

Game-based E-Learning Platform

**T. N. DODANGODA
2023**



Game-based E-Learning Platform

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

T. N. DODANGODA
University of Colombo School of Computing
2023



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: T. N. Dodangoda

Registration Number: 2018/MIT/016

Index Number: 18550166



Signature

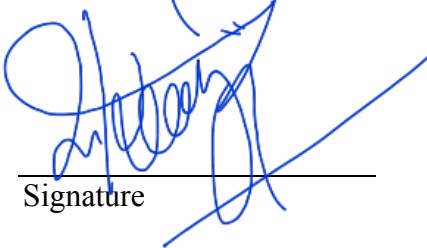
2023-02-12

Date

This is to certify that this thesis is based on the work of Mr. T. N. Dodangoda under my supervision. The thesis has been prepared according to the format stipulated and is of acceptable standard.

Certified by: Dr. M. I. E. Wickramasinghe

Supervisor Name: Dr. M. I. E. Wickramasinghe



Signature

2023-03-02

Date

Abstract

The game-based e-learning platform (codenamed, EDGE) is a system to build and manage game-based learning experiences. It is a web-based system employing the client-server architecture and uses technologies from the JavaScript stack. The system can be used by game creators, school academic staff, students and their parents. It provides a flexible template-based game development environment, ability to host and join game play sessions, user management and performance reporting capabilities.

Through the use of modern Agile methodologies and tools such as Feature Driven Development and Jira, the EDGE system was developed iteratively to address a critical problem prevailing in the current Sri Lankan education system: not incorporating modern teaching techniques in school education systems.

Existing systems in game-based and online learning were studied to build a solid understanding of requirements for this project. Several technical and architectural design approaches were evaluated by weighing their pros and cons. Afterwards, the candidate designs were implemented using modern technologies such as Angular, Node & Express, Socket.IO, HTML canvas and Docker containers. Finally, the system was extensively tested using a variety of methods such as exploratory testing, validation & verification and unit testing with the Jasmine and Karma frameworks.

The final solution is a system that has all the basic tools for building, managing and experiencing game-based learning right from a user's web browser. Whilst there were many hurdles to overcome due to the system's scale and complexity, the EDGE system proves that a platform of this nature can address present-day issues in delivering education content.

Acknowledgements

My heartiest gratitude to everyone who supported to achieve the completion of this Master's Thesis project. I would like to thank my supervisor, Dr. M. I. E. Wickramasinghe for his continuous guidance and feedback to shape the project into its current standing. I would also extend my gratitude to Dr. (Ms.) K. H. E. L. W. Hettiarachchi who supervised my initial attempt and provided invaluable feedback.

I would also like to thank the academic staff of University of Colombo School of Computing for their support, specially throughout the days of the 2020 / 2021 Coronavirus pandemic. I would also like to acknowledge the global community of developers who work passionately to create many of the open-source projects used by web systems such as EDGE.

Finally, my gratitude extends to my friends and family who supported me throughout hardships of the past two years to achieve the completion of this project.

Table of Contents

Declaration	3
Abstract	4
Acknowledgements	5
Table of Contents	6
List of Figures	9
List of Tables	12
List of Abbreviations	13
Chapter 1 - Introduction	14
1.1. Background and Motivation	14
1.2. Objectives	15
1.3. Dissertation Structure	16
Chapter 2 - Background	17
2.1. Similar Systems Review	17
2.1.1. Game-based Learning Software	18
2.1.2. E-Learning Software	29
2.1.3. Two-dimensional Game Development Systems	34
2.1.4. Related Technologies	38
2.1.5. Related Design Strategies	40
2.2. Use-case Diagram	42
2.2.1. Key Concepts	43
2.3. Requirement Specification	44
2.3.1. Functional Requirement Specification	44
2.3.2. Non-functional Requirements	48
Chapter 3 - Design	49
3.1. System Architecture	50
3.2. Designing the Game Editor	51
3.3. Dual Storage Engine Approach	52
3.4. Main Classes of the System	53
3.4.1. User Class Cluster	53
3.4.2. Game Sessions Class Cluster	54
3.5. Main Flows of the System	55
3.5.1. User Login and Registration	55
3.5.2. Game and Game Template Creation	56
3.6. Main Interfaces	58
3.6.1. Landing Page	58
3.6.2. Teacher Dashboard	59

3.6.3.	Gameplay Window	60
Chapter 4 -	Implementation	61
4.1.	Development Process	61
4.2.	Technology Stack.....	62
4.3.	Development Setup	63
4.4.	Relational Schema	65
4.5.	Interfaces and Code Structures	66
4.5.1.	Group Invite System	66
4.5.2.	Game Editor	68
4.5.3.	Performance Reports.....	74
Chapter 5 -	Testing and Evaluation	77
5.1.	Test Documentation	78
5.1.1.	‘Test’ Issue Types	78
5.1.2.	‘Bug’ Issue Types	79
5.1.3.	Jira Workflows.....	80
5.1.4.	Jira Workflow Scheme.....	81
5.2.	Testing Strategy	82
5.2.1.	Requirements Verification	82
5.2.2.	Development Testing	83
5.2.3.	User Testing	84
5.2.4.	Automated Testing Tools.....	85
5.2.5.	Manual Testing Methods	86
5.3.	Test Cases	88
5.3.1.	Exploratory Testing	88
5.3.2.	Backend Unit Testing	91
5.3.3.	Angular UI Testing	96
5.4.	Test Result Summary	98
5.5.	Requirements Verification	99
5.5.1.	Requirements Verification Summary	99
5.5.2.	Requirement Verification Breakdown	100
5.6.	User Testing	102
5.6.1.	Finalized User Roles	103
5.6.2.	Informal User Feedback.....	103
5.6.3.	Online Survey	106
5.6.4.	Survey Results	107
Chapter 6 -	Conclusion	112
6.1.	Retrospective.....	112
6.2.	Future Work	113
References	114

Appendix A – Prodigy System Teacher Reports	118
Appendix B – Oodlu Platform Student Analytics	119
Appendix C – System Requirement Specification	120
C.1. Game Editor Requirements	120
C.2. Game Play Requirements	121
C.3. Game Sessions Requirements	122
C.4. Groups Requirements	123
C.5. Performance Reports Requirements	124
C.6. Dashboard Requirements	125
C.7. User Requirements	126
Appendix D – Group Members Test Script	127
Appendix E – Online Survey Results	131

List of Figures

Figure 2.1 - Prodigy mathematics game battle screen	18
Figure 2.2 - Prodigy mathematics popup question screen	19
Figure 2.3 - Prodigy mathematics skills upgrade screen	19
Figure 2.4 - Oodlu platform teacher dashboard	21
Figure 2.5 - Oodlu platform game screen	21
Figure 2.6 - Oodlu platform question screen	21
Figure 2.7 - Jumpstart math blaster game play	23
Figure 2.8 - Jumpstart math blaster boss fight screen.....	23
Figure 2.9 - Educandy word search activity game.....	24
Figure 2.10 - Educandy dashboard my activities section	24
Figure 2.11 - Four types of blocks in Kahoot	26
Figure 2.12 - Kahoot classroom gameplay	26
Figure 2.13 - Kahoot game editor	27
Figure 2.14 - PhET projectile motion simulation startup screen	30
Figure 2.15 - PhET projectile motion simulation game play	30
Figure 2.16 - BuildBox main user interface.....	34
Figure 2.17 - BuildBox node editor	35
Figure 2.18 - CT.JS visual game editor interface	36
Figure 2.19 – Conceptual layered architecture of the backend components	41
Figure 2.20 - Conceptual layered architecture of the frontend components.....	41
Figure 2.21 - Top-level UML use case diagram	42
Figure 2.22 - UML component diagram for high-level features	44
Figure 3.1 - High-level system architecture diagram.....	50
Figure 3.2 - Main activities of the game editor component.....	51
Figure 3.3 - Prototype game project object.....	52
Figure 3.4 - User class cluster diagram.....	53
Figure 3.5 - Game session class cluster diagram	54
Figure 3.6 - User invite link join processes activity diagram	55
Figure 3.7 - Game and game template creation sequence diagram	57
Figure 3.8 - Game-based e-learning platform landing page design.....	58
Figure 3.9 - EDGE system teacher dashboard page design	59
Figure 3.10 - Game play window design	60

Figure 4.1 - Customized FDD development process	61
Figure 4.2 - Implementation technology stack	62
Figure 4.3 - Development setup folder structure	63
Figure 4.4 - Docker Desktop managing the MySQL and Mongo containers for the system ..	64
Figure 4.5 - Designer view of the EDGE system's relational database	65
Figure 4.6 - Copy invitation link option in the group overview page.....	66
Figure 4.7 - Code structure for creating the group invite link	67
Figure 4.8 - Join group from invite link page	67
Figure 4.9 - Game editor page overview section	68
Figure 4.10 - Game editor page resources section	69
Figure 4.11 - Multer library file upload configuration	70
Figure 4.12 - Game editor page levels section.....	70
Figure 4.13 - Game editor level scene editor page	71
Figure 4.14 - Game editor level properties editor page	72
Figure 4.15 - Game editor level script editor page	73
Figure 4.16 - JavaScript interop library code.....	73
Figure 4.17 - Available report types in the group page's reports section.....	74
Figure 4.18 - Database schema for tracking game objective progress data.....	75
Figure 4.19 - User objective report with graph and tabular presentation	75
Figure 4.20 - Backend code for processing user objective progress graph data	76
Figure 5.1 - Jira issue types used to document tests	78
Figure 5.2 - Jira workflow for Test issue types	80
Figure 5.3 - Jira workflow for Bug issue types.....	81
Figure 5.4 - Jira project workflow scheme	81
Figure 5.5 - Requirements verification process diagram	82
Figure 5.6 - Development testing process.....	83
Figure 5.7 - Jasmine test scripts in the Angular sub project	85
Figure 5.8 - Game editor page of the EDGE system	88
Figure 5.9 - Jira ticket documenting a game canvas exploratory test.....	89
Figure 5.10 - Jira bug ticket related to the game canvas camera bounding box	90
Figure 5.11 - Backend unit test helper code	91
Figure 5.12 - Overview of the group members DAO test.....	92
Figure 5.13 - Group members page showing user associations.....	93
Figure 5.14 - Jira ticket related to the group members unit test	94

