

IT Ticketing System with A Chatbot

**W G A G P Sanjeewa
2020**



IT Ticketing System with a Chatbot

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

**W G A G P Sanjeewa
University of Colombo School of Computing
2020**



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: W G A G Poorna Sanjeewa

Registration Number: 2017/MIT/070

Index Number: 17550706



Signature:

Date: 11/11/2020

This is to certify that this thesis is based on the work of

Mr. W G A G Poorna Sanjeewa

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

Certified by:

Supervisor Name: Damitha D.Karunaratna

Signature:

Date:

Acknowledgements

I would like to express my sincere gratitude to all the individuals who supported me throughout this project. First, I wish to express my sincere gratitude to my supervisor, Dr. Damitha D.Karunaratna, for his enthusiasm, patience, insightful comments, helpful information and ideas that have always helped me tremendously in writing of this thesis.

I also wish to express my sincere thanks to UCSC for accepting me into the MIT program as well as to all the lectures at UCSC who had given me helps technically and mentally throughout my journey of completion this project.

finally, I would also like to thank my parents and friends who helped me a lot and provide unending inspiration.

Contents

1. Introduction.....	1
1.1. Motivation	1
1.2. Objectives	2
1.3. Scope.....	2
1.4. Dissertation Organization	3
2. Background.....	4
2.1. Analysis	4
2.1.1. Functional requirements.....	4
2.1.2. Non-functional requirements	5
2.2. Overview of currently available systems in the market	6
2.3. Review of similar systems.....	7
2.3.1. Open source solutions	7
2.3.1.1. UVdesk.....	7
2.3.1.2. Zammad Community [6].....	8
2.3.1.3. OSTicket [8]	9
2.3.2. Enterprise solutions	10
2.3.2.1. HelpDesk[10]	10
2.3.2.2. Vision Helpdesk [11].....	11
2.3.3. Helpdesk Ticketing Systems developed by a Sri Lankan companies.	12
2.3.3.1. Extremeweb Ticket Support System [12].....	12
2.3.3.2. Tryo service desk System[13]	12
2.3.4. Comparison of features between available solutions and proposed solution	13
2.3.5. Other systems vs the proposed system	13
2.4. Design Strategy	14
2.4.1. Alternate Solutions	14
2.4.2. Software Development Methodology Used	16
2.4.2.1. Iterative Waterfall Model.....	17
2.4.3. Tools Used to improve software development process.....	18
3. Methodology	19
3.1. Use case diagrams	19
3.2. Activity Diagrams	21
3.3. Class Diagram.....	25
3.4. Sequence Diagram	26
3.5. ER (Entity Relationship) Diagram.....	26

3.6.	Database table structure	28
3.6.1.	What is SQLAlchemy?	28
3.7.	Chatbot	30
3.7.1.	Development.....	30
3.7.2.	Training	32
3.8.	User Interface design	33
3.8.1.	Template used.....	34
3.8.1.1.	What is Jinja?.....	34
3.8.1.2.	Usage of Jinja to achieve reusability	34
3.8.2.	Dynamically populate drop downs	37
3.8.3.	Send Email.....	39
3.8.4.	Dashboard Charts.....	40
3.9.	Security Measures	41
3.9.1.	Querying Database tables	41
3.9.2.	Menu Authority Structure.....	42
3.9.3.	Password Hashing	42
3.10.	Test Plan.....	43
3.11.	Implementation environment	44
4.	Evaluation	45
4.1.	Additional capabilities and features	46
4.2.	Lessons learnt	47
4.3.	Problems Encountered	47
4.4.	Overall comment	48
5.	Conclusion	49
	References	50
	Appendix A - MYSQL tables created by python models	52
	Appendix B – Test Cases	54
	Appendix C – User Manual	56
	Appendix D – Admin Manual.....	60

List of figures

Figure 2.1 Use case of common help desk systems	6
Figure 2.2 UVdesk create ticket screen.....	7
Figure 2.3 Zammad Dashboard	8
Figure 2.4 Zammad Ticket view.....	8
Figure 2.5 OSTicket Ticket filter	9
Figure 2.6 HelpDesk ticket view	10
Figure 2.7 Vision Helpdesk Incidents view	11
Figure 2.8 Vision Helpdesk Mobile App	11
Figure 2.9 Tryo service desk System	12
Figure 2.10 Projection of future traffic of major programming languages [14]	14
Figure 2.11 How AJAX Works [16].....	15
Figure 2.12 Iterative Waterfall Model.....	17
Figure 2.13 Git Hub repository of the project.....	18
Figure 2.14 Trello board of the project.....	18
Figure 3.1 Trello board of the project.....	19
Figure 3.2 All users use case diagram	20
Figure 3.3 login to system and user base menu activity diagram.....	21
Figure 3.4 create ticket activity diagram.....	19
Figure 3.5 reply to ticket activity diagram	23
Figure 3.6 create user activity diagram.....	24
Figure 3.7 Class diagram of the system and user types	25
Figure 3.8 Sequence diagram of the chatbot.....	26
Figure 3.9 ER diagram of the chatbot tables.....	26
Figure 3.10 ER diagram of the core tables	27
Figure 3.11 User Model.....	19
Figure 3.12 Process flow of chatterbot	30
Figure 3.13 Chatterbot configuration	31
Figure 3.14 Chatterbot training data	32
Figure 3.15 Chatterbot training python code	32
Figure 3.16 how Chatterbot training data store in the database	33
Figure 3.17 Dashboard view on a larger screen.....	33
Figure 3.18 Dashboard view on a smaller screen	34
Figure 3.19 Jinja usage to display menu	35
Figure 3.20 Jinja usage to display active menu item.....	35
Figure 3.21 selected menu item.....	36
Figure 3.22 JavaScript and python usage of dynamic dropdown population.....	37
Figure 3.23 call “create_reportto” function when department change.....	37
Figure 3.24 “create_reportto” function	38
Figure 3.25 “reportto” python function.....	38
Figure 3.26 “sendemail” python function	39
Figure 3.27 pass parameters to dashboard HTML page using python	40
Figure 3.28 create bar chart using chart.js using received parameters.....	40
Figure 3.29 created chart using received parameters	41
Figure 3.30 simple SQLAlchemy query with table joins.....	41
Figure 3.31 if non admin user tries to access admin URL, he will see this.....	42
Figure 3.32 python bcrypt for password hashing	42
Figure 3.33 hashed passwords in the database	42
Figure 3.34 System code and folder structure.....	43
Figure 4.1 Low priority ticket	46
Figure 4.2 Medium priority ticket	46
Figure 4.3 High priority ticket.....	47

Figure A.1 Department Model	52
Figure A.2 Ticketmaster Model	53
Figure C.1 Login Screen	56
Figure C.2 Dashboard	57
Figure C.3 Ticket creation page	58
Figure C.4 Ticket creation success message.....	58
Figure C.5 Chat window	59
Figure D.1 Admin menu	60
Figure D.2 Add user page	60
Figure D.3 User creation success message	61
Figure D.4 The E-mail received by the user.....	61

List of Tables

Table 2.1 Comparison of features between available solutions and proposed solution.....	19
Table 3.1 MYSQL User table	28
Table 3.2 User login test case.....	43
Table 3.3 All Software, Hardware and tools used.....	44
Table 4.1 Objective evaluation.....	45
Table A.1 MYSQL Department table	52
Table A.2 MYSQL Ticketmaster table	53
Table B.1 Ticket creation testcase.....	54
Table B.2 Reply to an assign ticket testcase.....	55

List of Abbreviations

AJAX	-	Asynchronous JavaScript and XML
CSS	-	Cascading Style Sheet
ER	-	Entity Relationship
HTML	-	Hyper Text Markup Language
ORM	-	Object Relational Mapper
OS	-	Operation System
RDBMS	-	Relational Database Management System
SDLC	-	Software Development Life Cycle
SQL	-	Structured Query Language
UI	-	User Interface
UML	-	Unified Modelling Language
URL	-	Uniform Resource Locator
YAML	-	Yet Another Markup Language

1. Introduction

In recent years, there has been an immense increase in using information systems by businesses. Most of the businesses nowadays use several information systems for different types of operations. As an example, an Insurance company can have core insurance system for their main insurance activities as well as distribution management system for monitor their adviser performance, customer service software for manage inbound and outbound communication, HR system for manage employee-related information and tasks. All these systems are accessible via company network and this entire enterprise IT environment plays a major role in the daily business operations. So that, keep these systems up and running and prompt solutions for issues are very important. That is where the IT help desk software comes to the picture. IT helpdesk ticketing software is being used by companies to provide a centralized facility to troubleshoot and facilitate solutions to IT-related issues.

1.1. Motivation

Information systems are made up of hardware, software, databases and networks. But the internal end-users of those systems are mostly non-technical people. If a technical issue arises, end users must have a way of informing those to the IT department. The purpose of an IT help desk is to make sure the high availability of information systems by providing prompt actions to end-users' technical issues. This is a very critical piece of any company.

Based on the company size, the number of issues per day could be varied. For large companies like banks, insurance companies etc. this could be hundreds or thousands per day. Most of those are regular issues which can resolve easily. Spending time on that kind of issues unnecessarily will cause delay on most important ones.

Following are the main problems every organization face when it comes to managing IT help desk activities.

- How to manage a large number of user request efficiently?
- How to document and keep track of the issues?
- How to check whether service level agreements are met or not?
- How to get user feedback about the service they received?
- How to prioritize requests? Which one to attend first?
- Can customer service agent resolve the issue immediately? If not, how can customer service agent assign that issue to relevant person? And keep a track of it?
- How to check statuses of the issues raised by each user and what is the progress of it?
- Is there a way to resolve end-users' issues without human interaction?
- How to check weekly monthly or yearly activities of end-users, help desk agents as well as other supporting staff?

- How managers can view their subordinates' activities?
- How to reduce the cost and time taken to resolve technical issues?

1.2. Objectives

The overall aim of this project is to provide an innovative solution for most of the issues faced by businesses when maintaining internal helpdesk activities. To achieve this, the following are some important objectives:

- Solve the recurring issues using an automated way (A chatbot) which reduce the time and cost taken for issue solving. Which makes more time to focus on important or urgent issues. Users can chat with the chatbot first and try to find a solution before raising a ticket.
- Reduce the number of calls received to help desk agents by letting end-users to raise a ticket using the online system.
- Train the chatbot based on user feedback to increase the accuracy of the chatbot.
- Make it easy for relevant parties to attend urgent issues first based on prioritized tickets.
- Self-evaluate the tickets being assigned or work has been completed.
- Manage the issue solving workflow with a more transparent way which all the involved parties can check the progress.
- Improve the end-user satisfaction about overall technical issues solving process.
- Ease management decision making regarding IT helpdesk tasks.
- Enhance the quality of service and meet the service level agreements of IT help desk.
- Improve the efficiency of overall IT help desk functions.

1.3. Scope

- This project involves developing a web-based IT ticketing system with a chatbot. Which would help to manage IT helpdesk functions efficiently and cost-effective way in the medium to large organizations.
- This system will only manage IT-related issues which occur within the organization. Customer support features are out of scope.
- When a user starts to chat with the bot, he will ask a set of questions to understand what the issue is. Then chatbot will ask the user to follow some instructions to resolve the issue. If chatbot doesn't have an answer, he will ask to raise a ticket.
- At the end of the project, the chatbot will be trained to solve some recurring issues. Not very complex issues.

1.4. Dissertation Organization

The rest of this dissertation is organized as follows:

Chapter 2. Background

In this chapter, we give a summary of background information relevant to the implementation of the project. Also, this consists of analysis about the requirements as well as a review about the similar systems and technologies. After we have a comparison of alternative design strategies, this includes development strategies, hardware and software strategies.

Chapter 3. Methodology

This chapter mainly describes the design and structure of the system. Design of the system includes various types of diagrams which explains the database design, sequence of the system functionalities, different type of users and the functionalities they use. The major codes and modules, as well as the system test plans, are also explained here.

Chapter 4. Evaluation

This chapter explains whether the project objectives satisfied or not, what are the lessons learnt during the project, what kind of problem we faced, have all aspects of the system tested and are those up to the expected level.

Chapter 5. Conclusion

This chapter gives a summary of the results of the project, some of the future work that could be done for further enhancement of the system.

2. Background

There are few researches about the helpdesk systems. The askspoke.com's help desk research [1] shows that help desk software has a big impact on ticket resolution time, and overall productivity of the business functions. This research shows how important a Ticketing system for a company. But unlike other software systems, there are not so many IT-specific Ticketing systems available in the market. This project aims to add extra features which not available in the market.

Why do we use a chatbot for this IT ticketing system?

According to salesforce.com's chatbot statistics 2019 [2], 77% of customers say chatbots will transform their expectations of companies in the next five years. Also, as they mention "53% of service organizations expect to use chatbots within 18 months — a 136% growth rate that foreshadows a big role for the technology in the near future". "The 'State of Service' research found that 80% of service decision-makers believe AI is most effective when deployed with — rather than in place of — humans."

These statistics clearly show the importance of a helpdesk system, as well as AI technologies, are the future of all kind of technologies. This project focus on getting benefit from AI chatbot. Which provide business users to more time to focus on critical business functions rather than wasting time on technical issues.

2.1. Analysis

System analysis fills the understanding gaps between technical and non-technical project stake holders. It describes how the current available systems operates and the what are the requirements of the proposed system.

2.1.1. Functional requirements

Functional requirements describe the functions of the system. What types of task the system performs. Following are the functional requirements of the proposed IT ticketing system.

- Create and maintain user profiles
- Reset password on first login.
- Ability to Set up the initial parameters of the system. Initial parameters include SLAs, Priority Levels etc.
- Menu access based on user roles.
- Dashboard with pie charts which represents the details about the tickets they got assigned, tickets they assigned to others and tickets they closed.
- Create tickets, assign to users and manage tickets.
- Add comments, reassign, close tickets.
- Filter and view tickets.
- Relevant users get Email notifications for all the activities which requires his/her attention.
- A trained chatbot with solutions to recurring simple IT issues.

2.1.2. Non-functional requirements

Non-functional requirements are the specification that describes the system's operation capabilities and constraints that enhance its functionality.[3] Following are the non-functional requirements of the system.

- Privacy – All User password are hashed before save to the database. No one can view or use those unless user share password.
- Security – System use flask SQLAlchemy extension for querying the database. So, SQL injection attacks are almost impossible.
- Efficiency – Because of the system uses the flask SQLAlchemy extension, it never renders a literal value in a SQL statement. Bound parameters are used to the greatest degree possible, allowing query optimizers to cache query plans effectively. [4] Because of that, SQL query performance are high. Also, THE System uses thread-based processes for sending email and some other tasks. So, system users do not have to wait till those tasks competed to perform another task.
- Accessibility – Because this is responsive web-based system, users can access this form anywhere using any device like PC, Tablet or mobile.
- Availability – Because this is web-based system, Availability is high. Also testing process ensures the minimum system issues.
- Efficiency – System design in high efficiency in mind. As an example
- Accuracy – System developed with high level of validations both in data input screens as well as database. Accuracy of the data available in the database are high.
- Maintainability – There are number of configurable parameters available. Business does not need a programmer to maintain the system.

2.2. Overview of currently available systems in the market

Following use case represents the currently available systems. It shows the general view of all the available system in the market and interaction of the users with those.

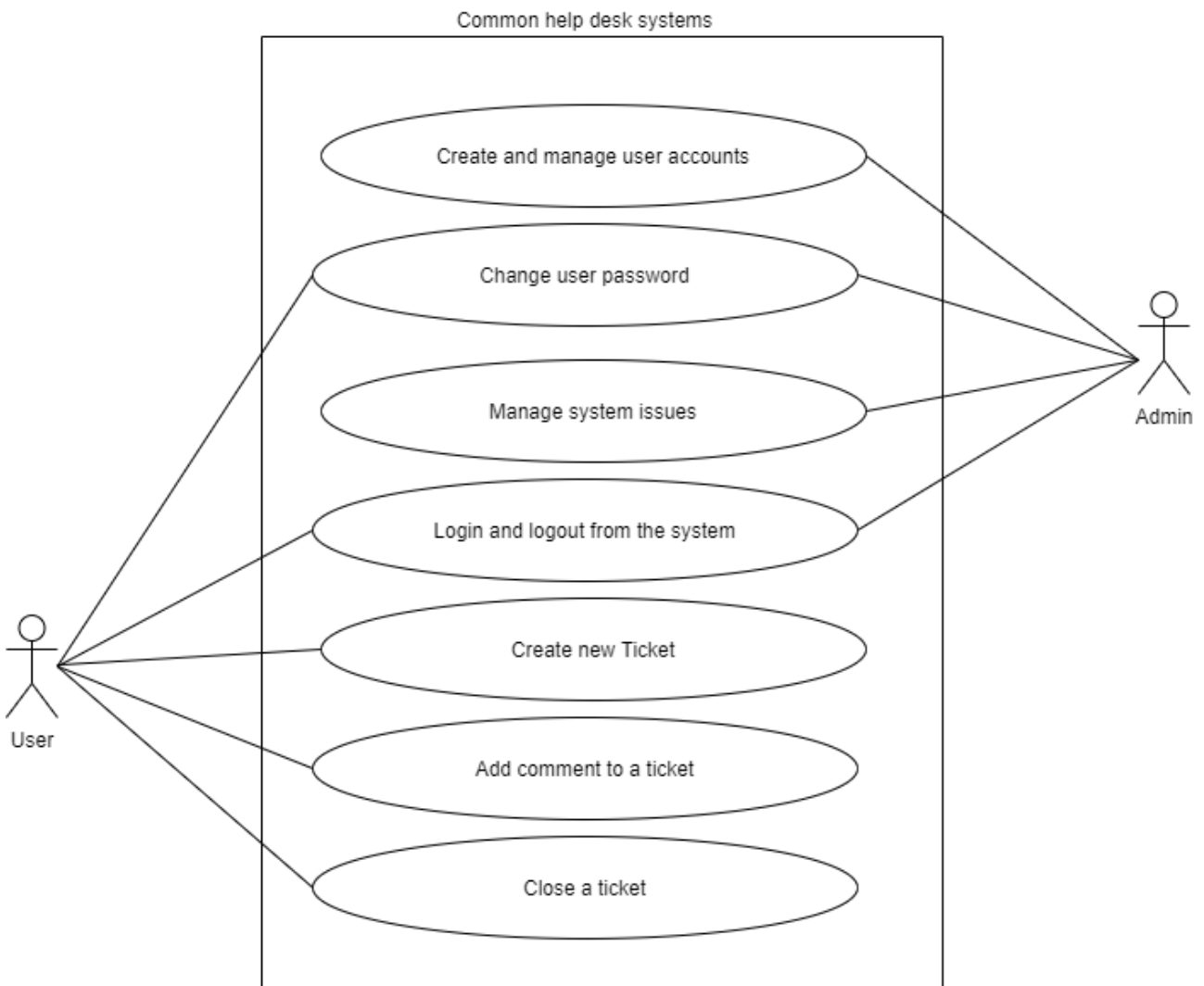


Figure 2.1 Use case of common help desk systems

2.3. Review of similar systems

There are a number of customer relationship management (CRM) systems and Help desk systems available in the market. But IT-related helpdesk ticketing systems are less compared to CRM systems. Here we have divided the similar systems into three categories. That are open source systems, Enterprise systems and Systems developed by Sri Lankan companies.

2.3.1. Open source solutions

Open source solutions are available publicly for free. These kinds of software are developed collaboratively by many developers. Most of the open-source systems have its own enterprise version as well. Usually, opensource systems come with limited features. Following are some most popular open-source helpdesk ticketing solutions and its features.

2.3.1.1. UVdesk [5]

UV desk is a popular open-source, PHP bases helpdesk system. This solution has an enterprise edition as well. Following are some of the useful feature of the open-source system.

- Insight reports
- Ticket management
- Mailbox
- Knowledgebase
- Agents and customers management

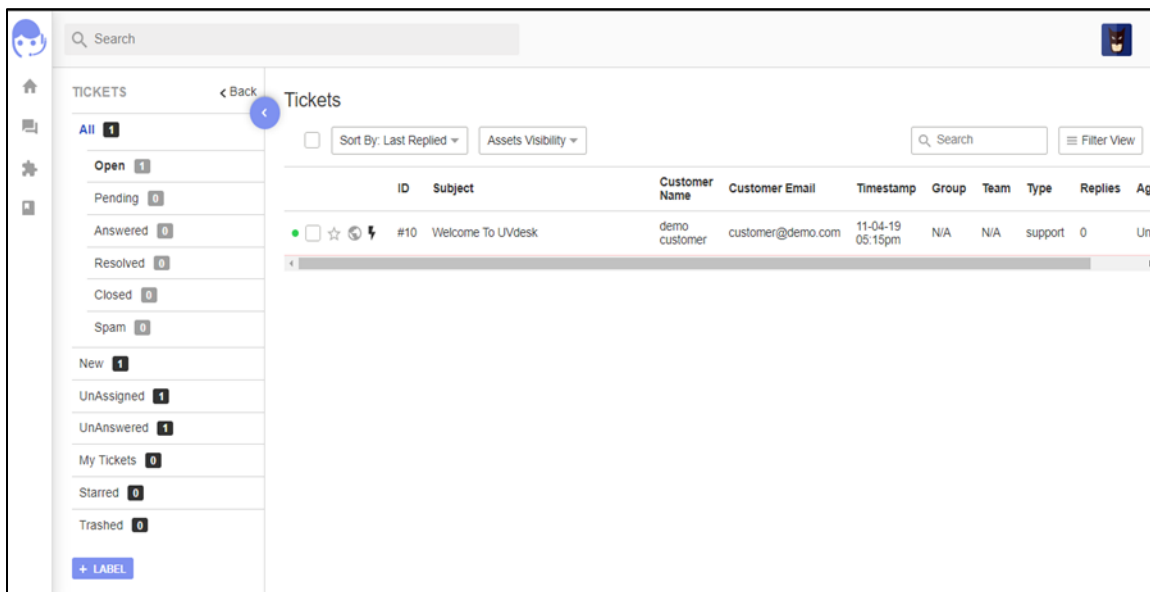


Figure 2.2 UVdesk create ticket screen

2.3.1.2. Zammad Community [6]

Zammad is an open-source user support and ticketing system. This is a simple web-based system which is initially developed by a Germans. This system is developed using the Ruby programming language. Following are the main features of the system [7].

