Metadata binds the class of a component to Angular framework and instructs Angular where to retrieve the building blocks that are specified in a component. Metadata is attached to a class using the @Component decorator.

Data binding

Data binding is the mechanism that coordinates parts of a template with parts of a component. This could be in the form of interpolation (retrieving a property value), property binding (passing a property value to a component), event binding (attaching an event to a piece of data) or a combination of property and event binding.

Directives

Directive is a class with directive metadata instructing how to transform DOM of a template dynamically when rendered by Angular. I.e A component is a directive with a template.

Services

A service is a class with one or many methods with well defined purposes. Components are the consumers of services. Services are responsible for fetching data from the server, validating user inputs etc.

Dependency Injection

Dependency injection provides new components with required services. When a component requires a service, the service should possess an injector for it to be used by the component.

4.2.2 Use of Angular 2 in the UI Layer

Angular version 2, ie Angular 2 was used in the UI layer implementation. Angular enforces single page applications. The author made use of an 'Angular2 single page seed project', which consists of basic structural components including Angular 2 libraries, and pre-configured scripts that build up a basic Angular 2 application. The UI layer is composed of an Angular 2 application which contains a single Angular module, with several components. Navigation among components is handled by App router (i.e. app.routing.ts). The Angular module (i.e. UI layer) communicates with business process and business services layers in the form of JSON messages. Figure 4.1 contains the high level view of solution at the UI layer.

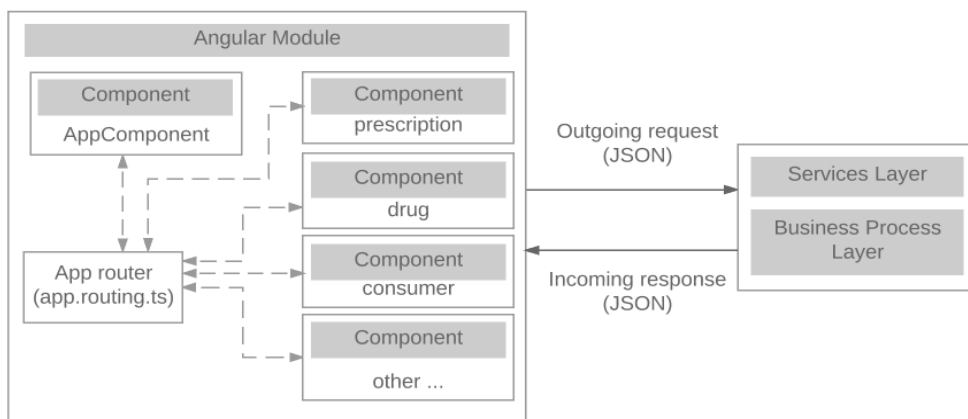


Figure 4.1 High level view of UI layer

The structure of the Angular2 application in the solution is as follows.

```

the-pharmacist
  |__ .../
  |__ src/
     |__ app/
        |__ home/
        |__ login/
        |__ drug/
        |__ consumer/
        |__ onLinePrescription/
        |__ prescription/
        |__ ...
        |__ app.component.html
        |__ app.component.ts
        |__ app.module.ts
        |__ app.routing.ts
  |__ index.html
  
```

The 'app' directory consists of all the components that the application is composed of.

app.component.ts and app.component.html

app.component.ts is a typescript file which defines the structure of the application (menus, header, footer etc) using its template file 'app.component.html'. It defines the 'AppComponent' class, which implements 'OnInit' and 'OnDestroy' interfaces. The methods 'ngOnInit()' and 'ngOnDestroy()' belonging to the above interfaces are implemented within this class, to define actions to perform on component initialization and component unloading respectively.

```
export class AppComponent implements OnInit, OnDestroy {
  public loggedIn: boolean;
  private loginSub;
  constructor(private loginService: LoginService) { }
  ngOnInit(): void {
    this.loginSub = this.loginService.loggedInObservable.subscribe(val => {
      console.info("AppComponent login status updated: " + val);
      this.loggedIn = val;
    });
  }
  ngOnDestroy() {
    this.loginSub.unsubscribe();
  }
}
```

'ngOnInit()' method determines session availability and sets the value of the boolean variable 'loggedIn'. This is checked in the 'app.component.html' page in order to determine display of 'Sign In', 'Sign Out' and 'Sign Up' menu items.

The following menu items defined in 'app.component.html' are loaded only if the variable 'loggedIn' is set to 'true'.

```
<ul id="nav-2" class="right hide-on-med-and-down" *ngIf="loggedIn">
  <li><a routerLink="/login" routerLinkActive="active ">Sign Out</a></li>
</ul>
<ul id="nav-3" class="right hide-on-med-and-down" *ngIf="!loggedIn">
  <li><a routerLink="/login" routerLinkActive="active ">Sign In</a></li>
  <li><a routerLink="/consumer-register" routerLinkActive="active">Sign Up</a></li>
</ul>
```

app.module.ts

For each module, it is required to declare and import all components used. This file declares and imports all components used by the root module, which in the author's solution is the only module.

All required components are imported initially.

```
import './rxjs-extensions';
import { BrowserModule } from '@angular/platform-browser';
import { NgModule } from '@angular/core';
import { FormsModule } from '@angular/forms';
import { HttpClientModule, JsonpModule } from '@angular/http';
import { AppComponent } from './app.component';
...
import { PrescriptionComponent } from './prescription/prescription.component';
import { PrescriptionDetailComponent } from './prescription/prescription-detail.component';
...
```

In order to identify this as a module, the `@NgModule` decorator is used.

```
@NgModule({
  declarations: [
    AppComponent, HomeComponent, LoginComponent, PrescriptionComponent,
    PrescriptionDetailComponent,
    ....
  ],
  imports: [
    BrowserModule, routing, FormsModule, HttpClientModule, MaterializeModule, JsonpModule
  ],
  providers: [
    PrescriptionService, LoginService, AuthGuard, ManufacturerService,
    ....
  ],
  bootstrap: [AppComponent]
})
```

All component classes of the module are listed under 'declarations' and third party modules are specified as 'imports'. 'providers' are the service classes that expose application functions. 'bootstrap' specifies the root component, which in this solution is the AppComponent itself.

app.routing.ts

This defines the navigation from one page to another within the application.

```
{ path: 'prescription', component: PrescriptionComponent, canActivate: [AuthGuard], data: { roles:
['admin', 'receiver', 'collector', 'verifier', 'cashier', 'issuer'] } },
{ path: 'prescription-detail/:prescId', component: PrescriptionDetailComponent, canActivate:
[AuthGuard], data: { roles: ['admin', 'receiver', 'collector', 'verifier', 'cashier',
'issuer'] } },
```

A 'component' is attached to a specific 'path' or a context. The context `/prescription` in the above code segment, loads the component `PrescriptionComponent`, and

context '/prescription-detail/:prescId' loads the ' PrescriptionDetailComponent', with its html page populated with prescription details of the prescription matching the prescription ID in the path parameter. As the access to various application components is role based, loading certain components involves a role check as given in the above code segment.

Prescription Component

The component 'prescription' is the most important custom component in the Angular 2 application. Other modules are mainly supportive modules that help enable the prescription component's functionality. Therefore the use of Angular 2 in the UI layer development will be explained based on the prescription component.

The prescription component consists of the following content.

```
prescription
  |__ prescription.ts
  |__ prescription-item.ts
  |__ prescription-state.ts
  |__ prescription.service.ts
  |__ prescription.component.ts
  |__ prescription.component.html
  |__ prescription-detail.component.ts
  |__ prescription-detail.component.html
```

Entity Classes (prescription.ts, prescription.item.ts, prescription.state.ts)

Angular is based on object oriented concepts. The entity classes representing the prescription, prescription items and prescription states are maintained in the files prescription.ts, prescription.item.ts and prescription.state.ts respectively.

The concept of 'prescription' is maintained as an entity in the file 'prescription.ts'.

```
export class Prescription {
  id: number;
  type: string;
  date: string;
  state: string;
  medicalPractitioner: string;
  pharmacistComment: string;
  consumerComment: string;
  consumer: Consumer;
```



```
    itemList: PrescriptionItem[] = [];  
    stateList: PrescriptionState[] = [];  
    newPrescription: boolean;  
}
```

The attributes defined in the Prescription Class, correspond to columns of the database table PM_PRESCRIPTION. Each prescription is associated with one or more drugs/items, held in the database table PM_PRESCRIPTION_ITEM. The class PrescriptionItem defined in prescription.item.ts represents prescription item entity and it corresponds to columns of this table. Each prescription is associated with one or more states held in the database table PM_PRESCRIPTION_STATE. The class PrescriptionState defined in prescription.state.ts represents the prescription state entity and it corresponds to the columns of this table.

Service Class (prescription.service.ts)

The class PrescriptionService defined in this file, declares methods that enable prescription related functionality based on CRUD operations. The back end service URL for the Prescription service exposed by the service layer (i.e. WSO2 Microservices for Java) is specified in the variable 'presServiceURL' and accessed by methods defined in this class. Angular 2 passes data in the form of JSON objects to the service layer.

```
private                                presServiceURL                                =  
'http://localhost:8090/msf4j/pharmacy/prescriptions';
```

The service class itself is defined with the decorator '@Injectable()'. This marks the class as available to be injected to other components, when the its methods are invoked from within.

The return type of methods is set to 'Observable'. An observable enables asynchronous operations, returning data over time. If the operations occur synchronously, a component will not be loaded until the associated operations have returned all data requested for. Eg: In the solution, when the user clicks on 'Prescription' menu item to view the prescription list, the relevant UI will not be loaded until the associated method finishes returning all available prescriptions. But the use of Observables makes the application more user friendly, as it loads the requested component first and then the requested data eventually.

```

getAllPrescriptions(): Observable<Prescription[]> {
    if (!this.prescriptions) {
        this.prescriptions = this.http.get(this.presServiceURL + "?state=NEW")
            .map(this.extractData).do(ignored => console.info("fetched Prescriptions"))
            .catch(this.handleError);
    }
    return this.prescriptions;
}

```

The class consists of methods to :

- Retrieve all prescriptions on Prescription component loading

```
getAllPrescriptions(): Observable<Prescription[]> {}
```

- Retrieve details of a selected prescription

```
getPrescription(id: number): Observable<Prescription> {}
```

- Save a prescription

```
savePrescription (prescription:Prescription): Observable<Prescription> {}
```

- Promote a prescription to the next state in prescription life cycle

```
promotePrescription(id:number, state:PrescriptionState): Observable<Prescription>
    {}
```

- Demote a prescription to the previous state in prescription life cycle

```
demotePrescription(id:number, state:PrescriptionState): Observable<Prescription>
    {})
```

prescription.component.ts and prescription.component.html

The PrescriptionComponent class of prescription.component.ts determines the content of prescription list UI. It refers to its template 'prescription.component.html'.

```
@Component({
  selector: 'prescription',
  templateUrl: './prescription.component.html',
})
```

The service methods in the PrescriptionService class of prescription.service.ts are invoked by the prescription.component.ts to handle prescription data. Eg: Invoke getAllPrescriptions() method in the PrescriptionService class to load a list of prescriptions on component loading.

```
getPrescriptions(): void {
  this.prescriptionService.getAllPrescriptions()
    .subscribe(prescriptions => this.prescriptions = prescriptions);
}
```

The component has to subscribe to the observable returned by the getAllPrescriptions() method, so that the data can be accessed when retrieved. In order for this method to be invoked on component loading, the getPrescriptions() method invocation itself is invoked within ngOnInit() method which is invoked during component initialization.

```
ngOnInit(): void {
  this.getPrescriptions();
}
```

Apart from this PrescriptionComponent class itself defines a few methods to enable adding new prescriptions and view details of a selected prescription.

gotoNew() method passes a prescription object with the ID set to -1 to the router, which will evaluate to a new prescription object that enables adding a new prescription.

```
gotoNew(): void {
  this.router.navigate(['/prescription-detail', -1]);
}
```

The application also allows selecting a prescription from the list and loading details of that prescription. The onSelect() method determines the selected prescription and gotoDetail() method enables loading the prescription detail UI.

```

onSelect(prescription: Prescription): void {
  this.selectedPrescription = prescription;
}
gotoDetail(): void {
  this.router.navigate(['/prescription-detail', this.selectedPrescription.id]);
}

```

These methods are invoked within the `prescription.component.html` file on user click event, in the prescription list.

```

<tbody>
  <tr *ngFor="let prescription of prescriptions" (click)="onSelect(prescription)"
      (click)="gotoDetail()">
    <td>{{prescription.id}}</td>
    <td>{{prescription.type}}</td>
    .....
  </tr>
</tbody>

```

'ngFor' directive lists items in a list. 'prescriptions' array is defined in the `prescription.component.ts`, of which the items are accessed iteratively (i.e. 'let prescription of prescriptions') to populate the prescription list.

prescription-detail.component.ts and prescription-detail.component.html

`prescription-detail.component.ts` determines the structure of the prescription detail page when viewing an existing prescription or adding a new prescription. It refers to its template '`prescription-detail.component.html`' and invokes required methods defined in `prescription.service.ts` in order to save , delete, promote and demote prescriptions.

`gotoDetail()` method of `prescription.component.ts` passes the prescription ID when loading `PrescriptionDetailComponent` , which in turn invokes the '`getPrescription(id:number): Observable<Prescription>{}`' method of Service class on initialization. In case a prescription ID set to '-1' is received, the method initializes a new prescription object there by enabling adding a new prescription.

4.3 Services Implementation

4.3.1 Microservices implementation with WSO2 MSF4J

Back-end services are broken down into a set of independent services, i.e. microservices functioning together realizing the overall application capabilities. These services are implemented as RESTful web services, using WSO2 Microservices Framework for Java (WSO2 MSF4J). These services make use of Java Persistence API (JPA) for data manipulation.

The service layer consists of the following Microservices.

- auth : to handle login/logout functionality
- user: to handle authorizations
- consumer : to handle consumer profiles
- drug :to handle drug details and search
- prescription :to handle prescriptions

In general each service implementation consists of the following.

- One or many entity classes
- Service Class implementing RESTful methods
- Classes handling JPA implementations

Additionally, WSO2 MSF4J makes use of annotations, which are meta data to communicate certain information to the compiler. By default it supports a subset of JAX-RS (Java API for RESTful Web Services) and additionally the use of JPA (Java Persistence API) annotations.

4.3.2 Entity Class Implementation

Entity classes define an entity which represents an object. i.e. Prescription entity class represents the Prescription object. Each of these objects directly maps to a row in the relevant database table. i.e. Prescription object maps to a row in PM_PRESCRIPTION table. This class defines the mapping of each attribute in the entity to a column in the relevant database table.

```
@Entity
@Table(name = "PM_PRESCRIPTION")
public class Prescription {
    @Id
    @TableGenerator(name = "prescription_id_gen", table = "PM_ID_GEN", pkColumnName =
"PM_GEN_NAME", valueColumnName = "PM_GEN_VAL", allocationSize = 1)
    @GeneratedValue(strategy = GenerationType.TABLE, generator = "prescription_id_gen")
    @Column(name = "PM_PRESC_ID")
    private int id;
    @Column(name = "PM_PRESC_TYPE")
    private String type;
    @Column(name = "PM_PRESC_C_STATE")
    private String state;
    @Column(name = "PM_PRESC_DATE")
    private Date date;
    @OneToOne(targetEntity = Consumer.class)
    @JoinColumn(name="PM_CONSUMER_ID")
    private Consumer consumer;
    .....
}
```

JPA Annotations `@Entity`, `@Table` and `@Column`, are used to specify that the Class is an entity class, the database table to which the entity class maps and the column in the table each variable in the entity Class maps to , respectively. `@Id` indicates the primary key, which is the column `PM_PRESC_ID` in the `PM_PRESCRIPTION` table of the database.