Figure 5.15 - Group members DAO response highlighting incorrect fields.....	95
Figure 5.16 - Final state of the Jira ticket for the group members test	95
Figure 5.17 - Terminal output of the group members unit test after correction	95
Figure 5.18 - Code snippet for setting up the Angular UI test for the game resources page...96	
Figure 5.19 - Jira ticket for the game resources UI test.....	97
Figure 5.20 - Karma UI test results for the UI tests.....	97
Figure 5.21 - post evaluation survey form preview	107
Figure A.1 - Prodigy mathematics game teacher reports.....	118
Figure B.1 - Oodlu platform analytics screen	119
Figure E.1 - Online survey responses for system impression.....	131
Figure E.2 - Online survey responses for system usability	131
Figure E.3 - Online survey responses for system performance	132
Figure E.4 - Online survey response for future expansion	132
Figure E.5 - Online survey responses for feature set.....	133

List of Tables

Table 2.1 - JavaScript frameworks summary	39
Table 5.1 - Automated testing tools	86
Table 5.2 - Manual testing software	87
Table 5.3 - Test result summary table.....	98
Table 5.4 - Requirements verification summary table.....	99
Table 5.5 - Unfulfilled game editor requirements	100
Table 5.6 - Unfulfilled game play requirements	100
Table 5.7 - Unfulfilled game session requirements	100
Table 5.8 - Unfulfilled groups feature requirements	101
Table 5.9 - Unfulfilled performance report requirements.....	101
Table 5.10 - Unfulfilled dashboard feature requirements.....	101
Table 5.11 - User testing action plan	102
Table 5.12 - User testing volunteer user roles	103
Table 5.13 - Informal feedback collected during user testing	105
Table 5.14 - User testing online survey questions	106
Table 5.15 - Survey responses for system impression.....	108
Table 5.16 - Survey responses for system usability.....	109
Table 5.17 - Survey responses for system performance	109
Table 5.18 - Survey responses for system feature set.....	110
Table 5.19 - Survey responses for future expansion.....	111
Table C.1 - Project game editor requirements	120
Table C.2 - Project game play requirements.....	121
Table C.3 - Project game sessions requirements	122
Table C.4 - Project groups requirements	123
Table C.5 - Project performance reports requirements.....	124
Table C.6 - Project dashboard requirements.....	125
Table C.7 - Project user requirements.....	126

List of Abbreviations

MEAN	- Mongo, Express, Angular, & Node (technology stack)
FDD	- Feature Driven Development
STEM	- Science, Technology, Engineering & Mathematics (areas of study)
LMS	- Learning Management Software
PhET	- Physics Education Technology
UI	- User Interface
DOM	- Document Object Model
MVC	- Model-View-Controller architecture
MVVM	- Model-View-View model architecture
MVP	- Model-View-Presenter architecture
UML	- Unified Modeling Language
AES	- Advanced Encryption Standard
CBC	- Cipher Block Chaining
PII	- Personally Identifiable Information
JSON	- JavaScript Object Notation
DAO	- Data Access Object

Chapter 1 - Introduction

1.1. Background and Motivation

‘Game based learning’ describes the act of conducting pedagogical work with a game or game mechanic as the core educational tool (BERG, 2015). With the widespread adoption of web and mobile technologies in recent years many products have emerged catering game-based learning experiences to the global market (Kapuler, 2020). However, it is yet to be introduced into the Sri Lankan education system.

The Sri Lankan school curriculum focus on general science and mathematics during early secondary school grades (eduLanka, n.d.). Fundamental understanding of these subjects helps students pursue careers in STEM fields later in life.

Even with a clearcut requirement for game-based teaching/learning methods within the Sri Lankan school curriculum, students are still forced to follow traditional methods. The effective evaluation of students’ knowledge is further hindered due to political, economic and social factors. The absence of suitable tools as well as incompatibility, cost and complexity of existing tools contribute to the delayed adoption of new learning methods.

Education providers have not identified the severity of this problem, and as a result does not attempt to restructure their teaching methods to address the learning needs of young students.

Attempting to solve this problem I propose combining game-based learning and e-learning concepts to create a tailor-made solution, for students in early secondary school grades and their education providers; a ‘Game-based E-Learning Platform’ to create, play and manage game-based learning experiences.

The proposed platform is a web application that contains pre-made, customizable game templates. Education providers choose a game template, customize it and host game-sessions in their virtual classrooms managed inside the platform. Students join these sessions and engage in game-based learning experiences. Game progress is automatically tracked and used to generate detailed reports to all parties in the classroom, including parents.

1.2. Objectives

The main goal of this project is to develop a hosted web application that is able to create, play and manage game-based e-learning experiences. Additionally, the project should further achieve the following objectives.

- Develop (at least) five game templates.
- Develop a fully playable game-based learning experience using the developed system.
- Host the final version of the system on a public server.
- Host a gaming session with real users and a game developed using the system.
- Generate reports from a real gaming session.

1.3. Dissertation Structure

- **Abstract**

Concise abstract of the entire project.

- **Acknowledgements**

Acknowledging people and factors which contributed to the project's success.

- **Introduction**

Establishes the foundation for the project and its requirements. The overview of the project, its objectives and dissertation structure are discussed.

- **Background**

A detailed review about existing systems, design principles and alternative approaches are documented. Further, the system requirement is specified.

- **Design documents**

Important design decisions are discussed and interface prototypes of the system are attached in this chapter.

- **Implementation details**

Documents the implementation process, technologies used and code examples of the system are presented in this chapter.

- **Testing & evaluation**

The evaluation processes used are documented in this chapter along with test result reports.

- **Conclusion**

A retrospective on the planning, key decisions and final outcome of the project is reported in this final chapter.

Chapter 2 - Background

This chapter discusses in-depth the state-of-the-art in game-based learning and e-learning systems. It later provides an overview of the new system's main entities by using both formal diagrams and text descriptions. Finally, the high-level requirements of the system are defined.

2.1. Similar Systems Review

In order to understand the functionalities, shortcomings and notable aspects of the vast array of different existing systems, the similar systems review was carried out in three main categories:

1. Game-based learning software
2. E-learning software
3. 2-D game development software

2.1.1. Game-based Learning Software

2.1.1.1. Prodigy Mathematics Games

Prodigy (Prodigy Education Inc, n.d.) is a comprehensive suite of turn-based mathematics and English games. Initially, teachers must define a set of learning objectives and the subject topic. The system will then select challenges from a pool of template questions best suited for each learning outcome and present them as games to the user.

The turn-based game mechanic is shown in Figure 2.1 below. The left and right characters are controlled by the student and computer respectively. In each round, the student is presented with a question as shown in Figure 2.2. Incorrect answers are not accepted, and instead a hint is displayed. Upon entering the correct answer, the student gets to inflict damage to the opponent. Afterwards the computer-controlled character gets a turn, and then the round repeats until one player loses.



Figure 2.1 - Prodigy mathematics game battle screen

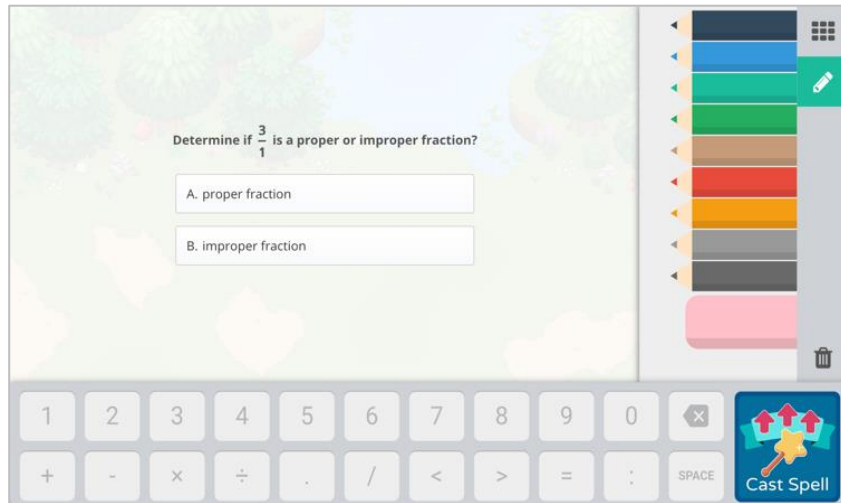


Figure 2.2 - Prodigy mathematics popup question screen

The main progression mechanic of the game is to spend ‘skill points’. Skill points are acquired after completing tasks or special move combinations. Spending points increase a player’s abilities. The interface related to this section is shown in Figure 2.3



Figure 2.3 - Prodigy mathematics skills upgrade screen

Notable features of the Prodigy system are listed below.

- Supports only mathematics curriculums from grades 1 through 8.
- Primary game mechanic is ‘magic spells’ acquired by the user by winning challenges.
- Teachers are provided with a dashboard to inspect student progress.
- Teachers have access to comprehensive reports (Appendix A).
- Students can be manually added via a ‘class code’.
- Teachers can configure student placement tests. This allows the Prodigy system to cater challenge difficulty according to the student’s skill level.

Limitations of the Prodigy system are as follows.

- Game worlds allow students to interact with each other in real time. However, the combat system only allows two players at once.
- Game content is not customizable.
- External party is responsible for the randomly picked pool of questions.
- Does not facilitate any form of communication within the platform.

2.1.1.2. Oodlu Educational Games

Oodlu is a platform which lets teachers create game-based learning sessions where questions are interspersed with mini games. Oodlu is recognized as having a proven approach to game-based learning (Oodlu Ltd., n.d.).

With the Oodlu platform, teachers can create games, manage groups of students and view student progress (as seen in Figure 2.4). Teachers are able to select several types of games, and an example type of game is shown in Figure 2.5. Educational questions are shown in an intermittent manner during gameplay. An example multiple choice question is shown in Figure 2.6.