- Dashboard
- Ticket management
- Supports individual escalation or setting client solution time limit
- Multilingual support
- Two-factor-authentication

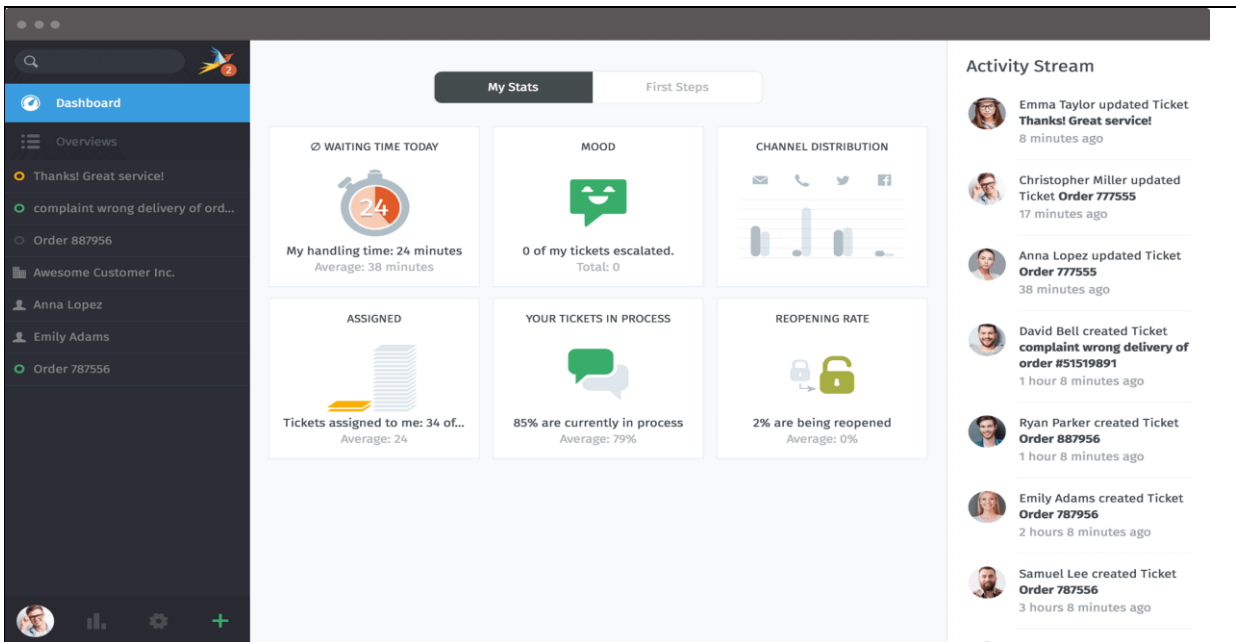


Figure 2.3 Zammad Dashboard

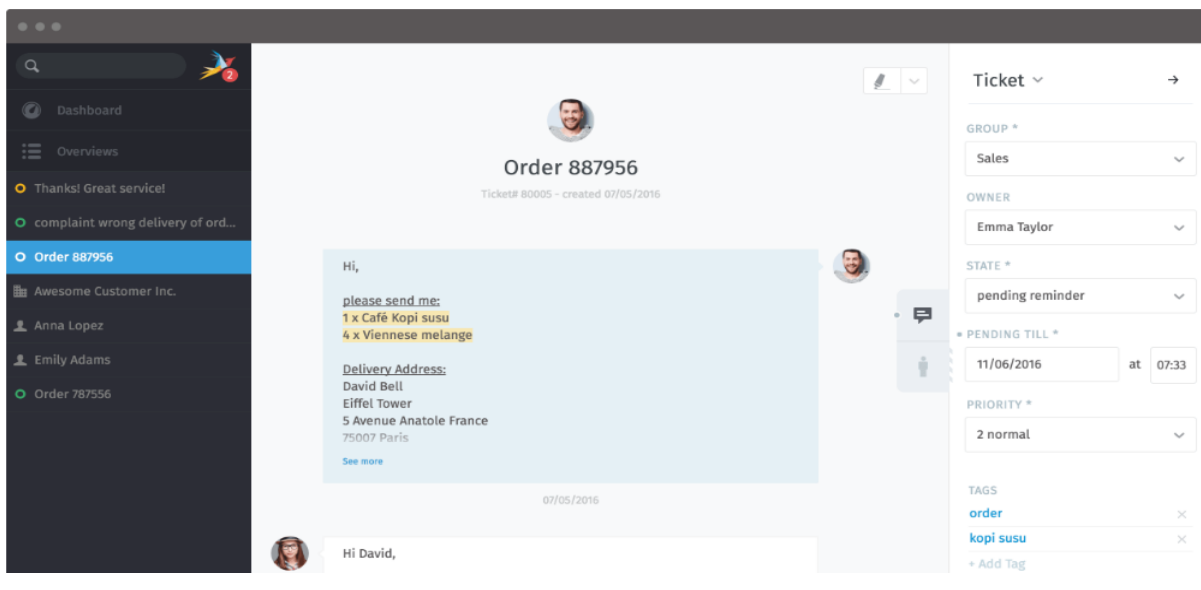


Figure 2.4 Zammad Ticket view

2.3.1.3. OSTicket [8]

This is another popular open source-system available for users. This system also has a community edition. This system is developed using PHP language. Following are some interesting features of this system.

- Custom Fields
- Ticket Filters
- Agent Collision Avoidance
- Assign, Transfer, & Referral
- Thread Action
- Service Level Agreements
- Tasks

Add New Filter

Filters are executed based on execution order. Filter can target specific ticket source.

Filter Name: *

Execution Order: (1...99) * Stop processing further on match! ⓘ

Filter Status: Active Disabled *

Target Channel: ⓘ *

Filter Rules | Filter Actions | Internal Notes

Filter Rules: *Rules are applied based on the criteria.* *

Rules Matching Criteria: Match All Match Any * (case-insensitive comparison) ⓘ

ⓘ ⓘ

ⓘ ⓘ

Figure 2.5 OSTicket Ticket filter

2.3.2. Enterprise solutions

Enterprise solutions are commercially developed, usually owned by a company and users have to purchase a license or a subscription to use the system. These systems sell to customers with certain conditions and are restricted from modifying or redistributing. There are so many helpdesk ticketing systems available in the market. Following are the top two Help Desk ticketing systems according to capterra.com [9].

2.3.2.1. HelpDesk[10]

This is an online ticketing system which aims to simplify work within the team. It also helps to save the time of the customer service agents and enable them to provide the highest level of customer service. Following are some interesting features of the system.

- Categories
- Filters
- Smart search
- Ticket details
- Attachments
- Tagging
- Email notification

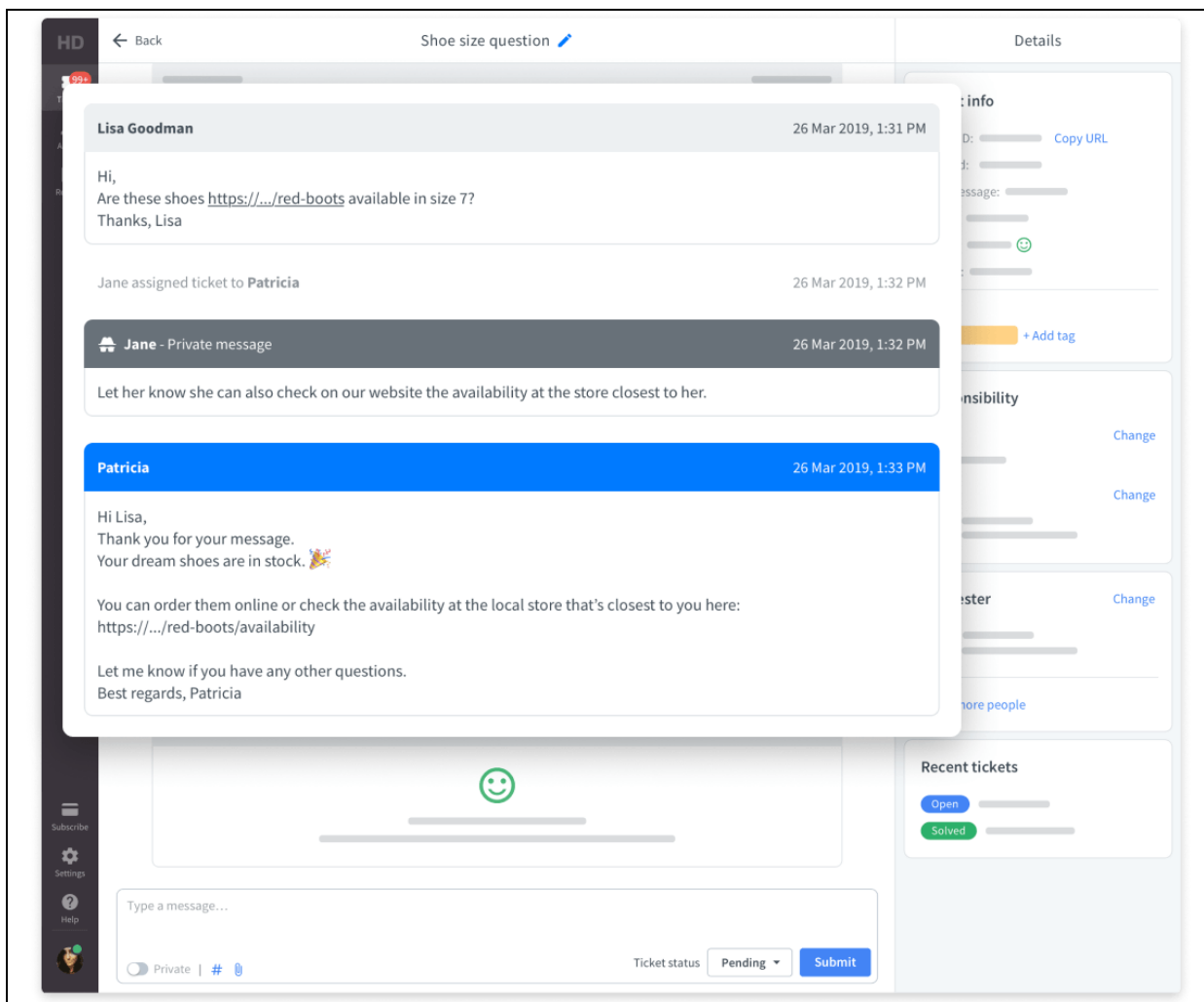


Figure 2.6 HelpDesk ticket view

2.3.2.2. Vision Helpdesk [11]

Vision Helpdesk is all in one customer support help desk software. It allows managing various channels like e-mail, web forums, twitter, facebook and calls. This product is used by more than 1500 companies across the globe. Following are some interesting features available in this system.

- Workflow
- SLA & Escalation
- Notifications
- Ticket based billing
- Task Management
- Mobile App

The screenshot shows the 'Incidents' view in the Vision Helpdesk web application. The interface includes a navigation sidebar on the left with categories like SOLO™ Telecom, Research and Development, General, Open (13), Awaiting (0), Closed, Resolved, Pending (0), Sales, Technical Support, Billing, Marketing, SOLO™ Automation, Customer Support Department, Billing Department, Open (0), Awaiting (0), Closed, and Resolved. The main area displays a table of 12 incidents with columns for Incident hash, Subject, Email, Department, and Priority. The table is paginated to show 1 to 12 of 12 items.

Incident hash	Subject	Email	Department	Priority
SBBL-704899	Can I host the Helpdesk on my own servers	yogesh.patil2436@gmail.com	Technical Support	Critical
BNMO-426289	Are you ITIL certified	john_gudhino@gmail.com	Sales	High
BALW-032197	Helpdesk Annual Billing	boris_rebello@hotmail.com	Billing	High
GZBW-157740	GDPR Compliant	renusharma19@rediffmail.com	Sales	Low
HLFD-426470	Urgent Help for helpdesk installation	Priyanka@yahoo.com	General	Low
SJRI-580141	I have query about 4G plans	max@iamdemo.co	Research and Development	Medium
HBRL-827243	Call @ Sales team		General	Medium
REVF-766389	Meeting with AsiaPacific Business		Sales	Low
RMNF-743051	Best of luck for new team	demo@demo.com	Research and Development	Medium
GBJK-993199	Looking for Helpdesk	braganza.dennis@gmail.com	General	Low
RMPR-286228	V5 - compatibility with php 7	adolftthopil20@gmail.com	General	High
WTHJ-363716	Your request for Helpdesk on live chat	adolfvhd@gmail.com	Sales	High

Figure 2.7 Vision Helpdesk Incidents view

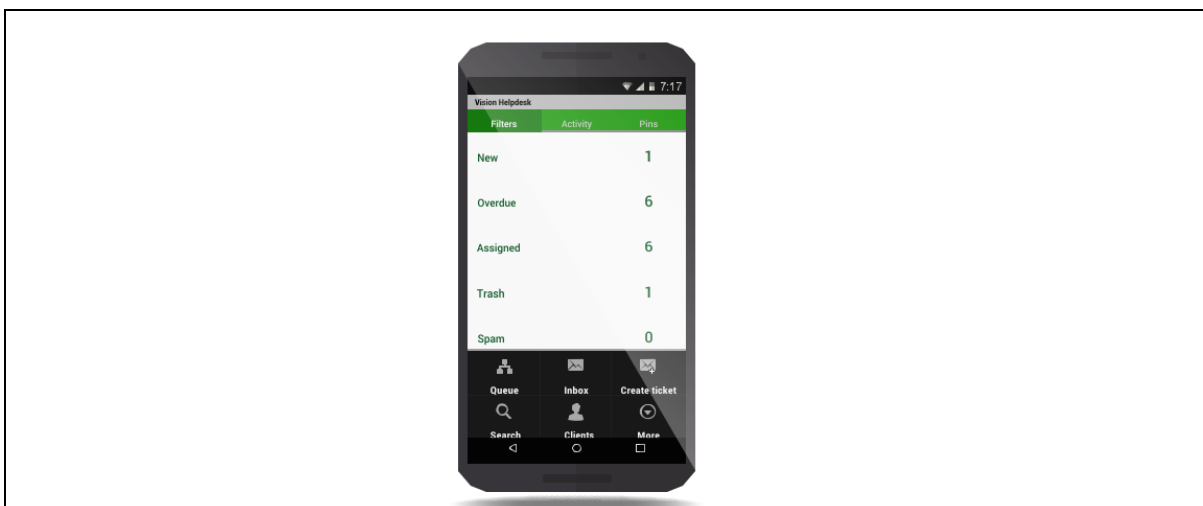


Figure 2.8 Vision Helpdesk Mobile App

2.3.3. Helpdesk Ticketing Systems developed by a Sri Lankan companies.

There is a limited number of helpdesk Softwares developed by Sri Lankan companies. Following are two software solutions available in the Sri Lankan market and its features.

2.3.3.1. Extremeweb Ticket Support System [12]

This is an online help desk software solution for handling client requests through a web-based support ticket system. Following are the features available in the system.

- Online Help Desk Software with a Role-Based Access
- Email Notifications
- Unique Ticket ID
- Custom Ticket Status & Auto Status Update
- Allow / Disallow File Upload
- Ticket ID Recovery

2.3.3.2. Tryo service desk System[13]

This is also an online ticketing system which is developed by Tryonics(PVT) Ltd. It has very limited features like,

- Log reported incidents.
- Maintenance of service standards.
- Report generation for issue analysis.
- Alerts



Figure 2.9 Tryo service desk System

2.3.4. Comparison of features between available solutions and proposed solution

Feature	Open Source Solutions			Enterprise solutions		Sri Lankan Developed		Our solution
	UVdesk	Zammad	OSTicket	HelpDesk	Vision Helpdesk	Extremeweb	Tryo	
Insight reports / Dash Board	YES	YES	YES	YES	YES	Not mentioned	YES	YES
Ticket management	YES	YES	YES	YES	YES	YES	YES	YES
Mailbox	YES	NO	NO	YES	YES	NO	NO	NO
Knowledgebase / FAQ	YES	NO	NO	YES	YES	NO	NO	YES
Agents and customers management	YES	NO	YES	YES	YES	NO	NO	NO
Multilingual support	NO	YES	NO	NO	NO	NO	NO	NO
Two-factor-authentication	NO	NO	NO	NO	NO	NO	NO	Future
Custom Fields	NO	NO	NO	YES	NO	NO	NO	NO
Ticket Filters	YES	YES	YES	YES	YES	YES	YES	YES
Thread Action	NO	NO	YES	NO	NO	NO	NO	YES
Service Level Agreements	NO	YES	YES	YES	YES	NO	NO	YES
Priority Levels	NO	NO	NO	YES	YES	NO	NO	YES
Tagging	NO	NO	NO	YES	YES	NO	NO	YES
Email notifications	NO	NO	NO	YES	YES	NO	NO	YES
Workflow	NO	NO	NO	YES	YES	NO	NO	YES
Mobile App	NO	NO	NO	NO	YES	NO	NO	Future
Alerts	NO	NO	NO	YES	YES	NO	NO	YES
Web based	YES	YES	YES	YES	YES	YES	YES	YES
Chat Bot	NO	NO	NO	NO	NO	NO	NO	YES

Table 2.1 Comparison of features between available solutions and proposed solution

2.3.5. Other systems vs the proposed system

The proposed system is specifically for IT-related issue solving within the company or organization. But all the commercially available helpdesk systems are focus on general Customer Relationship Management (CRM) activities. So that instead of features like the agent and customer management available on commercially available systems, the proposed system has role-based employee profile management feature.

None of the available systems has used Artificial Intelligence (AI) related technologies for better issue solving. But most of the systems have a live chat feature, which is required another employee to respond to the messages. But the proposed system uses a chatbot to resolve issues quickly and efficiently with minimum errors.

2.4. Design Strategy

There are multiple Software development models, tools and technologies available today. For this project, there are some open-source software components used as well as there are some components developed from scratch.

2.4.1. Alternate Solutions

There is plenty of hardware and software available for design developed software systems. Following are the software, hardware and tools used for the development of this system.

Desktop application or Web App?

Web-based systems can be accessed via the internet using any device with internet access. But the desktop application must be installed on a computer before using it. It also depends on the operating system installed on the computer. Web-based applications can be scale very easily when using cloud-based solutions. So, for this project, we develop a web app.

Programming languages?

There are different types of programming languages available for web app developments. There are pros and cons for each of these languages. The main programming language selected for this project is Python. Following are the main reasons to choose python over other web development languages.

- Python is considered as easy to learn a language.
- Keep gaining popularity as backend web development language
- Lots of framework choices
- Available more tools to support with compared to other languages (E.g.: Debugger packages)
- Python usually considered as modern, versatile and simple language compared to languages like PHP
- Python has a large community support.

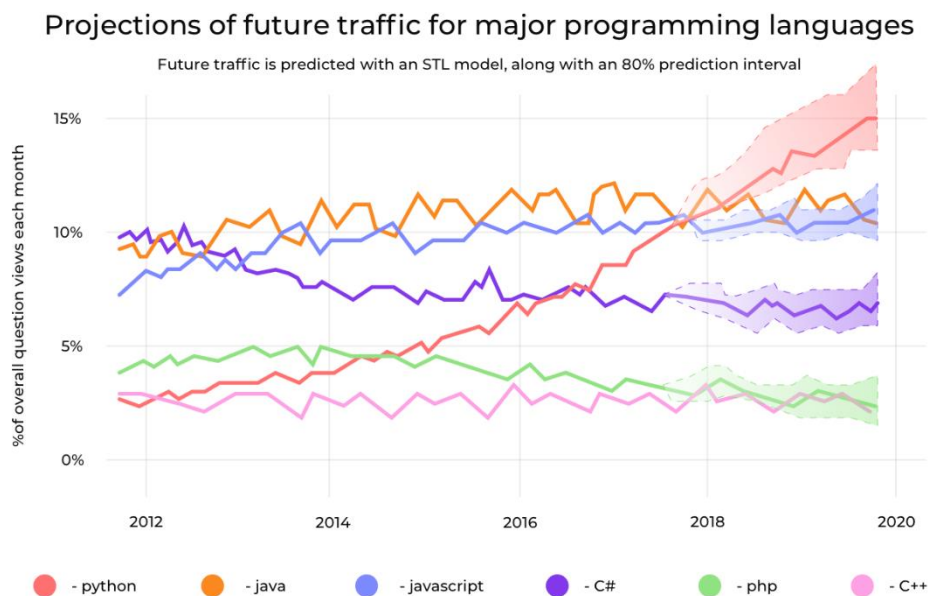


Figure 2.10 Projection of future traffic of major programming languages [14]

Python framework?

Flask python framework used to develop this project. There are multiple python frameworks are available. Most popular options are Django and flask. There are many other python frameworks as well. As an example, Tornado, web2py, Flacon and Pyramid can be considered. Flaks is generally considered as easy to learn framework compared to Django. Also, Django is more suitable for large projects like E-commerce sites.

Database?

MySQL selected as the database of the project. There are plenty of options available when it comes to RDBMS. MySql is the most popular open-source RDBMS system available today. But the current trend shows the decreasing of its popularity. The main considerations for selecting MySql for this project are,

- MySql opens source.
- Large community support.
- High reliability.
- Still the world’s most popular opensource database.

UI Framework?

Bootstrap used as a front-end framework for this project. Without any argument, Bootstrap is the world’s most use frontend web framework currently available. It uses HTML, CSS and JavaScript. Bootstrap has many articles, tutorials, plugging and templates available.