Certain columns in a tables are foreign keys that map to different tables. Eg; 'Consumer'.

```
    @OneToOne(targetEntity = Consumer.class)
    @JoinColumn(name="PM_CONSUMER_ID")
    private Consumer consumer;
```

The above code segment specifies the cardinality of the relationship among Prescription and Consumer entities using standard JPA annotations. One

prescription can have only one consumer. Therefore the annotation `@OneToOne` is used specify that each Prescription object corresponds to only one object of the Consumer class. This relationships is maintained in the mapping on 'consumer' variable to the `PM_CONSUMER_ID`, which acts as a foreign key in the `PM_PRESCRIPTION` table which is also the primary key of `PM_CONSUMER` table.

Constructors are defined in entity classes for the initialization of the relevant object.

Eg: Prescription Entity Class defines the following constructors.

1. Default constructor for new Prescriptions

```
public Prescription() {}
```

2. A constructor with arguments for viewing existing prescriptions

```
public Prescription(String type, String state, Date date, String medicalPractitioner,
String pharmacistComment, String consumerComment) {
    this.type = type;
    this.state = state;
    this.date = date;
    this.medicalPractitioner = medicalPractitioner;
    this.pharmacistComment = pharmacistComment;
    this.consumerComment = consumerComment;
}
```

In addition entity classes define a set of getter and setter methods for individual variables which are required by other classes/ services.

Eg:

```
public int getId() {
    return id;
}

public String getState() {
    return state;
}

public void setState(String state) {
    this.state = state;
}
```

The following segment defines the table ID generation strategy.

```
@Id
@TableGenerator(name = "prescription_id_gen", table = "PM_ID_GEN", pkColumnName = "PM_GEN_NAME",
valueColumnName = "PM_GEN_VAL", allocationSize = 1)
@GeneratedValue(strategy = GenerationType.TABLE, generator = "prescription_id_gen")
@Column(name = "PM_PRESC_ID")
```

The annotation `@Id` identifies the primary key of the entity. The solution involves JPA

ID generation strategy, 'Table strategy' [23] which generates and maintains a unique ID for a table. For this purpose , the database consists of the table PM_ID_GEN, which holds the table name and the next primary key value of each table.

4.3.3 Service Class Implementation

Service classes expose resources for data manipulation, which are accessed by the UI layer for RESTful manipulation of data.

```
@Path("/pharmacy/prescriptions")
public class PrescriptionService {
.....

    @POST
    public Response addPrescription(Prescription prescription) {
        prescription.setDate(Calendar.getInstance().getTime());
        PrescriptionState newState = prescription.getStateList().get(0);
        newState.setUpdatedTime(Calendar.getInstance().getTime());
        newState.setStatus("NEW");
        dbLink.addPrescription(prescription);
        return Response.ok()
            .entity("Prescription added successfully.").build();
    }
.....
}
```

The supported REST method is specified by annotations @GET, @POST, @PUT and @DELETE. @Path annotation specifies the resource path, where as the resource path '/' is supported by methods that do not specify a path. Eg : Path for the class is defined as '@Path("/pharmacy/prescriptions")', Each resource specifies its path within the Class. i.e. the path for addPrescription() method is '/'. Therefore the complete resource path for addPrescription() method is 'http://{host>:{port}}/pharmacy/prescriptions/'.

Each of these methods are associated with a method in JPA implementation class of the relevant service. When a service class method is invoked, in turn it invokes the associated method in JPA implementation class. The above code segment sets the Prescription date, Prescription state and then invokes the addPrescription() method in the JPA implementation class using the instance 'dbLink' of the JPA implementation

class instance.

```
dbLink.addPrescription(prescription);
```

This method triggers the database operations required for resource manipulation. (JPA implementation details are explained in the next section). Once the method is executed, a response is returned indicating the method execution status.

```
return Response.ok(.entity("Prescription added successfully.")).build();
```

Service invocation

The UI layer (Angular 2 application) communicates data to the service via a JSON message. For example when a prescription is added, Angular2 sends the prescription data to the Microservices layer in the form of a JSON object. Eg:

```
{
  "id":641,
  "type":"Manual",
  "state":"Collect",
  "date":"Dec 30, 2016 8:50:37 PM",
  "consumer":{
    "id":28,
    "firstName":"Peter",
    "lastName":"Clinton",
    "contactNo":"077734534",
    "email":"peterc@gmail.com",
    "address":"No 20, Hill Street, Colombo 5",
    "isRegisteredUser":false
  },
  "medicalPractitioner":"A.B.C. George",
  "consumerComment":"",
  "itemList":
  {
    "id":638,
    "drugId":0,
    "drugName":"Betamethazone Ointment",
    "drugStrengthId":4,
    "drugStrength":"0.10%",
    "quantity":3,
    "quantityUnit":"pill",
    "noOfTimes":1,
    "frequency":1,
    "perUnitTime":"Day(s)",
    "beforeOrAfterMeal":"before",
    "duration":1,
    "durationUnit":"Week(s)",
    "instructions":""
  }
  "stateList":[
    {
      "id":15,
      "status":"NEW",
      "updatedBy":"AsanthiK",
      "updatedAt":"Dec 30, 2016 8:50:37 PM",
      "comment":""
    },
    "comment":"verify prmoted"
  ],
}
]
```

Microservices engine serializes the incoming JSON message into a Prescription Java

object and invokes 'addPrescription' method in PrescriptioService class, which in turn invokes 'addPrescription' method of PrescriptionDBLink class by passing the Prescription object. addPrescription' method of PrescriptionDBLink Class (i.e. the JPA layer) then creates a SQL query to add the prescription and adds prescription details to relevant database tables.

4.3.4 Use of Java Persistence API (JPA) for data manipulation

JPA is a Java specification for manipulating data among relational databases and Java objects or classes. It creates a bridge between Java objects and relational database tables by exposing high level APIs and Interfaces to access, persist and manipulate data in databases there by eliminating the need for the services at service layer to specify SQL queries.

The Microservices engine converts the incoming message to a Java object and invokes a specified service. The service should define an entity class which maps to the relevant database tables to be manipulated and access/implement required JPA APIs/Interfaces in order to communicate with the database layer.

The JPA layer accepts the Java Object passed by the service layer and converts these Java objects to SQL queries and accesses the database to perform the required operations.

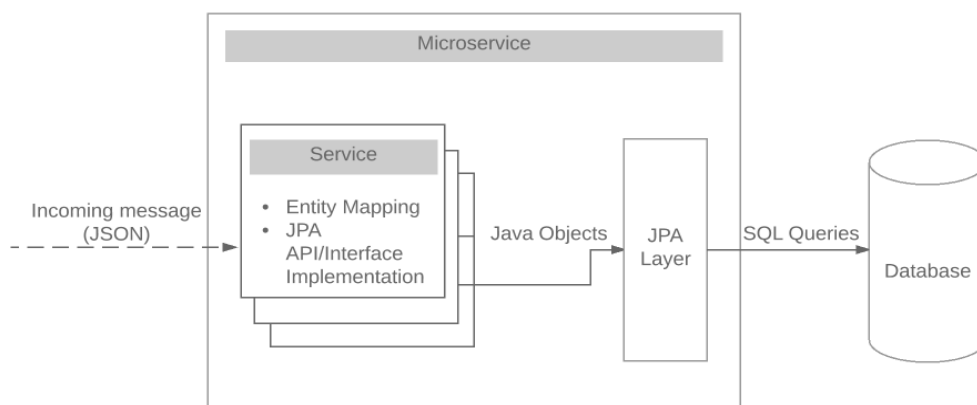


Figure 4.2 Microservices

Implementation of JPA is as follows.

EntityManager

In JPA, an instance of the interface 'EntityManager', which provides data manipulation functionality represents a database connection, . Every HTTP request makes use of an EntityManager instance. An instance of the interface 'EntityManagerFactory' is required to instantiate EntityManager instances. In order to create an EntityManagerFactory instance, the method 'createEntityManagerFactory()' of 'Persistence' class is made use of. It accepts database connection details.

```
EntityManagerFactory emf = Persistence.createEntityManagerFactory (
    PERSISTENCE_UNIT_NAME, dbConfigs);
```

This emf object is used to invoke the createEntityManager() method of 'EntityManager' to create a new EntityManager instance.

```
EntityManager em = emf.createEntityManager();
```

Database transactions (EntityTransaction) and query related operations (Query) are performed using an EntityManager instance obtained in this manner.

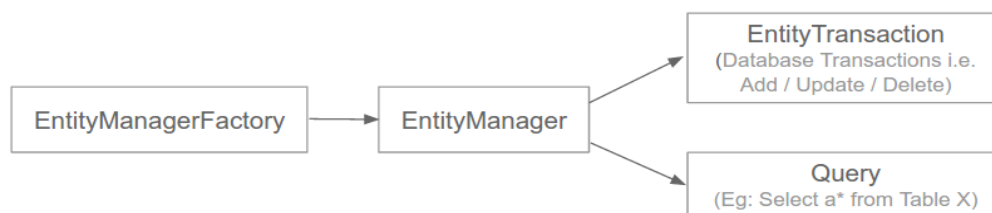


Figure 4.3 JPA basic flow

Performing Transactions

The obtained EntityManager instance 'em' is used to access the database to perform various transactions.

Operations that make database modifications (add, update, delete) are associated with the interface 'EntityTransaction'. These operations should be performed using the EntityManager's getTransaction() method.

```
Eg: em.getTransaction().begin();
```

```
em.getTransaction().commit();
```

```
em.getTransaction().rollback();
```

Also a simple search based on the primary key value can be performed using the find() method of EntityManager.

```
try {
    return em.find(Prescription.class, id);
}
finally {
    em.close();
}
```

It is also required to invoke the close() method, in order to release resources back to the EntityManagerFactory at the end of the database operation performed.

Executing Queries

There are requirements to build queries during runtime. These queries may be complex and not based on a simple primary key based search. Such queries in JPA are built using the interface 'Query'. When the result type of the query is unknown or in case it returns polymorphic results, the query is assigned to an instance of interface type 'Query'. On the other hand, if the result type is known, an instance of the interface 'TypedQuery', which is a sub interface of 'Query', is used to hold the query. The service layer makes use of TypedQueries as the query result types are always known.

A TypedQuery instance cannot be acquired directly. This is done via the EntityManager.

Initially, an instance of the interface 'CriteriaBuilder' is created using the operation 'getCriteriaBuilder' of EntityManager instance 'em'.

```
CriteriaBuilder cb = em.getCriteriaBuilder();
```

The CriteriaBuilder operation 'createQuery' is used to build a 'CriteriaQuery' instance 'cq'. By associating it with the required entity type (i.e. Prescription in this case).

```
CriteriaQuery<Prescription> cq=criteriaBuilder.createQuery(Prescription.class);
```

'Root' is the entity which defines the basis for all attributes, joins and paths in the query. A root instance is generated using the from() method of CriteriaQuery instance accepting the relevant class.

```
Root<Prescription> rootEntry = cq.from(Prescription.class);
```

The select operations are performed by passing the Root instance to the select()

method, to create a new CriteriaQuery instance, using the previously created CriteriaQuery instance.

```
CriteriaQuery<Prescription> all = cq.select(rootEntry);
```

(Here, all elements in the Root instance 'rootEntry' , i.e all Prescription entries are selected.)

Once the CriteriaQuery instance holding all information required for the query is obtained in this manner, it is used to create the query using createQuery() method of the EntityManager instance. The result type of the query is known, therefore it is held in a 'TypedQuery' object instance.

```
TypedQuery<Prescription> tq = em.createQuery(all);
```

The operation getResultList() of 'TypedQuery' is invoked using the above created instance to hold the results. The retrieved results can be returned as follows.

```
return tq.getResultList();
```

The Figure 4.5 depicts the flow of control form EntityManager to Query.

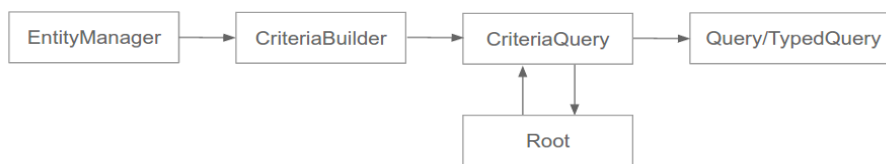


Figure 4.4 JPA Query control flow

4.3.5. JPA implementation within a Microservice

Each Microservice has a service class and JPA implementation class. The JPA implementation class handles database operations required for the relevant microservice.

In the prescription microservice, this is PrescriptionDBLink class. Methods in this class handle database operations corresponding to resource manipulation methods in the PrescriptionService Class. When a resource manipulation method in PrescriptionService class is invoked, it in turn invokes the corresponding method in PrescriptionDBLink Class that performs database operations.

In order for data manipulation methods to function, an EntityManagerFactory instance should be available. This is passed on to the JPA implementation class by the main class (Application class) during service initialization at start up via the service class.

The Application class refers to a set of constants that refer to the application and its database connection details in a separate class named 'Constants';

```
public class Constants {
    public static class Database {
        public static final String PERSISTENCE_JDBC_URL =
            "javax.persistence.jdbc.url";
        public static final String PERSISTENCE_JDBC_USER = "javax.persistence.jdbc.user";
        public static final String PERSISTENCE_JDBC_PASSWORD =
            "javax.persistence.jdbc.password";
        public static final String C3P0_MAX_CONNECTION_POOL_SIZE = "hibernate.c3p0.max_size";
    }
    public static final String PERSISTENCE_UNIT_NAME = "org.abc.pharmacy.unit";
}
}
```

These constants are passed into a Map named 'dbConfigs' defined in the Application class to form the database connection details.

```
public class Application {
    public static void main(String[] args) {
        Map<String, Object> dbConfigs = new HashMap<>();
        String url = "jdbc:mysql://localhost:3306/pharmacy_db?autoReconnect=true";

        if (url != null) {
            dbConfigs.put(Constants.Database.PERSISTENCE_JDBC_URL, url);}
        String user = "root";
        if (user != null) {
            dbConfigs.put(Constants.Database.PERSISTENCE_JDBC_USER, user);}
        String password = "root";
        if (password != null) {
            dbConfigs.put(Constants.Database.PERSISTENCE_JDBC_PASSWORD, password);}
        String maxSize = "100";
        if (maxSize != null) {
            dbConfigs.put(Constants.Database.C3P0_MAX_CONNECTION_POOL_SIZE, maxSize);}

        .....
    }
}
```

The Application class also creates an EntityManagerFactory instance, by invoking the createEntityManagerFactory() method of Persistence class. This method accepts the 'dbConfigs' which contains database connection details related to the application which is identified by the constant ' PERSISTENCE_UNIT_NAME ' .

```
public class Application {
    public static void main(String[] args) {
        ....
        EntityManagerFactory emf;
        try {
            emf = Persistence.createEntityManagerFactory(
                PERSISTENCE_UNIT_NAME,
                dbConfigs);
        }
        .....
    }
}
```

When starting the microservices, each service class is instantiated via its constructors

within the main class, which is the Application class.

```
public class Application {
    ....
    EntityManagerFactory emf;

    new MicroservicesRunner()
        .deploy(new PrescriptionService(emf))
        .start();
    ...
}
```

Application class passes the EntityManagerFactory object, 'emf' to service class constructor. For this, service class defines a constructor accepting an EntityManagerFactory object.

```
public class PrescriptionService {
    private PrescriptionDBLink dbLink;
    public PrescriptionService(EntityManagerFactory emf) {
        dbLink = new PrescriptionDBLink(emf);
    }
    ...
}
```

This constructor instantiates an object of the JPA implementation class (eg: PrescriptionDBLink class) named 'dbLink' , by invoking a constructor accepting an EntityManagerFactory implemented in the JPA implementation class. This way, the EntityManagerFactory instance is passed over to the JPA implementation class.

```
public class PrescriptionDBLink {
    private EntityManagerFactory emf;
    public PrescriptionDBLink(EntityManagerFactory emf) {
        this.emf = emf;
    }
}
```

When the microservice is started , objects of the service class and the JPA implementation class are instantiated in this manner.