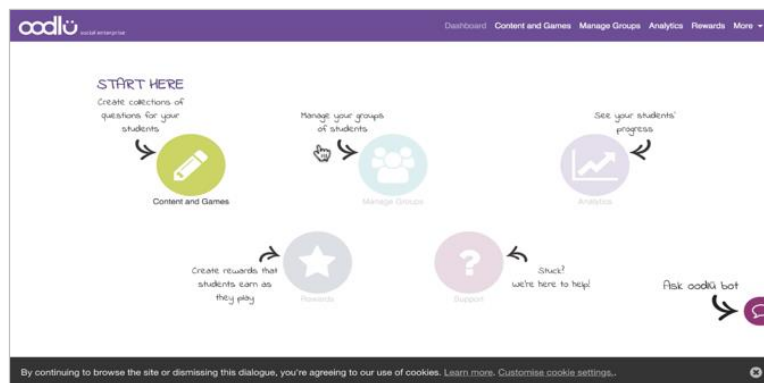


Figure 2.4 - Oodlu platform teacher dashboard

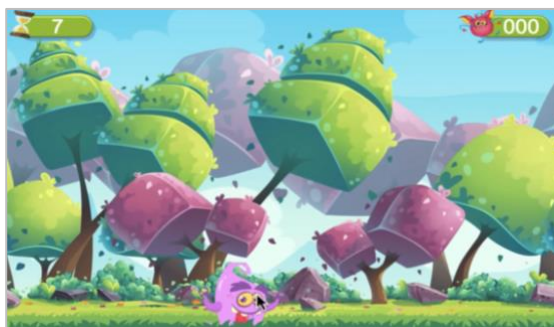


Figure 2.5 - Oodlu platform game screen

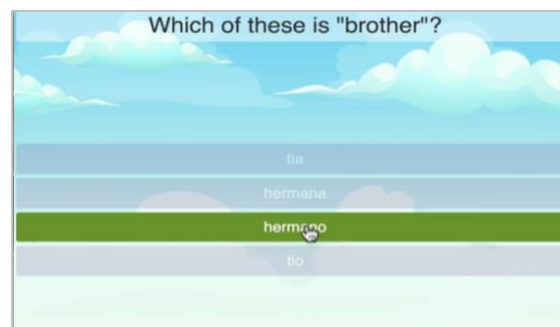


Figure 2.6 - Oodlu platform question screen

Notable aspects of the Oodlu platform are mentioned below.

- Students are given a small window of time to play a game as a reward for correctly answering a question. The game is pre-defined for a quiz session by the teacher.
- Number of hints displayed when incorrect answers are given can be configured.
- Provides in-depth analytics about student performance (refer Appendix B).

Shortcomings of the platform are mentioned below.

- The games are not customizable, only the questions displayed between game play sessions can be changed.
- Games are not related to any educational content. Instead, they are stimuli to engage the student into answering more questions. The two approaches seem contradictory.
- No communications channels built into the platform.

2.1.1.3. *Jumpstart Math Blaster HyperBlast 2*

Jumpstart Math Blaster (JumpStart, n.d.) is a modern 3D game designed to be played on mobile devices and targeted for users older than six years. The game incorporates mathematics challenges in between gameplay sessions similar to Oodlu. However, unlike Oodlu the challenges and gameplay occur in tandem, and not jarringly different screens.

Math blaster is a three-dimensional ‘space shooter’ game (Figure 2.7). At the end of each level a character appears, presenting questions for the user as shown in Figure 2.8. The user must correctly answer in a separate prompt within the given time limit, in order to progress.



Figure 2.7 - Jumpstart math blaster game play



Figure 2.8 - Jumpstart math blaster boss fight screen

The notable difference of this software in comparison to others, is the use of three-dimensional graphics and touch-based input. It is an example of blending the learning and game-play aspects harmoniously, rather than having the two different types of screens.

One of the main disadvantages of this system is that it's focused entirely around casual game play. While it still tries to provide educational content, the content cannot be controlled by an education provider. Further, the game play is slightly addictive. The sudden change of pace at the end of the level may be perceived as a disturbance by the user.

2.1.1.4. Educandy

Educandy (Educandy, n.d.) is a website allowing teachers to create interactive learning activities. Similar to Prodigy and Oodlu, users are presented with a set of game templates. The questions for each template must be filled in by the user – typically an education provider. Afterwards, the platform shuffles the questions for each game.

Each game template belongs to one of the three activity types; word search, picture match up or quizzes. An example of a word search game is shown in Figure 2.9. The dashboard provided for managing games is shown in Figure 2.10.



Figure 2.9 - Educandy word search activity game

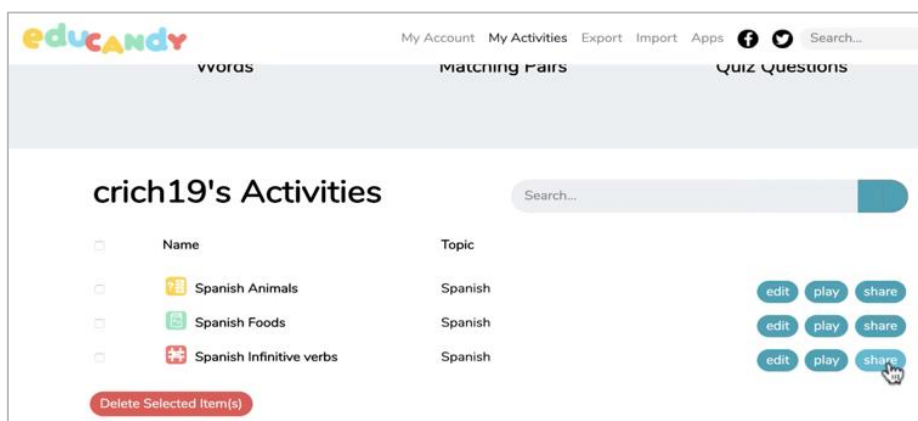


Figure 2.10 - Educandy dashboard my activities section

Notable features of the Educandy platform are mentioned below.

- Ability to play the games as a classroom activity or individual student activity.
- Export and import question data across games of the same activity type.
- Sharing games using 4-digit codes or links.
- Certain games can be played in two-player mode or against the computer.

In contrast to other systems, the Educandy platform has the following limitations.

- Does not facilitate performance reports.
- Number of game activities are limited and non-extensible by the teacher.
- Games are much more suited for testing associative knowledge rather than critical thinking. This is limited by the degree of interaction permitted.

2.1.1.5. Kahoot

Kahoot (Kahoot! Corporation, n.d.) is a quiz-based game platform that can be used in school or work environments, as well as in casual settings. Within the Kahoot platform, ‘a Kahoot’ is a term used to describe a single game. For example, a math Kahoot.

Similar to Oodlu and Educandy, users can input the questions for the type of game they wish to make. The system shuffles the questions, and presents them to the user. A notable distinction here is that users are never allowed to provide answers as raw input (such as via keyboard). Answering is done by selecting or re-ordering four types of blocks shown in Figure 2.11.



Figure 2.11 - Four types of blocks in Kahoot

Kahoot both encourages and supports multi-player interactions. For example, Kahoot allows its users to publish the created games to be used by others. If a teacher wants to use an existing game but need to customize the content to suit their students better, Kahoot allows the teacher to “clone” the existing games.

Game play in Kahoot differ from other systems. Students and teachers play Kahoots in classrooms (hosted sessions). As shown in Figure 2.12 the teacher’s screen (right) displays the questions and answers and the student’s personal device (left) shows only the answers.

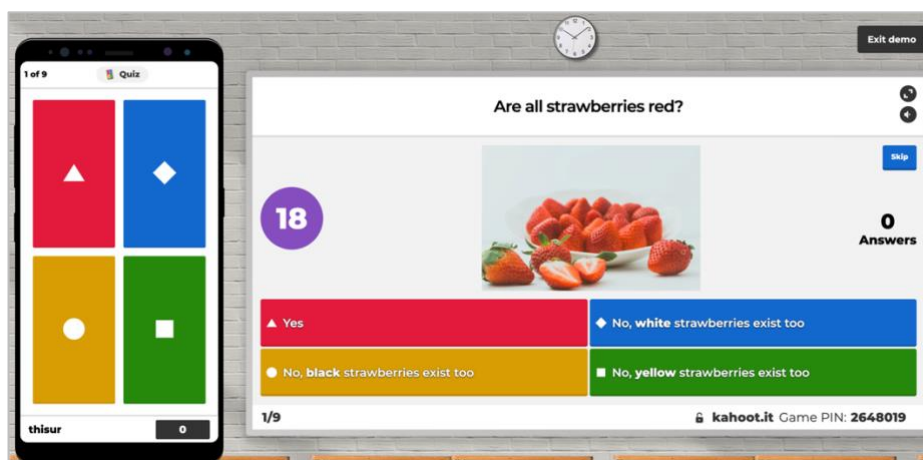


Figure 2.12 - Kahoot classroom gameplay

If a student wants to play a game in a self-paced manner, the system will display both the question and the answers on their device. Multiplayer aspects do not apply in this scenario.

Kahoot questions have a time limit. After the timer runs out, the teacher's screen shows the overall answers for the question. The student's screen shows the correct answer as well as their score. Kahoot also provides well integrated multiplayer gaming and seamless synchronization.

The editor (shown in Figure 2.13) allows users to create new games. It also lets users to duplicate existing Kahoots.

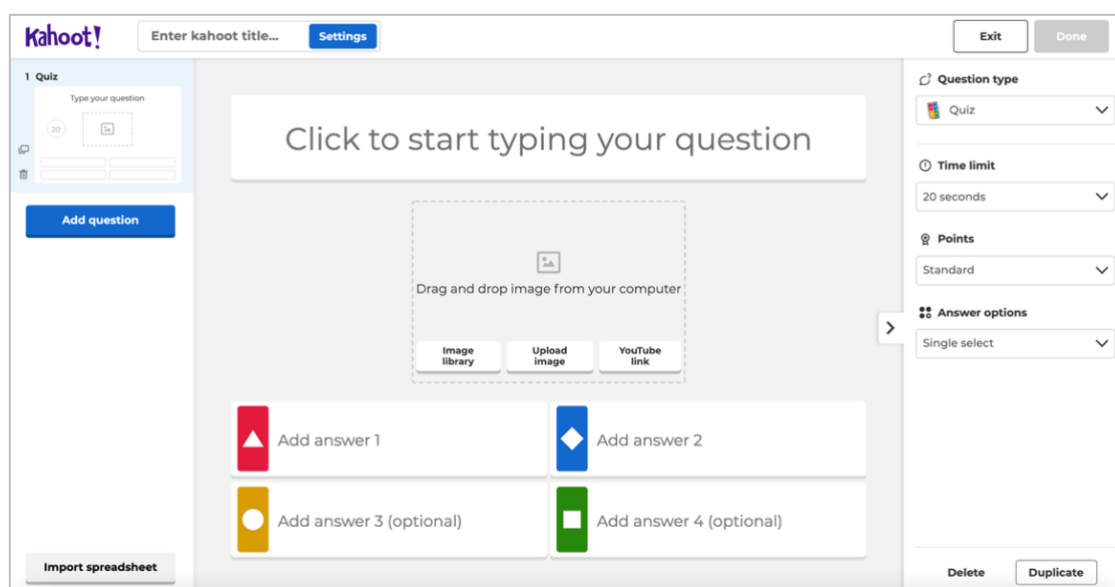


Figure 2.13 - Kahoot game editor

Since Kahoot games can be easily shared and discovered through the platform itself, education providers can re-use content. This provides much freedom for education providers to create and share game-based learning content without investing significant effort. However, there are a few limitations of the Kahoot system as mentioned below.