Chatbot development?

ChatterBot python library used as the main framework for the chatbot of this project. “ChatterBot is a Python library that makes it easy to generate automated responses to a user’s input. ChatterBot uses a selection of machine learning algorithms to produce different types of responses. This makes it easy for developers to create chatbots and automate conversations with users.” [15] There are many commercially available chatbots are there. ChatterBot is easy to integrate with Flask.

Other?

There are few other open-source tools/plugins being used for this project.

- AJEX - AJEX used to send data between Flask and web browser. Following figure (Figure 2.10) Shows how the AJEX works.

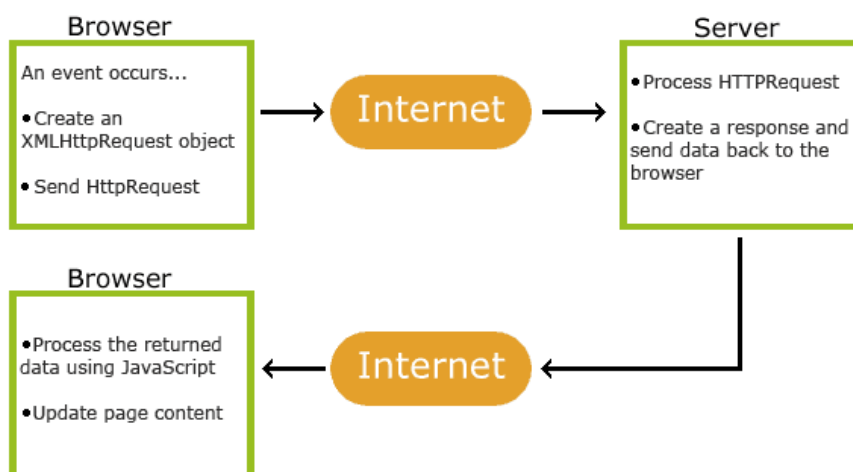


Figure 2.11 How AJAX Works [16]

- DataTables – “DataTables is a plug-in for the jQuery Javascript library. It is a highly flexible tool, built upon the foundations of progressive enhancement, that adds all of these advanced features to any HTML table.” This jQuery plug-in used for tables development in the frontend. [17]
- Chart.js – This is a plug-in for Simple, clean and engaging HTML5 based JavaScript charts. [18] This plug-in used to create charts on the dashboard page.

2.4.2. Software Development Methodology Used

Following describe the thinking behind the decision of selecting Iterative waterfall methodology as the development methodology of this project.

The Software development life cycle (SDLC) model used for this project is the iterative waterfall methodology. Following are the main reasons to choose iterative waterfall methodology.

- This is a mid-sized software project and the requirements are clear.
- Product definition is less dynamic
- Since only one developer is working on the project, complete each phase one at a time is easy.

Following are the reasons for not using other SDLC Methodology.

- Traditional waterfall model – In this model, there is no way to do that changes. Once a step is done changes doesn't accept.
- V-model (Validation and Verification model) – This methodology mostly suits for the projects where failures and downtimes are minimal. (Example – Medical software)
- Spiral model – This methodology suits for a project where requirements are unclear at the beginning, very large and complex projects. (Example – Research and development projects)
- The Rational Unified Process (RUP) – This is mostly suited for Large, high-risk projects.
- Scrum, extreme programming and other agile methodologies – Agile is the most popular SDCL nowadays. These methodologies are the best suit for projects when a group of people work on.

2.4.2.1. Iterative Waterfall Model

Iterative waterfall model contains the same steps as the traditional waterfall model. But the difference is the iterative model allows changes to the previous phase. “The iterative waterfall model provides feedback paths from every phase to its preceding phases, which is the main difference from the classical waterfall model.” [19]

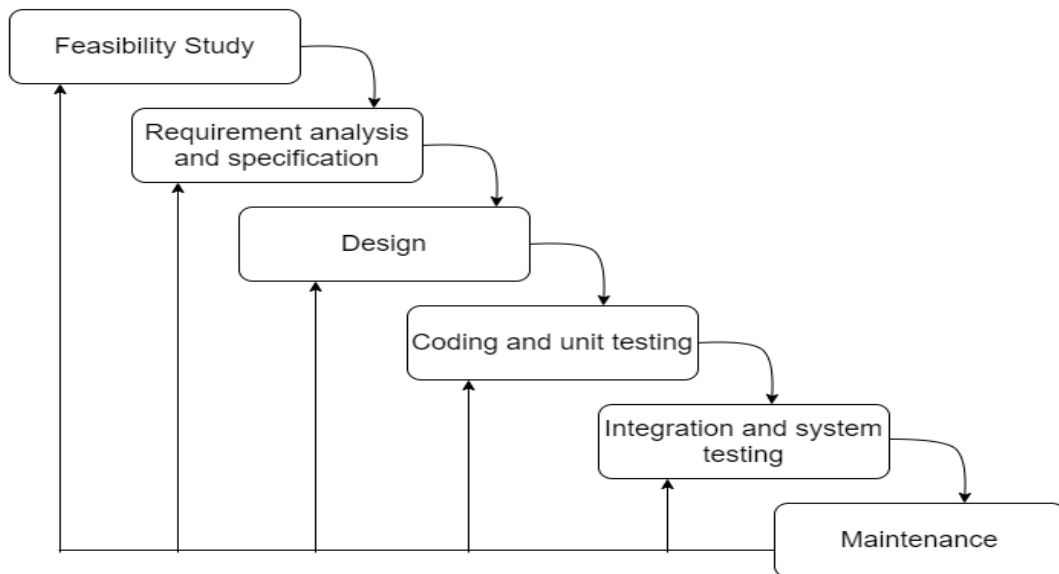


Figure 2.12 Iterative Waterfall Model

- **Feasibility Study:** The main goal of this phase is to determine whether it would be financially and technically feasible to develop the software. The feasibility study involves understanding the problem and then determine the various possible strategies to solve the problem. These different identified solutions are analysed based on their benefits and drawbacks, the best solution is chosen and all the other phases are carried out as per this solution strategy.
- **Requirements analysis and specification:** The aim of the requirement analysis and specification phase is to understand the exact requirements of the customer and document them properly.
- **Design:** The aim of the design phase is to transform the requirements specified in the Software requirement specification (SRS) document into a structure that is suitable for implementation in some programming language.
- **Coding and Unit testing:** In coding phase software design is translated into source code using any suitable programming language.
- **Integration and System testing:** Integration of different modules are undertaken soon after they have been coded and unit tested. Integration of various modules is carried out incrementally over several steps.
- **Maintenance:** Maintenance is the most important phase of a software life cycle. The effort spent on maintenance is 60% of the total effort spent to develop full software.

2.4.3. Tools Used to improve software development process.

Following are two software tools used throughout the project.

Git used for version control. As the Git structure, only two branches being used. One as development and the other one is the master branch. There is no need for having so many branches as only one developer working on this project. The main reason for using git here is to keep track of the changes made during the project life cycle.

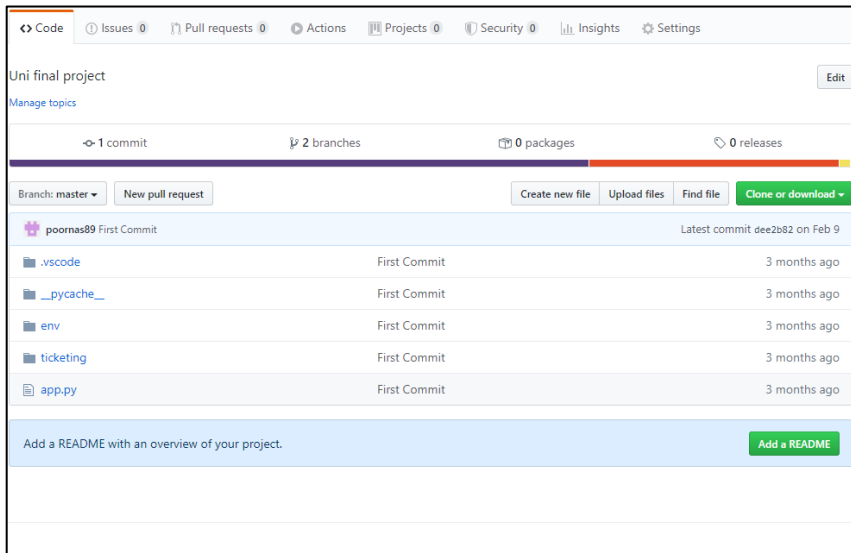


Figure 2.13 Git Hub repository of the project

Also, Trello [18] (A work management tool) board used to keep track of the tasks which need to be performed, which has completed and what are the task currently InProgress. Trello is a collaboration tool that organizes project task into board stickers. This mainly used by agile development. Even though this project uses the iterative development model, this software tool is used to manage small tasks.

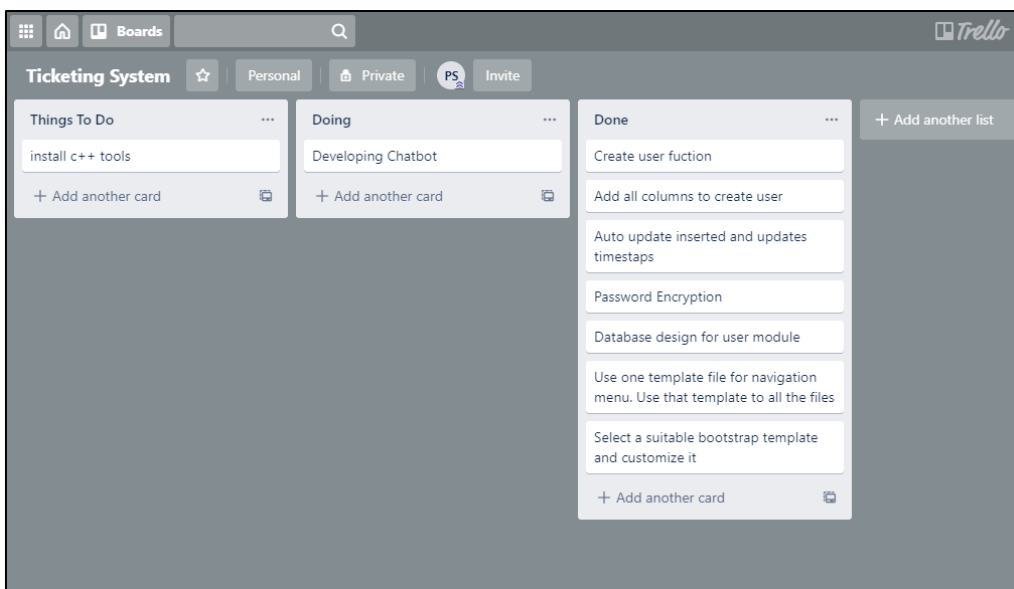


Figure 2.14 Trello board of the project

3. Methodology

This chapter will explain in detail the methodology and the design is being used to develop the project. How the system processes work, what are the software and tools being used, what is the database structure looks like and what are the mechanism being used to develop some security aspects of the system. System developed and designed using object-oriented concepts and modelling. UML diagrams are being used to visually represent the system.

3.1. Use case diagrams

Use case diagrams show the interactions between users and the system using structured text and visuals. Following figure 3.1 and 3.2 show the interaction between system users and different functions of the system.

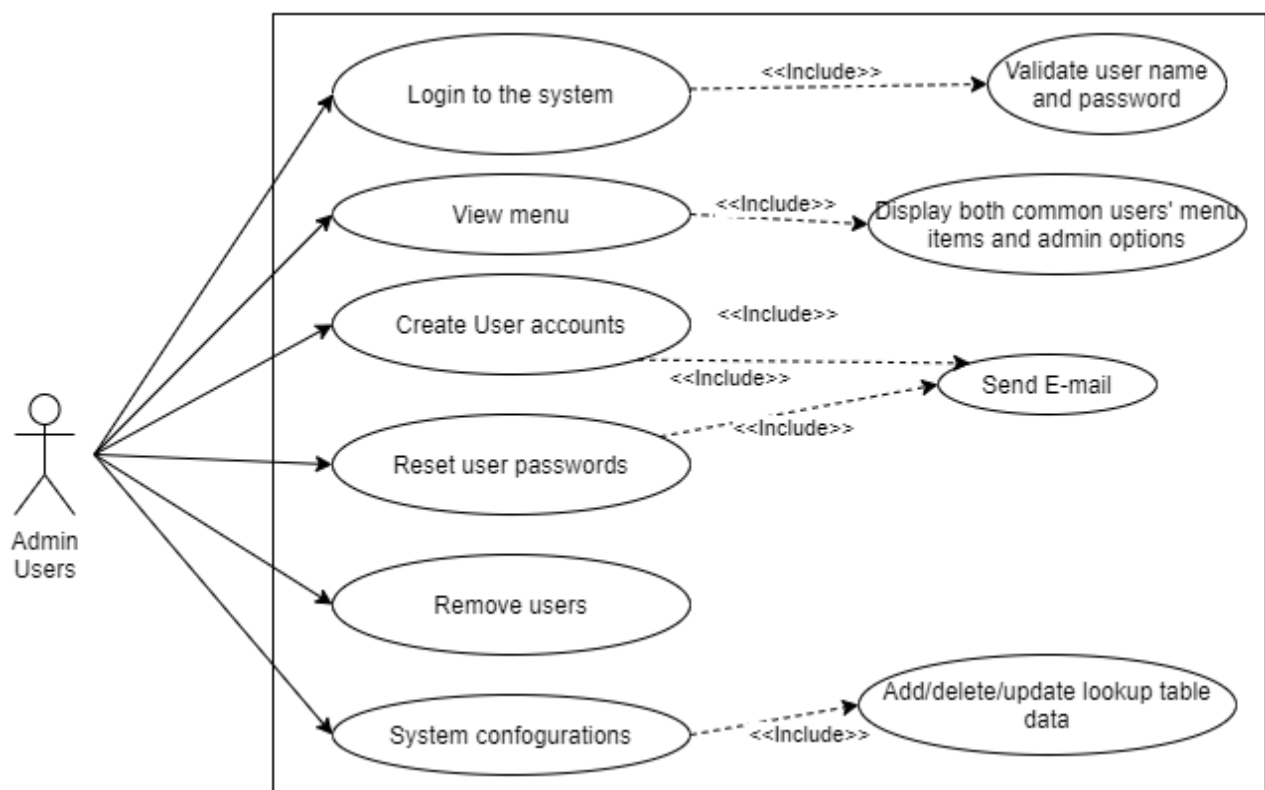


Figure 3.1 Trello board of the project

- After login to the system Admins users has more options in the menu. That options are not accessible for other users.
- Admin can create user accounts and it will trigger an email to the user account holder.
- Admin has authority to rest user password and once password resets an email will be sent to that user.
- Admin can remove user accounts.
- Admin can add, delete or update lookup tables (E.g.: Department table, User grades table)

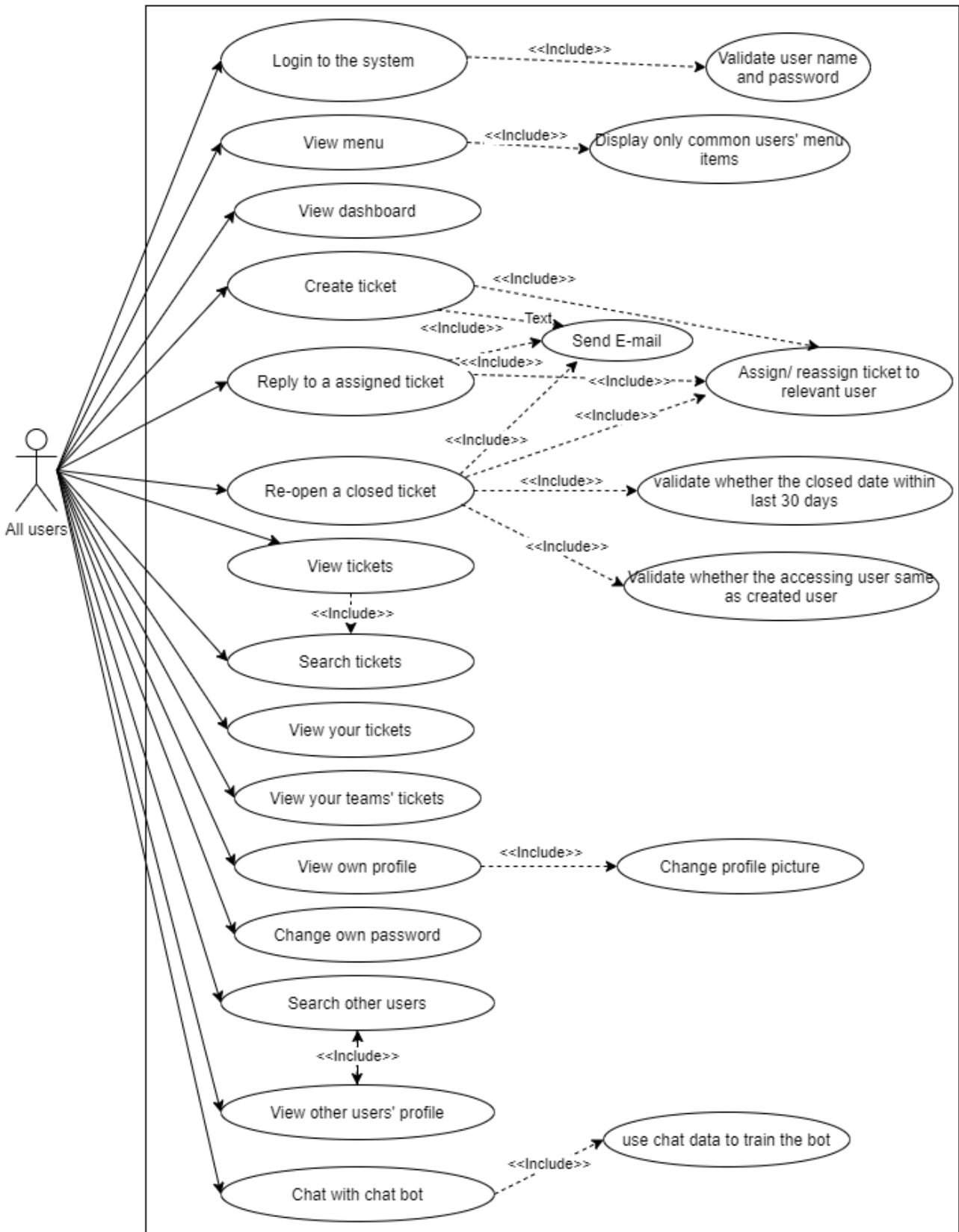


Figure 3.2 All users use case diagram

- After login to the system, all other users apart from admins see the same menu options. Admin sees more admin options.
- Users can see their dashboard as soon as login to the system.
- Users can create a new helpdesk ticket. Assign that to a user can be done while creating the ticket. Once user click create button, the ticket will be created, and an email will trigger to the ticket assigned user.

- Users can see all the tickets assigned to them on the dashboard and reply. Tickets can be re-assign if needed. An email will trigger once the reassignment happened.
- Users can re-open a closed ticket within the 30 days of the ticket closer. But only the ticket created user can re-open a ticket.
- Users can view all the ticket created in the system.
- Users can search for tickets by different parameters.
- Users can view all the tickets which are created by themselves.
- Users can view Tickets which is created by his/her team.
- Users can view their own user profile details and change their profile picture.
- Users can change their password
- Users can view and search all the other users in the system.
- Users can chat with the chatbot and get answers to most of their issues.

3.2. Activity Diagrams

An activity diagram shows the flow of one activity to another activity in the system. Activity diagrams are a bit similar to flow charts. Start of the process is denoted by a filled circle and end of the process denoted by a filled circle inside another circle. Round cornered rectangles show the activities.

Figure 3.3 Activity diagram shows the login process and user-based menu control of the system.

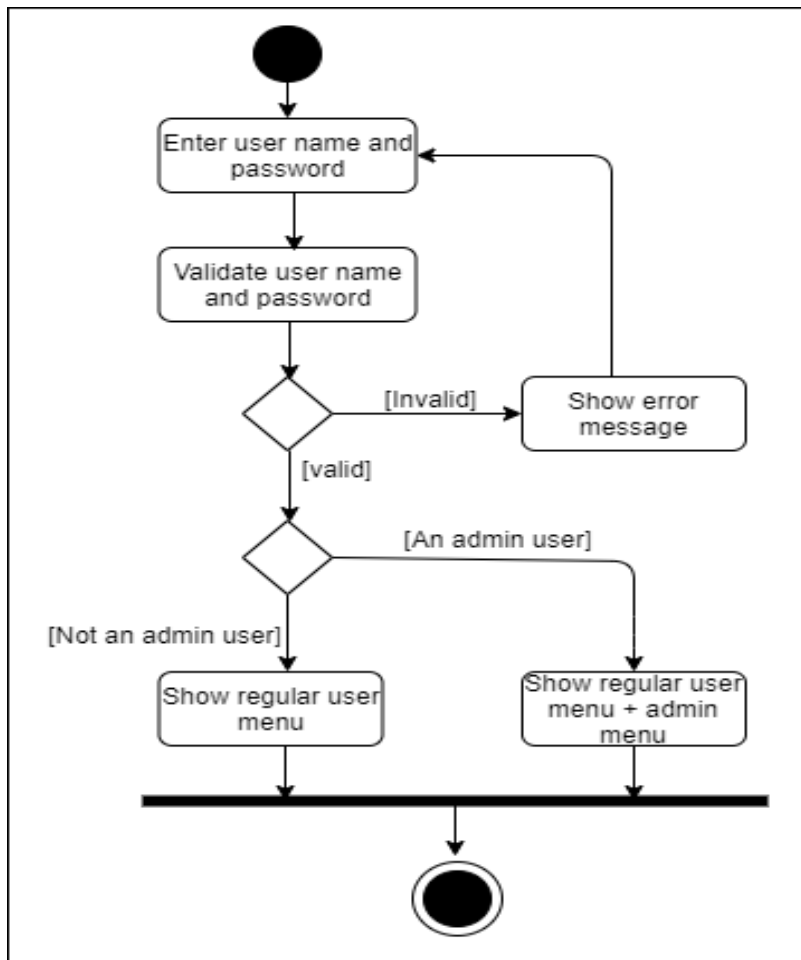


Figure 3.3 login to system and user base menu activity diagram

Figure 3.4 activity diagram shows the ticket creation process and its validations. It represents the high-level logic behind each validation layers.