When a service method is invoked it invokes the relevant JPA implementation class method which triggers the database transaction related to the operation.

```
public class PrescriptionService {
    .....
    public Response addPrescription(Prescription prescription) {
        .....
        dbLink.addPrescription(prescription);
        return Response.ok()
            .entity("Prescription added successfully.")
            .build();
    }
    .....
}
```

This method invokes the relevant operation in the JPA implementation class using the JPA implementation class object instantiated at the service start up.

```
public class PrescriptionDBLink {
    .....
    public void addPrescription(Prescription prescription) {
```

```

EntityManager entityManager = null;
try {
    entityManager = emf.createEntityManager();
    entityManager.getTransaction().begin();
    entityManager.persist(prescription);
    if (prescription.getItemList() != null) {
        for (PrescriptionItem item : prescription.getItemList()) {
            item.setPrescription(prescription);
            entityManager.persist(item);
        }
    }
    if (prescription.getStateList() != null) {
        for (PrescriptionState state : prescription.getStateList()) {
            state.setPrescription(prescription);
            entityManager.persist(state);
        }
    }
    if (prescription.getConsumer() != null) {
        entityManager.persist(prescription.getConsumer());
    }
    entityManager.getTransaction().commit();
} catch (PersistenceException e) {
    if (entityManager != null) {
        entityManager.getTransaction().rollback();
    }
    throw new RuntimeException("Exception occurred while connecting to the database: ", e);
} finally {
    if (entityManager != null && entityManager.isOpen()) {
        entityManager.close();
    }
}
}
}
}

```

During the method invocation, an EntityManager object , 'entityManager' is created by invoking the createEntityManager() method of EntityManagerFactory interface.

This object is used to acquire and begin the database transaction using getTransaction() and begin() methods.

```
entityManager.getTransaction().begin();
```

The persist() method persists an instance of an Entity class. Prescription items and the prescription state is also persisted as in this method implementation.

```
entityManager.persist(prescription);
```

Once data persistence is done, the transaction is committed.

```
entityManager.getTransaction().commit();
```

In case an exception occurs the transaction needs to rollback.

```
entityManager.getTransaction().rollback();
```

Finally, it is closed to release all EntityManager resources acquired.

```
entityManager.close();
```

JPA implementation class also consists of methods performing various database queries, such as filtering a list of Prescriptions.

```

public class PrescriptionDBLink {
.....
    public List<Prescription> getAllPrescriptions() {
        EntityManager entityManager = emf.createEntityManager();

```



```

CriteriaBuilder criteriaBuilder = entityManager.getCriteriaBuilder();
CriteriaQuery<Prescription> pCriteriaQuery =
    criteriaBuilder.createQuery(Prescription.class);
Root<Prescription> rootEntry = pCriteriaQuery.from(Prescription.class);
CriteriaQuery<Prescription> all = pCriteriaQuery.select(rootEntry);
TypedQuery<Prescription> allQuery = entityManager.createQuery(all);
.....
}
.....
}

```

As explained in the previous section on JPA query implementation, these methods make use of an EntityManager instance to build a CriteriaBuilder instance which in turn is used to build a new CriteriaQuery instance. This CriteriaQuery instance is used to build a Root instance. The Root instance passed to the previous CriteriaQuery instance to build another CriteriaQuery instance, containing all details required for the query. This instance is used to build the query and is assigned to an instance of TypedQuery. The query formed is executed and assigned to a List of type 'Prescription', as there are multiple results returned.

```

public class PrescriptionDBLink {
.....
    public List<Prescription> getAllPrescriptions() {
        .....
        List<Prescription> prescriptionList = allQuery.getResultList();
        .....
        return prescriptionList;
    }
.....
}

```

4.3.6. Service Start up

The Class 'Application' is the main class of the microservices implementation project. As explained in the previous section, in order for a service to start the service constructors accepting EntityManagerFactory objects need to be instantiated via the MicroservicesRunner().deploy() method. As follows.

```

new MicroservicesRunner()
    .deploy(new DrugManufacService(emf))
    .deploy(new DrugMgtService(emf))
    .deploy(new PrescriptionService(emf))
    .deploy(new ConsumerService(emf))
    .deploy(new AuthService(emf))
    .start();

```

Once the service is added here, and the application is run, added services will be accessible.

4.3.7. Sample service

Eg: 'prescriptions' service in the application contains the following classes.

The 'prescription' service consists of the following Classes.

- Entity classes : Prescription, PrescriptionItem, PrescriptionState, OnlinePrescription
- RESTful resource implementation class : PrescriptionService
- JPA implementation class : PrescriptionDBLink

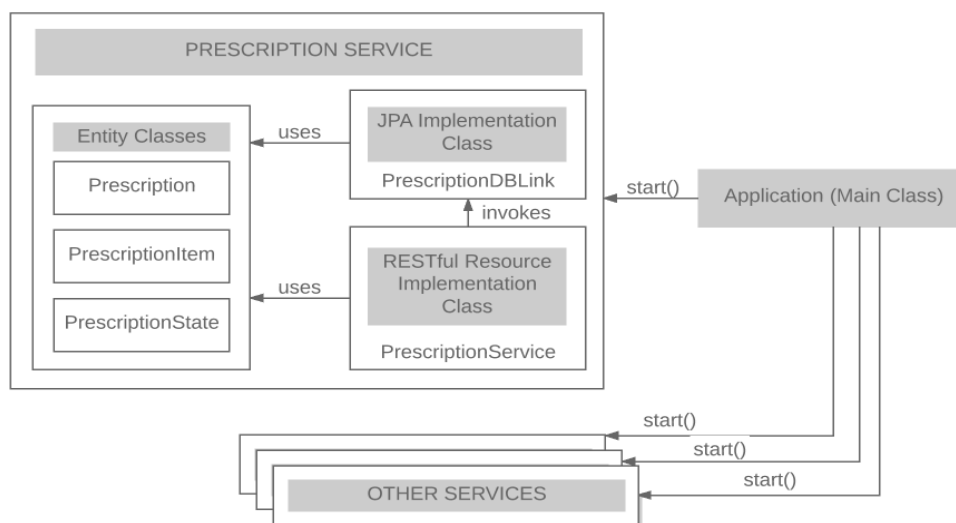


Figure 4.5 Microservices Structure

4.3.8. Exposed resources

Given below are the resources and sample endpoint URLs of back end resources exposed via these services.

Service	CRUD operation	Description	Resource Endpoint URL
Drug	GET	All drugs	http://localhost:8080/pharmacy/drugs/
		Strengths of a drug	http://localhost:8080/pharmacy/drugs/6/strengths
Prescription	GET	All prescriptions	http://localhost:8080/pharmacy/prescriptions
		Specific prescription	http://localhost:8080/pharmacy/prescriptions/639
	POST	Promote a prescription	http://localhost:8080/pharmacy/prescriptions/644/promote
Demote a prescription		http://localhost:8080/pharmacy/prescriptions/644/demote	
Consumer	GET	All consumers	http://localhost:8080/pharmacy/consumer
		Specific consumer	http://localhost:8080/pharmacy/consumer/39
	POST	Consumer	http://localhost:8080/pharmacy/consumer/register
Auth	GET	Login	http://localhost:8080/pharmacy/auth/login
		Logout	http://localhost:8080/pharmacy/auth/logout

Table 4.1 Exposed resources

4.4 Business Process Implementation

4.4.1 Prescription processing business process

The involvement of the business process within the application is shown in Figure 4.8 below.

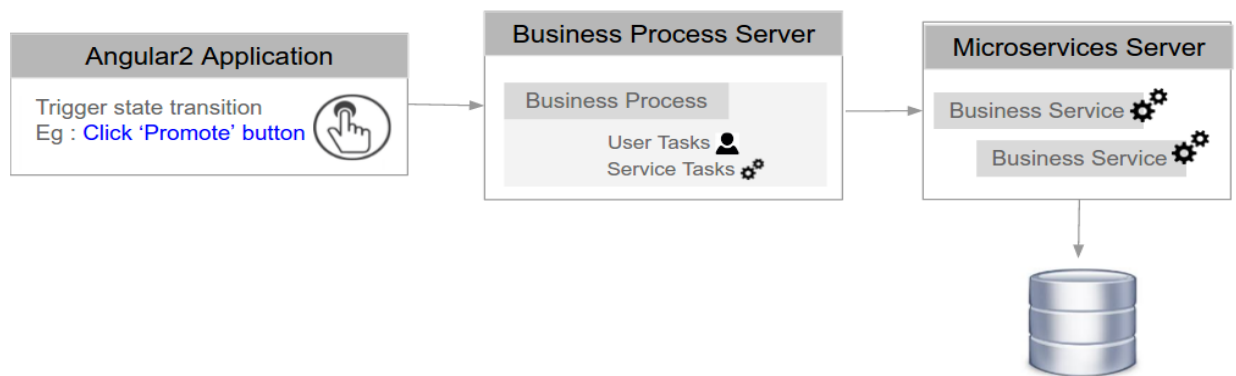


Figure 4.6 Business process involvement in the solution

Using BPMN2.0 notation the business process can be modeled in the form of an illustration similar to a state transition diagram. A business process defined this way acts as a blue print representing the life cycle of a particular process, which in the case of this project, the prescription life cycle.

How the business process works

For each prescription created, a new instance of the business process is generated, which is associated with the prescription ID, also known as the 'Correlation ID' in the context of BPMN notation.

A business process instance created in this manner stays active from start to the completion of the defined prescription life cycle. This instance is aware of its current state in the life cycle.

When the status of a prescription needs to be updated, a service call is made to the business process engine (i.e. Activiti engine), which is the container of business processes.

The business process engine identifies, the relevant business process depending on the

request and then the request is dispatched to the business process instance associated with the prescription by mapping the Correlation ID.

The service call updates status of the instance and invokes the back-end services associated with it to update the prescription state in the database.

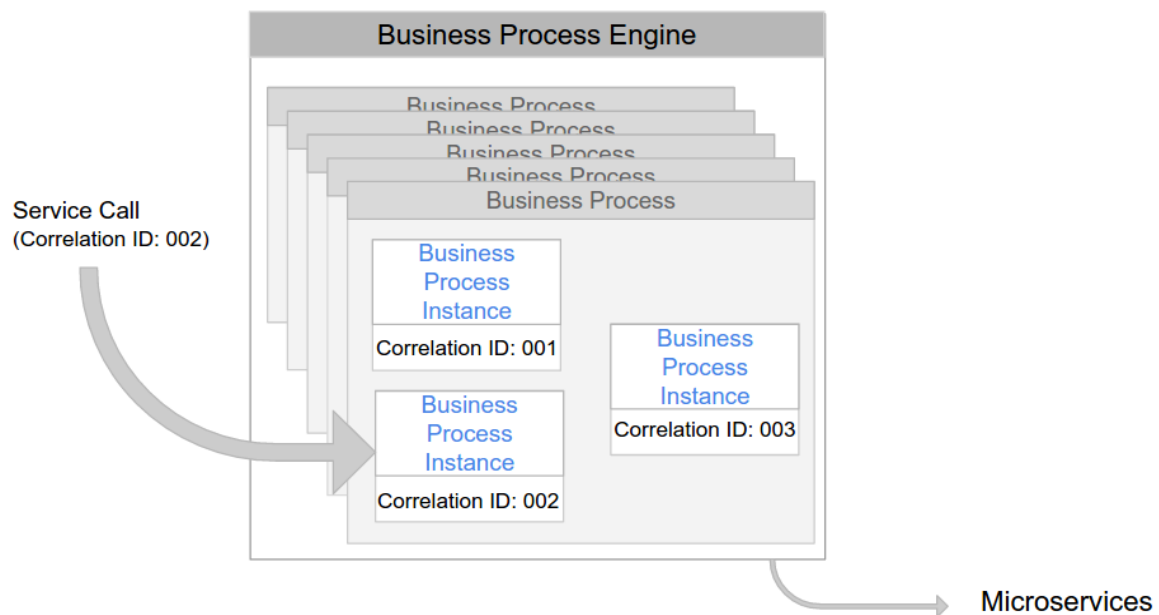


Figure 4.7 Business Process Engine

4.4.2 Business Process Illustration using Activiti Eclipse 2.0 Designer

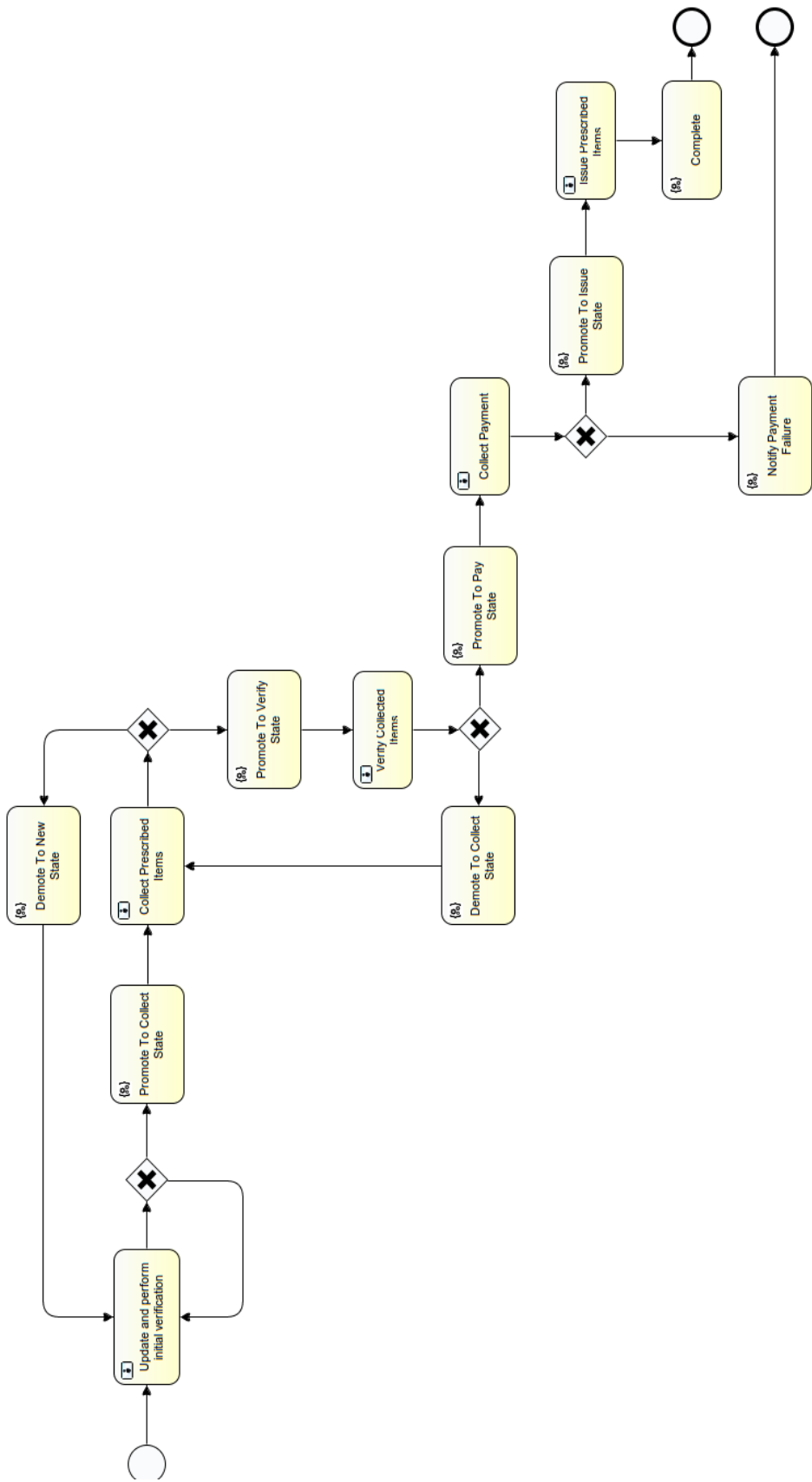
A typical business process is composed of a set of sequentially executed tasks. Activiti Engine, which forms the basis of the Business Process Server supports several task types namely,

- Start Task : The task that marks the beginning of the flow
- Java Service Task : Tasks that are used to invoke external Java classes
- User Task : Tasks that are used to model an action carried out by a human actor. (Eg: Promoting or demoting the prescription)
- Exclusive Gateway : A gateway controls the execution flow. It uses process variables to determine the path to continue the process.