- Many of the advanced Kahoot features are only available in the paid version.
- The paid version does not have a wide range of game templates (limited to seven at the time of writing).
- Complex teacher training is required be used effectively in classroom environments, in contrast to platforms such as Oodlu and JumpStart games which have a clear user flow.
- The platform design assumes schools (as well as users) have access to the necessary infrastructure (networking, devices, uninterrupted power).

2.1.1.6. Conclusion

The study of existing game-based learning systems has provided great insight into the previous accomplishments in a global context. These systems can be categorized based on their mode of teaching as follows.

- Autonomous (self-directed) learning
- Supervised (teacher-directed) learning systems

Autonomous learning systems does not involve the role of a ‘supervisor’ in the learning process. Prompts and natural organization of the system’s interface guides the user. Further, all the educational content required for the learning process is bundled with the system.

In contrast, supervised learning software require a supervisor (in many cases, a teacher) to host the learning session. They would setup the necessary questions, rewards and constraints for each session. They might also be responsible for setting up the user accounts for students. Students interact with supervisors fulfil the learning objectives of the session.

The EDGE system can be categorized as a supervised learning system, since it bears similarity to existing systems such as Kahoot.

2.1.2. E-Learning Software

Learning software not involving gamification can be considered e-learning software. For example, Learning Management Systems (LMSs). However, personal tools and social software such as grade tracking, and classroom chat/video conferencing systems can also be considered as e-learning systems (Dalsgaard, n.d.).

Studying existing e-learning software helps to understand the common baseline features, what areas can be improved and how non-gamified e-learning systems achieve their goals.

2.1.2.1. PhET Interactive Simulations

Physics Education Technology (PhET) provides interactive simulations for physics experiments. A notable aspect is that the system was created by Nobel laureate Carl Wieman and supported by the University of Colorado. PhET simulations are proven to be practical and effective (Astutik & Prahani, 2018).

Even though the initial goal of the project website (University of Colorado, n.d.) was to create physics simulations, at present users can access content for chemistry, mathematics, earth science, biology and many more subjects. The simulations can be accessed through any modern web browser. They can also be downloaded or embedded as raw HTML code in a third party's website.

Figure 2.14 shows the startup screen for a projectile motion simulation. Figure 2.15 shows the simulation in action. There are no obvious text blocks or message prompts. Users can experiment and learn through observing the outcomes of the simulation parameters.

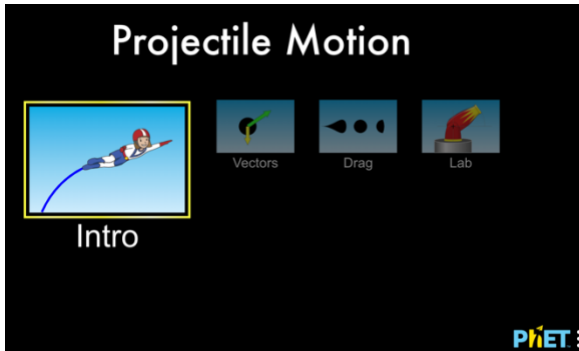


Figure 2.14 - PhET projectile motion simulation startup screen

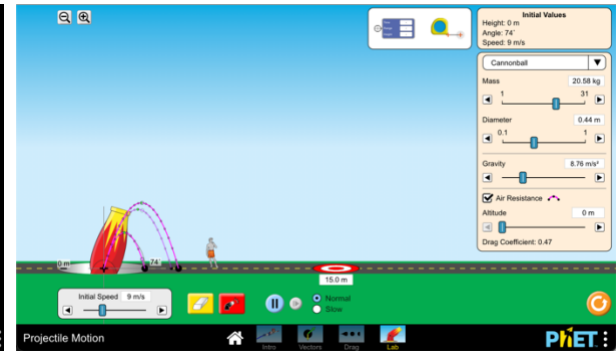


Figure 2.15 - PhET projectile motion simulation game play screen

Notable aspects of this system are as follows.

- Learning content is peer reviewed and published by a reputable institute.
- Content is not distributed as-is, and assumes user is capable of setting up the simulation.
- Content can be freely distributed by users and other education providers.
- Physics simulations are performed with real mathematical models.

Limitations of this system are as follows.

- Content cannot be customized. Even though the simulation is distributed as a simple web page, the logic is obfuscated in JavaScript code.
- Promotes self-directed learning or learning through demonstration. Does not contribute a guided interactive game-based learning approach.

2.1.2.2. Google Classrooms

Google Classrooms is a suite of tools designed to promote collaboration between students and teachers. It is a learning management system, and allows education providers to share course material as well as conduct assignments (Martínez-Monés, et al., 2017).

Google Classrooms is one of the few solutions that provide real-time communications out of the box. This is in part due to Google G-Suite services such as Google Hangouts and Google Meet being integrated into Google Classroom.

Notable features of this system are mentioned below.

- Streamlined user interface.
- Strong privacy protection settings.
- Presence of a class updates dashboard (named ‘Class Stream’) where students and teachers can see and share casual information.
- Students can obtain ‘originality reports’ on their coursework before handing over to teachers. Google Classrooms will automatically compare the content uploaded by the student to identify uncited content and unintentional plagiarism (Google, n.d.).

A few disadvantages of Google Classrooms (in terms of an LMS) are listed below.

- Parents are not considered a first party within Google Classrooms. They are not allowed to engage or see the interactions between their children and teachers. Parents are only allowed to receive announcements and notification about specific activities.
- The platform only supports manual grading of students’ pedagogical work. For example, student reports must be graded by teachers manually after student submission.
- Not suited for organization-wide usage (for example, throughout a school). Even though it is argued previously that this type of software is used primarily for administrative purposes, users who wish to invest in Google Classrooms may expect it to replace their existing LMS system.

2.1.2.3. Moodle Learning Management System

Moodle is a popular learning management system choice due to its open-source nature, easy implementation and extensive customizability. It is reported that Moodle was being used frequently even as early as 2013 (Cavus & Zabadi, 2014). Notable features implemented in the Moodle LMS are as follows.

- Real-time chat within a Moodle course.
- Internal organization hosted e-mail support exists in Moodle since it implements its own mail service without the need for setting up mail servers.
- Video conferencing support is built in. Moodle itself does not support video conferencing as a feature but supports third party integrations with services such as Zoom, Skype, BlindSideNetworks, etc.
- Extensive community support is present.

Along with a great feature set, implementing Moodle still presents a few drawbacks.

- Moodle's extensive feature set means that it is useful only when paired with a technical team that understands how to use the system to its full potential.
- The Moodle LMS is open-source software. If an organization chooses to host it themselves the organization has to bear its risks.
- Minor usability issues exist. As per a discussion on the Moodle user forum (Anon., n.d.), Moodle has minor usability issues since it is developed as an open-source project. This is in comparison with services such as Google Classrooms which are considered more 'polished'.

2.1.2.4. Conclusion

Software other than LMS solutions may be limited to simulation and demonstration software. This is due the integration of gamification in to an e-learning platform, categorizes it under game-based learning software instead.

Besides simulation software, LMSs do not provide varying functionalities or product offerings. Instead, many existing solutions provide overlapping features with varying degrees of usefulness. Concluding the study, it is now possible to answer the following questions.

- What features do e-learning software have in common?

The ability to manage and administer learning activities is common to most LMS software. For example, features such as managing students in an organization, providing online course material and assignments, creating performance reports, organizing video conferences, conducting discussion forums, etc. With regards to simulation type software, the ability to change parameters and re-test outcomes is present.

- What areas can be improved in e-learning software?

In LMS software, the approach to implementing the solution can be improved. Instead of providing user-hosted solutions, the solutions can be provided as a cloud-hosted solution. Maintaining access to low-level modifications in the instance can also be provided, even though it is managed by the cloud provider. With regards to simulation software, it would be beneficial for students to persist their experiments and outcomes instead of performing them in a one-off manner. For example, a platform where students are able to examine and compare their history of different experiments would be valuable.

2.1.3. Two-dimensional Game Development Systems

The EDGE system requires the implementation of game-development components. Under the similar game-based learning software systems study, it was concluded that currently existing solutions provide little to no support for comprehensive editing of gaming content. Therefore, this section expands on features of comprehensive two-dimensional game editors.

2.1.3.1. BuildBox

BuildBox is a prototype-oriented game development software package. It provides a set of tools to define game logic, create sound & graphic assets, add animations and edit levels. BuildBox runs as a standalone application. Games created with BuildBox can be published on mobile and desktop platforms. This is because the underlying technology uses a proprietary JavaScript game engine. The platform also allows creators to monetize their games. Figure 2.16 shows the main interface of the BuildBox application.