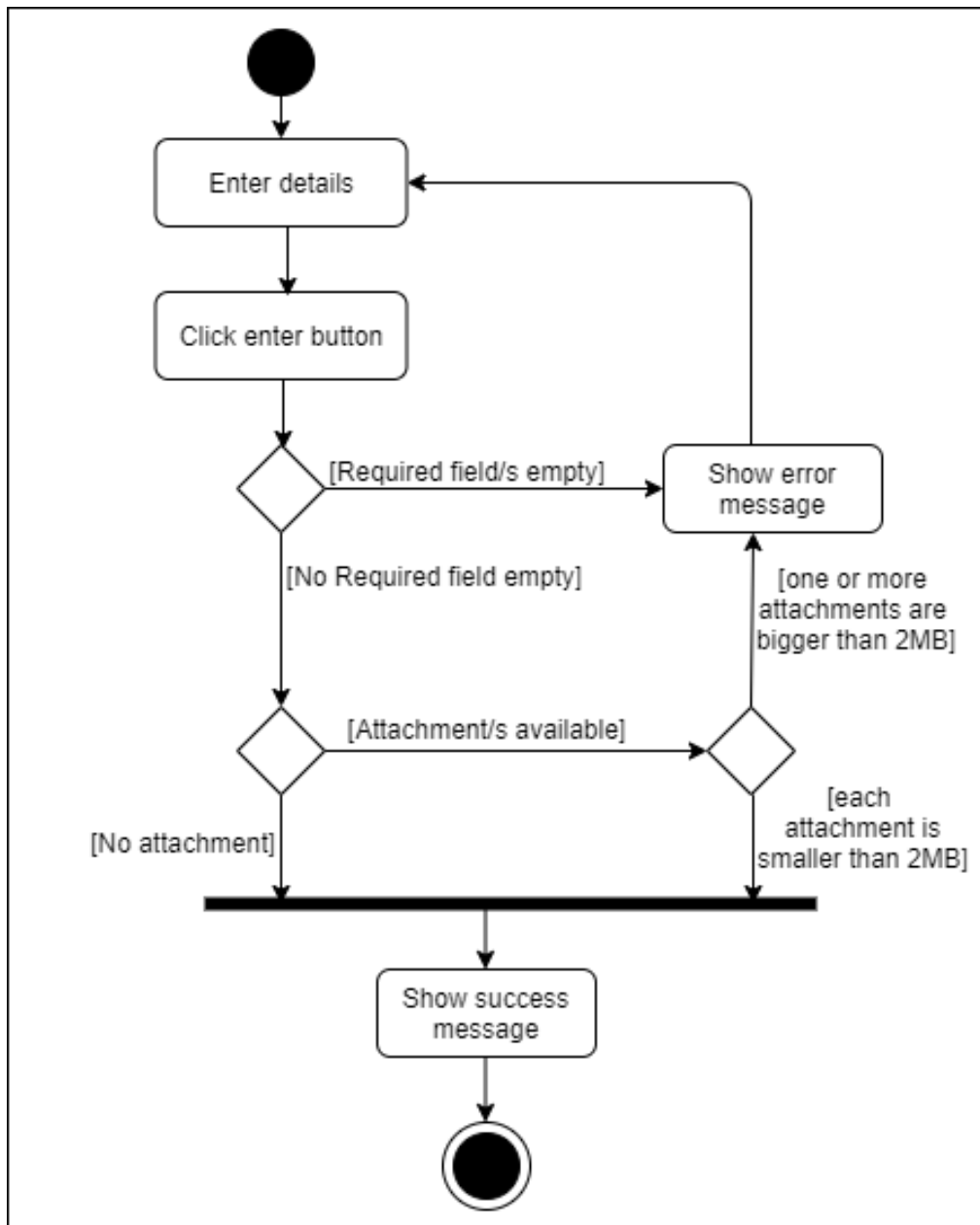


Figure 3.4 create ticket activity diagram

Figure 3.5 activity diagram represents reply to a ticket activity. It contains assign ticket to the correct user. The default selection of the reassign user is the last reply user. If that's not the correct user to reassign this time, reassign to required user.

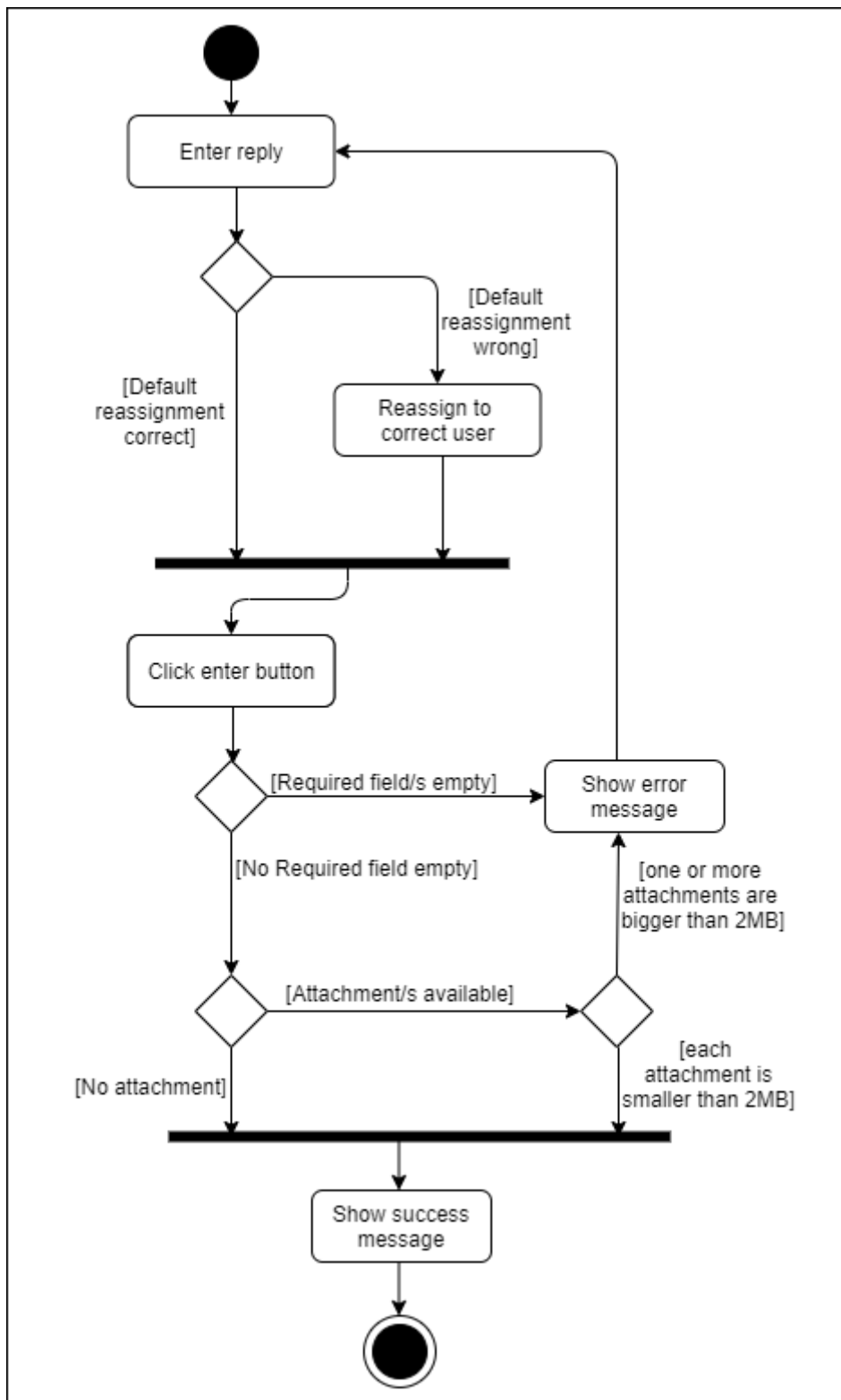


Figure 3.5 reply to ticket activity diagram

Figure 3.6 activity diagram shows the create user activity. It contains the initial password setup, Username validation and other validation activities.

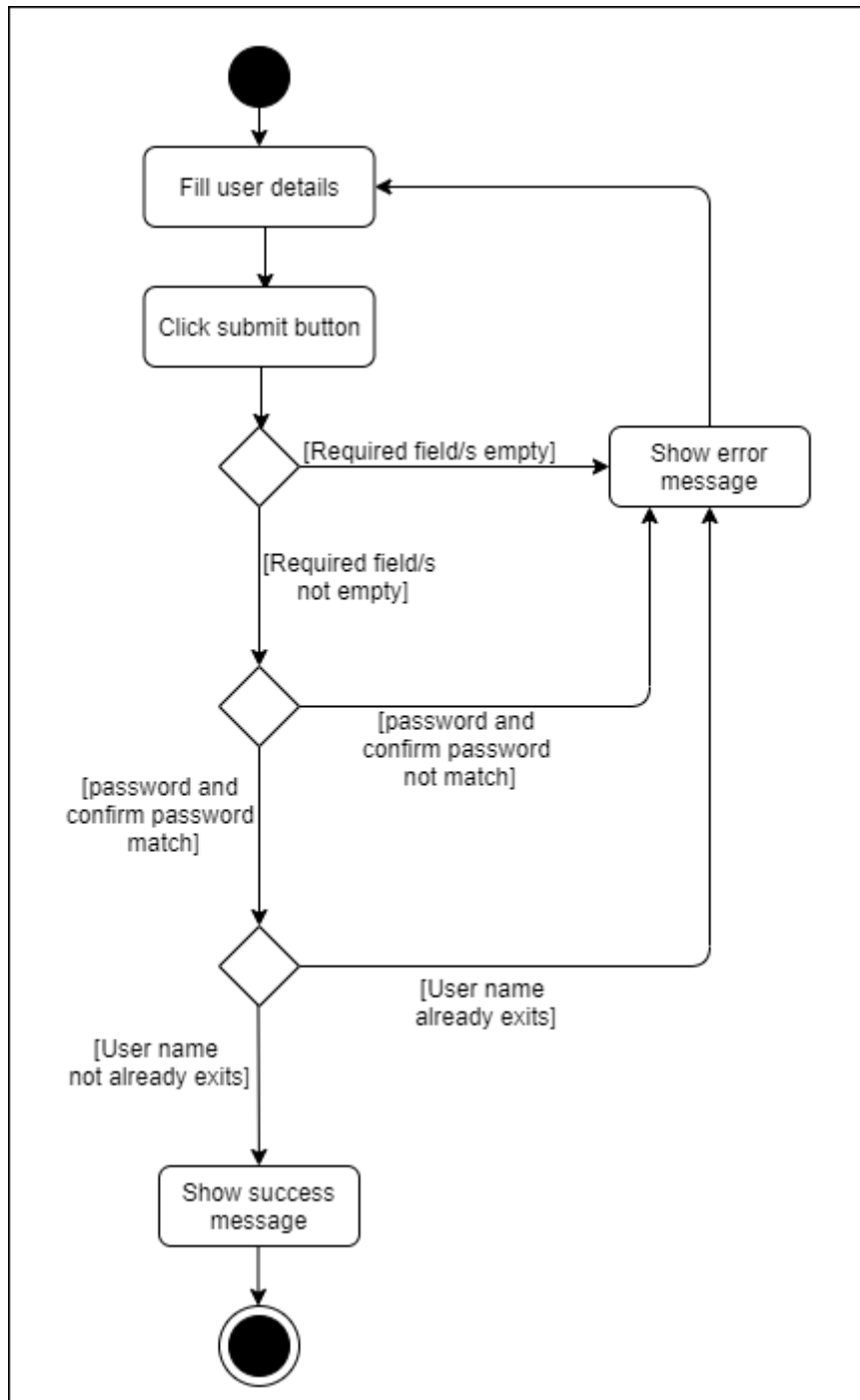


Figure 3.6 create user activity diagram

3.3. Class Diagram

“Class diagrams are used when developing an object-oriented system model to show the classes in a system and the associations between these classes. Loosely, an object class can be thought of as a general definition of one kind of system object. An association is a link between classes indicating that some relationship exists between these classes. Consequently, each class may have to have some knowledge of its associated class.” [21]

Figure 3.7 shows the high-level view of system classes and it’s main methods.

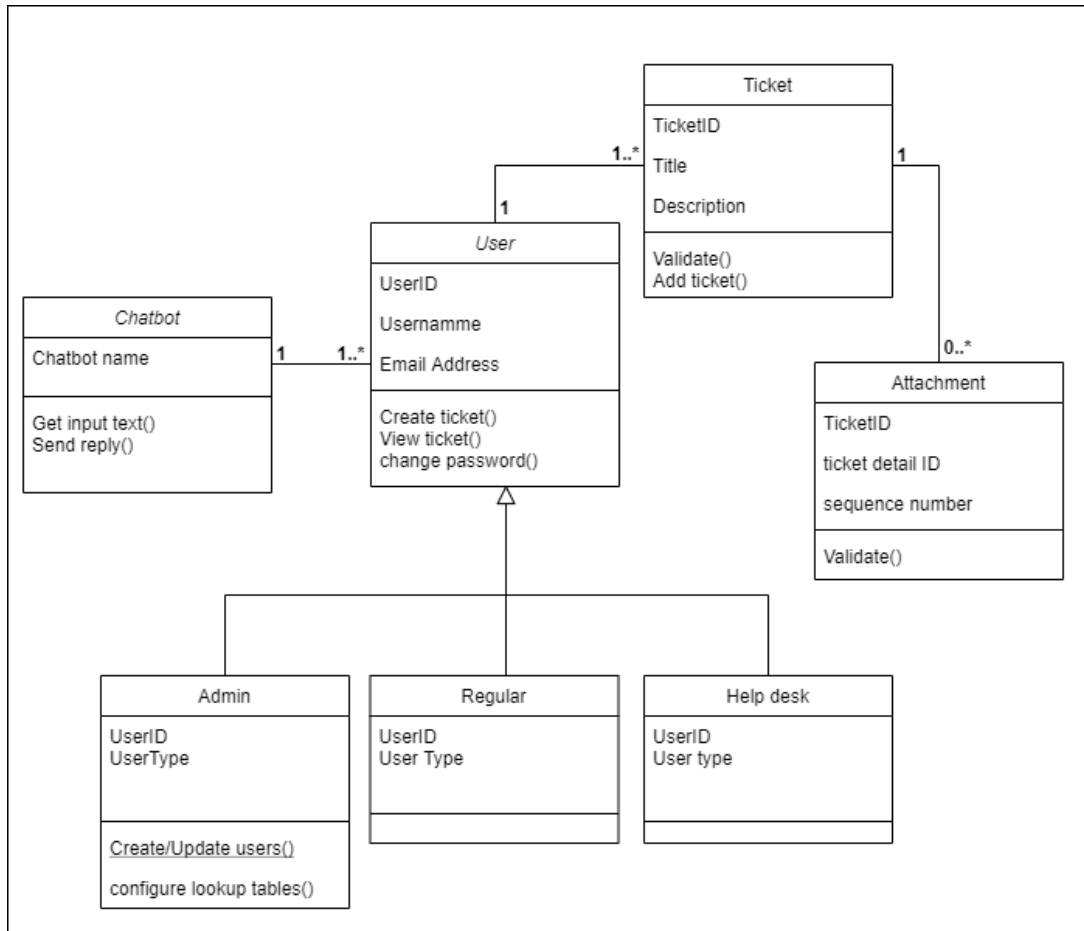


Figure 3.7 Class diagram of the system and user types

3.4. Sequence Diagram

Sequence diagrams represent the interaction between objects. It shows the sequence of those interaction.

Figure 3.8 shows the interaction between the chat window and the trained chatbot in a sequence.

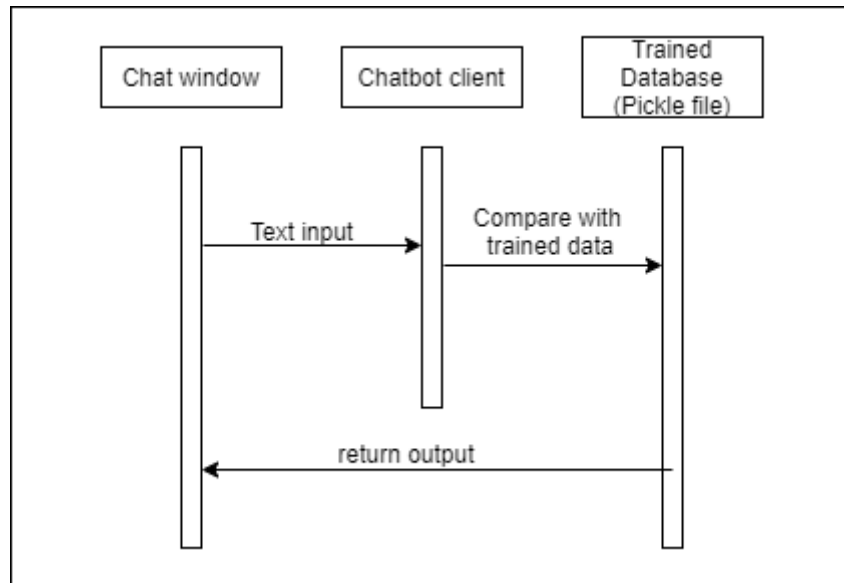


Figure 3.8 Sequence diagram of the chatbot

3.5. ER (Entity Relationship) Diagram

ER diagram displays the relationship between entities stored in a database. In a DBMS, an entity can be a table or an attribute of a table. It explains the logical structure of the database. ER diagrams can be used as the blueprint of the database design.

Figure 3.9 shows the database structure of the chatbot tables. These tables use to keep the data about chatbot training as well as user conversations with the chatbot.

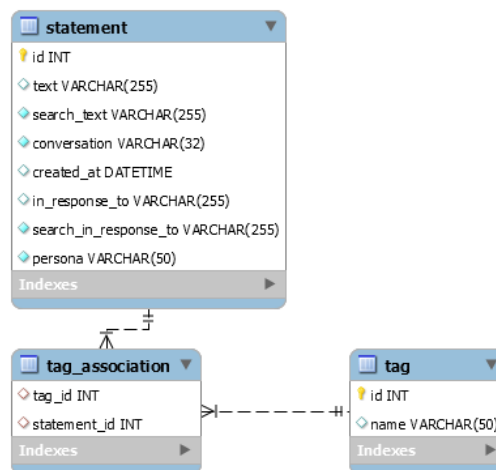


Figure 3.9 ER diagram of the chatbot tables

Figure 3.10 explains the database structure of the core application. It shows the relationships between tables, Main attributes of the tables, Primary keys and foreign keys.

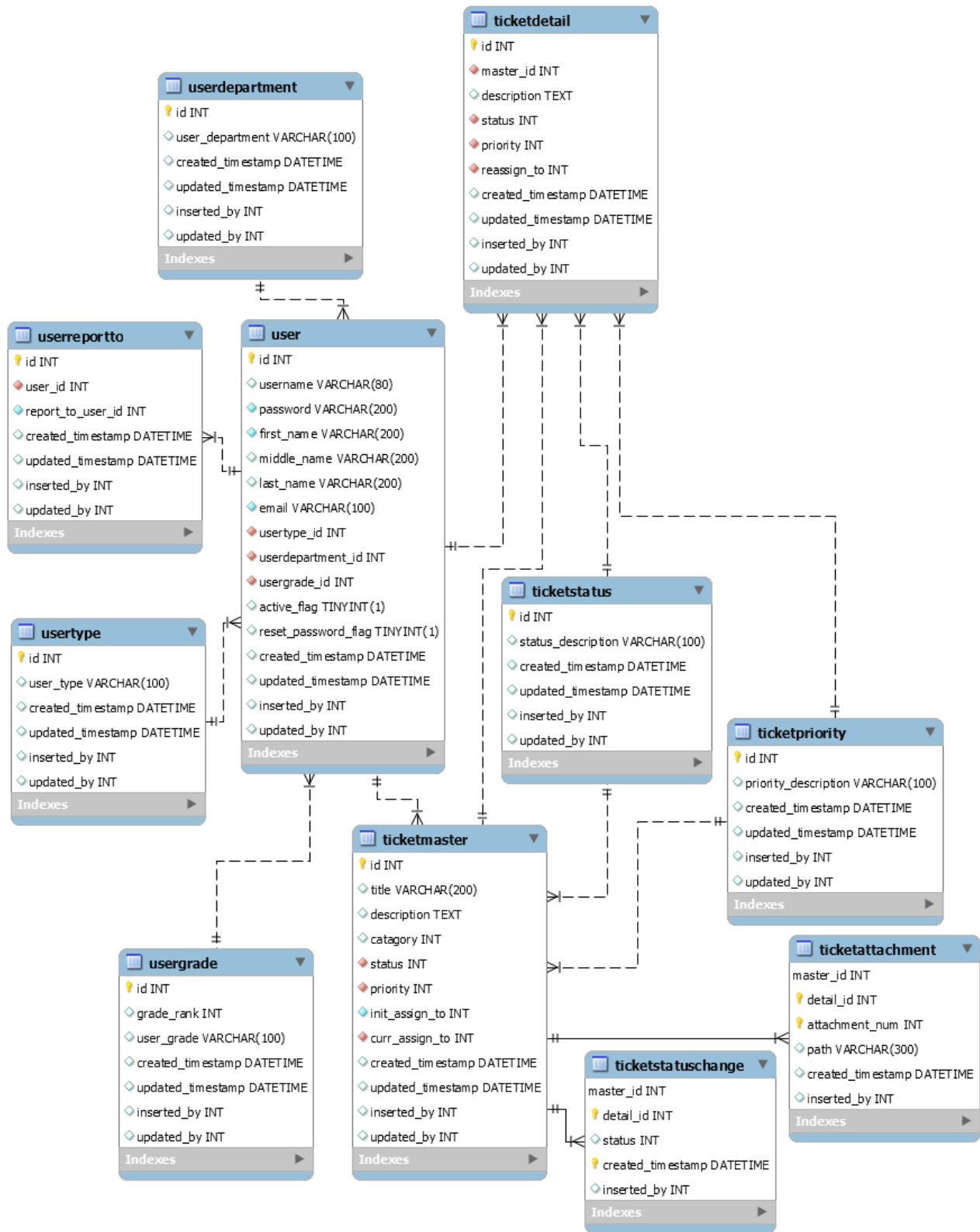


Figure 3.10 ER diagram of the core tables

3.6. Database table structure

All the database tables related activities such as table creation with relationships, querying tables, insert, update and delete data are done using SQLAlchemy ORM. Tables created with the best normalized form possible.