When the business process required for the solution is considered it requires all of the above task types.

Given in Figure 4.10 is the Business Process implemented using the Eclipse BPMN 2.0 designer plug-in on eclipse.

When the business process is illustrated using BPMN2.0 notation, the business process code is automatically generated in XML



4.8 Prescription handling business process in BPMN2.0 notation

Figure

Task list

Each user task is mapped to an associated service task and each service task is mapped to the relevant back end Microservice endpoint URL in their property definitions.

Given below is a list of user tasks and associated service tasks identified for the process.

Start Task	User Task	Service Task	End Task
Start process	Update and perform initial verification	Promote to collect state	
	Collect prescribed items	Promote to verify state	
		Demote to new state	
	Verify collected items	Promote to pay state	
		Demote to collect state	
	Collect payment	Promote to issue state	
		Notify payment failure	End process
	Issue prescribed items	Complete	End process

Table 4.2 Business Tasks

Next section contains an explanation of how different components of the business process were implemented.

Process Start

Process start event is added by specifying a name.

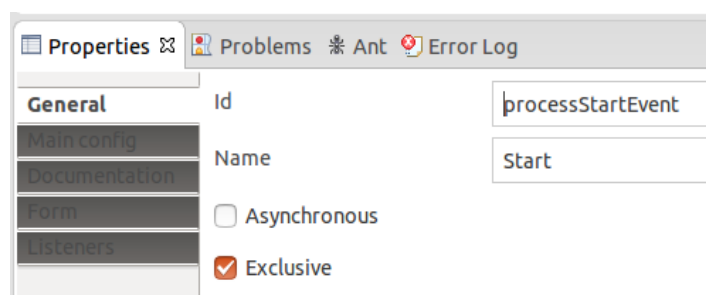
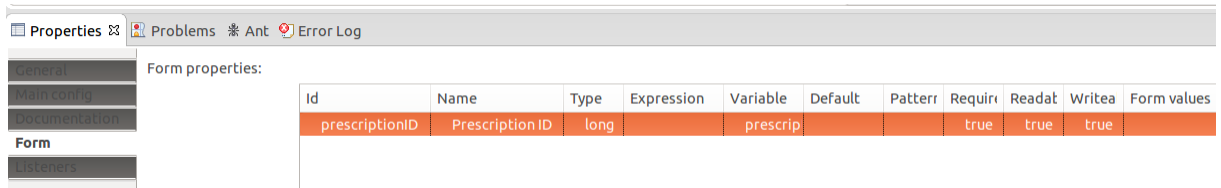


Figure 4.9 Process Start – General



Id	Name	Type	Expression	Variable	Default	Pattern	Required	Readable	Writable	Form values
prescriptionID	Prescription ID	long		prescrip			true	true	true	

Figure 4.10 Process Start - Form

When a business process instance is started for a prescription, the prescription ID is set.

User tasks

The service calls coming from the UI layer (Angular 2 Application), invoke the user tasks.

Each user tasks should be given an ID and a task name.

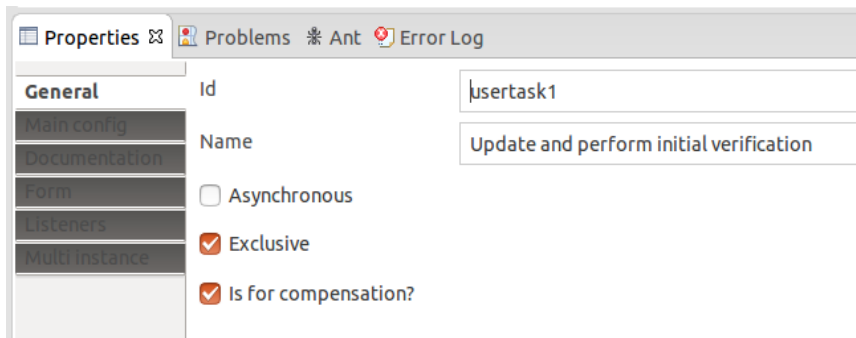
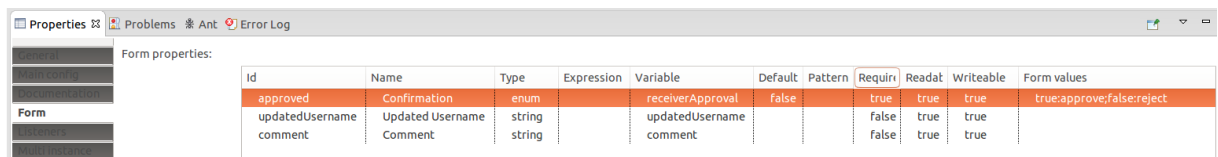


Figure 4.11 User task - General

Each user task in the business process is associated with 3 parameters as follows.

- Approved : A boolean value indicating that the user task is triggered
- updatedUsername: User name of the user who performed the user task
- comment: User comments



Id	Name	Type	Expression	Variable	Default	Pattern	Required	Readable	Writable	Form values
approved	Confirmation	enum		receiverApproval	false		true	true	true	true:approve,false:reject
updatedUsername	Updated Username	string		updatedUsername			false	true	true	
comment	Comment	string		comment			false	true	true	

Figure 4.12 User task - Form

Autogenerated XML content for the user task is as follows.

Eg:

```
<userTask id="usertask1" name="Update and perform initial verification" activiti:assignee="admin"
activiti:candidateGroups="admin">
  <extensionElements>
    <activiti:formProperty id="approved" name="Confirmation" type="enum"
variable="receiverApproval" default="false" required="true">
      <activiti:value id="true" name="approve"></activiti:value>
      <activiti:value id="false" name="reject"></activiti:value>
    </activiti:formProperty>
    <activiti:formProperty id="updatedUsername" name="Updated Username" type="string"
variable="updatedUsername"></activiti:formProperty>
    <activiti:formProperty id="comment" name="Comment" type="string"
variable="comment"></activiti:formProperty>
  </extensionElements>
</userTask>
```

Service tasks

Each service tasks should be given an ID and a task name.

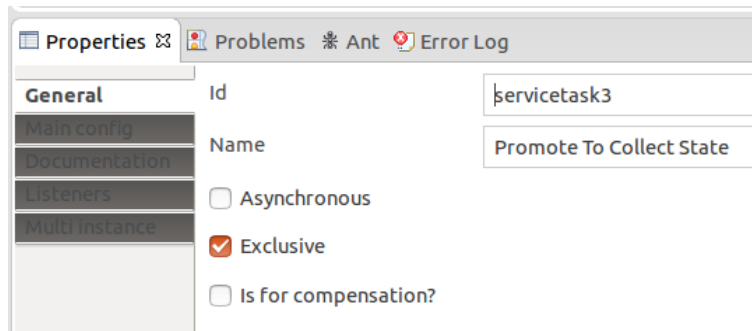


Figure 4.13 Service task – General

Each triggered service task invokes the back end operation updating the database based on the promote/demote action performed.

Eg: When the 'Promote to collect state' service task is triggered, it invokes the back end microservice operation with the URL

['http://localhost:8090/msf4j/pharmacy/prescriptions/\\${prescriptionID}/promote'](http://localhost:8090/msf4j/pharmacy/prescriptions/${prescriptionID}/promote).

The HTTP request with the input payload and transport headers are constructed in this configuration.

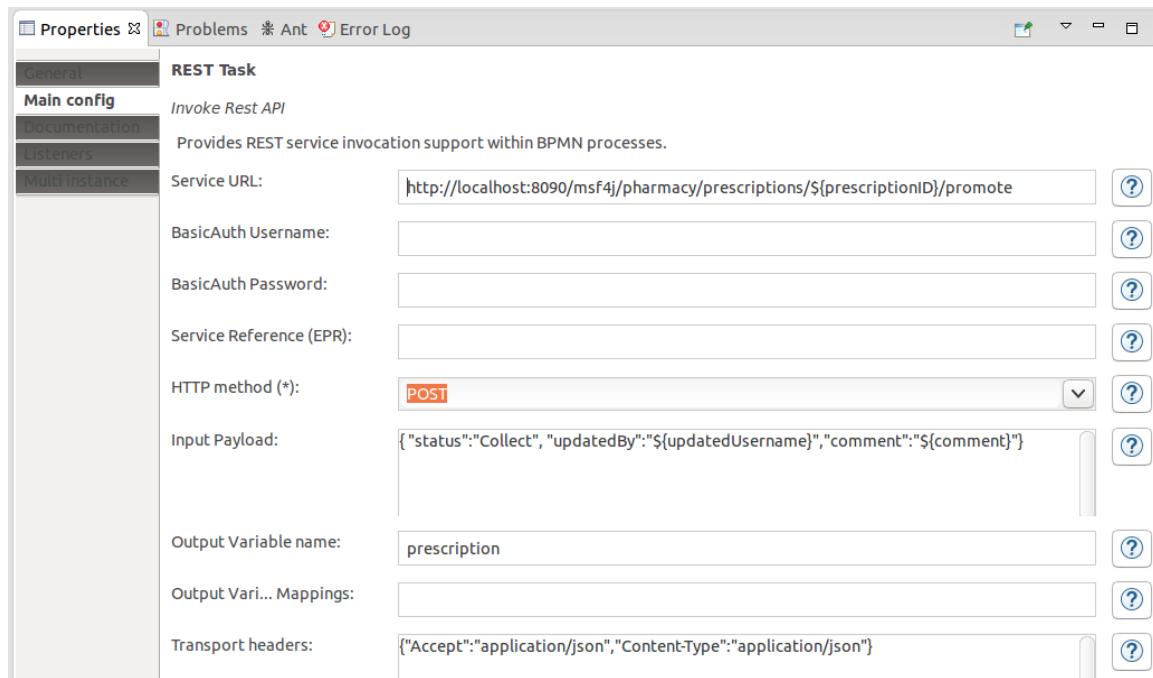


Figure 4.14 Service task – Main config

Auto generated XML content for the service task is as follows.

Eg:

```
<serviceTask id="servicetask3" name="Promote To Collect State"
  activiti:class="org.wso2.carbon.bpmn.extensions.rest.RESTTask"
  activiti:extensionId="org.wso2.developerstudio.bpmn.extensions.restTask.RESTTask">
  <extensionElements>
    <activiti:field name="serviceURL">
      <activiti:expression>http://localhost:8090/msf4j/pharmacy/prescriptions/{
prescriptionID}/promote</activiti:expression>
    </activiti:field>
    <activiti:field name="method">
      <activiti:string>POST</activiti:string>
    </activiti:field>
    <activiti:field name="outputVariable">
      <activiti:string>prescription</activiti:string>
    </activiti:field>
    <activiti:field name="input">
      <activiti:expression>{ "status": "Collect", "updatedBy": "${updatedUsername}", "comment": "${
comment}" }</activiti:expression>
    </activiti:field>
    <activiti:field name="headers">
      <activiti:string>{ "Accept": "application/json", "Content-
Type": "application/json" }</activiti:string>
    </activiti:field>
  </extensionElements>
</serviceTask>
```

Exclusive Gateway

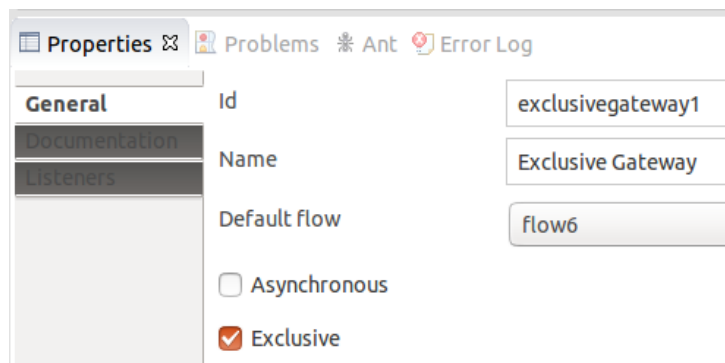


Figure 4.15 Exclusive gateway – General

An exclusive gateway decides on the actions to perform on a condition. The negative flow is marked as the default.

Auto generated XML content for the exclusive gate is as follows.

Eg:

```
<exclusiveGateway id="exclusivegateway1" name="Exclusive Gateway"
default="flow6"></exclusiveGateway>
```

Message flow

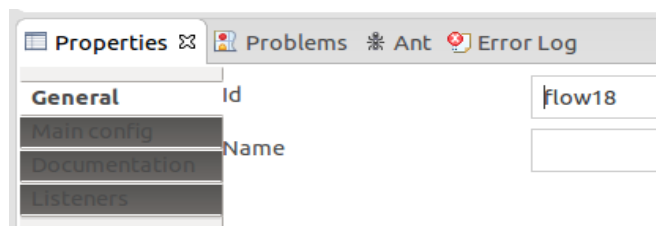


Figure 4.16 Message Flow – General

All message flows that are associated with a 'promote' action, are marked with a condition, which requires the 'approved' parameter value to be 'true'.

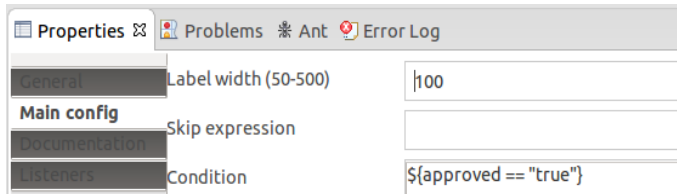


Figure 4.17 Message Flow – Main config

Auto generated XML content for the message flow is similar to the following.

Eg:

```
<sequenceFlow id="flow18" sourceRef="exclusivegateway1" targetRef="servicetask3">
  <conditionExpression xsi:type="tFormalExpression"><![CDATA[${approved ==
"true"}]]></conditionExpression>
</sequenceFlow>
```

Process End

Process end event is defined as follows.

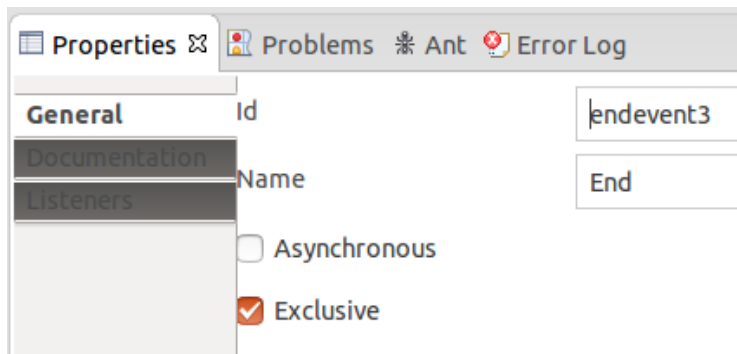


Figure 4.18 End event – General

4.4.3 Invocation and deployment of Business Process

A deployment artifact was created with .bar extension for the implemented business process and deployed on the WSO2 Business Process Server 3.6.0.