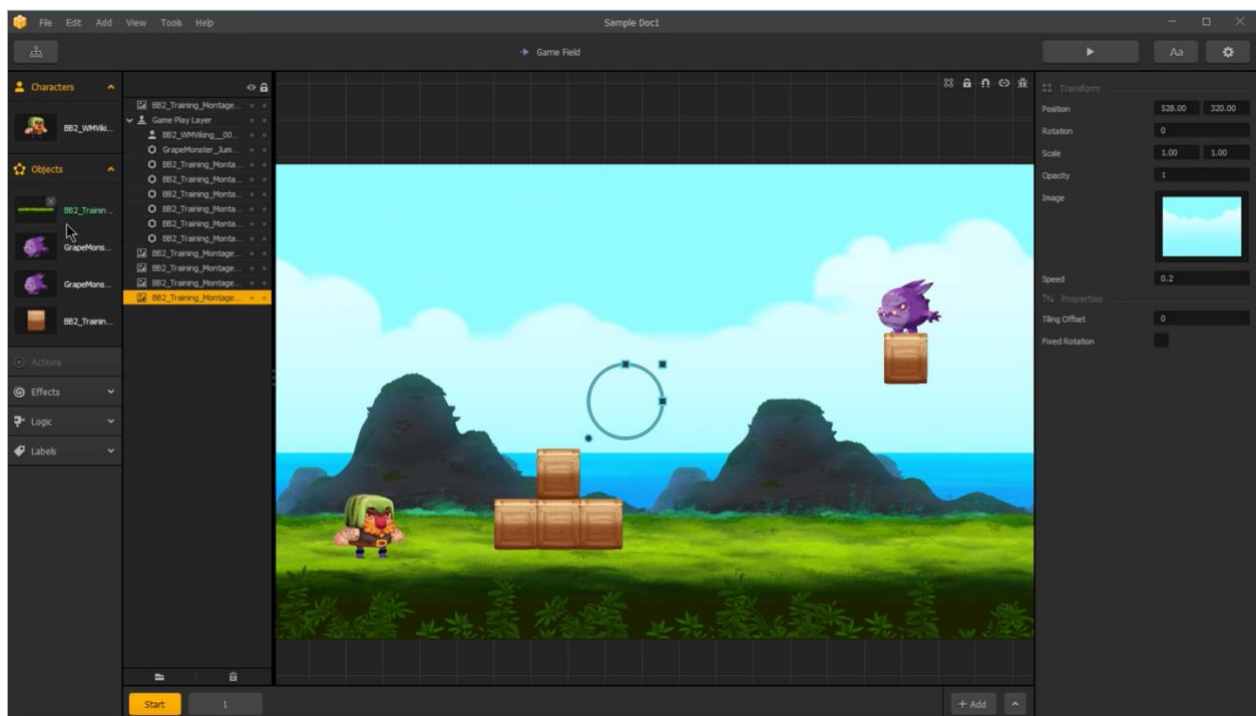


Figure 2.16 - BuildBox main user interface

Adding logic to the game world entities is done through a separate screen named, 'Asset Nodes'. Similar to other node systems (such as the one found in Blender (Blender Foundation, n.d.)), inputs and outputs are connected between different nodes. The nodes define aspects such as movement constraints, collisions, logic and animation triggers. A screenshot of this node editor is shown in Figure 2.17 below.

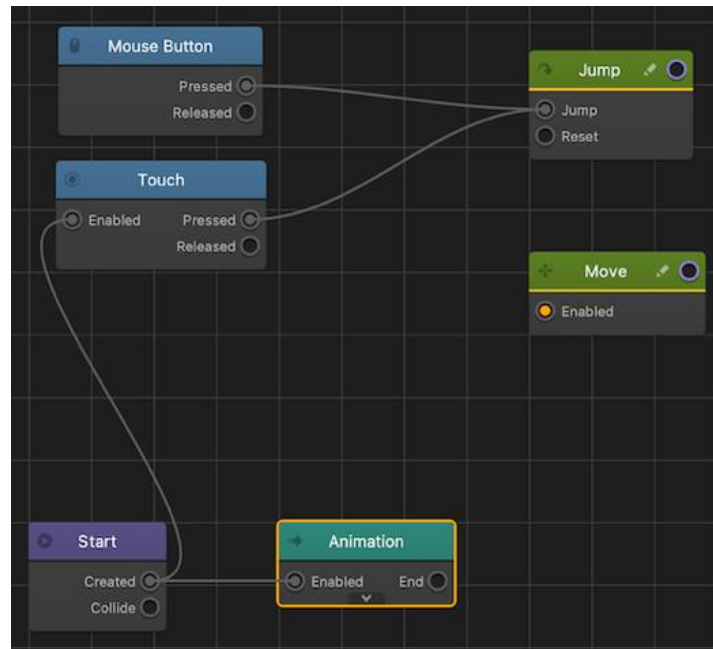


Figure 2.17 - BuildBox node editor

Node editors cause complex workflows according to my own personal experience. However, since BuildBox only shows the nodes that apply to a specific asset, sufficient simplicity can be maintained. Other notable aspects of BuildBox are mentioned below.

- The node-based editor covers almost all combinations of actions that can be performed by hand-written code.
- 2D and 3D editing capabilities are present.
- BuildBox bundles pre-made game assets so users can get started quickly.
- Monetization through ad platforms (BuildBox, n.d.) and in-app purchasing is made simple.
- Allows non-coders or small teams to iteratively create well designed games.

There are a few limitations of the BuildBox platform. These are listed below.

- Even though BuildBox is a complete package for non-coders, it is not a replacement for complex game development systems. It is targeted at an audience who are beginners wanting to create games, fast.
- The tool requires a high-performance host machine to be responsive.
- Stability issues when running the BuildBox application for long periods of time (Anon., n.d.) (Anon., n.d.).
- BuildBox lacks support for custom plugins and addons.

2.1.3.2. CT.JS

CT.JS is an open source 2D game editor built to be used in web browsers (Gorynych, n.d.). It is extendable, well documented and contains common features that are used in two-dimensional games. A screenshot of its interface is shown in Figure 2.18 below.

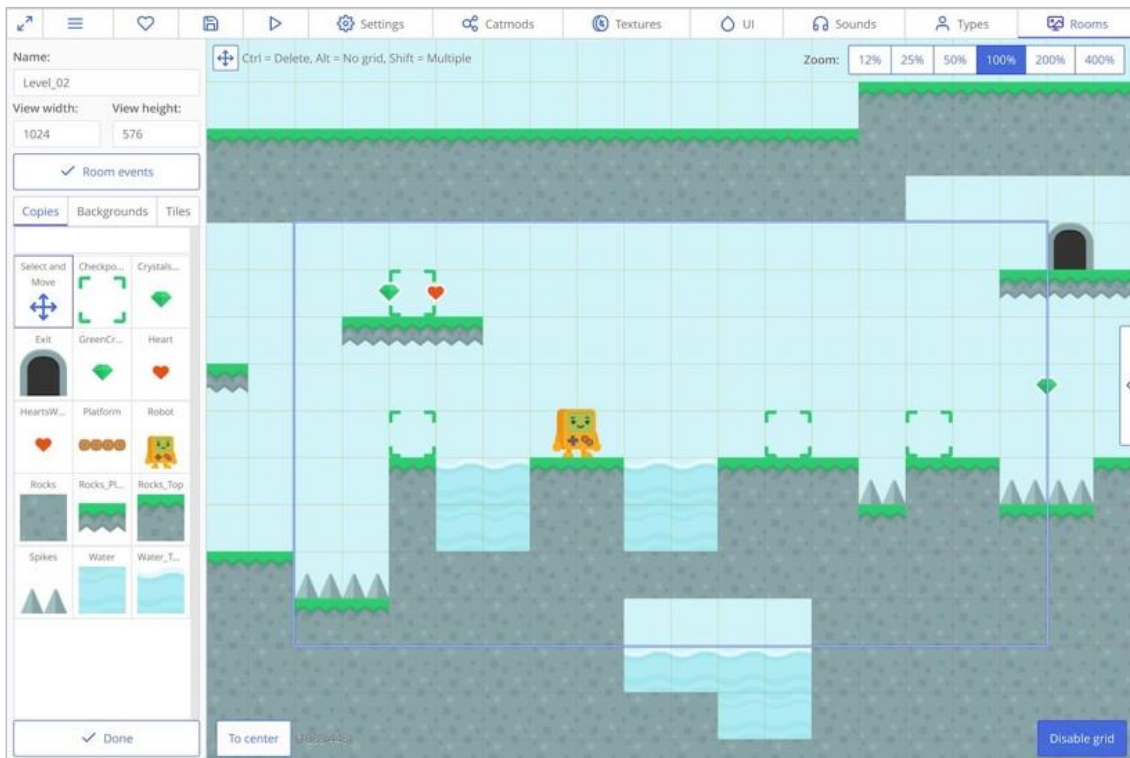


Figure 2.18 - CT.JS visual game editor interface

Unlike BuildBox, it is not a complete solution that aims to be used as a standalone software. Instead, it is an extensible component for creating games. It contains a visual editor, script editor and several other useful features. Whereas in no-code solutions all logic-based programming happens through nodes, CT.JS encourages the use of code to create the game. A few of the notable aspects of CT.JS are mentioned below.

- Highly customizable through third party plugins.
- Fully open-source project code.
- Sprite editor supports tiled sprites to create landscapes.
- Games built with CT.JS are standalone JavaScript applications.

The following are limitations of the CT.JS system.

- Has a learning curve as it uses its own game editor user interface conventions.
- Feature set is fairly limited (but extensible).
- Since it is an open-source project, the system does not guarantee any continued support or bug-free feature updates.
- It may be a risk for a user to invest in this type of open-source projects (Giera, 2014)

2.1.3.3. Conclusion

Through the study of the aforementioned existing systems, I was able to understand the complexity involved in designing a game development system. While it is possible to invent new methods to design games, it is important to understand the constructs already in place by leading projects in the industry of two-dimensional game development. I believe the appropriate contextual information was obtained through this study.

2.1.4. Related Technologies

2.1.4.1. Data Persistence

This project has two implicit requirements for types of data persistence:

- High-performance storage for real-time data processing (fast access rate)
- Reliable storage for on-demand data (at a slower access rate)

The two most popular database paradigms were considered for addressing these requirements. They are relational databases and NoSQL databases. While relational databases ensure proper organization of data at the cost of noticeable performance overhead (Davoudian, et al., 2018), NoSQL databases offer the opposite value proposition.

While most NoSQL databases store data in volatile memory, some products store data in non-volatile (disk) storage. For example, popular key-value storage technologies such as Redis and LevelDB store in memory while being accessed (Redis.IO, n.d.) (LevelDB, n.d.). In comparison, technologies such as, RocksDB and LMDB store data mostly in secondary storage. They flush in-memory data to persistent disk storage frequently (RocksDB, n.d.) (LMDB, n.d.).

2.1.4.2. JavaScript Frameworks

JavaScript is one of the most commonly used languages in the Asia and United States regions (StackOverflow, 2020). For this project, it plays a vital role as the entire system was developed using the TypeScript language, the successor to the existing JavaScript language. The following frameworks were reviewed with the goal of understanding potential application in this project.

- Phaser game development framework
- Node JavaScript runtime
- React.JS frontend framework
- Angular frontend framework

The aforementioned frameworks have flourished into complex, mature development components since their inception. It is not feasible to provide in-depth reviews of each, and thus a summary with categorical information is shown in Table 2.1 below.