3.6.1. What is SQLAlchemy?

“SQLAlchemy is a library that facilitates the communication between Python programs and databases. Most of the times, this library is used as an Object Relational Mapper (ORM) tool that translates Python classes to tables on relational databases and automatically converts function calls to SQL statements. SQLAlchemy provides a standard interface that allows developers to create database-agnostic code to communicate with a wide variety of database engines.” [22]

In SQLAlchemy ORM each table is referred to as a model. A model is a python class with attributes. These attributes are the columns. There are multiple models available in the database. These models sometimes have a relationship with other model/models.

Table 3.1 shows User table created in MYSQL database and figure 3.11 shows how the User model coded using python.

Full list of MYSQL table and python model comparison given in the Appendix.

Column Name	Description	Data Type and length	Key
id	ID	int	Primary
username	Username	varchar(80)	Unique
password	Password	varchar(200)	
first_name	First Name	varchar(200)	
middle_name	Middle Name	varchar(200)	
last_name	Last Name	varchar(200)	
email	E-Mail	varchar(100)	
usertype_id	User Type ID	int	Foreign
userdepartment_id	User Department ID	int	Foreign
usergrade_id	User Grade ID	int	Foreign
active_flag	Active Flag	tinyint(1)	
reset_password_flag	Reset password flag	tinyint(1)	
created_timestamp	Created date time	datetime	
updated_timestamp	Updated date time	datetime	
inserted_by	Record inserted user ID	int	
updated_by	Record updated user ID	int	

Table 3.1 MYSQL User table

```

class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(80), unique=True)
    password = db.Column(db.String(200), nullable=False)
    first_name = db.Column(db.String(200), nullable=False)
    middle_name = db.Column(db.String(200))
    last_name = db.Column(db.String(200))
    email = db.Column(db.String(100), nullable=False)
    usertype_id = db.Column(db.Integer, db.ForeignKey(
        'usertype.id'), nullable=False)
    userdepartment_id = db.Column(db.Integer, db.ForeignKey(
        'userdepartment.id'), nullable=False)
    usergrade_id = db.Column(db.Integer, db.ForeignKey(
        'usergrade.id'), nullable=False)
    active_flag = db.Column(db.Boolean)
    reset_password_flag = db.Column(db.Boolean)
    created_timestamp = db.Column(db.DateTime, default=func.now())
    updated_timestamp = db.Column(
        db.DateTime, default=func.now(), onupdate=func.now())
    inserted_by = db.Column(db.Integer)
    updated_by = db.Column(db.Integer)
    userreportto = db.relationship('Userreportto', backref='user', lazy=True)
    ticketmaster = db.relationship('Ticketmaster', backref='user', lazy=True)
    ticketdetail = db.relationship('Ticketdetail', backref='user', lazy=True)

    def __init__(self, username, password, first_name, middle_name,
                 last_name, email, usertype_id, userdepartment_id,
                 usergrade_id, active_flag,
                 reset_password_flag, inserted_by, updated_by):
        self.username = username
        self.password = password
        self.first_name = first_name
        self.middle_name = middle_name
        self.last_name = last_name
        self.email = email
        self.usertype_id = usertype_id
        self.userdepartment_id = userdepartment_id
        self.usergrade_id = usergrade_id
        self.active_flag = active_flag
        self.reset_password_flag = reset_password_flag
        self.inserted_by = inserted_by
        self.updated_by = updated_by

```

Figure 3.11 User Model

3.7. Chatbot

3.7.1. Development

Chatterbot python library has been used to develop the chatbot of this project. Chatterbot library using machine learning algorithms to generate responses based on previously trained conversations. Chatbot's accuracy gets higher as it trains more. But when training the chatbot, the training data needs to be carefully examined whether we train the chatbot with correct data. If chatbot trained with wrong data accuracy gets low.

Chatterbot selects the response based on the closest matching trained sentence. It generates a confidence value based on trained data. Following figure 3.12 shows how the chatterbot select response based on the confidence value.

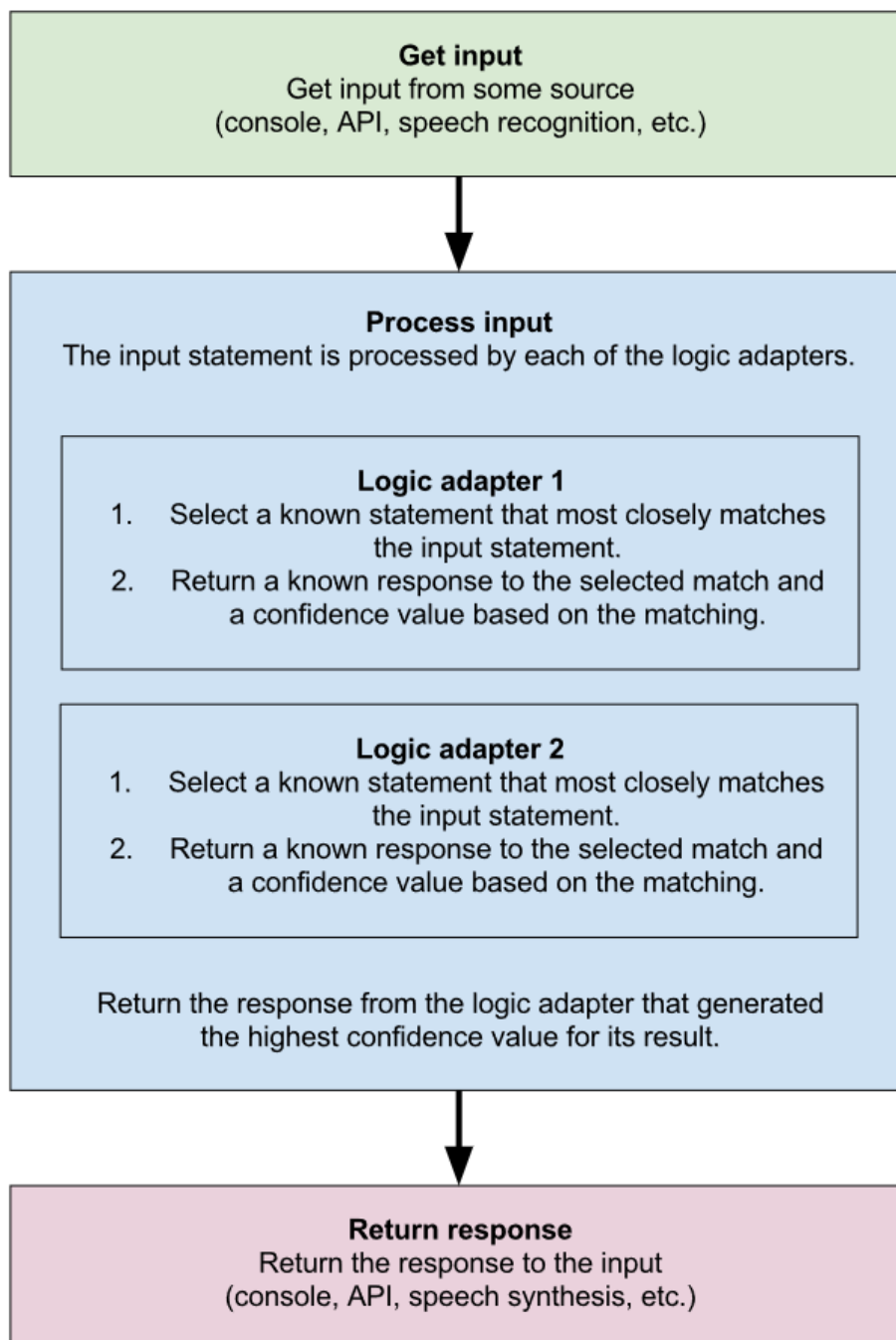


Figure 3.12 Process flow of chatterbot

There are Multiple configurations needs to be done on chatterbot software in order to get the preferred response from the chatbot. Following figure 3.13 shows the configuration made for this project to get the optimal response.

```
# Creating ChatBot Instance
chatbot = ChatBot(
    'TicDeskbot',
    storage_adapter='chatterbot.storage.SQLStorageAdapter',
    logic_adapters=[
        'chatterbot.logic.BestMatch',
        {
            'import_path': 'chatterbot.logic.BestMatch',
            'default_response': 'I am sorry, but I do not understand.\
                Please <a href="{{ url_for("newticket") }}">Click here</a> to\
                create a ticket'
        }
    ],
    database_uri=app.config['SQLALCHEMY_DATABASE_URI']
)
```

Figure 3.13 Chatterbot configuration

Let's look at these configurations one by one.

- Storage adapter – Storage adapter used to connect to different databases. Because of this project uses MYSQL database, storage adapter setup as SQL storage adapter. There are different storage adapters can be used for chatterbot based on the requirement.
- Logic adapters – Logic adapter select the response to the given input. Here we selected the best match logic adapter. There are few other logic adapters available for chatterbot. Those are Time Logic Adapter, Mathematical Evaluation Adapter, Specific Response Adapter. But test results give low confidence when using those adapters for this project.
- Default response – This is the default response users get if the best match response not found
- Database URI – Connection string for the storage adapter. Here in this project, all the connection strings are stored as configurations.

3.7.2. Training

Chatterbot is a corpus-based chatbot solution. Which means it uses a list of written texts as input for the training. Chatterbot comes with the inbuilt tools to help simplify the chatbot training process. For this project, all the corpus data save as YAML files. These files should have a proper format. Following figure 3.14 shows the YAML data format.

```
! blue_screen.yml ×
ticketing > static > chatbot > ! blue_screen.yml
1  categories:
2  - Blue Sreen issue
3
4  conversations:
5  - - I'm getting a blue screen
6    - Simply rebooting can fix the blue screen of death (or STOP error, as it is otherwise known).
7    - Rebooting can help the issue.
8  - - I have got the dreaded blue screen of death!
9    - Simply rebooting can fix the blue screen of death (or STOP error, as it is otherwise known).
10   - Rebooting can help the issue.
11 - - Blue screen issue.
12   - Simply rebooting can fix the blue screen of death (or STOP error, as it is otherwise known).
13   - Rebooting can help the issue.
14 - - Blue screen
15   - Simply rebooting can fix the blue screen of death (or STOP error, as it is otherwise known).
16   - Rebooting can help the issue.
17 - - I am getting a blue screen
18   - Simply rebooting can fix the blue screen of death (or STOP error, as it is otherwise known).
19   - Rebooting can help the issue.
20 - - I'm getting a blue screen
21   - Simply rebooting can fix the blue screen of death (or STOP error, as it is otherwise known).
22   - Rebooting can help the issue.
```

Figure 3.14 Chatterbot training data

Following figure 3.15 shows the Chatbot training python code. It takes all the corpus data set as inputs and store them in the database. As graph data structure. Figure 3.16 explains how chatbot training data save in the database.

```
# Training with Corpus Data
trainer_corpus = ChatterBotCorpusTrainer(chatbot)
trainer_corpus.train(
    './ticketing/static/chatbot/endings.yml',
    './ticketing/static/chatbot/greetings.yml',
    './ticketing/static/chatbot/options.yml',
    './ticketing/static/chatbot/blue_screen.yml'
)
```

Figure 3.15 Chatterbot training python code

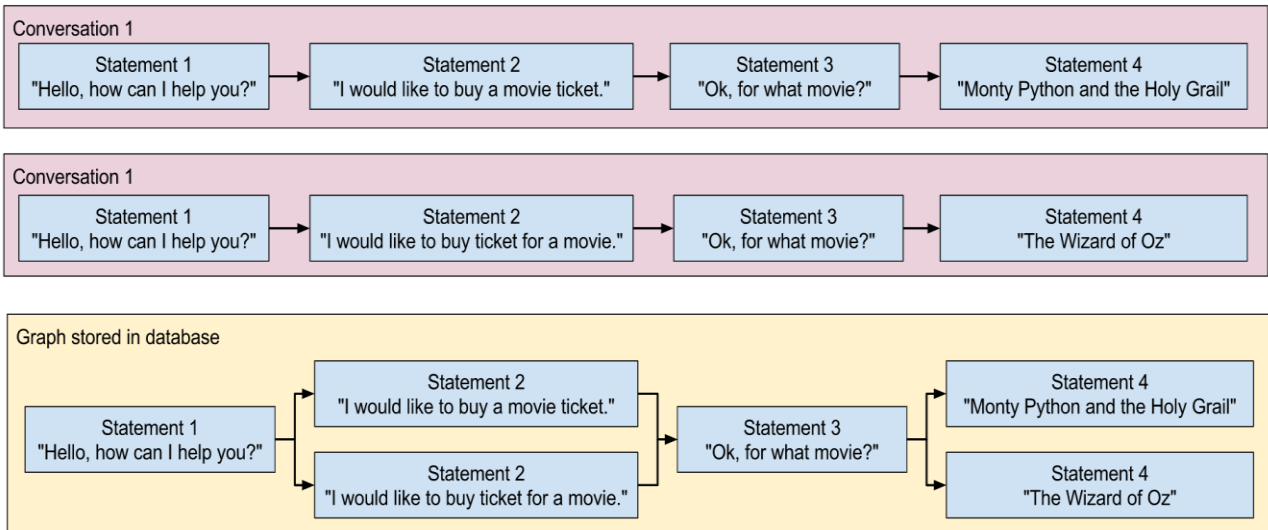


Figure 3.16 how Chatterbot training data store in the database

3.8. User Interface design

All User interfaces of this project designed using responsive web development approach. All the interfaces response perfectly for all the user behaviors, different screen sizes and different platforms. Figure 3.17 and figure 3.18 shows how the system dashboard adjusts according to the screen size.

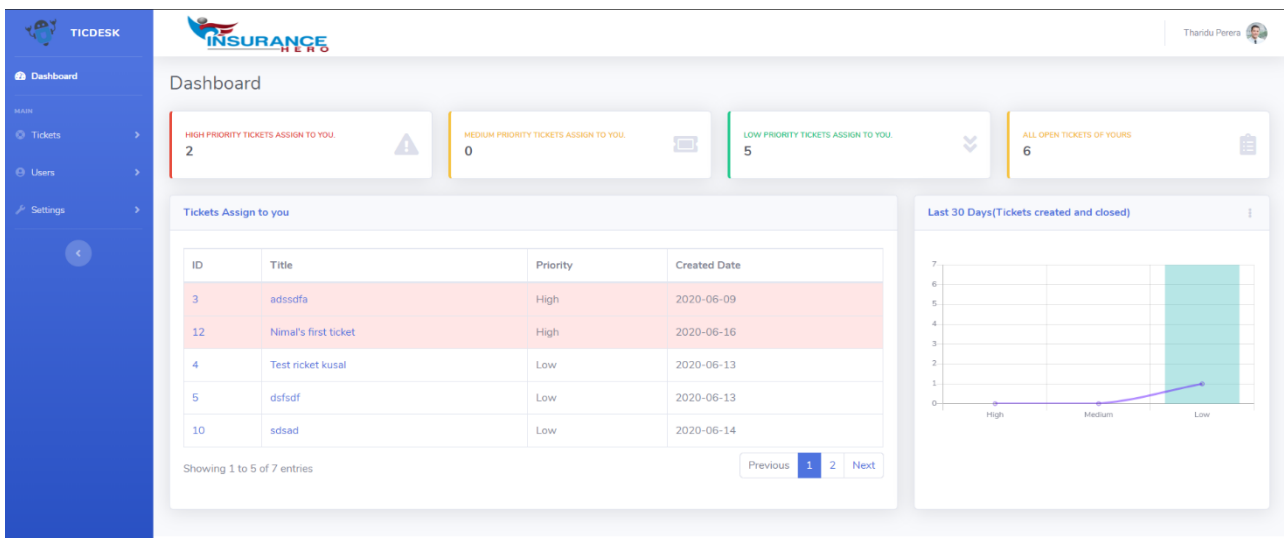


Figure 3.17 Dashboard view on a larger screen

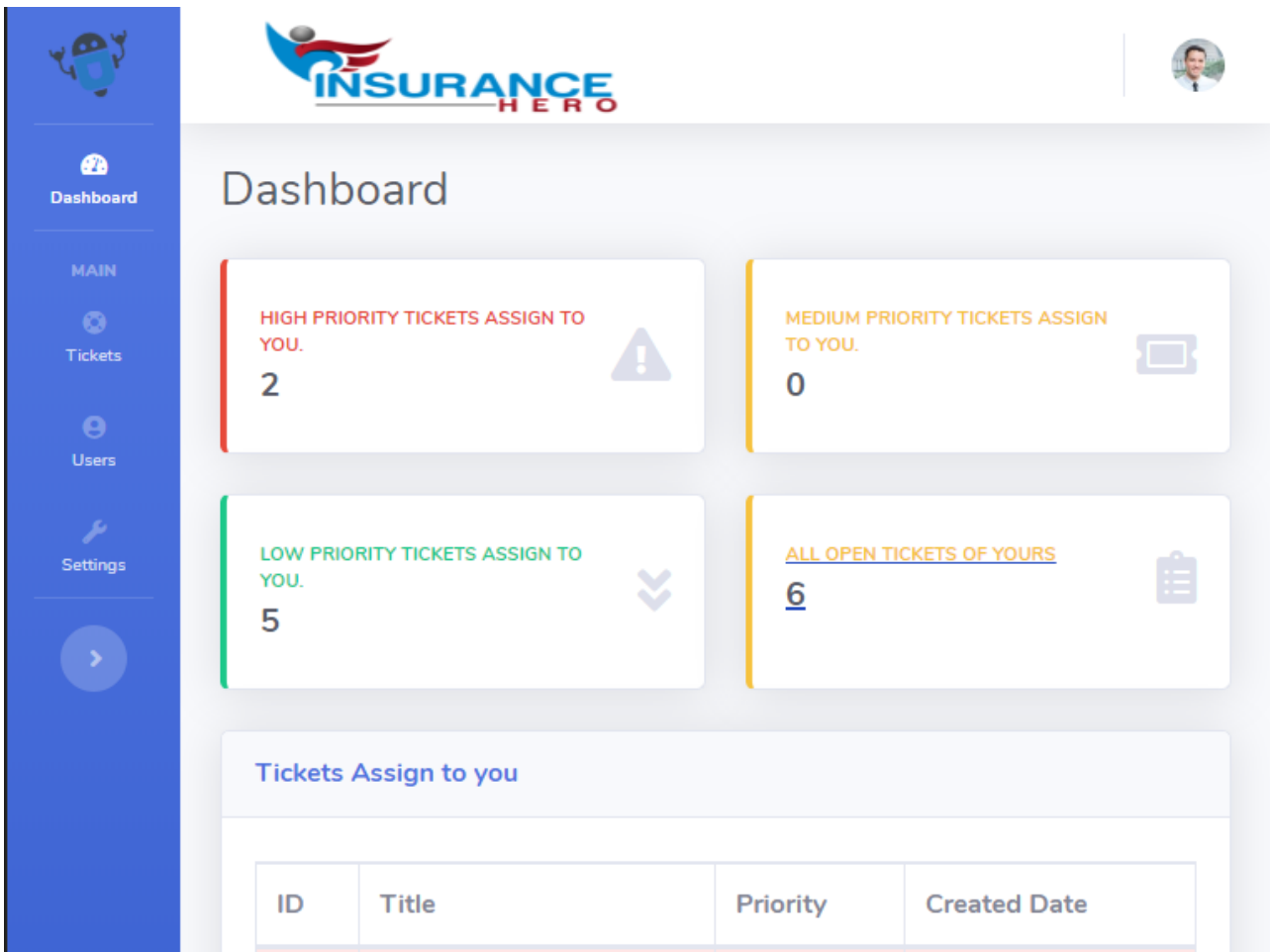


Figure 3.18 Dashboard view on a smaller screen

3.8.1. Template used

SB Admin 2 template used as the based template for the project. “SB Admin 2 is a free, open source, Bootstrap 4 based admin theme” [23]. This template use bootstrap 4 and CSS. For this project, this template highly customized and use with python jinja templates.

3.8.1.1. What is Jinja?

“Jinja is a modern and designer-friendly templating language for Python.” [24] HTML is a static template but Jinja provides dynamic capabilities for HTML tables.

3.8.1.2. Usage of Jinja to achieve reusability

When creating a user-based menu for this project, Jinja capabilities are heavily used. For this project, there is only one HTML document which contains the code for the user menu. Each menu item redirects the user to different HTML page bus menu remains the same with the selected option keep highlighted. These capabilities were not there in the original template. But using Jinja capabilities that were achieved by passing parameters between python and HTML.