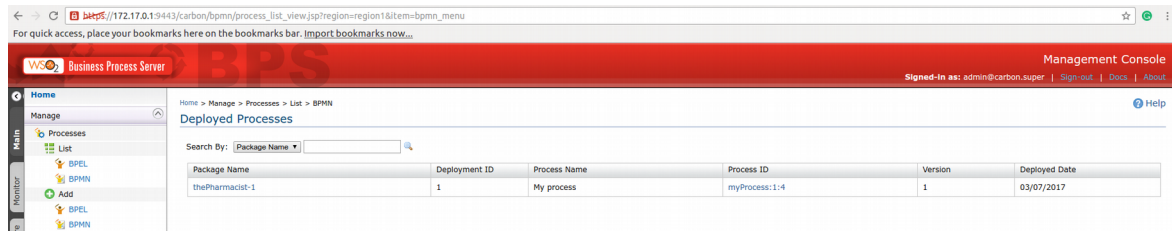


Figure 4.19 Business process running on WSO2 Business process server

The service calls coming from the UI layer (Angular 2 Application), invoke the users tasks on this business process deployed on WSO2 BPS 3.6.0.

The implementation details from business process invocation to prescription status update in the database are as follows.

Initially a business process instance associated with a prescription is started. For this the Angular 2 application contains the 'addPrescription()' method in 'prescription-detail.component.ts' which invokes the 'initBPMNProcess(id:string)' method of 'prescription.service.ts' along with a prescription ID to initiate a business process instance for a prescription.

Then, when a status update is made, the relevant business process instance and the associated user task needs to be identified and invoked. For this, the 'prescription-detail-component.ts' invokes 'promotePrescription(id: number, state: PrescriptionState)' or 'demotePrescription(id: number, state: PrescriptionState)' methods in 'prescription.service.ts' according to the type of status update made. These methods retrieve the business process instance ID associated with the prescription ID. Using the process instance ID the next user task in the sequence is identified. This user task is invoked by invoking the following method in 'prescription.service.ts'.

```
invokeBPMNTask(taskID: number, approve: string, updatedUsername: string, comment: string)
```

The triggered user task in turn triggers the associated service task according to its type of status update made (promote or demote). i.e. If the user task is 'Update and perform initial verification', the service task 'Promote to collect state' is triggered.

Then the invoked service task invokes the associated microservice operations to update the prescription service in the database by constructing a request with the details provided in the service task configuration.

Eg:

Service URL : <http://localhost:8090/msf4j/pharmacy/prescriptions/{prescriptionID}/promote>

{prescriptionID}/promote

HTTP method : POST

Input payload : {"status":"Collect","updatedBy":"\$

{updatedUsername}","comment":"\${comment}"}

Transport headers : {"Accept":"application/json","Content-Type":"application/json"}

4.5 Hosting the application

4.5.1 Use of Nginx

NodeJS (Run time environment of the Angular2 Application) exposes its services via HTTP port 4200. (i.e. <http://localhost:4200>) WSO2 MSF4J exposes its Java services via HTTP port 8080. (i.e. <http://localhost:8080>) and WSO2 BPS exposes its services via HTTP port 9763.

The browser communicates with NodeJS, WSO2 MSF4J and WSO2 BPS when operating the application. But the browser is not capable of dealing with requests from 3 different origins, due to the same origin policy. The origin is the combination of URI scheme, host name and port. Port differences create the difference in origins of NodeJS, WSO2 MSF4J and WSO2 BPS. This creates the need to expose all 3 endpoints to the browser via the same port. Nginx comes into play, to facilitate this.

Nginx is an open source reverse proxy server that is used for load balancing. In the solution it exposes services of NodeJS, WSO2 MSF4J and WSO2 BPS' via a

common defined port, so that requests tall 3 origins appear to be to and from the same port to the browser.

Nginx Configuration

```
server {
    listen      8090;
    server_name localhost;

    location /msf4j/ {
        proxy_pass      http://localhost:8080/;
        proxy_redirect  http://localhost:8080 http://localhost:8090/msf4j;
    }
    location /bpmn/ {
        proxy_pass      http://localhost:9763/;
        proxy_redirect  http://localhost:9763 http://localhost:8090/bpmn;
    }
    location / {
        proxy_pass      http://localhost:4200/;
        proxy_redirect  http://localhost:4200 http://localhost:8090;
    }
}
```

'server' configuration specifies the host and port of requests, Nginx listens to. In the configuration , the Nginx server listens to port 8090 from the localhost.

'proxy pass' specifies which URL to forward the requests hitting a specified 'location' of the above host/port combination. '. I.e. Requests targeting <http://localhost:8090/msf4j/> will be forwarded to , r<http://localhost:8080> requests targeting <http://localhost:8090/> will be forwarded to <http://localhost:4200> and requests targeting <http://localhost:8090/bpmn/> will be forwarded to <http://localhost:9763>.

'proxy redirect' specifies the mapping of requests originating from other servers to the host/port exposed via Nginx. I.e Requests coming from <http://localhost:8080> will be mapped to the target URL <http://localhost:8090/msf4j/>, requests coming from <http://localhost:4200> will be mapped to the target URL <http://localhost:8090/> and requests coming from <http://localhost:9763> will be mapped to the target URL <http://localhost:8090/bpmn/>.

This way, Nginx exposes the port 8090 to the browser, so that the browser is perceived to think that it only deals with port 8090. Nginx takes care of port mapping of requests with ports of NodeJS, WSO2 MSF4J and WSO2 BPS.

4.5.2 Deployment summary

The summary of the complete deployment is as follows.

Layer / Proxy Server	Application Type	Hosted Server
UI	Angular 2	NodeJS v6.9.1
Business Process	BPMN2.0 application	WSO2 Business Process Server 3.6.0
Business Services	Java (JDK 1.8.0)	WSO2 Microservices Framework for Java 2.1.0
Database	-	MYSQL Server 5.6.33
Proxy server	-	Nginx 1.4.6

Table 4.3 Deployment summary

The application consists of an Angular 2 application , BPMN Application and a Java Application interacting together via Nginx while reading and writing data from and to a MySQL database.

4.6 User Interfaces and User Manual

Appendix A contains the User manual for the developed application.

4.7 Summary

The chapter consisted of implementation details of the solution with relevance to each layer along with the used third party components and solution specific implementation details. In each layer, the implementation approach followed is explained within the context of the used third party component and how interconnections among layers is implemented. Next the author approaches testing and evaluation of the implemented solution.

Chapter 5 : Evaluation & Testing

5.1 Introduction

In this chapter, the approach followed to test the application is described and the implemented application is assessed with a set of identified test cases. Issues identified throughout the process were fixed and the results of the final round of tests is included in the test results. The application is also subjected to an evaluation process of which the results are analyzed in order to identify the status and potential for improvement.

5.2 Testing

5.2.1 Test Approach

A test plan was derived to identify and document the test scope, test requirements and test approaches. (Appendix B).

Based on the test plan, the following testing technique were involved.

- Black box testing (Functional a security scenarios)

This was identified as the most suitable testing technique to test the application. This technique tests the application from a user perspective, which is important as majority of the target audience will be non-technical (general public and pharmaceutical staff), which makes the accuracy and ease of operation a priority.

- White box testing (Back end service validation)

This was used to validate the back end Micro service (Drug service, Prescription service , Consumer service and Auth Service) operations.

5.2.2 Test Results

Functional Tests

The following test cases were identified to verify application functionality (Appendix C).

- Generic login/logout functionality
- Consumer UI features for consumer registration, prescription upload and prescription viewing and status verification
- Pharmacy UI features for prescription listing, prescription creation
- Prescription processing work flow

ID	Test case	Status
TC 1	Consumer Registration	Passed
TC 2	Login with valid credentials	Passed
TC 3	[Negative] Login with invalid credentials	Passed
TC 4	Logout from the system	Passed
TC 5	Consumer uploads a prescription online	Passed
TC 6	Prescription state visibility to consumers	Passed
TC 7	Consumer loads a prescriptions in his prescription list	Passed
TC 8	Role based authorization	Passed
TC 9	Receiver views online prescription	Passed
TC 10	Receiver rejects online prescription	Passed
TC 11	Receiver adds new prescription in the system	Passed
TC 12	Verification of role based prescription list visibility to Pharmacy staff	Passed
TC 13	Verification of prescription state promotion	Passed
TC 14	[Negative] Verification of prescription state demotion	Passed

Table 5.1 Test Cases

Back-end service Validation

The tests were carried out using a test script developed using open source testing tool, Apache JMeter 2.13. (Appendix D). The back-end service resources specified under

section '4.3.3. Exposed resources' are validated via a JMeter script invoking the back-end services.

The test summary report generated by JMeter is given in Figure 5.1.

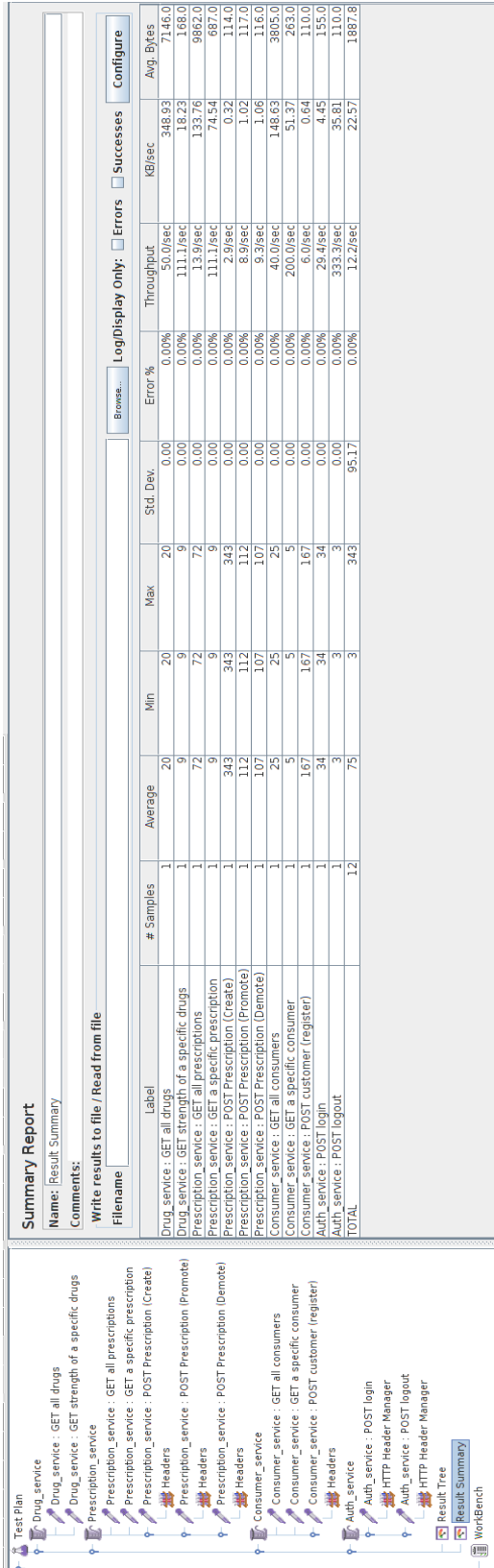


Figure 5.1 JMeter summary report

5.3 Evaluation

5.3.1 Evaluation approach

The evaluation technique used to evaluate the project are as follows

- Questionnaire /demonstration based interviews targeting pharmaceutical professionals
- Questionnaire followed by application description targeting 20 potential consumers

The questionnaires used can be found in Appendix E.

5.3.2 Analysis of evaluation feedback by pharmaceutical professionals

The evaluation was carried out with the feedback of pharmaceutical professionals running the following pharmacies.

- Sams Pharmacy, Stanley Tilakaratne Mawatha, Nugegoda
- Chemicine Pharmacy, Colombo 10, Sri Lanka

Evaluation of the application was carried out based on the following criteria.

Comprehensiveness

The application was perceived as sufficient or fulfilling minimum requirements of the process by evaluators. Features such as Stock handling, billing and alerting drug conflicts (Eg: Allergies) when adding drugs to prescriptions were suggested as improvements in serving consumers better.

Usefulness

Evaluators rated the application as useful to their business, in the events where there are many prescriptions to be processed simultaneously. The opinion is also biased

based on the evaluator and pharmaceutical staff's familiarity with computerized applications and computer literacy and pharmacy's workload.

Discrepancy avoidance

As per the feedback, prescriptions with a large number of drug items are more error prone when handled by several staff members. The application addresses a clear separation of tasks among various parties involved in the process, there by giving more focus and attention to a specific stage of the prescription processing task at a time. The application was perceived as a solution to avoid discrepancies by the evaluators. The application logs all users involved with various stages of a prescription. This was identified as useful, to evaluate mistakes and discrepancies made during the process.

Time saving factor

Evaluators stated that the application saves time by avoiding common discrepancies such as human errors with handling prescriptions with many items. They also stated that the application may not save much time, compared to the manual process, as there will be a learning curve for the staff to mature to a point where the application can be used efficiently. In addition manual handling of prescriptions with one or two items, may be more efficient than handling it via the application through a life cycle process.

It should be noted that the evaluator opinion on this is biased based on the workload and staff availability at the pharmacy. As the solution targets large scale pharmaceutical businesses, where a large number of prescriptions are handled simultaneously, the process needs to be regulated. Therefore even though a step by step process may not be required for such small prescriptions, the application is a good solution for simultaneous handling of prescriptions..

Relevance of customer UI

It was revealed that computer literate regular customers with frequent access to the Internet would make use of the customer UI. Depending on their familiarity and reliance on computerized applications and preference to share medical details over

Internet. Evaluators also stated that most of the customers would prefer manual submission/process as their reliance on the pharmacy is mostly built based on of face-to-face interaction.. The evaluators exhibited a certain amount of reluctance to accept a prescription without seeing the actual physical prescription to start the process. Despite the reluctance, they mentioned that it may not be an issue for prescriptions with routine/common, non-addictive or non-abusive drugs.

User friendliness

The application was found user friendly by evaluators, as they already use computerized applications for billing. It should be noted that the familiarity with handling the process using a computerized system affects the opinion on user friendliness.

5.3.3 Analysis of evaluation feedback by potential consumers

Application evaluation by potential consumers was carried out using a questionnaire distributed among 15 potential consumers belonging to different professions including medical, IT and other professions.

Evaluators representing consumers provided positive feedback and preference for consumer related features. Remote accessibility including online prescription upload and status tracking features was perceived as a solution for high time consumption in manual process.

The evaluation results were analyzed under the following criteria.

Consumer preference to upload prescriptions online

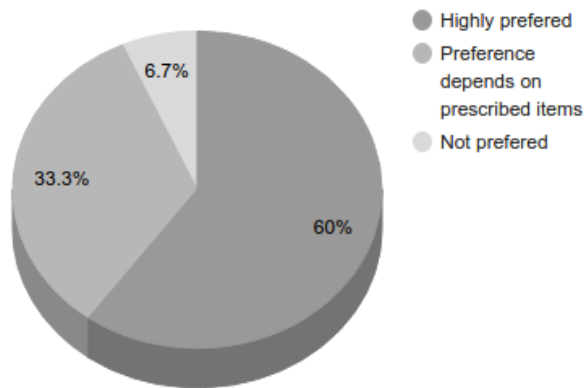


Figure 5.2 Customer preference to upload prescriptions online

Evaluators preferred the online prescription submission feature either always or depending on the prescribed items. A very small percentage did not seem comfortable with using the feature stating their preference for direct interaction with the pharmacy. It should be noted that evaluators who may have queries on prescribed drugs/items may prefer manual hand over of prescriptions to create opportunity to resolve queries before prescription processing is started.