Framework	Applicable Labels	Summary
Phaser	Game development, Graphics	Phaser is a fully featured JavaScript game development framework capable of creating HTML5 games. It provides features such as physics simulation, path finding AI, WebGL shaders, audio handling, animation and much more. Phaser works best in web browsers that support WebGL, the standard JavaScript API for rendering graphics (Khronos, n.d.).
Node JS	Runtime, Server environment	<p>JavaScript requires a language runtime to execute. In web browsers the runtime is built-in by its vendor. For example, in the case of Google Chrome, it uses a system known as V8 (Google LLC, n.d.).</p> <p>Node is a re-write of this system to allow JS to be used outside of the browser. It is a strong candidate to be used in this project as it integrates well with existing JavaScript tools, projects & frameworks.</p>
React.JS	Frontend	React.JS extends the browser Document Object Model (DOM) manipulation capabilities of JavaScript by providing a ‘Virtual Dom’ (Aggarwal, 2018). The idea behind the framework (developed by social network giant – Facebook), is a coding paradigm for plain JavaScript, that makes it easy for developers to handle interactions between logical UI components.
Angular	Frontend	Angular is a web application framework that uses the TypeScript programming language to build single-page mobile and web apps. This type of application framework is similar to the React.JS framework but provides much more built-in functionality.

Table 2.1 - JavaScript frameworks summary

2.1.5. Related Design Strategies

Design strategies that best suit the type and scope of the proposed project are discussed in this section. The term ‘design strategies’ refer to the design of the system architecture, deployment architecture and general design of the flow and interfaces of the system.

2.1.5.1. Design Patterns

Organized things are easier to understand and explain (LeFever, n.d.). The applicability of MVC, MVVM and Model-View-Presenter (MVP) patterns to this project was evaluated, based on their compatibility with design technologies.

In frameworks such as Node and React, there is no enforcement to select a design pattern. However, in the case of technologies such as .NET or Angular, MVC is strictly enforced. For the purposes of this project, I have chosen to adopt MVVM as the design pattern based on the following rationale.

- MVVM prevents application logic from being interleaved with business logic.
- Views are light weight representations of the interface and does not contain heavy logic.
- The application is free to use any organization of components, but aspects related to business logic can be organized using the MVVM approach.

2.1.5.2. Logical Structuring of Components

The project is developed using a layered architecture. It allows for changing parts of the system, without affecting unrelated areas. This is useful since the project has many different components. A drawback of employing such a system is that well designed interfaces need to be maintained between components. This adds extra development overhead, but is beneficial in the case of adopting un-planned changes. A conceptual overview of the frontend and backend components using the layered architecture are shown in Figure 2.19 and Figure 2.20 respectively.

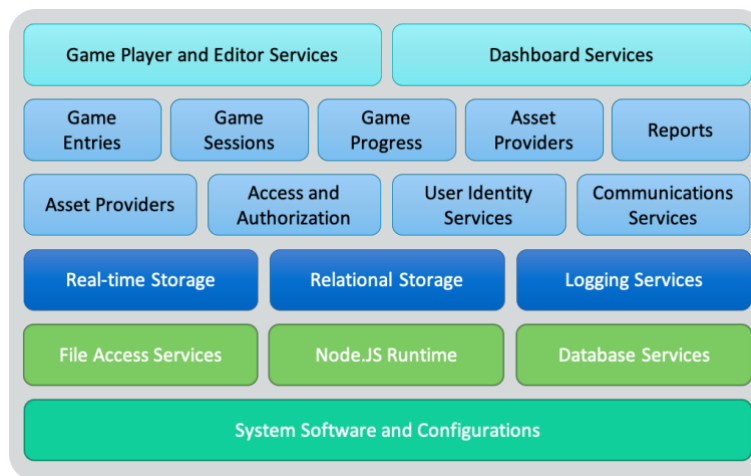


Figure 2.19 – Conceptual layered architecture of the backend components

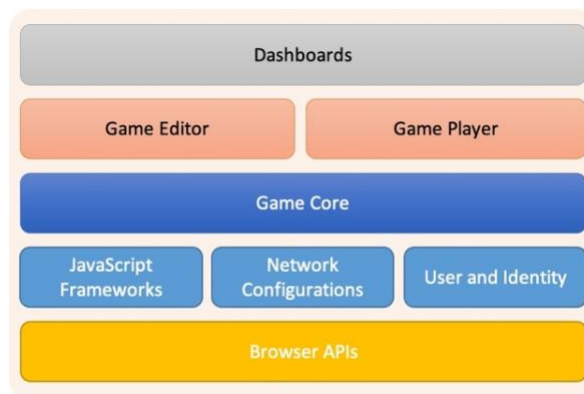


Figure 2.20 - Conceptual layered architecture of the frontend components

2.2. Use-case Diagram

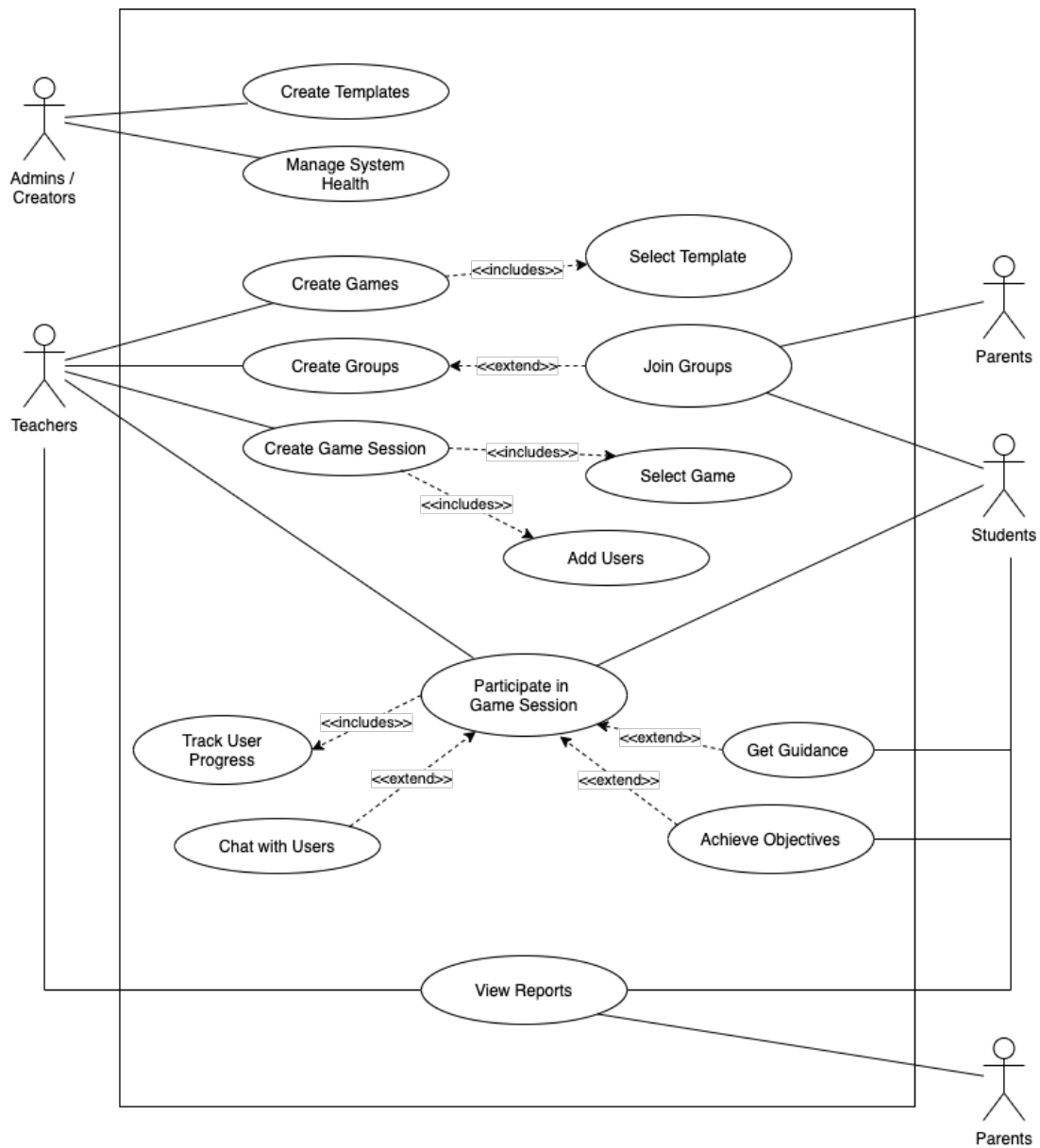


Figure 2.21 - Top-level UML use case diagram

Figure 2.21 above shows the top-level UML use case diagram for this project. The four main actors are administrators (also known as game creators), teachers, students and parents. The key concepts for terms mentioned in the use case diagram are explained below.

2.2.1. Key Concepts

2.2.1.1. Game Templates

Game templates are customizable starter projects for games. They are created by administrators. When a teacher selects a game template, its resources are cloned into a fully independent version of itself. This cloned version is the project file for the actual game.

2.2.1.2. Game Sessions

A game session is analogous to a pre-scheduled online video conference. It is an allocated time slot where students and teachers can play a pre-selected game (or test a game template). Game sessions are scheduled for entire collections of members called, Groups.

2.2.1.3. Groups

Groups help organize teachers, students and parents. For example, “Grade 9F” could be considered a group. A game session’s lifetime is managed by its parent group. Sessions in a group are scheduled for all the members in the group.

2.2.1.4. Guidance and Objectives

Guidance and objectives are two metrics for tracking user progress during a game session. The number of guidance hints required to complete a game, as well as number of objectives to complete are set by the teacher during the creation of the game. The system uses these metrics later for generating reports.

2.3. Requirement Specification

Due to the complexity of this system, it is impractical to include a detailed specification in this chapter. Instead, requirements are grouped by feature and documented as simplified bullet-points. The full requirement specification can be found in Appendix C.

2.3.1. Functional Requirement Specification

Requirements are captured according to these six areas; game editor, game play & communications, game sessions, users & groups, performance reports and overall dashboards. The relationships between these areas (excluding dashboards) are shown in the UML component diagram below (Figure 2.22).