Figure 3.19 shows the usage of Jinja templates. “layout.html” contains the HTML code related to the menu and index.html page import all the menu code in.

```
<> index.html X
ticketing > templates > <> index.html > ...
1  {% extends "layout.html" %}
2  {% set active_page = "menu_index" %}
3
4  {% block title %}
5  TicDesk
6  {% endblock %}
7
8  {% block content %}
9  {% block style %}
10 <!-- Custom styles for this page -->
11 <link href="{{ url_for('static', filename='css/chat.css')}}" rel="stylesheet">
12 <!-- Custom styles for this page -->
13 <link href="{{ url_for('static', filename='vendor/datatables/dataTables.bootstrap4.min.css')}}" rel="stylesheet">
14 <!-- Custom styles for page load -->
15 <link href="{{ url_for('static', filename='css/pageload.css')}}" rel="stylesheet">
16 <!-- Custom styles for page load -->
17 <link href="{{ url_for('static', filename='css/ticket-row-color.css')}}" rel="stylesheet">
18 <!-- Custom styles for page load -->
19 <link href="{{ url_for('static', filename='css/Chart.css')}}" rel="stylesheet">
20 <!-- Custom styles for page load -->
21 <link href="{{ url_for('static', filename='css/Chart.min.css')}}" rel="stylesheet">
22 {% endblock %}
23 <!-- Page Heading -->
24 <div class="d-sm-flex align-items-center justify-content-between mb-4">
25 | <h1 class="h3 mb-0 text-gray-800">Dashboard</h1>
26 </div>
```

Figure 3.19 Jinja usage to display menu

Figure 3.20 shows how “layout.html” use Jinja capabilities to view the active menu item. These parameters pass to HTML from python and based on the parameter menu item will show as active and collapsed.

```
<> layout.html X
ticketing > templates > <> layout.html > html > body#page-top > div#wrapper > ul#accordionSidebar.navbar-nav.bg-gradient-primary.sidebar
121
122 <!-- Nav Item - Pages Collapse Menu -->
123 <li class="nav-item {{ 'active' if active_page in ('menu_adduser') else '' }}">
124 | <a class="nav-link {{ 'active' if active_page in ('menu_adduser') else 'collapsed' }}" href="#"
125 | data-toggle="collapse" data-target="#collapsePages"
126 | aria-expanded="{{ 'true' if active_page in ('menu_adduser') else 'false' }}" aria-controls="collapsePages">
127 | <i class="fas fa-fw fa-user-circle"></i>
128 | <span>Users</span>
129 | </a>
130 | <div id="collapsePages" class="collapse {{ 'show' if active_page in ('menu_adduser') else '' }}"
131 | aria-labelledby="headingPages" data-parent="#accordionSidebar">
132 | <div class="bg-white py-2 collapse-inner rounded">
133 | | <h6 class="collapse-header">Edit Users:</h6>
134 | | <a class="collapse-item {{ 'active' if active_page == 'menu_adduser' else '' }}"
135 | | href="{{ url_for('adduser') }}">
```

Figure 3.20 Jinja usage to display active menu item

Figure 3.21 shows how a menu item shows active and main menu category keeps collapsed once user click. According to the figure, the user has clicked the “All tickets” menu item and Tickets main category display as collapsed.

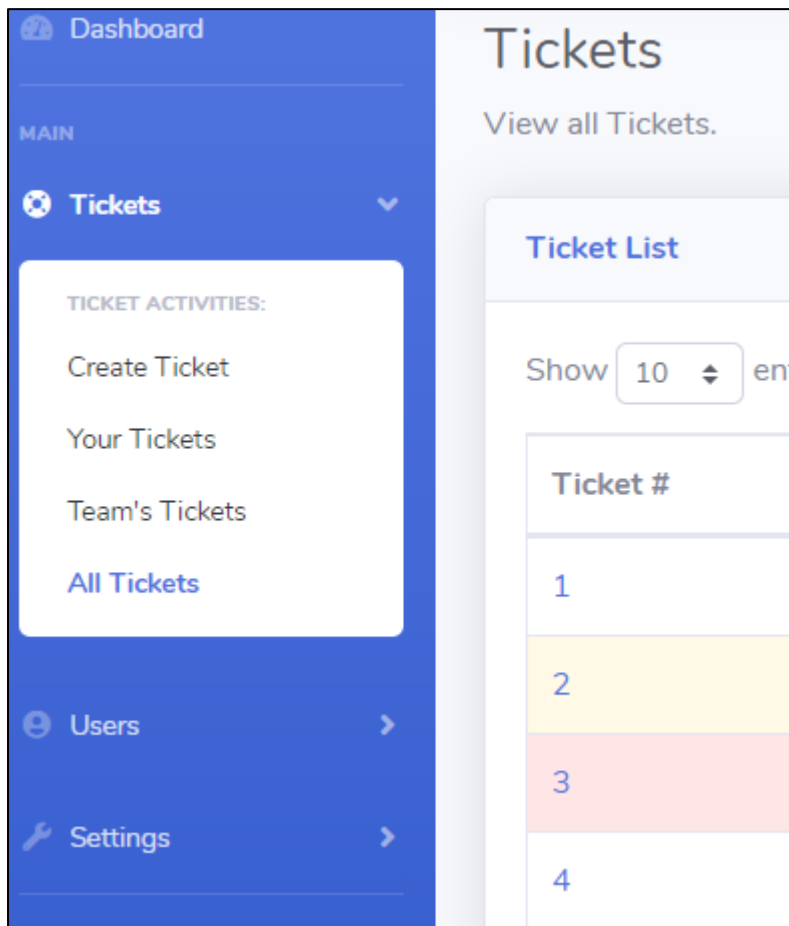


Figure 3.21 selected menu item

3.8.2. Dynamically populate drop downs

Dynamically populate dropdowns which means a dropdown populated by a selection of another dropdown, are used for some input forms of the system. Following figure 3.22 chart shows how Javascript and python are being used together to achieve this.

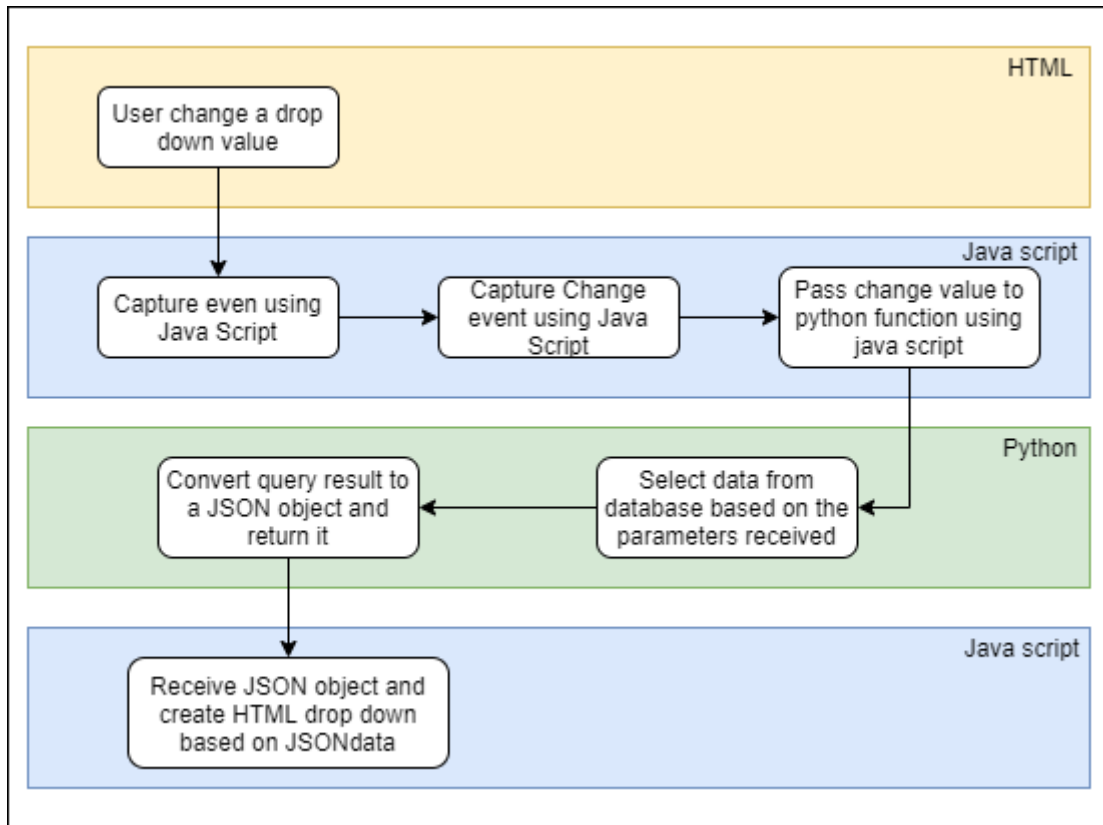


Figure 3.22 JavaScript and python usage of dynamic dropdown population

Figure 3.23 shows the on-change event of the department drop-down. This Javascript code triggers when department dropdown changes and call a function named “create_reportto” figure 3.24 shows the “create_reportto” fiction. 3.25 Shows the python function which returns query results as a JSON object.

```
userdepartment.onChange = create_reportto;
```

Figure 3.23 call “create_reportto” function when department change

```

fetch("/reportto/" + userdepartment_val + "/" + usergrade_val).then(function (response) {
  response.json().then(function (data) {

    let opitonalHTML = '<option value="0">--None--</option>';

    for (let user of data.users_list) {
      opitonalHTML += '<option value="' + user.id + '>' + user.full_name + '</option>';
    }
    userreportto.innerHTML = opitonalHTML;
  });
});

```

Figure 3.24 "create_reportto" function

```

@app.route("/reportto/<department>/<grade>")
def reportto(department, grade):
    if grade != 'None':
        user_list = User.query.with_entities(
            User.id, func.concat(User.first_name, ' ',
                                User.last_name, ' (', User.email, ')')
            .label("full_name")).filter(and_(
                User.userdepartment_id == department,
                User.usergrade_id > grade, User.username != 'Admin'
            )).all()
    else:
        user_list = User.query.with_entities(
            User.id, func.concat(User.first_name, ' ',
                                User.last_name, ' (', User.email, ')')
            .label("full_name")).filter(and_(
                User.userdepartment_id == department,
                User.username != 'Admin'
            )).all()

    user_array = []

    for user in user_list:
        userobj = {}
        userobj['id'] = user.id
        userobj['full_name'] = user.full_name
        user_array.append(userobj)

    return jsonify({'users_list': user_array})

```

Figure 3.25 "reportto" python function

3.8.3. Send Email

Flask-email extension has used for sending emails. Flask itself has a powerful extension to send an email. Using a few configurations, an email can send easily. But for this project, there are some modifications done to the default features. The main modification is introducing the threading for the email function. Because of that email can be triggered separately without delay any activity which the user performs.

Following figure 3.26 shows the send email python function.



```
sendemail.py X
ticketing > sendemail.py > ...
1  from ticketing import mail
2  from flask_mail import Message
3  from threading import Thread
4  from ticketing import app
5
6
7  def send_async_email(app, msg):
8      with app.app_context():
9          mail.send(msg)
10
11
12 def send_mail(subject, to, txt_body):
13     try:
14         msg = Message(subject,
15                       sender="tic.desk.21@gmail.com",
16                       recipients=[to])
17         msg.html = txt_body
18         thr = Thread(target=send_async_email, args=[app, msg])
19         thr.start()
20         print('Mail sent!')
21     except Exception as e:
22         print(str(e))
23
```

Figure 3.26 “sendemail” python function

This function accepts 3 input parameters. Those are email subject, email address and email body. Then a new thread will create for sending the mail.

3.8.4. Dashboard Charts

Dashboard Graphs created using Charts.js opens source flexible charts design solution. To populate data into these graphs data has been pass between python and JavaScript. Blow figures show how the User last 30 days ticket created, and closed graph works.

```
# User is loggedin show them the index page
return render_template('index.html',
    username=session['username'],
    high_priority=high_priority,
    medium_priority=medium_priority,
    low_priority=low_priority,
    all_created=all_created,
    all_assigned=all_assigned,
    created_high_last30=created_hight_last30,
    created_medium_last30=created_medium_last30,
    created_low_last30=created_low_last30,
    closed_high_last30=closed_high_last30,
    closed_medium_last30=closed_medium_last30,
    closed_low_last30=closed_low_last30)
```

Figure 3.27 pass parameters to dashboard HTML page using python

```
var Created = [created_hight_last30, created_medium_last30, created_low_last30];
var Closed = [closed_high_last30, closed_medium_last30, closed_low_last30];
// Pie Chart Example
var ctx = document.getElementById("mybarChart");
var mybarChart = new Chart(ctx, {
  type: 'bar',
  data: data = {
    labels: [Priority[0], Priority[1], Priority[2]],
    datasets: [
      {
        type: 'line',
        label: "Closed",
        fill: false,
        data: [Closed[0], Closed[1], Closed[2]],
        borderColor: 'rgba(102, 26, 255, 0.5)',
      },
      {
        label: 'Ceated', //1D2939
        data: [Created[0], Created[1], Created[2]],
        backgroundColor: [
          'rgba(255, 99, 132, 0.4)',
          'rgba(255, 206, 86, 0.4)',
          'rgba(75, 192, 192, 0.4)'],
        hoverBorderWidth: '3',
      },
    ],
  },
}
```

Figure 3.28 create bar chart using chart.js using received parameters

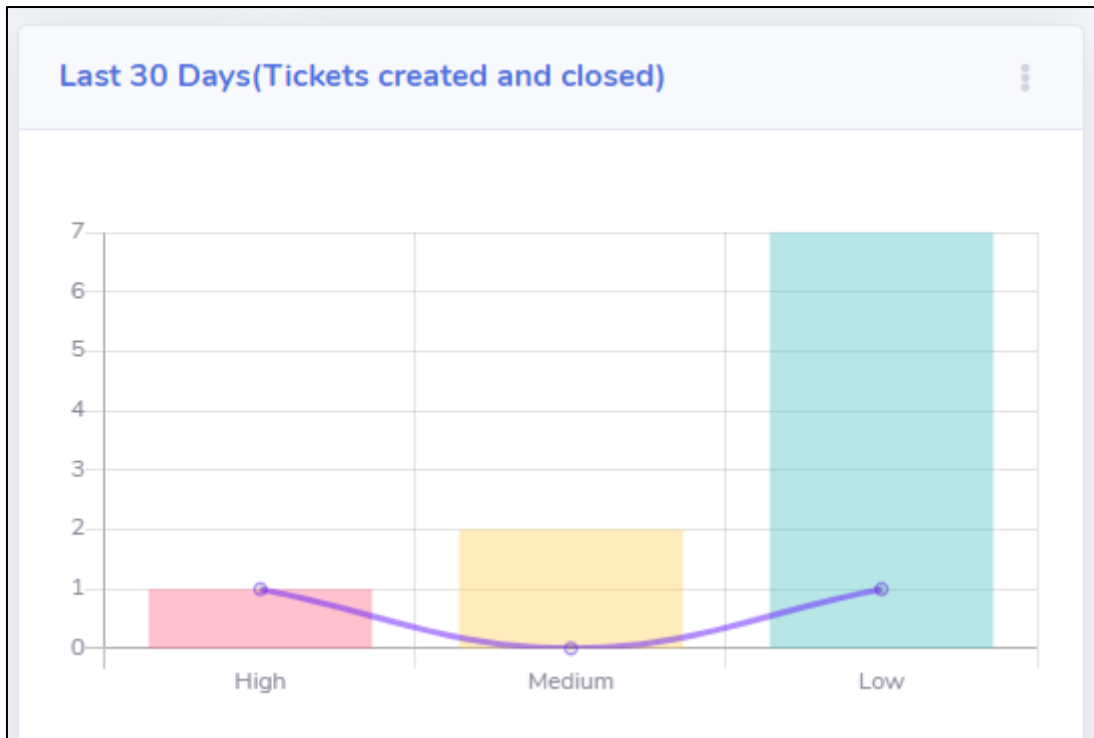


Figure 3.29 created chart using received parameters

3.9. Security Measures

There are 3 main security measures considered when developing the system.

3.9.1. Querying Database tables

Using plain SQL queries make a security concern of SQL injection attacks. This is one of the main reason for SQLAlchemy ORM selection. SQLAlchemy query engine automatically put quotation marks to all the special characters which make almost impossible to do a SQL injection attack. In this project, all the queries use SQLAlchemy ORM.

```

user_team = Userreportto.query.filter(
    Userreportto.report_to_user_id == user_id).join(
    User, User.id == Userreportto.user_id).join(
    Usergrade, User.usergrade_id == Usergrade.id,
    isouter=True).add_columns(
    func.concat(User.first_name, ' ',
    User.last_name, ' (', User.email, ')')
    .label("member_name"), Usergrade.user_grade).order_by(
    Usergrade.id)

```

Figure 3.30 simple SQLAlchemy query with table joins

3.9.2. Menu Authority Structure

There are two types of views on the system menu. Admin view and other users' view. Admin view is not visible for other users. But this authority level is not enough as users can type URL manually and gain access. To prevent that from happening, each admin menu item checks for user profile whether the user has admin access or not. If some non-admin user tries to access admin menu by typing URL he will see a page not found error.

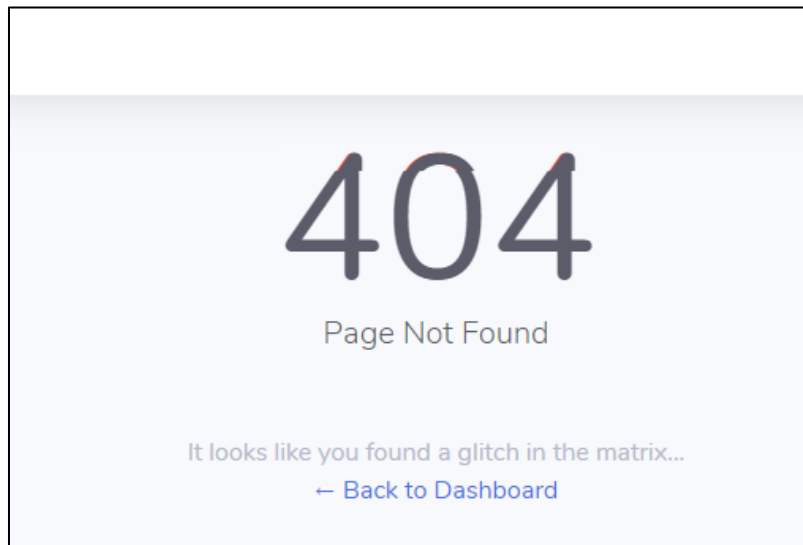


Figure 3.31 if non admin user tries to access admin URL, he will see this

3.9.3. Password Hashing

All passwords available in the database are hashed. To do this python bcrypt hashing function has been used.

```
password = bcrypt.generate_password_hash(request.form['password'])
```

Figure 3.32 python bcrypt for password hashing

id	username	password
1	Admin	\$2b\$12\$dgGj1.Hh1sHKwalplPG0LuEbjgHqfAM5M2NXkGUpOpDna4baRf97G
2	tharidup	\$2b\$12\$X2eLI2FyNdkTNQ2IyAbIXeN6wY7NzvDYwgw4r/ZSna3EzB3TBnP5u
3	nimalp	\$2b\$12\$UHYNyNkMgVUbD8eiJuzzYOq0as2UGPV5yjt96a.dolO3dGcPWfpzS
4	kelumu	\$2b\$12\$euRDQogLhvEglqL9IDzT3eLDBfC/EO3VwBofB9Av3fPXwymqIicC
5	sadunu	\$2b\$12\$Y6R2Ewkghaq6iSNmPhAE5O7yONrG0qMoHoQPkm6sMKAekyLXmfCR6
6	kusalf	\$2b\$12\$kJu8YIXbJA1HMvYyLgxyY9.ZPK6JPZKxyf.GmQS8hCpicqUl2pyCy
7	hemanthak	\$2b\$12\$9w4PaGrPkqDFmEz4VUuX7ukTC2Jx2J4KuHLBY2TRfkoLIHavKOqAq
8	upuin	\$2b\$12\$WWPrxOfCHmr95P6dOX48levh7p9CpnXpP3FC7p4CCvs4EGMb.0bla
9	kusalas	\$2b\$12\$vkTRW6/7sGKYzX15Q.pGfuMkcRoKm9UjRur4T51pAV2MRWqQoXeOK
10	tharidug	\$2b\$12\$8WJmLZ8jXyYjhpPyBWaa7OOGY857NktUK3jECnqRgTdwHvt67/.hK

Figure 3.33 hashed passwords in the database

3.10. Test Plan

Testing is one of the most important aspects of any software system. Testing ensures all initially planned milestones achieved with the expected outputs. This system was tested using different types of test cases. The following table shows the first test case tested when developing the system. Full list of test cases given in the Appendix.

Test Case #	Test Case Description	Expected Result	Actual Result	Pass/Fail
1	Test login with invalid username and valid password	Incorrect username/password message	Incorrect username/password	Pass
2	Test login with invalid password and valid username	Incorrect username/password message	Incorrect username/password	Pass
3	Test login with blank password	Enter password message	Enter password message	Pass
4	Test login with both blank username	Enter Username message	Enter Username message	Pass
5	Test login with both blank username and password	Both fields show enter username and enter password message	Both fields show enter username and enter password message	Pass
6	Test login with both valid username and password	Successful login	Successful login	Pass

Table 3.2 User login test case

Following figure 3.34 shows the code and folder structure of the project. Static folder contains subfolders for all the CSS, Image files, Javascript files, user attachments and chatbot corpuses. Template folder contains all the HTML templates being used in the system.