Consumer preference to use online prescription submission feature

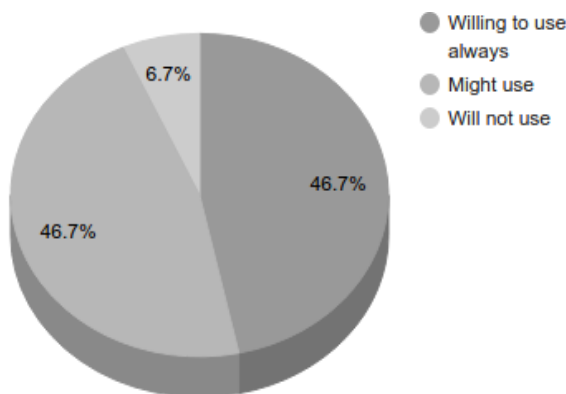


Figure 5.3 Customer preference to use online prescription submission feature

Evaluators rated that this feature is useful as it eliminates the need to physically be present at the pharmacy to hand over a prescription. Evaluators who seemed comfortable with uploading prescriptions online preferred this feature, where as those who were not comfortable with uploading prescriptions online did not find this feature

useful. It should be noted that these preferences are based on the evaluator's experience and reliance on similar applications.

Consumer preference to use online status tracking feature

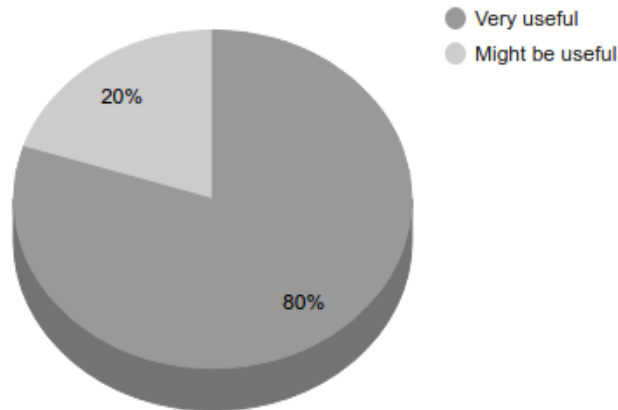


Figure 5.4 Customer preference to track status online

All evaluators found the online status tracking feature useful, regardless of their preference to use online prescription uploading feature. A suggestion was made to add SMS notifications to notify consumers on the prescription status.

5.4 Summary

The application was subjected to a set of tests verifying UI, features, security aspects and back end services and conformance of application to expected results were verified. Deviations from expected outputs were addressed and fixed. The application was also evaluated by pharmaceutical staff possessing expertise in the domain and potential consumers. The evaluation revealed that the application possess many useful features. Additionally, improvements were revealed, which can be incorporated as future work or extensions of the application. The next chapter discusses the project conclusion and potential for future work.

Chapter 6 : Conclusion & Future Work

6.1 Introduction

This chapter concludes the thesis by summarizing the project in terms of work carried out, revelations and limitations. The author also reveals potential for future work that can be carried out by extending and improving the implemented application.

6.2 Work carried out

Through out the project the author carried out following tasks.

- Researched into the pharmaceutical prescription handling process.
- Identified issues, limitations and areas of improvement associated with prevailing prescription handling processes.
- Identified technology based solutions for the above identified issues and limitations.
- Designed and implemented a application to handle the prescription management process associating the identified technology based solutions.
- Evaluated the implemented application based on feedback from pharmaceutical professionals and potential consumers and identified areas of improvement and extensions.

6.3 Revelations

Based on the work carried out and the evaluation, the following revelations were made.

- Pharmaceutical companies and staff are now leaning towards computerized process handling with the understanding that it makes the process efficient and reduces room for human errors. Staff is willing to use computerized applications

as it will have recorded evidence of each step in the process including employee involvement and time. In contrast there are certain pharmaceutical companies that are more comfortable with handling the prescriptions manually.

- Potential consumers are very much biased towards using services of this application as it saves a lot of time from consumer perspective.
- The author's solution does make its contribution to optimize the prescription handling process and reduce human errors. It can also be improved with additional features and integration with an enterprise grade applications addressing the medical domain.
- Use of tools with dedicated purposes (Business Process Management , Microservices Server etc) makes it easy to build a flexible and extensible solution and it keeps the solution focused and dedicated to the requirements of addressed problem and its domain.

6.4 Lessons Learnt

The author was able to gather the following lessons learnt at project completion.

- As a result of not having sufficient hands on experience on integration of different tools used, a lot of time has to be spent and it costed a certain amount of rework to integrate different tools (Eg: Angular 2 with WSO2 Microservices server etc) during the implementation phase. Therefore it was realized that a lot of hands on experience is required when using multiple tools/applications for an application, before carrying out the actual integration of the output of one tool with the input of another.
- Incremental development is the most suitable approach in implementing projects where the requirements keep evolving. Since the requirements kept evolving when the design phase was started, the design had to be changed repeatedly. Since certain features were implemented with in depth detail from the very beginning, changing the application with requirement changes was cumbersome. Therefore it

was realized that a detailed, in depth design and implementation during initial stages is not practical in a project with evolving requirements. Rather the design should be kept minimal, until requirements are evolved.

6.5 Future Work

- Stock handling and billing related functionality can be incorporated with the application to complete the work flow related functionality. A stock inventory should be maintain along with item costs to enable this.
- The application can be improved to display drug conflict alerts and allergy warnings when conflicting drugs are added in a prescription. For this, drug conflict information should be maintained in the database, against which each prescription is checked.
- Customer communication can be improved by integrating a SMS sending feature apart from online status tracking..
- The application can be integrated with a hospital management system where the prescriptions can be directly submitted to the pharmacy by the practitioner for processing. This eliminates the need to carry hard copies of prescriptions and will make the prescription submission process even more efficient.
- It can also be improved to integrate all pharmacy outlets of a particular chain. The list of outlets can be made available and the consumers can be given the option to select a pharmacy outlet to search for drugs or submit prescriptions online. In order to enable this feature , all stock and prescription records maintained in the database should be associated with a specific pharmacy outlet.
- The solution can be integrated with a dedicated identity management application to delegate user and role management and separated them from the main purpose of the application. This enables a variety of authorization options based on role/permission.
- The solution can also be integrated with a dedicated API management application

which enables securing back end service/resources and applying 'Quality of Service' features (throttling, caching etc)

- Caching mechanisms such as data caching and browser caching can be implemented in order to improve response time. Eg: When staff and a large number of consumers access the services instead of making the same database call repeatedly, the data retrieved can be cached in the service layer and returned to users.
- Finally, usage patterns can be monitored to figure out users of different APIs exposed in the solution. Eg: If consumers frequently request for a new drug via prescriptions which the pharmacy does not have, monitoring such data would allow the pharmacy to be aware of consumer requirements and cater to those requirements.

6.6 Summary

This chapter summarized the project work carried out, revelations and limitations. Lessons learnt revealed difficulties faced by the author during project implementation, which can be avoided in future to help develop applications efficiently by avoiding rework as much as possible. The future work identified can be carried out to add more sophisticated features and to integrate the application with enterprise grade applications to widen its scope.

References

Research Articles

- [1] K. Leggett, "Trends 2016: The Future Of Customer Service," in <http://b2b.cbsimg.net/>, 2016. [Online]. Available: http://b2b.cbsimg.net/whitepapers/Forrester_s_Top_10_Customer_Service_Trends_for_2016-_The_Future_of_Customer_Service.pdf. Accessed: Jul. 3, 2016.
- [2] S. Priya, R. Sumathy, and D. Anuradha, "HUMAN RESOURCE MANAGEMENT SYSTEM USING SPRING 'REST'FUL WEB SERVICE," 2015. [Online]. Available: <http://ijoer.in/3.2.15/30-34%20CH.SWATHI.pdf>. Accessed: Jul. 3, 2016.
- [3] A. Menkudle, S. Sonawane, and A. Jagtap, "Extracting Application Model from Restful Web Services for Client Stub Generation," in <http://www.ijcta.com/>, 2014. [Online]. Available: <http://www.ijcta.com/documents/volumes/vol5issue1/ijcta2014050134.pdf>. Accessed: Jul. 3, 2016.
- [4] J. Lewis and M. Fowler, "Microservices," in martinfowler.com, Martin Fowler, 2014. [Online]. Available: <http://martinfowler.com/articles/microservices.html>. Accessed: Jul. 19, 2016.

Web References

- [1] Y.-F. Chen, K. E. Neil, A. J. Avery, M. E. Dewey, and C. Johnson, "Prescribing errors and other problems reported by community pharmacists," vol. 1, no. 4, Dec. 2005. [Online]. Available: <http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1661637/>. Accessed: Jul. 2, 2016.
- [2] AIIM, "What is business process management?," 2016. [Online]. Available: <http://www.aiim.org/What-is-BPM>. Accessed: Jul. 2, 2016.
- [3] N. Palmer, "What is BPM?," 2015. [Online]. Available: <http://bpm.com/what-is-bpm>. Accessed: Jul. 2, 2016.
- [4] A. Mar, "6 steps to business process management success," Simplicable, 2011.

- [Online]. Available: <http://arch.simplicable.com/arch/new/6-steps-to-business-process-management-success>. Accessed: Jul. 2, 2016.
- [5] sandra, "Business process modeling techniques with examples - Creately Blog," in diagrams, Creately Blog, 2014. [Online]. Available: <http://creately.com/blog/diagrams/business-process-modeling-techniques/>. Accessed: Jul. 2, 2016.
- [6] C. M. 4 Inc, "Introduction to data services," InfoQ, 2016. [Online]. Available: <https://www.infoq.com/articles/narayanan-soa-data-services>. Accessed: Jul. 2, 2016.
- [7] "Collaborative solutions,". [Online]. Available: <https://home.meditech.com/en/d/newmeditech/pages/collaborativesolutions.htm#epres>. Accessed: Jul. 2, 2016.
- [8] "HP OpenVMS systems - partners,". [Online]. Available: <http://h71000.www7.hp.com/partners/sms/>. Accessed: Jul. 2, 2016.
- [9] "Horizon Meds Manager™," 2016. [Online]. Available: <http://www.health-care-it.com/company/559028/products/205076/horizon-meds-manager>. Accessed: Jul. 2, 2016.
- [10] 2016, "AngularJS," 2010. [Online]. Available: <https://docs.angularjs.org/tutorial>. Accessed: Jul. 2, 2016.
- [11] jq. Foundation, "jQuery UI," 2016. [Online]. Available: <https://jqueryui.com/>. Accessed: Jul. 2, 2016.
- [12] "Apache ODE – Apache ODE™," 2008. [Online]. Available: <http://ode.apache.org/index.html>. Accessed: Jul. 2, 2016.
- [13] "Activiti,". [Online]. Available: <http://activiti.org/>. Accessed: Jul. 2, 2016.
- [14]]W. 2 Inc, "WSO2 Business Process Server Documentation," 2005. [Online]. Available: <https://docs.wso2.com/display/BPS351/WSO2+Business+Process+Server+Documentation>. Accessed: Jul. 2, 2016.
- [15] Oracle, "REST data services," 2016. [Online]. Available: <http://www.oracle.com/technetwork/developer-tools/rest-data->

- services/overview/index.html. Accessed: Jul. 2, 2016.
- [16] W. 2 Inc, "WSO2 Data Services Server Documentation," 2005. [Online]. Available: <https://docs.wso2.com/display/DSS350/WSO2+Data+Services+Server+Documentation>. Accessed: Jul. 2, 2016.
- [17] camunda community, "BPMN Tutorial - BPMN 2.0 Tutorial for beginners - learn BPMN," 2016. [Online]. Available: <https://camunda.org/bpmn/tutorial/>. Accessed: Sep. 18, 2016.
- [18] T. O. Group, "Business process layer," 1995. [Online]. Available: https://www.opengroup.org/soa/source-book/soa_refarch/busproc.htm. Accessed: Sep. 18, 2016.
- [19] "What is Microservices architecture?," 2016. [Online]. Available: <https://smartbear.com/learn/api-design/what-are-microservices/>. Accessed: Sep. 20, 2016.
- [20] A. Azeez, "Performance advantages WSO2 Microservices framework for java 2.0 brings to spring," in Articles, JAXenter, 2016. [Online]. Available: <https://jaxenter.com/performance-advantages-wso2-microservices-framework-for-java-2-0-to-spring-128340.html>. Accessed: Sep. 20, 2016.
- [21] The Angular core team, "ARCHITECTURE OVERVIEW," in angular.io, 2015. [Online]. Available: <https://angular.io/docs/ts/latest/guide/architecture.html>. Accessed: Jul. 01, 2016.
- [22] M. Mohtashim, "JPA - Criteria API," in tutorialspoint.com. [Online]. Available: https://www.tutorialspoint.com/jpa/jpa_criteria_api.htm. Accessed: Nov. 16, 2016.
- [23] P. Kharkar, "Automatic ID creation using jpa table generator," in <http://www.thejavageek.com/>, 2014. [Online]. Available: <http://www.thejavageek.com/2014/01/14/automatic-id-creation-using-jpa-table-generator/>. Accessed: Nov. 15, 2016.
- [24] wso2.com, "Creating a BPMN Process", in docs.wso2.com, 2016. [Online]. Available: <https://docs.wso2.com/display/BPS360/Creating+a+>

BPMN+Process#CreatingaBPMNProcess-Creatingtheprocessandservicetask.

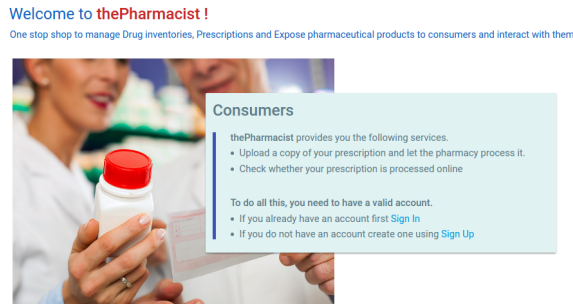
Accessed: Oct. 10, 2016

- [25] B. Al-Sarori, "Activiti User Guide," in [activiti.org](http://www.activiti.org), 2016. [Online]. Available: <https://www.activiti.org/userguide/>. Accessed: Nov. 29, 2016.
- [26] ObjectDB Software Ltd, "Chapter 3 - Using JPA," in ObjectDB, 2010. [Online]. Available: <http://www.objectdb.com/java/jpa/persistence>. Accessed: Oct. 04, 2016.
- [27] A. Bein, "Selecting all JPA entities as criteria query," in Adam Bein's Weblog, 2013. [Online]. Available: http://www.adam-bien.com/roller/abien/entry/selecting_all_jpa_entities_as. Accessed: Oct. 4, 2016.

Appendices

Appendix A – User Manual

Application landing page



Appendix A : Figure 1 Home Page

User Login

This is the login UI for both pharmaceutical staff and consumers.

1. Click on 'Sign' In' menu item on the landing page and enter log in credentials to log in to the application.
 - Once logged in, the menu items relevant to the user role is loaded.

Sign In

Enter your username and password

Username
andy

Password
.....|

Appendix A : Figure 2 Login

Home page for consumers is as follows.

Prescription ID	Prescription Type	Consumer Name	Submitted Date	Status
16	Online	anne hathaway	Mar 8, 2017 11:21:43 PM	New
17	Online	anne hathaway	Mar 8, 2017 11:47:00 PM	New
18	Online	anne hathaway	Mar 8, 2017 11:49:47 PM	New

Appendix A : Figure 3 Consumer Home page

Home page for pharmaceutical staff is as follows.