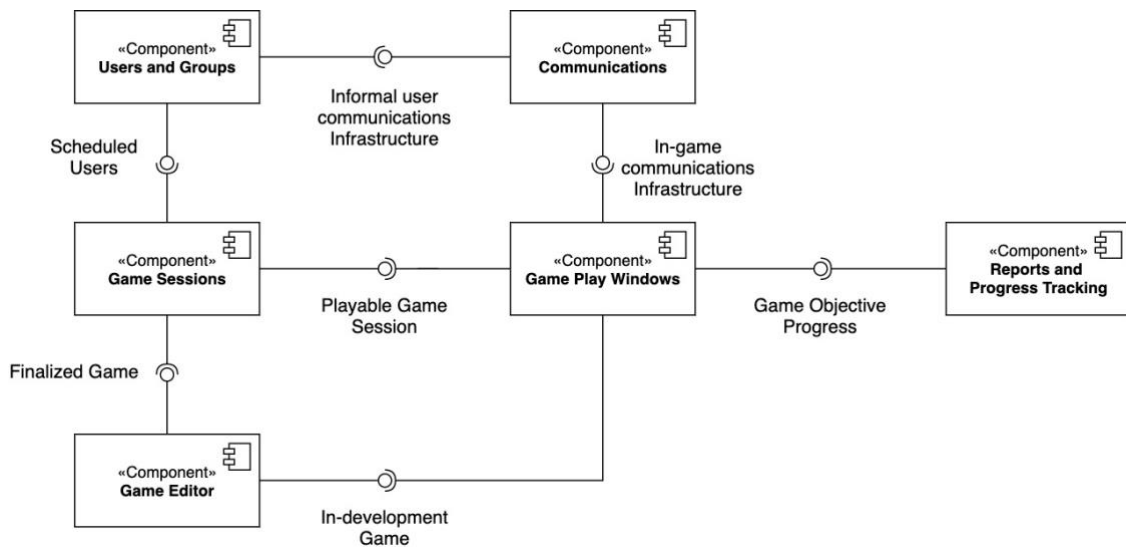


Figure 2.22 - UML component diagram for high-level features

2.3.1.1. Game Editor Requirements

1. Add, manipulate and program objects in a level scene.
2. Customize a level using a friendly UI without coding.
3. Have the option to add behavior to objects via code, but should not be required.
4. Create, duplicate and delete game levels.
5. Set game/template metadata (e.g., title, description, etc.)
6. Select a game template when creating a game.
7. Set objectives and guidance trigger points for game.
8. Specify supported report types for a game.

2.3.1.2. Gameplay and Communications Requirements

1. Students and teachers can participate in game play sessions.
2. Get notified about network errors as soon as they occur.
3. View and interact with the game.
4. Games should accept keyboard and mouse input from the host device.
5. Pause game if user backgrounds the session.
6. Track user game progress and events for reports.
7. Prompt user with hints to progress in the game, if they are struggling.
8. Provide indicator about the progress of each objective in the game.
9. Facilitate a group chat for each game play session.
10. Group chat must persist across play sessions for each session.

2.3.1.3. *Game Session Requirements*

1. Teachers should be able to schedule sessions for a group.
2. When scheduling a session, specify the game, time and users.
3. Add and remove users to/from a session that is already scheduled.
4. Users should get an email notification when a game is scheduled.
5. Users in a group (except parents) should be able to participate in a game session.

2.3.1.4. *Users and Groups Requirements*

1. Users should be able to login with a unique email and password.
2. Users must be able to logout of the system at any time.
3. New users must be able to register in the system.
4. Users cannot enter emails of existing user accounts
5. Registered users should be able to join a group using an invite link.
6. Unregistered or logged out users should be able to register / login before joining a group via an invite link.
7. Users must be able to leave a group they joined.
8. Only users of the group must have access to its access it.
9. Through the groups page, users should be able to see its details, members, scheduled sessions and reports.
10. Teachers must be able to delete groups they created.
11. Teachers must be able to create new groups.
12. Teachers must be able to associate students and parents.
13. Teachers must be able to schedule new sessions for the group.

2.3.1.5. Performance Report Requirements

1. Teachers, students and parents must be able to view performance reports.
2. The system must collect performance metrics of users in game sessions.
3. The system must not collection unwanted information such as PII.
4. Reports should be visualized as graphs and detailed using a tabular presentation.

2.3.1.6. Dashboard Requirements

1. A ‘summary’ dashboard should be available for all users.
2. The summary dashboard should display session updates and chat updates.
3. When a user clicks on a list item in the summary dashboard (e.g., a chat message) they should be navigated to the correct group the chat is made in.
4. Administrators and teachers should have access to a ‘templates / games’ dashboard.
5. From the templates / games dashboard they should be able to view existing entries and create new entries.
6. Clicking on a template / game entry should navigate them to the respective editing screen.
7. All users must have access to a ‘groups’ dashboard.
8. From the groups dashboard, users should be able to see a list of groups they are part of.
9. Clicking a list item in the groups dashboard should navigate the user to the correct groups page.
10. Administrators and teachers should be able to create new groups from the groups dashboard.
11. A search functionality must be present in both templates / games and groups dashboards.

2.3.2. Non-functional Requirements

Qualitative non-functional requirements of system are documented in this section.

- Real-time communication should be established reliably between users. Text messages should arrive in the same order as they are sent.
- Privacy of users should be maintained. Secure protocols must be implemented to prevent unauthorized parties from accessing user data.
- Interfaces should be readable, clear and easy to comprehend without overwhelming the user. Further, interfaces such as reports must be properly responsive on all screen dimensions.
- The application should not be device resource intensive.

Chapter 3 - Design

This chapter documents the design aspects of the system. The overall architecture of the system as well as the states, action sequences, object-oriented classes and data flow are discussed. The initial architecture builds on the scope of the system. Informal notations are used to describe the system architecture. The subsequent sections provide UML 2.5 diagrams.

3.1. System Architecture

The components between a single client and server instance are documented in this section. A diagram with informal notation of the components is shown below in Figure 3.1 below.

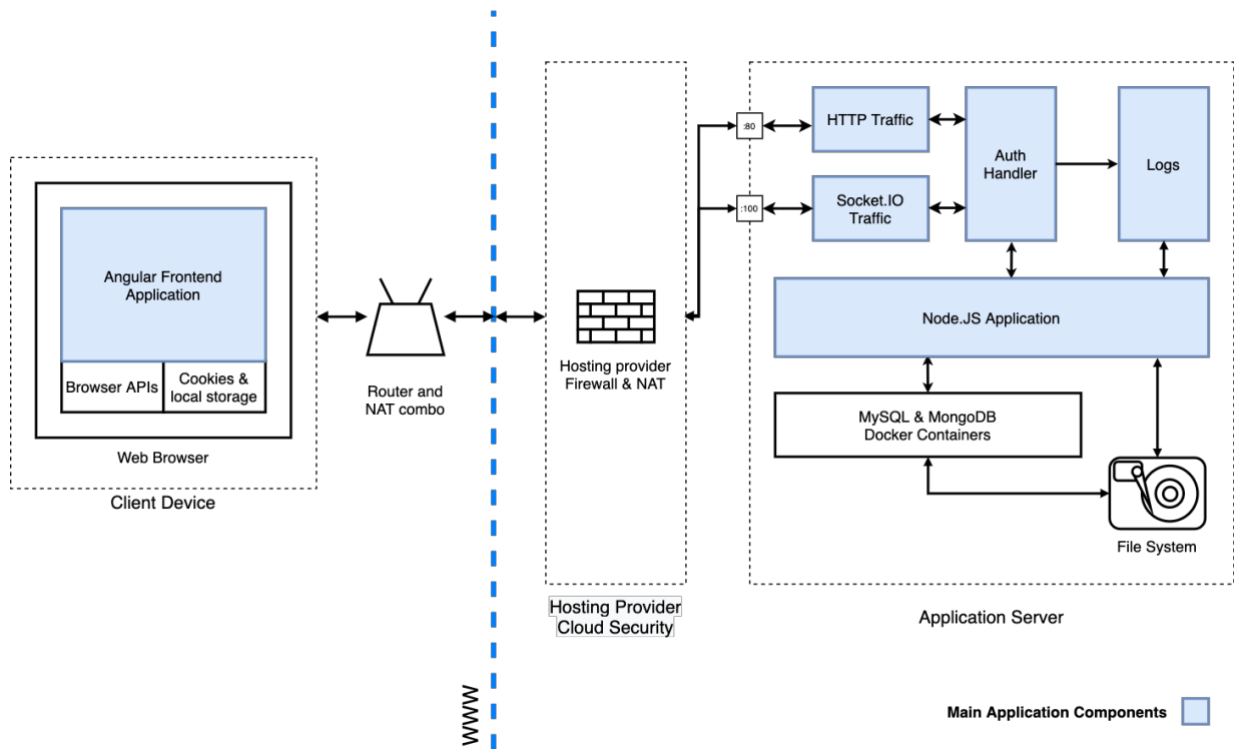


Figure 3.1 - High-level system architecture diagram

Connection between clients and users are authenticated where necessary. Communication via the HTTP protocol is handled by the default HTTP port 80, and Socket.IO protocol traffic is handled at port 100.

Docker containers are used to orchestrate database servers. This allows to replicate the development environment perfectly on the remote application server when deploying.

3.2. Designing the Game Editor

After several design iterations the template system was conceived. A dedicated group of users create general game templates, that are adapted into games by teachers. This lifts the technical burden of having to use coding to design a game from teachers.

Games and game templates are similar entities. They share attributes and logical connections with all parts of the system. As a result, the same processes used for creating the game template is used for customizing the game.

The game editor has many complex functionalities that made it a challenge to design in an approachable manner. The editor was designed from a top-down approach. Figure 3.2 shows the high-level activities the user is allowed to navigate to within the game editor. This activity-based structure helped align other actions into well-defined screens, making the game editor design (which sounds complicated) be approachable for even laymen users.