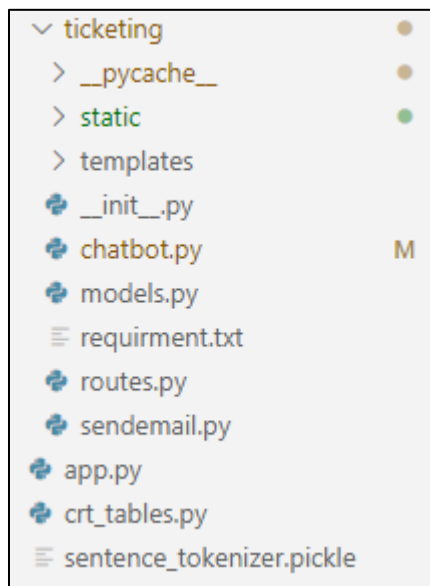


Figure 3.34 System code and folder structure

3.11. Implementation environment

Tool, Software and hardware used	Description
Windows 10	All the development was done using a Microsoft windows 10 OS
RAM	16GB
Hard drive	256GB SSD
Processor	Intel Core i7-8565U
Anaconda	Conda virtual environment is used as the python interpreter for this project. Anaconda is free and open source distribution.
MYSQL	MySQL 8.0.19 community server is used and the database for this project.
MySQL workbench	MySQL workbench a visual tool for MySQL database. This software used to execute ad-hoc queries when developing the system.
Visual Studio Code	VS code used as the main IDE and debugging tool for this project.
Python version	Python 3.7.6
Flask version	1.1.1
Chatterbot version	ChatterBot 1.0.5
Bootstrap Version	Bootstrap 4
draw.io	draw.io is a free online diagram creation software. This software used to draw all the diagrams of this project

Table 3.3 All Software, Hardware and tools used

4. Evaluation

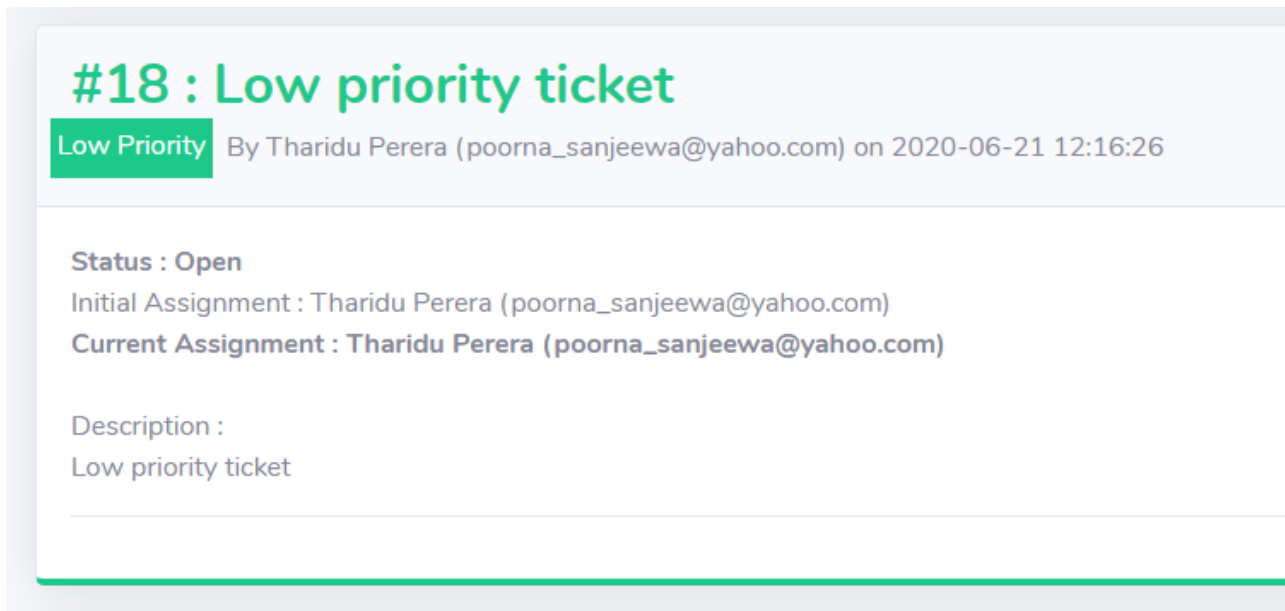
The best way to evaluate the project is to check whether it meets the objectives. Let's Take the initial Objectives one by one and check.

Initial Objectives	Comment
Solve the recurring issues using an automated way (A chatbot) which reduce the time and cost taken for issue solving. Which makes more time to focus on important or urgent issues. Users can chat with the chatbot first and try to find a solution before raise a ticket.	The chatbot is in place and ready. More conversations bot gets trained during the system usage, more accurate it gets.
Reduce the number of calls received to help desk agents by letting end-users to raise a ticket using the online system.	A Very efficient system is available now. Based on the priority issues get solved. Priorates are visually present in the dashboard.
Train the chatbot based on user feedback to increase the accuracy of the chatbot.	This will happen during usage. All customer interactions are gets save in the database.
Make it easy for relevant parties to attend urgent issues first based on prioritized tickets.	Tickets are ordered based on the priority and show in the dashboard with color codes. So that system users never missed the important ones.
Self-evaluate the tickets being assigned or work has been completed.	All active tickets are visible in the dashboard.
Manage the issue solving workflow with a more transparent way which all the involved parties can check the progress.	All involved and relevant parties get an email notification for each integration of the tickets. Also, the dashboard is available to view the status of the current months' ticket.
Improve the end-user satisfaction about overall technical issues solving process.	Uses get email notifications, Tickets are prioritized and view by color codes. So, the chance of a ticket solving getting delay is minimum. Which will improve user satisfaction.
Ease management decision making regarding IT helpdesk tasks.	Managers can view their teams' tickets and all the information separately.
Enhance the quality of service and meet the service level agreements of IT help desk.	All the tickets which do not meet the SLAs are visible over the dashboard. Necessary actions can be taken accordingly.
Improve the efficiency of overall IT help desk functions.	There are additional benefits also available with this system. Those described below.

Table 4.1 Objective evaluation

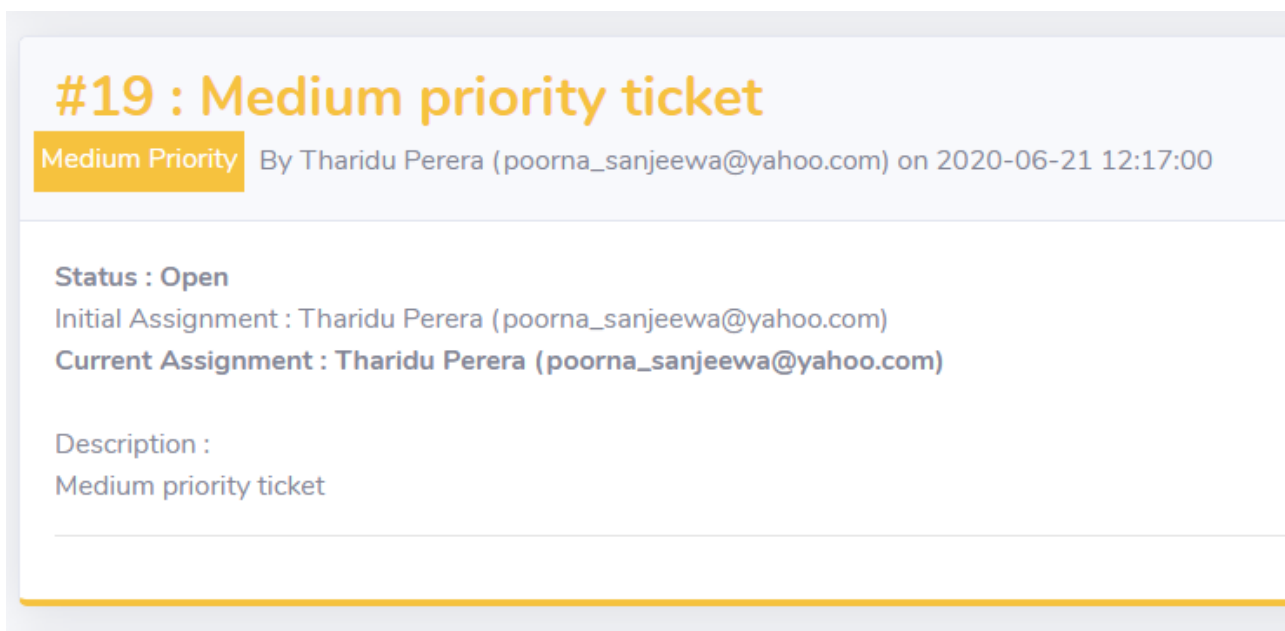
4.1. Additional capabilities and features

- The system initially planned to solve IT helpdesk issues. But this system can also be used to do important issue solving between departments. The ticket assignment works as a workflow and it's possible to assign tickets to any employee who registered in the company. So that additional capability kept as it is so it will be beneficial for the business operations.
- Three types of priority levels are display using 3 colour codes (Low – green, Medium – yellow, High - red) throughout the system. These colour codes are available in the dashboard, ticket viewing screens as well as ticket commenting areas. Users never miss out most important, prioritized tickets.



The screenshot shows a ticket interface for a low priority ticket. At the top, the ticket ID "#18 : Low priority ticket" is displayed in green. Below it, a green box contains the text "Low Priority" followed by "By Tharidu Perera (poorna_sanjeewa@yahoo.com) on 2020-06-21 12:16:26". The status is "Open". The initial and current assignments are both "Tharidu Perera (poorna_sanjeewa@yahoo.com)". The description is "Low priority ticket". A green horizontal line is at the bottom of the ticket card.

Figure 4.1 Low priority ticket



The screenshot shows a ticket interface for a medium priority ticket. At the top, the ticket ID "#19 : Medium priority ticket" is displayed in orange. Below it, an orange box contains the text "Medium Priority" followed by "By Tharidu Perera (poorna_sanjeewa@yahoo.com) on 2020-06-21 12:17:00". The status is "Open". The initial and current assignments are both "Tharidu Perera (poorna_sanjeewa@yahoo.com)". The description is "Medium priority ticket". An orange horizontal line is at the bottom of the ticket card.

Figure 4.2 Medium priority ticket

#20 : High priority ticket

High Priority By Tharidu Perera (poorna_sanjeewa@yahoo.com) on 2020-06-21 12:17:27

Status : Open

Initial Assignment : Tharidu Perera (poorna_sanjeewa@yahoo.com)

Current Assignment : Tharidu Perera (poorna_sanjeewa@yahoo.com)

Description :

High priority ticket

Figure 4.3 High priority ticket

4.2. Lessons learnt

Following are the list of lessons learnt during the project lifecycle.

- Able to acquire practical understanding about the software development process and its' stages.
- Able to learn new technologies and tools.
- Gain knowledge about how to use different programming languages and use them, passing data between them more efficiently
- Improved project documentation skills.
- Learnt more about software testing process.
- Learnt how to apply software engineering practises and how to apply them on the real project.
- Improved skills in software debugging and issues fixing.
- How to work with tight deadlines.

4.3. Problems Encountered

There is a set of problems encountered during the development of the project. But hard-working and well planning gets through most of the issues. Following are some of those problems.

- At the initial stages of the project, the development was done using a Linux environment (Laptop with ubuntu OS installed). But suddenly the laptop motherboard failure occurred. The entire project had to move to a new laptop which installed windows. Project coding didn't lose because of git integration. But all the environment setup and debugging configurations had to be done on the windows PC which takes lots of time.
- Chatbot training took more time than expected. So, accuracy is a bit lower than expected. But this can be sorted during the actual usage of the chatbot.
- Initially, SQLite database used as the main database. But it has limited capabilities and had to change it to much larger RDBMS. At that time all the queries were written in plain SQL and had to rewrite. But rather than rewrite SQLs, SQLAlchemy was introduced and not changing a database as just the matter of changing the connection string. But to do all these changes it took some considerable effort and time.

4.4. Overall comment

When considering all the aspects together and the test results, the system met all the specifications and tested properly. Even the look, as well as functionalities of the system, is sometimes beat the commercially available expensive Help desk systems also.

5. Conclusion

As a helpdesk ticketing system, this system has all the capabilities it required. But because of the way this system design was done, this system has the additional capability of use as a workflow management system. Because users can assign tasks to the users within the organization and see the progress of those tasks as well. This is a two in one kind of system for an organization.

If I had to mention any deficiencies of the system that would be the accuracy of the chatbot. But the chatbot accuracy is not something which can be obtained within a few months. It needs more inputs and more training. Even the training data might differ from one organization to the other.

There are few areas which can be enhanced as future improvements.

- A mobile app – A mobile app can be developed for this system so users can access the system much easier. Even though the system developed using very responsive UIs, a mobile app is much easier to use.
- Two-factor authentication – System security is becoming a major issue organizations face today. So rather than just using a password to login to the system a pin code via SMS or some authentication service can be used.
- Active Directory integration – Most of the organizations today use Microsoft products with their active directory service. By integrating the system with Microsoft active directory service, users can log in to the system using the same Microsoft credentials.

References

- [1] Jessica Greene, “35 IT Help Desk Statistics: How Does Your Team Stack Up?”, *askspoke.com*, 2018. [Online], Available: <https://www.askspoke.com/blog/it/it-help-desk-statistics/> [Accessed: Jan. 20, 2020].
- [2] Mathew Sweezey, “Key Chatbot Statistics to Know in 2019”, *salesforce.com*, Aug, 2019, [Online]. Available, <https://www.salesforce.com/blog/2019/08/chatbot-statistics.html> [Accessed: Jan. 20, 2020].
- [3] Altexsoft, “Non-functional Requirements: Examples, Types, How to Approach”, *altexsoft.com* [Online], Available: <https://www.altexsoft.com/blog/non-functional-requirements/> [Accessed: Feb. 23, 2020].
- [4] pypi, “SQLAlchemy - Introduction”, *pypi.org* [Online], Available: <https://pypi.org/project/SQLAlchemy/> [Accessed: Feb. 23, 2020].
- [5] UVdesk , “UVdesk open source features”, *uvdesk.com* [Online], Available: <https://www.uvdesk.com/en/opensource-features/> [Accessed: Jan. 20, 2020].
- [6] Zammad community, “Zammad’s documentation”, *zammad.org*. [Online], Available, <https://docs.zammad.org/en/latest/about/zammad.html> [Accessed: Jan. 21, 2020].
- [7] Aaron Kili, “An Open Source Help Desk and Support Ticket System”, *tecmint.com*, 2018, May. [Online]. Available, <https://www.tecmint.com/install-zammad-ticket-system-in-centos-ubuntu-debian/> [Accessed: Jan. 21, 2020].
- [8] OSTicket, “OSTicket Features”, *osticket.com*. [Online]. Available, <https://osticket.com/features/> [Accessed: Jan. 21, 2020].
- [9] Capterra, “Help Desk Software”, *capterra.com*, 2020 [Online]. Available, <https://www.capterra.com/help-desk-software/>, [Accessed: Jan. 22, 2020].
- [10] HelpDesk, “Discover HelpDesk features that will simplify your team’s efforts”, *helpdesk.com*, 2020 [Online]. Available, <https://www.helpdesk.com/features/>, [Accessed: Jan. 23, 2020].
- [11] Vision Helpdesk, “Multi Channel Help Desk Software”, *visionhelpdesk.com*, 2020 [Online]. Available, <https://www.visionhelpdesk.com/products/help-desk-software>, [Accessed: Jan. 23, 2020].
- [12] Extremeweb, “Ticket Support Systems”, *extremewebdesigners.com*, 2020 [Online]. Available, <https://www.extremewebdesigners.com/services/ticket-support-systems/>, [Accessed: Jan. 23, 2020].
- [13] Tryonics (PVT) Ltd, “Tryo Service Desk”, *tryonics.com*, 2019 [Online]. Available, <http://www.tryonics.com/project/tryo-service-desk/>, [Accessed: Oct. 31, 2019].

- [14] Steel Kiwi, "Top 13 Python Web Frameworks to Learn in 2020", *steelkiwi.com*, 2019 [Online]. Available, <https://steelkiwi.com/blog/top-10-python-web-frameworks-to-learn/>, [Accessed: May. 17, 2020].
- [15] ChatterBot, "About ChatterBot", *chatterbot.readthedocs.io*, 2019 [Online]. Available, <https://chatterbot.readthedocs.io/en/stable/>, [Accessed: May. 17, 2019].
- [16] W3schools, "AJAX Introduction", *w3schools.com*, 2019 [Online]. Available, https://www.w3schools.com/xml/ajax_intro.asp, [Accessed: May. 17, 2019].
- [17] DataTable, "DataTable", *datatables.net*, 2019 [Online]. Available, <https://datatables.net/>, [Accessed: May. 17, 2019].
- [18] Charts.js, "Chart.js", *chartjs.org*, 2019 [Online]. Available, <https://www.chartjs.org/>, [Accessed: May. 17, 2019].
- [19] GeeksforGeeks, "Software Engineering | Iterative Waterfall Model", *geeksforgeeks.org*, 2019 [Online]. Available, <https://www.geeksforgeeks.org/software-engineering-iterative-waterfall-model/>, [Accessed: May. 17, 2019].
- [20] Trello, "What is Trello", *computerworld.com*, 2018 [Online]. Available, <https://www.computerworld.com/article/3226447/what-is-trello-a-guide-to-atlassians-collaboration-and-work-management-tool.html>, [Accessed: May. 17, 2020].
- [21] I. Sommerville, "System modeling," in *Software Engineering*, 10th ed, Marcia Horton, Ed. England: Pearson Education Limited, 2016, pp. 149
- [22] Bruno Krebs, "SQLAlchemy ORM Tutorial for Python Developers", *auth0.com*, 2017 [Online]. Available, <https://auth0.com/blog/sqlalchemy-orm-tutorial-for-python-developers/>, [Accessed: May. 20, 2020].
- [23] Bruno Krebs, "SQLAlchemy ORM Tutorial for Python Developers", *startbootstrap.com*, 2019 [Online]. Available, <https://startbootstrap.com/themes/sb-admin-2/>, [Accessed: May. 20, 2020].
- [24] Jinja, "Jinja", *jinja.palletsprojects.com*, 2019 [Online]. Available, <https://jinja.palletsprojects.com/en/2.11.x/>, [Accessed: May. 20, 2020].

Appendix A - MYSQL tables created by python models

- Department model and table

```
class Userdepartment(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    user_department = db.Column(db.String(100), unique=True)
    created_timestamp = db.Column(db.DateTime, default=func.now())
    updated_timestamp = db.Column(
        db.DateTime, default=func.now(), onupdate=func.now())
    inserted_by = db.Column(db.Integer)
    updated_by = db.Column(db.Integer)
    user = db.relationship('User', backref='userdepartment', lazy=True)

    def __init__(self, user_department, inserted_by, updated_by):
        self.user_department = user_department
        self.inserted_by = inserted_by
        self.updated_by = updated_by
```

Figure A.1 Department Model

Column Name	Description	Data Type and length	Key
id	Department id	int	Primary
user_department	Department name	varchar(100)	Unique
created_timestamp	Created date time	datetime	
updated_timestamp	Updated date time	datetime	
inserted_by	Record inserted user ID	int	
updated_by	Record updated user ID	int	

Table A.1 MYSQL Department table

- Department model and table

```

class Ticketmaster(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    title = db.Column(db.String(200))
    description = db.Column(db.Text)
    catagory = db.Column(db.Integer)
    status = db.Column(db.Integer, db.ForeignKey(
        'ticketstatus.id'), nullable=False)
    priority = db.Column(db.Integer, db.ForeignKey(
        'ticketpriority.id'), nullable=False)
    init_assign_to = db.Column(db.Integer, nullable=False)
    curr_assign_to = db.Column(db.Integer, db.ForeignKey(
        'user.id'), nullable=False)
    created_timestamp = db.Column(db.DateTime, default=func.now())
    updated_timestamp = db.Column(
        db.DateTime, default=func.now(), onupdate=func.now())
    inserted_by = db.Column(db.Integer)
    updated_by = db.Column(db.Integer)
    ticketdetail = db.relationship(
        'Ticketdetail', backref='ticketmaster', lazy=True)
    ticketdetail = db.relationship(
        'Ticketattachment', backref='ticketmaster', lazy=True)
    ticketdetail = db.relationship(
        'Ticketstatuschange', backref='ticketmaster', lazy=True)

    def __init__(self, title, description, catagory, status, priority,
        init_assign_to, curr_assign_to, inserted_by, updated_by):
        self.title = title
        self.description = description
        self.catagory = catagory
        self.status = status
        self.priority = priority
        self.init_assign_to = init_assign_to
        self.curr_assign_to = curr_assign_to
        self.inserted_by = inserted_by
        self.updated_by = updated_by

```

Figure A.2 Ticketmaster Model

Column Name	Description	Data Type and length	Key
id	Ticket Id	int	Primary
title	Title	varchar(200)	
description	Description	text	
category	Category	int	
status	Status	int	Foreign
priority	Priority	int	Foreign
init_assign_to	Initially assigned user	int	
curr_assign_to	Currently assign user	int	Foreign
created_timestamp	Created date time	datetime	
updated_timestamp	Updated date time	datetime	
inserted_by	Record inserted user ID	int	
updated_by	Record updated user ID	int	

Table A.2 MYSQL Ticketmaster table

Appendix B – Test Cases

- Ticket creation test case

Test Case #	Test Case Description	Expected Result	Actual Result	Pass/Fail
7	Submit with blank Title	Message saying fill the required fields	Message saying fill the required fields	Pass
8	Submit with blank description	Message saying fill the required fields	Message saying fill the required fields	Pass
9	Submit by filling title and description	Successfully submitted message	Successfully submitted message	Pass
10	Attach a file 1 bigger than 2MB	Error message saying file is too big	Error message saying file is too big	Pass
11	Attach a file 2 bigger than 2MB	Error message saying file is too big	Error message saying file is too big	Pass
12	Attach a file 3 bigger than 2MB	Error message saying file is too big	Error message saying file is too big	Pass
13	Attach a file 4 bigger than 2MB	Error message saying file is too big	Error message saying file is too big	Pass
14	Attach a file less than 2MB, Keep the title blank and submit	Message saying fill the required fields	Message saying fill the required fields	Pass
15	Attach a file less than 2MB, Keep the description blank and submit	Message saying fill the required fields	Message saying fill the required fields	Pass
16	Attach one files less than 2MB, fill both title and description then submit	Successfully submitted message	Successfully submitted message	Pass
17	Attach four files less than 2MB, fill both title and description then submit	Successfully submitted message	Successfully submitted message	Pass
18	Attach two files less than 2MB, to random slots, fill both title and description then submit	Successfully submitted message	Successfully submitted message	Pass