Prescription ID	Prescription Type	Consumer Name	Submitted Date	Status	Item Count
670	Manual	Anne Hathaway	Mar 8, 2017 5:00:14 PM	Complete	2
671	Manual	John Grisham	Mar 8, 2017 5:41:40 PM	Collect	1
672	Manual	Anne Hathaaway	Mar 8, 2017 11:14:42 PM	Verify	1
675	Manual	Peter Kuruwita	Mar 8, 2017 11:55:22 PM	New	1

Appendix A : Figure 4 Staff Home page

Consumer Registration

The menu item 'Sign-up', is available to allow registering in the application.

1. Click on the menu item 'Sign-up' on the landing page.
 2. Specify consumer details and click on 'Sign Up' to register the consumer.
- Once registered, consumers can log in and use the Drug search and Online

prescription upload related features.

Sign Up

It's free and always will be

Username Jane	Password *****
First Name Jane	Last Name Austin
Mobile Number 0777253146	Email janea@gmail.com
Address No 10, Deal Place, Colombo 10	

Appendix A : Figure 5 Sign up

Online Prescriptions

Registered consumers and pharmacy staff can view 'Online prescriptions'. For consumers this shows a list of prescriptions uploaded by the consumer himself. For pharmaceutical staff this shows a list of prescriptions submitted by consumers .

Upload Prescription Online

Consumers can uploaded a prescription as follows.

1. Click on menu item 'Online prescriptions'.
 - This will show a list of online prescriptions submitted by or relevant to the logged in user. i.e. For a consumer, this shows only the prescriptions submitted by him. For a pharmaceutical staff user with the role 'Admin' or 'Receiver', this shows a lost of online prescriptions submitted by all consumers.
 - The status of the prescription in the processing cycle is displayed along with each prescription.

PRESCRIPTIONS

ADD PRESCRIPTION

*** Please visit the pharmacy to collect your items when the status is 'Pay' ***

Prescription ID	Prescription Type	Consumer Name	Submitted Date	Status
15	Online	anne hathaway	Mar 8, 2017 11:24:58 PM	Issue
16	Online	anne hathaway	Mar 8, 2017 11:21:43 PM	New

Appendix A : Figure 6 Online prescription list

2. Click on 'Add Prescription'.

- This will load the UI to upload a new prescription.

3. Browse and select a scanned copy of the prescription to be uploaded.

- Specify any comments required to be added

4. Click on 'Submit prescription' to submit it to the pharmacy.

- Once submitted, this prescription will be listed under 'Online prescriptions' list for the pharmaceutical staff.

PRESCRIPTION

BROWSE Upload Prescription

Dr B. Who
Farmstreet 12
Kirkville
tel. 3876

R/ date 1 Nov ,

Digoxin 0.125 mg
tablets da no.7
S 1 da 1 tablet

B. WHO

Ms/Mr Patient 30
address:
age: 70 years

Consumer Comments

Consumer Details

First Name	Last Name	Contact No	Email	Address
anne	hathaway	077342394	anneh@gmail.com	No 30, Townhall, Colombo 10

Internal Prescription ID
0

BACK **SAVE** **REJECT**

Appendix A : Figure 7 Online prescription

The Admin or Receiver user should enter the prescriptions submitted online under 'Prescriptions' in order to start processing them.

Once the prescription is added, update the '**Internal Prescription ID**' with the prescription ID generated for the manually added prescription

- This associates the online prescription with the internal prescription added by the pharmacy to track the prescription state.

If a prescription cannot be accepted for processing it should be rejected using the '**Reject**' button.

Prescriptions

Prescription that have been added for processing are listed under 'Prescriptions'. Each prescription here is either related to a prescriptions submitted online or a prescription manually handed over to the pharmacy by a consumer.

1. Navigate to menu item 'Prescriptions' to view a list of prescriptions depending on the logged in user's role.

- The state of listed prescriptions depend on the pharmacy staff user role as follows.

Role	Prescription state
Admin	Prescriptions in any state.
Receiver	Prescriptions in state 'New'
Collector	Prescriptions in state 'Collect'
Verifier	Prescriptions in state 'Verify'
Cashier	Prescriptions in state 'Pay'
Issuer	Prescriptions in state 'Issue'
Consumer	Prescription uploaded by consumer himself

Appendix A : Table 1

- Given above it the view of prescription list to an Admin user.

PRESCRIPTIONS

Prescription ID	Prescription Type	Consumer Name	Submitted Date	Status	Item Count
670	Manual	Anne Hathaway	Mar 8, 2017 5:00:14 PM	Complete	2
671	Manual	John Grisham	Mar 8, 2017 5:41:40 PM	Collect	1
672	Manual	Anne Hathaaway	Mar 8, 2017 11:14:42 PM	Verify	1
675	Manual	Peter Kuruwita	Mar 8, 2017 11:55:22 PM	New	1

Appendix A : Figure 8 Prescription List

Add Prescriptions

1. Navigate to 'Prescriptions' menu item and click on 'Add Prescription'.
- To start processing a prescription it is necessary to add a prescription.

PRESCRIPTION

ID: _____ Type: **Manual** Date: _____ State: **New**

Consumer Comments

Prescribed Items

Drug	Brand	Strength	Dosage	Quantity	Instructions	Availability	Select
							<input type="button" value="ADD NEW"/> <input type="button" value="AVAILABILITY"/> <input type="button" value="REMOVE"/>

Consumer Details

First Name _____ Last Name _____ Contact No _____ Email _____ Address _____

Medical Practitioner

Comment

Prescription History

Status	Updated Time	Updated By	Comment
--------	--------------	------------	---------

Appendix A : Figure 9 Add prescription

2. To add prescription items , click on 'Add item' and specify drug, strength and dosage details and click on 'Add' .

Prescribed Items

Add Items Drug: Metformin Strength: 500mg Instructions: Consume as instructed below

Quantity: 1 Pill tblsp teasp ml

Times: 2 **Every** Frequency: 1 Hour(s) Before Meals Day(s) Week(s) Month(s)

For Duration: 1 Hour(s) Day(s) Week(s) Month(s) As Needed

Appendix A : Figure 10 Add prescription item

- Then the added item will be listed in the prescription.

Prescribed Items

Drug	Brand	Strength	Dosage	Quantity	Instructions	Availability	Select
Omeprazole		200mg	1 pill(s) 2 Time(s) for 1 Day(s) before Meals for 2 Week(s)	50	Consume as instructed below	Yes	
Metformin		500mg	1 pill(s) 2 Time(s) for 1 Day(s) after Meals for 1 Month(s)	50	Consume as instructed below	Yes	

Appendix A : Figure 11 Prescription items

3. Specify other required details and click on 'Save' to save the prescription.

- Once saved, this prescription will be listed with the state 'New' in the Prescription List.

Prescription ID	Prescription Type	Consumer Name	Submitted Date	Status	Item Count
670	Manual	Anne Hathaway	Mar 8, 2017 5:00:14 PM	New	2

Appendix A : Figure 12 Prescription in New state

Promoting Prescriptions

A 'New' prescription should be promoted to each state sequentially, in order to process and complete it.

To promote a prescription

1. Login as a user with 'Receiver' role and click on 'Prescriptions' to view a list of prescriptions in 'New' state similar to 'Appendix B : Figure 12 Prescription in New state'

Select the prescription required to be promoted to the next level and load it.

- Note the prescription history at the bottom of the prescription.

Click on 'Promote'.

Comment
Promote to Collect|

BACK SAVE PROMOTE

Prescription History

Status	Updated Time	Updated By	Comment
NEW	Mar 8, 2017 5:00:14 PM	renold	

Appendix A : Figure 13 Promote prescription to collect state

- Once promoted, this prescription will be moved to 'Collect' state, which will make it only viewable by the Admin or users with 'Collector' role.
- 2. Login as a user with 'Collector' role and click on 'Prescriptions' to view a list of prescriptions in 'Collect' state.

Prescription ID	Prescription Type	Consumer Name	Submitted Date	Status	Item Count
670	Manual	Anne Hathaway	Mar 8, 2017 5:00:14 PM	Collect	2

Appendix A : Figure 14 Prescription in Collect state

- Select the prescription required to be promoted to the next level and load it.
- Note the prescription history at the bottom of the prescription.
- Collect the prescribed items and click on 'Promote'.

Comment

Promote to Verify

[BACK](#) [SAVE](#) [PROMOTE](#) [DEMOTE](#)

Prescription History

Status	Updated Time	Updated By	Comment
NEW	Mar 8, 2017 5:00:14 PM	renold	
Collect	Mar 8, 2017 5:05:12 PM	renold	Promote to Collect

Appendix A : Figure 15 Promote prescription to verify state

- Once promoted, this prescription will be moved to 'Verify' state, which will make it only viewable by the Admin or users with Verifier role.

3. Login as a user with 'Verifier' role and click on 'Prescriptions' to view a list of prescriptions in 'Verify' state.

Prescription ID	Prescription Type	Consumer Name	Submitted Date	Status	Item Count
670	Manual	Anne Hathaway	Mar 8, 2017 5:00:14 PM	Verify	2

Appendix A : Figure 16 Prescription in Verify state

Select the prescription required to be promoted to the next level and load it.

- Note the prescription history at the bottom of the prescription.

Verify the collected items and click on 'Promote'.

Comment

Promote to Pay

BACK SAVE PROMOTE DEMOTE

Prescription History

Status	Updated Time	Updated By	Comment
NEW	Mar 8, 2017 5:00:14 PM	renold	
Collect	Mar 8, 2017 5:05:12 PM	renold	Promote to Collect
Verify	Mar 8, 2017 5:12:27 PM	collin	Promote to Verify

Appendix A : Figure 17 Promote prescription to pay state

- Once promoted, this prescription will be moved to 'Verify' state, which will make it only viewable by the Admin or users with 'Cashier' role.

4. Login as a user with 'Cashier' role and click on 'Prescriptions' to view a list of prescriptions in 'Pay' state.

Prescription ID	Prescription Type	Consumer Name	Submitted Date	Status	Item Count
670	Manual	Anne Hathaway	Mar 8, 2017 5:00:14 PM	Pay	2

Appendix A : Figure 18 Prescription in Pay state

Select the prescription required to be promoted to the next level and load it.

- Note the prescription history at the bottom of the prescription.

Make the payment transaction and click on 'Promote'.

Comment

Promote to Issue|

BACK SAVE PROMOTE

Prescription History

Status	Updated Time	Updated By	Comment
NEW	Mar 8, 2017 5:00:14 PM	renold	
Collect	Mar 8, 2017 5:05:12 PM	renold	Promote to Collect
Verify	Mar 8, 2017 5:12:27 PM	collin	Promote to Verify
Pay	Mar 8, 2017 5:16:59 PM	valerie	Promote to Pay

Appendix A : Figure 19 Promote prescription to pay state

- Once promoted, this prescription will be moved to 'Issue' state, which will make it only viewable by the Admin or users with 'Issuer' role.

5. Login as a user with 'Issuer' role and click on 'Prescriptions' to view a list of prescriptions in 'Issue' state.

Prescription ID	Prescription Type	Consumer Name	Submitted Date	Status	Item Count
670	Manual	Anne Hathaway	Mar 8, 2017 5:00:14 PM	Issue	2

Appendix A : Figure 20 Prescription in Issue state

Select the prescription and load it.

- Note the prescription history at the bottom of the prescription.

Issue the prescribed items and click on 'Promote' to mark it as 'Completed'.

Comment
Promote as Completed

BACK SAVE PROMOTE

Prescription History

Status	Updated Time	Updated By	Comment
NEW	Mar 8, 2017 5:00:14 PM	renold	
Collect	Mar 8, 2017 5:05:12 PM	renold	Promote to Collect
Verify	Mar 8, 2017 5:12:27 PM	collin	Promote to Verify
Pay	Mar 8, 2017 5:16:59 PM	valerie	Promote to Pay
Issue	Mar 8, 2017 5:27:14 PM	caleb	Promote to Issue

Appendix A : Figure 21 Promote prescription to complete state

- Once promoted, this prescription will be moved to 'Completed' state, which will make it only viewable by the Admin.

670	Manual	Anne Hathaway	Mar 8, 2017 5:00:14 PM	Complete	2
-----	--------	---------------	------------------------	----------	---

Appendix A : Figure 22 Completed prescription

- Prescription history will be visible as follows when loaded.

Prescription History

Status	Updated Time	Updated By	Comment
NEW	Mar 8, 2017 5:00:14 PM	renold	
Collect	Mar 8, 2017 5:05:12 PM	renold	Promote to Collect
Verify	Mar 8, 2017 5:12:27 PM	collin	Promote to Verify
Pay	Mar 8, 2017 5:16:59 PM	valerie	Promote to Pay
Issue	Mar 8, 2017 5:27:14 PM	caleb	Promote to Issue
Complete	Mar 8, 2017 5:34:41 PM	Issac	Promote as Completed

Appendix A : Figure 23 Completed prescription history

Demoting Prescriptions

If there is an issue with a prescription, it can be sent back to the previous state to rectify the issues. i.e. If there is an issue with the collected items, the verifier user may need to send the prescription back to 'Collect' state.

E.g.:

To demote the prescription

1. Login as a user with 'Verifier' role and click on 'Prescriptions' to view a list of prescriptions in 'Verify' state.

Select the prescription and load it.

Specify a comment describing why the prescription is demoted and click on 'Demote'.

Comment
Demoted as there is an issue with the collected items|

BACK SAVE PROMOTE DEMOTE

Prescription History

Status	Updated Time	Updated By	Comment
NEW	Mar 8, 2017 5:41:40 PM	renold	
Collect	Mar 8, 2017 5:42:06 PM	renold	Promote to Collect
Verify	Mar 8, 2017 5:42:50 PM	collin	Promote to Verify

Appendix A : Figure 24 Demote prescription to collect state

- Once demoted, this prescription will be moved to 'Collect' state, which will make it only viewable by the Admin or users with 'Cashier' role.

Prescription ID	Prescription Type	Consumer Name	Submitted Date	Status	Item Count
671	Manual	John Grisham	Mar 8, 2017 5:41:40 PM	Collect	1

Appendix A : Figure 25 Prescription demoted to collect state

- Prescription history will appear as below.

Prescription History

Status	Updated Time	Updated By	Comment
NEW	Mar 8, 2017 5:41:40 PM	renold	
Collect	Mar 8, 2017 5:42:06 PM	renold	Promote to Collect
Verify	Mar 8, 2017 5:42:50 PM	collin	Promote to Verify
Collect	Mar 8, 2017 5:46:07 PM	valerie	Demoted as there is an issue with the collected items

Appendix A : Figure 26 Prescription history of demoted prescription

Appendix B– Test Plan

Objectives

To identify and document the approach and plan for testing the implemented application.

The objectives of the test plan is to

- Identify the project deliverables that need to be tested
- Identify test requirements and testing strategy
- Identify Test deliverables

Scope

To conduct a comprehensive test to ensure feasibility of using the implemented application in production. The test should cover back end service testing and end-to-end scenario verification of the application from both pharmacy and customer perspective.

Test Coverage

Functional Testing of features and security scenarios

Features

- Consumer registration
- Prescription processing life cycle
 - For online submitted prescriptions
 - For manually added prescriptions

Security Scenarios

- User login
- Access restrictions based on user role

Back end service testing

The following REST services should be tested.

- Drug service
- Prescription service
- Auth services
- User services
- Consumer service

Test Strategy

The following approaches are followed to test the above identified test requirements.

- Black box testing : To cover all functional and security requirements specified above.
- White box testing : Test and validate back end micro services using a test script to maintain the tests and re-run the them easily.

Deliverables

Deliverables include a tested prescription processing application with bug fixes.

Appendix C – Test Cases

Test case ID : TC 1		
Test case : Consumer Registration		
Pre-conditions : The Prescription Processing Application should be loaded		
Steps		
1	Load Sign Up UI	Consumer registration UI loads
2	Specify all mandatory details required to register and register.	User registration should conclude successfully.