Table B.1 Ticket creation testcase

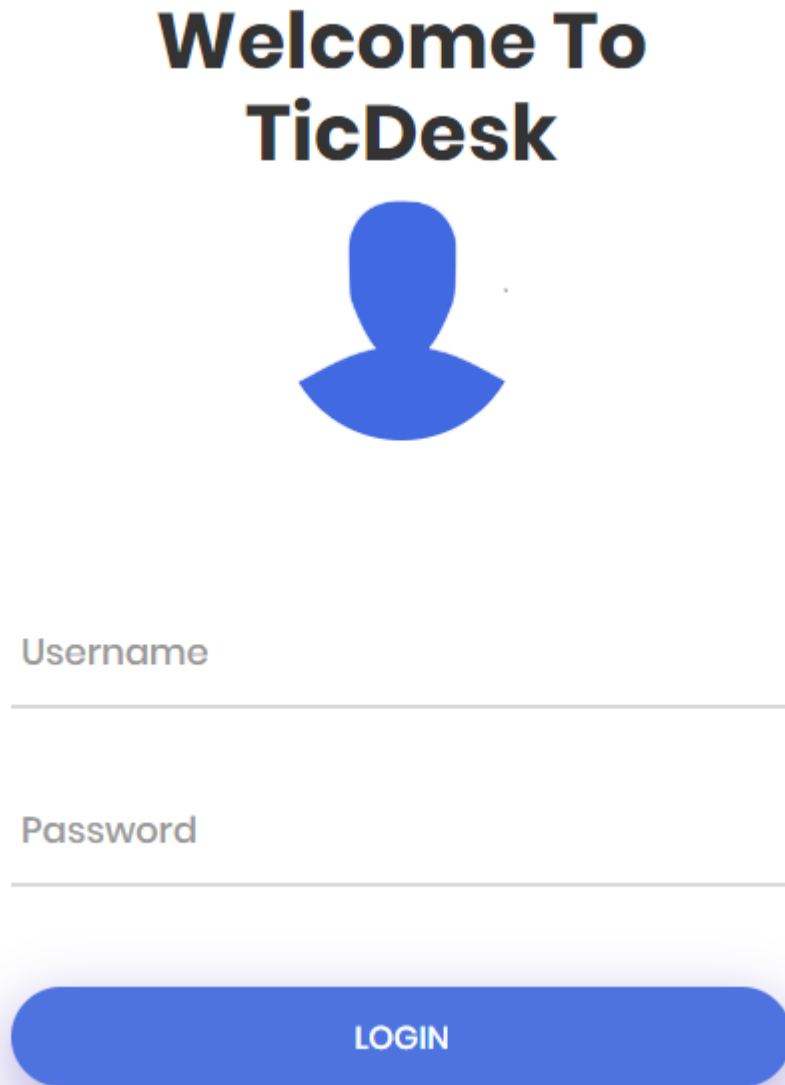
- Reply to an assign ticket test case

Test Case #	Test Case Description	Expected Result	Actual Result	Pass/Fail
19	Submit with blank description	Successfully submitted message	Successfully submitted message	Pass
20	Submit with filled description	Successfully submitted message	Successfully submitted message	Pass
21	Attach a file 1 bigger than 2MB	Error message saying file is too big	Error message saying file is too big	Pass
22	Attach a file 2 bigger than 2MB	Error message saying file is too big	Error message saying file is too big	Pass
23	Attach a file 3 bigger than 2MB	Error message saying file is too big	Error message saying file is too big	Pass
24	Attach a file 4 bigger than 2MB	Error message saying file is too big	Error message saying file is too big	Pass
25	Attach a file less than 2MB, Keep the description blank and submit	Message saying fill the required fields	Message saying fill the required fields	Pass
26	Attach one files less than 2MB,fill description then submit	Successfully submitted message	Successfully submitted message	Pass
27	Attach four files less than 2MB,fill description then submit	Successfully submitted message	Successfully submitted message	Pass
28	Attach two files less than 2MB, to random slots, fill description then submit	Successfully submitted message	Successfully submitted message	Pass
29	Change Status to 'Blocked' and submit	Successfully submitted message and status should display as blocked	Successfully submitted message and status should display as blocked	Pass
30	Change Status to 'Close - waiting approval' and submit	Successfully submitted message and status should display as closed - waiting for approval	Successfully submitted message and status should display as closed - waiting for approval	Pass
31	Change Assign to user and submit. Check currently assigned user is correct	Successfully submitted message and currently assigned user should be same as the one selected	Successfully submitted message and currently assigned user should be same as the one selected	Pass

Table B.2 Reply to an assign ticket testcase

Appendix C – User Manual

1) How to login to the system



The login screen features a large blue silhouette of a person's head and shoulders centered below the title. Below the silhouette are two input fields: the first is labeled 'Username' and the second is labeled 'Password'. At the bottom of the form is a prominent blue rounded rectangular button with the word 'LOGIN' in white capital letters.

Welcome To TicDesk

Username

Password

LOGIN

Figure C.1 Login Screen

- 1) Enter correct username and password.
 - Please note that username is case sensitive
- 2) Click login button or hit enter. If you provide the correct username and password, you will redirect to the dashboard.

2) Dashboard explanation

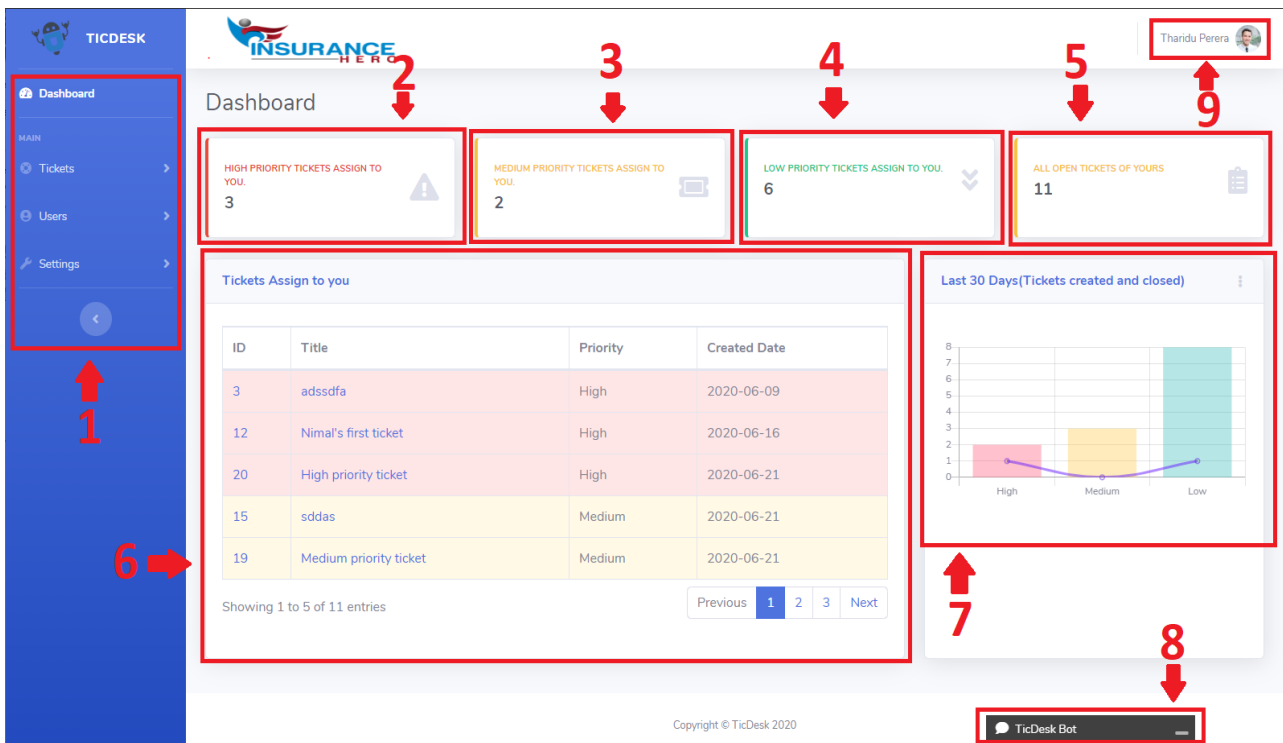


Figure C.2 Dashboard

1 – This is the menu items. You can collapse each menu category and select items.

2 – This card shows how many high priority items assign to you at the moment

3 – This card shows how many medium priority items assign to you at the moment

4 – This card shows how many low priority items assign to you at the moment

5 – This card shows how many tickets you created are still open

6 – This table includes all the tickets currently assign to you. This is order by the priority. You can click in the ticket ID or the name. It will take you to the ticket and you can work on that ticket.

7 – This graph shows the past 30 days activity of the user. Each bar represents each priority level and how many tickets user created for last 30 days. Line shows the how many tickets currently closed of those created tickets.

8 – This is the chat window for chat with the chatbot

9 – This shows the currently logged in your name and the profile picture. When user clicks on this profile picture drop appear with two options. One is to log out of the system and another for view user profile.

3) How to create a ticket

Figure C.3 Ticket creation page

- 1) Click the 'Create ticket' menu item. It will re-direct you to the ticket creation page.
- 2) Title and Description are mandatory.
- 3) You should select ticket priority based on your requirement.
- 4) Select the department you want to assign this ticket to. Based on your department selection, 'Assign to user' dropdown will change.
- 5) Select the you want to assign the ticket.
- 6) You can attach up to 4 attachments. Please note that each attachment should be less than 2MB of size.
- 7) Click submit button If ticket creation success, following message will pop up.

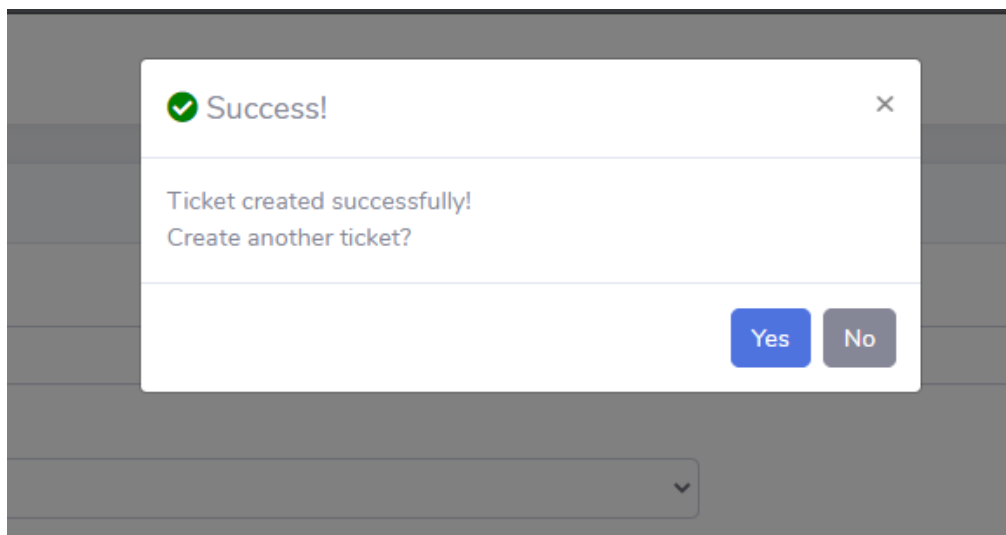


Figure C.4 Ticket creation success message

4) Chat with the chatbot

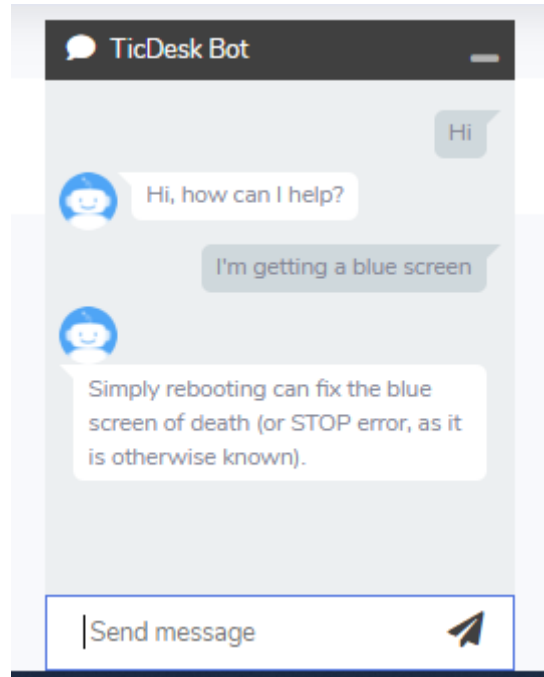


Figure C.5 Chat window

Chat window is available on dashboard page for users to chat with the chatbot.

Appendix D – Admin Manual

1) Additional menu

Apart from regular main menu items, Admin users have additional admin section. As shown on the below Figure.

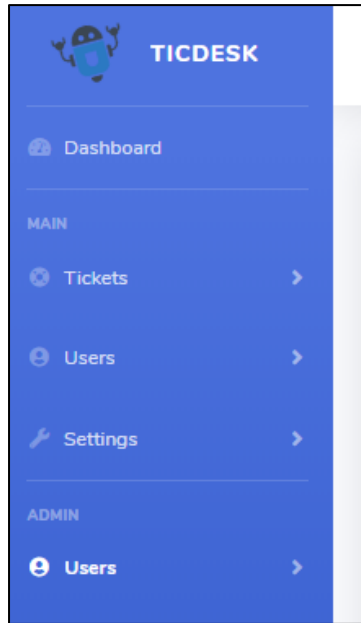
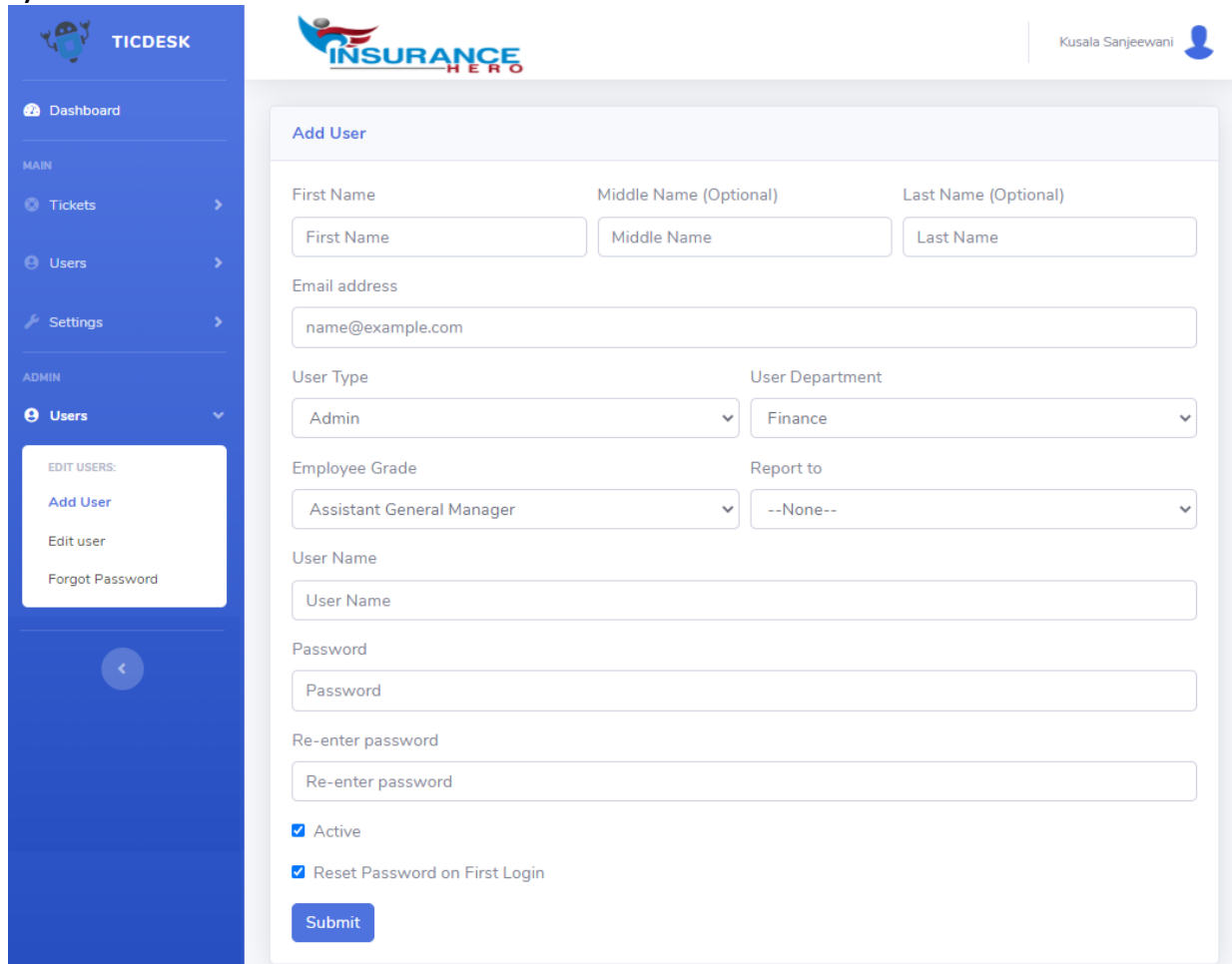


Figure D.1 Admin menu

2) Add a user



The 'Add User' page features a sidebar on the left with the TICDESK logo and a menu. The main content area is white and contains a form for adding a user. The form includes the following fields and options:

- First Name:
- Middle Name (Optional):
- Last Name (Optional):
- Email address:
- User Type:
- User Department:
- Employee Grade:
- Report to:
- User Name:
- Password:
- Re-enter password:
- Active
- Reset Password on First Login
-

Figure D.2 Add user page

- 1) Click add user option under Admin section of the menu. It will load the add menu page.
- 2) All the fields of this page are mandatory apart from the user's middle name and last name.
- 3) Then Type user's email address.
- 4) Then select User type (Admin, Regular or Help Desk)
- 5) Select user department.
- 6) Select User grader (Executive, Assistant Manager, etc.)
- 7) Then select who is the person user report to (User's immediate supervisor). This report to list will change based on the user department and the grade. Only higher-grade employees than the user will be displayed.
- 8) Type a username for the user. Always try keep the user's first name and last name first character as the username. If that username already taken, then try first two character of the last name and so on. Please note this username is unique key. If you try to add same username twice an error message will popup.
- 9) Then type the password.
- 10) Retype the same password for validation
- 11) Active tag is default ticked.
- 12) Reset password on first login is also default ticked. This will prompt a password reset window at first user login
- 13) Then click submit button. A success message will pop up on successful user creation. An email will be sent to user informing his user account is created. The email contains only the username. You must provide user password separately. Password is not sent with the email because of the security concerns.

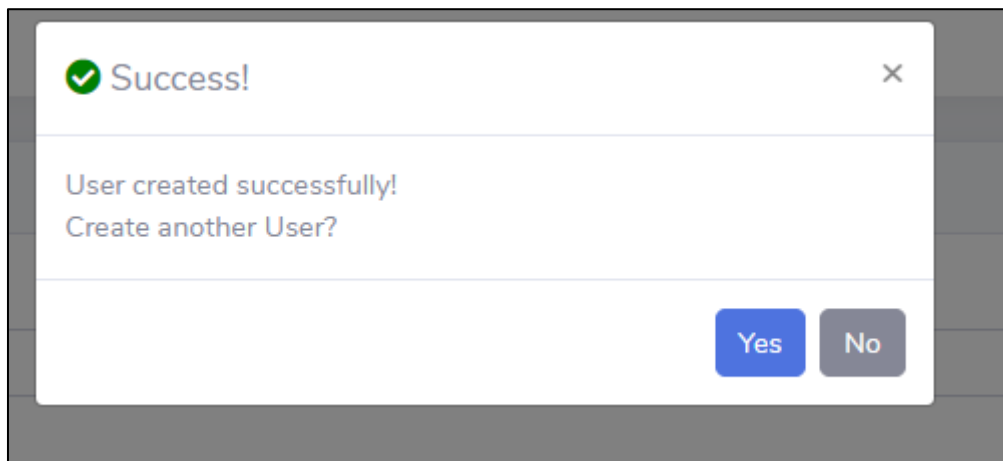


Figure D.3 User creation success message

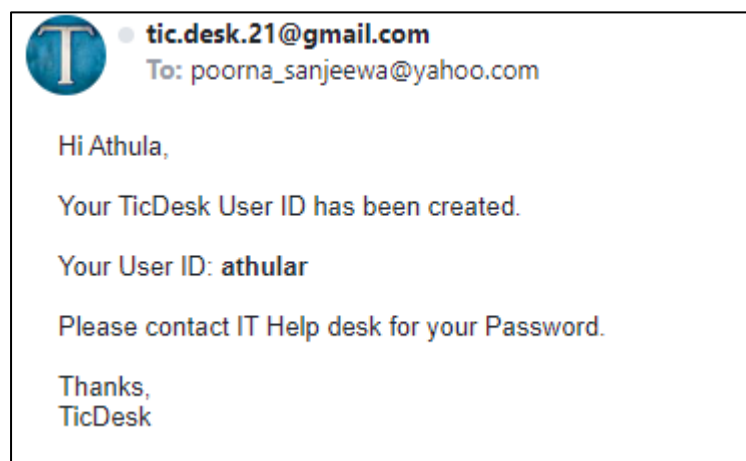


Figure D.4 The E-mail received by the user