Test case ID : TC 2		
Test case : Consumer Login with valid credentials		
Pre-conditions : The consumer should be registered in the application		
Steps		
1	Select Login UI	Login UI loads
2	Specify valid login credentials and log in.	The consumer should be successfully logged in. Only the prescription upload and prescription list UI should be visible to the logged in consumer.

Test case ID : TC 3		
Test case : [Negative] Login with invalid credentials		
Pre-conditions : The Prescription Processing Application should be loaded		
Steps		
1	Select Login UI	Login UI loads
2	Specify invalid login credentials and attempt to login.	The login should fail. The user should be notified that the credentials are invalid.
Test case ID : TC 4		

Test case : Logout from the system	
Pre-conditions : The user should be logged into the application.	
Steps	
1	Click on 'Sign-out' The user should be logged out of the system. The home page should be loaded.

Test case ID : TC 5	
Test case : Consumer uploads a prescription online	
Pre-conditions : The consumer should be logged into the application.	
Steps	
1	Select Prescription Upload UI Prescription Upload UI loads
2	Browse and add a scanned copy of a prescription and specify any additional comments required and save the uploaded prescription. The prescription should be successfully uploaded. A record for the uploaded prescription should be visible under the prescription list of the user with the status 'Pending'

Test case ID : TC 6	
Test case : Prescription state visibility to consumers	
Description : To ensure that consumers can see the prescription state online, as it traverses through the prescription life cycle.	
Pre-conditions : The consumer should be logged into the application. The consumer should have uploaded several prescriptions. The prescriptions should be in statuses, Pending , New, Collect, Verify, Issue, Pay, Completed and Rejected	
Steps: Log in to the application, load prescription list ensure that prescriptions in all the below states are visible to consumers.	
ID	Prescription State
6.1	Pending
6.2	New

6.3	Collect
6.4	Verify
6.5	Issue
6.6	Pay
6.7	Completed
6.8	Rejected

Test case ID : TC 7	
Test case : Consumer loads a prescription in his prescription list	
Pre-conditions : The consumer should be logged into the application. The consumer should have uploaded a prescription.	
Steps	
Select Prescription List UI	Prescription List UI loads with a list of available prescriptions for the logged in consumer.
Click on a prescription	Prescription should be loaded. This view should be the same as Prescription upload view.

Test case ID : TC 8		
Test case : Role based authorization		
Description : To ensure that a user can only view interfaces that are authorized by the role.		
Pre-conditions : The user should have a valid account and should have only the roles specified below..		
Steps: Log in to the application with a user with only the specified role assigned.		
ID	Role	Accessible interfaces
8.1	admin	Consumers / Drug / Prescriptions / Online Prescriptions
8.2	receiver	Consumers / Drug / Prescriptions / Online Prescriptions
8.3	collector	Consumers / Drug / Prescriptions
8.4	verifier	Consumers / Drug / Prescriptions
8.5	cashier	Consumers / Drug / Prescriptions
8.6	issuer	Consumers / Drug / Prescriptions

Test case ID : TC 9		
Test case : Receiver views online prescription		
Pre-conditions : The receiver should be logged into the application. There should be online prescriptions.		
Steps		
1	Select Online Prescription List UI	Online Prescription List UI loads with a list of available prescriptions online prescriptions.
2	Click on a prescription	Prescription should be loaded. This view should be the same as Prescription upload view.

Test case ID : TC 10		
Test case : Receiver rejects online prescription		
Pre-conditions : The receiver should be logged into the application. There should be online prescriptions in state 'Pending'		
Steps		
1	Select Online Prescription List UI	Online Prescription List UI loads with a list of available prescriptions online prescriptions.
2	Click on a prescription in state 'Pending'	Prescription should be loaded. This view should be the same as Prescription upload view.
3	Specify a comment and click on 'Reject'	The prescription state in the online prescription list should be updated as 'Rejected'

Test case ID : TC 11		
Test case : Receiver adds new prescription in the system		
Description : To ensure that it is possible for receivers to add a new prescription in the system		
Pre-conditions : The receiver should be logged into the application.		
Steps		
1	Select Prescriptions → Add New Prescriptions	UI for adding a new prescriptions should be displayed.
2	Specify details for the new prescriptions and save	The prescriptions details should be successfully saved.

	The added prescriptions should be available in the list of prescriptions with the status “New”.
--	---

Test case ID : TC 12		
Test case : Verification of role based prescription list visibility to Pharmacy staff		
Description : To ensure that prescriptions listed are filtered based on the user role.		
Pre-conditions : Prescriptions in the following states should existing in the system. - New / Collect/ Verify/ Pay / Issue/ Completed		
Steps		
ID	Role	Visible Prescription States
12.1	admin	New / Collect/ Verify/ Pay / Issue/ Completed / Rejected
12.2	receiver	New
12.3	collector	Collect
12.4	verifier	Verify
12.5	cashier	Pay
12.6	issuer	Issue

Test case ID : TC 13				
Test case : Verification of prescription state promotion				
Description : To ensure that prescription state can be promoted as required				
Pre-conditions :				
Prescriptions in the following states should existing in the system. - New / Collect/ Verify/ Pay / Issue/ Completed				
A user with only the specified role should be logged in and the prescriptions in the relevant state should be listed.				
Steps				
Log in from the role specified under 'Role1' and promote a prescription in the specified state. Login from a user with user role specified under 'Role2' and verify that the prescription state is updated to the state given under 'New state'				
ID	Role1	Prescription States	Role2	New state
13.2	receiver	New	collector	Collect
13.3	collector	Collect	verifier	Verify
13.4	verifier	Verify	cashier	Pay

13.5	cashier	Pay	issuer	Issue
13.6	issuer	Issue	admin	Completed

Test case ID : TC 14				
Test case : [Negative] Verification of prescription state demotion				
Pre-conditions :				
Prescriptions in the following states should existing in the system.				
- Collect/ Verify/ Pay				
A user with only the specified role should be logged in and the prescriptions in the relevant state should be listed.				
Steps				
Log in from the role specified under 'Role1' and demote a prescription in the specified state.				
Login from a user with user role specified under 'Role2' and verify that the prescription state is updated to the state given under 'New state'				
ID	Role1	Prescription States	Role2	New state
14.1	collector	Collect	receiver	New
14.2	verifier	Verify	collector	Collect
14.3	cashier	Pay	admin	Payment failure

Appendix D – Apache JMeter 2.13 Script

Please note that a collapsed version of the XML script file is given here for clarity, with only one GET resource and one POST resource invocation expanded.

```
▼<jmeterTestPlan version="1.2" properties="2.8" jmeter="2.13 r1665067">
  <hashTree>
    <TestPlan guiclass="TestPlanGui" testclass="TestPlan" testname="Test Plan" enabled="true">...</TestPlan>
    <hashTree>
      <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup" testname="Drug_service" enabled="true">...</ThreadGroup>
      <hashTree>
        <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Drug_service : GET all drugs" enabled="true">
          <elementProp name="HTTPSampler.Arguments" elementType="Arguments" guiclass="HTTPArgumentsPanel" testclass="Arguments" testname="User Defined Variables" enabled="true">
            <collectionProp name="Arguments.arguments">
              <elementProp>
                <stringProp name="HTTPSampler.domain"/>
                <stringProp name="HTTPSampler.port"/>
                <stringProp name="HTTPSampler.connect.timeout"/>
                <stringProp name="HTTPSampler.response.timeout"/>
                <stringProp name="HTTPSampler.protocol"/>
                <stringProp name="HTTPSampler.contentEncoding"/>
                <stringProp name="HTTPSampler.path">http://localhost:8080/pharmacy/drugs/</stringProp>
                <boolProp name="HTTPSampler.follow.redirects">true</boolProp>
                <boolProp name="HTTPSampler.auto.redirects">false</boolProp>
                <boolProp name="HTTPSampler.use.KeepAlive">true</boolProp>
                <boolProp name="HTTPSampler.DO_MULTIPART_POST">false</boolProp>
                <boolProp name="HTTPSampler.monitor">false</boolProp>
                <stringProp name="HTTPSampler.embedded.url.re"/>
              </elementProp>
            </collectionProp>
          </HTTPSamplerProxy>
          <hashTree/>
          <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Drug_service : GET strength of a specific drugs" enabled="true">...</HTTPSamplerProxy>
          <hashTree/>
          <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup" testname="Prescription_service" enabled="true">...</ThreadGroup>
          <hashTree>
            <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Prescription_service : GET all prescriptions" enabled="true">...</HTTPSamplerProxy>
            <hashTree/>
            <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Prescription_service : GET a specific prescription" enabled="true">...</HTTPSamplerProxy>
            <hashTree/>
            <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Prescription_service : POST Prescription (Create)" enabled="true">
              <elementProp name="HTTPSampler.Arguments" elementType="Arguments">
                <collectionProp name="Arguments.arguments">
                  <elementProp name="HTTPSampler.Arguments" elementType="HTTArgument">
                    <boolProp name="HTTArgument.always.encode">false</boolProp>
                    <stringProp name="Argument.value">
                      <itemList>
                        {
                          "quantityUnit": "pill", "perUnitTime": "Day(s)", "beforeOrAfterMeal": "after", "durationUnit": "Day(s)", "drugName": "Amoxicillin", "quantity": "1", "noOfTimes": "3", "frequency": "quantityUnit": "pill", "perUnitTime": "Day(s)", "beforeOrAfterMeal": "before", "durationUnit": "Week(s)", "drugName": "Omeprazole", "quantity": "1", "noOfTimes": "3", "frequency": "updateBy": "testReceiver"}], "type": "Manual", "state": "New", "consumer": {"firstName": "TestConsumerFN1", "lastName": "TestConsumerLN1", "contactNo": "07745633", "email address"}, "newPrescription": true, "medicalPractitioner": "Test_MedPractitioner"}
                    </stringProp>
                    <stringProp name="Argument.metadata"></stringProp>
                  </elementProp>
                </collectionProp>
              </elementProp>
            </HTTPSamplerProxy>
            <hashTree/>
            <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Prescription_service : POST Prescription (Promote)" enabled="true">...</HTTPSamplerProxy>
            <hashTree/>
            <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Prescription_service : POST Prescription (Demote)" enabled="true">...</HTTPSamplerProxy>
            <hashTree/>
            </hashTree>
          <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup" testname="Consumer_service" enabled="true">...</ThreadGroup>
          <hashTree>
            <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Consumer_service : GET all consumers" enabled="true">...</HTTPSamplerProxy>
            <hashTree/>
            <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Consumer_service : GET a specific consumer" enabled="true">...</HTTPSamplerProxy>
            <hashTree/>
            <HTTPSamplerProxy guiclass="HttpTestSampleGui" testclass="HTTPSamplerProxy" testname="Consumer_service : POST customer (register)" enabled="true">...</HTTPSamplerProxy>
            <hashTree/>
            </hashTree>
          <ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup" testname="Auth_service" enabled="true">...</ThreadGroup>
          <hashTree/>
          <ResultCollector guiclass="ViewResultsFullVisualizer" testclass="ResultCollector" testname="Result Tree" enabled="true">...</ResultCollector>
          <hashTree/>
          <ResultCollector guiclass="SummaryReport" testclass="ResultCollector" testname="Result Summary" enabled="true">...</ResultCollector>
          <hashTree/>
        </hashTree>
      </jmeterTestPlan>

```

Appendix D : Figure 1 JMeter script

Appendix E – Evaluation Questionnaire

Questionnaire for Pharmaceutical Professionals

Mark the appropriate answers with a 'X'.

1. How do you rate the comprehensiveness of this application to your pharmacy?

<input type="checkbox"/>	Contains enough functionality	<input type="checkbox"/>	Contains minimum required functionality	<input type="checkbox"/>	Need more features
--------------------------	----------------------------------	--------------------------	--	--------------------------	--------------------

2. How useful do you find this application to your pharmacy?

<input type="checkbox"/>	Very useful	<input type="checkbox"/>	Might be useful	<input type="checkbox"/>	Not useful
--------------------------	-------------	--------------------------	-----------------	--------------------------	------------

3. Do you think that this application will save time in processing prescriptions?

<input type="checkbox"/>	Saves time	<input type="checkbox"/>	Might save time	<input type="checkbox"/>	Does not save time
--------------------------	------------	--------------------------	-----------------	--------------------------	--------------------

4. Do you think this application will help avoid the discrepancies that sometimes happen during prescription processing?

<input type="checkbox"/>	Avoids discrepancies	<input type="checkbox"/>	Might avoid discrepancies	<input type="checkbox"/>	Does not avoid any discrepancies
--------------------------	-------------------------	--------------------------	---------------------------	--------------------------	-------------------------------------

5. Do you think that customers will find this application useful and make use of online prescription uploading and status checking features?

<input type="checkbox"/>	Will use	<input type="checkbox"/>	Might use	<input type="checkbox"/>	Will not use
--------------------------	----------	--------------------------	-----------	--------------------------	--------------

6. How do you rate the user friendliness of the application? (5 being the most user friendly)

<input type="checkbox"/>	1	<input type="checkbox"/>	2	<input type="checkbox"/>	3	<input type="checkbox"/>	4	<input type="checkbox"/>	5
--------------------------	---	--------------------------	---	--------------------------	---	--------------------------	---	--------------------------	---

7. Any suggestions to improve the application?

8. Please state any other comment you would like to add.

Questionnaire for potential consumers

When dealing with pharmacies, it is often experienced that consumers have to wait in line to handover prescriptions and to collect the items after the prescription is processed. With this solution, you can register at the pharmacy's website and upload your prescriptions online.

PRESCRIPTION

[BROWSE](#) Upload Prescription

Dr B. Who
Farmstreet 12
Kirkville
tel. 3876

R/ date 1 nov .
Digoxin 0.125 mg
tablets da no. 7
S 1 da 1 tablet

B. WHO

Ms/Mr Patient 30
address:
age: 70 years

Consumer Comments

Consumer Details

First Name	Last Name	Contact No	Email	Address
anne	hathaway	077342394	anneh@gmail.com	No 30, Townhall, Colombo 10

Internal Prescription ID
0

[BACK](#) [SAVE](#) [REJECT](#)

The pharmacy can validate the prescription submitted online and start to process it. When the prescription is being processed its status is updated and the updated status can be viewed by consumers.

PRESCRIPTIONS

ADD PRESCRIPTION

*** Please visit the pharmacy to collect your items when the status is 'Pay' ***

Prescription ID	Prescription Type	Consumer Name	Submitted Date	Status
15	Online	anne hathaway	Mar 8, 2017 11:24:58 PM	Issue
16	Online	anne hathaway	Mar 8, 2017 11:21:43 PM	New

Consumers can check the prescription status online, and come to the pharmacy, once the processing is done and the pharmacy is ready to accept payments and issue the items.

At this point the pharmacy can validate the actual prescription if needed.

Mark the appropriate answers with a 'X'.

1. Please state your profession.

--

2. Will the drug search feature be useful to you.

	I am comfortable uploading a prescription online
	It will depend on the prescribed items
I	I do not want to make copies of my prescriptions available online

3. How comfortable are you to upload a prescription online so that your pharmacy can start to process it?

	I am comfortable uploading a prescription online
	It will depend on the prescribed items
I	I do not want to make copies of my prescriptions available online

4. **How often do you think you would use this online prescription submission feature?**

	I'm willing to use it always
	I might use this feature
	I will not use this feature

5. **How useful do you find the prescription status tracking feature?**

	Very useful as I will get to know when exactly to go to the pharmacy to collect the items
	It may be useful, but I'm not willing to keep tracking prescription status always
	I don't find it very useful

6. Any suggestions to improve the application?

--