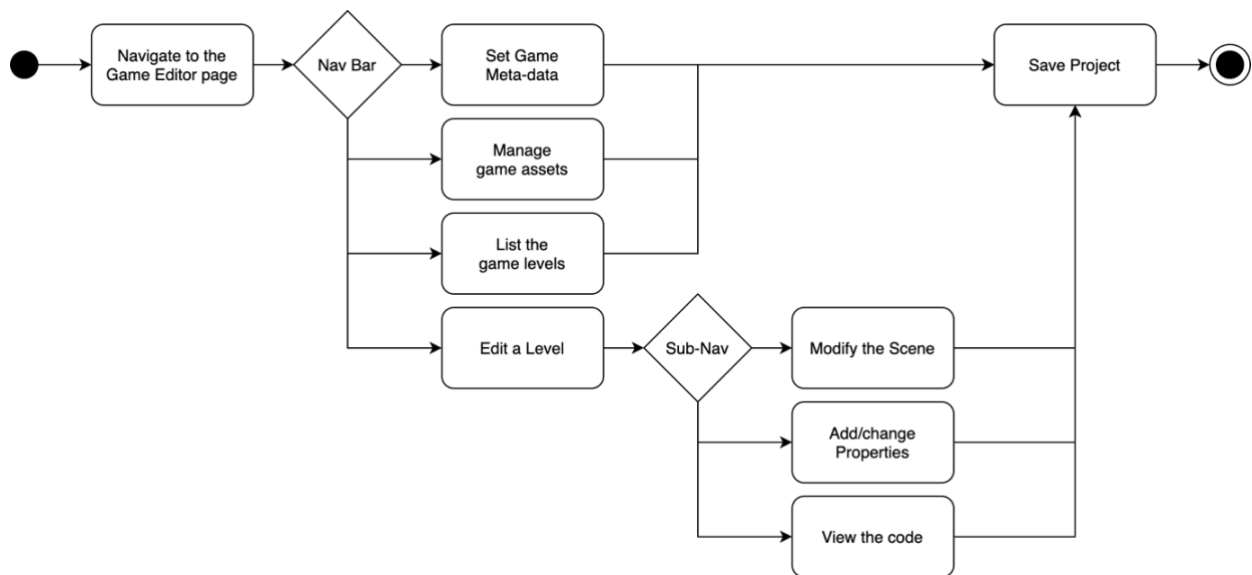


Figure 3.2 - Main activities of the game editor component

3.3. Dual Storage Engine Approach

Most of the system's data can be modeled using relational databases. This includes entities such as users, user associations, game entries, game progress, etc. However, during the process of modeling the game project and chat messages two requirements emerged.

1. Game projects are hierarchical. There are many nested objects which if stored in relational schema, require unnecessary table joins to store & retrieve data.
2. Chat messages are simple objects, but are created at a higher velocity than other entities. It would be better to store messages in a separate storage method.

To address this, two database types are employed within the system; MySQL (using InnoDB) and MongoDB. Using this method, object prototypes such as the one shown in Figure 3.3 of a complex game project can be easily stored and updated using NoSQL.

```
{
  "_id": "0001",
  "resources": [],
  "levels": [
    {
      "_id": "0001",
      "name": "Test Game",
      "type": "1",
      "displayMode": "on_top",
      "locked": false,
      "scene": {
        {
          "_id": "s0001",
          "type": "sprite",
          "name": "Title Screen",
          "frame": {
            "x": 0,
            "y": 0,
            "w": 400,
            "h": 300
          }
        }
      },
      "properties": {
        "formHTML": "",
        "values": []
      },
      "logic": {
        "setup": "",
        "perframe": "",
        "destroy": ""
      }
    }
  ]
}
```

Figure 3.3 - Prototype game project object

3.4. Main Classes of the System

There are several class-clusters within the system. These classes are associated with each other, and their models provide understanding on how the rest of the system must process them.

3.4.1. User Class Cluster

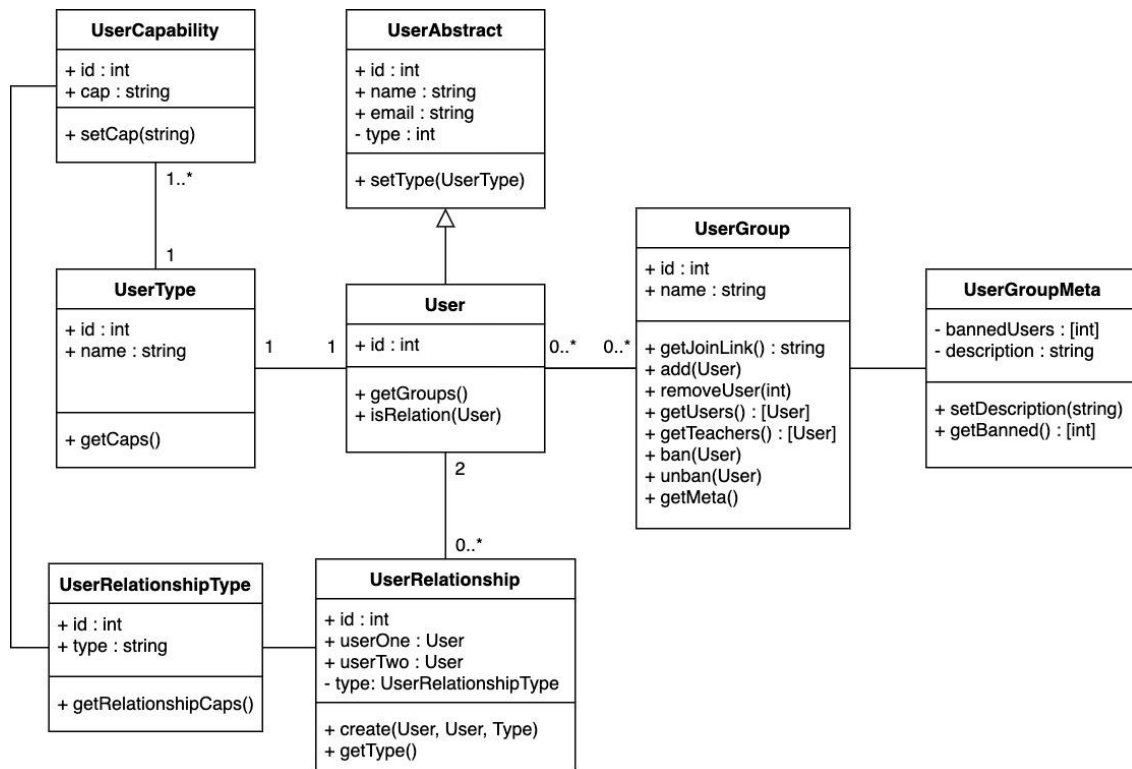


Figure 3.4 - User class cluster diagram

The ‘users and groups’ component manages user records, their access rights within the system, their relationships as well as their associations with groups. Groups are the largest unit of user organization within the system. It is logically equivalent to a traditional classroom members record. User relationships are used to identify associations between users.

If two users have for example, a student-parent relationship the parent is allowed to view progress reports of their child. The UML class diagram notation documenting the above associations is shown in Figure 3.4 above.

3.4.2. Game Sessions Class Cluster

A game session links users, games and reports together. Teachers create games that can be incorporated into pedagogical processes. Games played by students generate performance metrics that can be analyzed by all users. The design of classes associated with game sessions are shown in Figure 3.5 below.

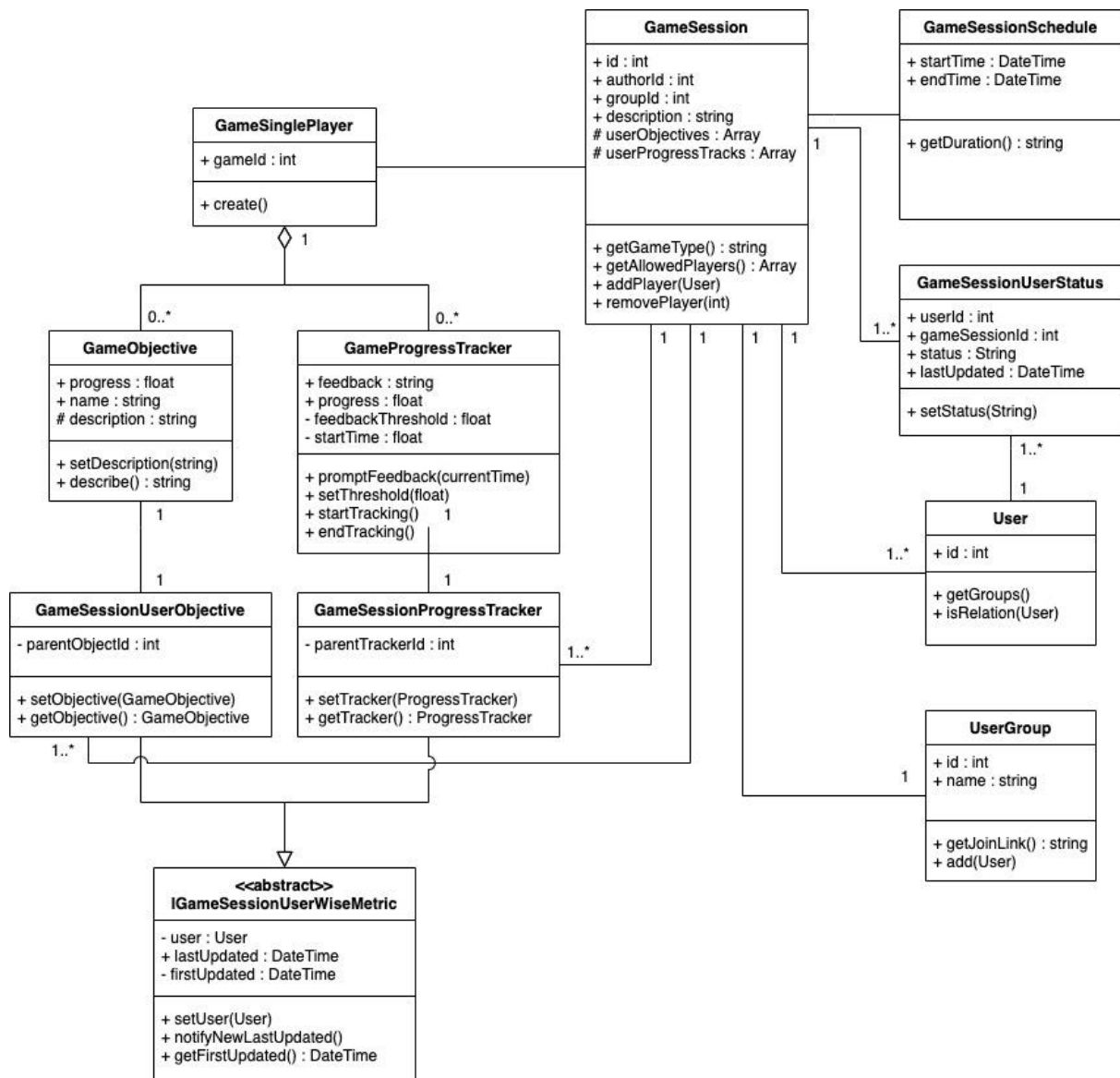


Figure 3.5 - Game session class cluster diagram

3.5. Main Flows of the System

There are four different user types in the system and they each have specific user journeys. Some are shared, such as the login process. However, flows such as creating games are unique to administrators and teachers. This section models a few main user flows in the system using UML activity and sequence diagrams.

3.5.1. User Login and Registration

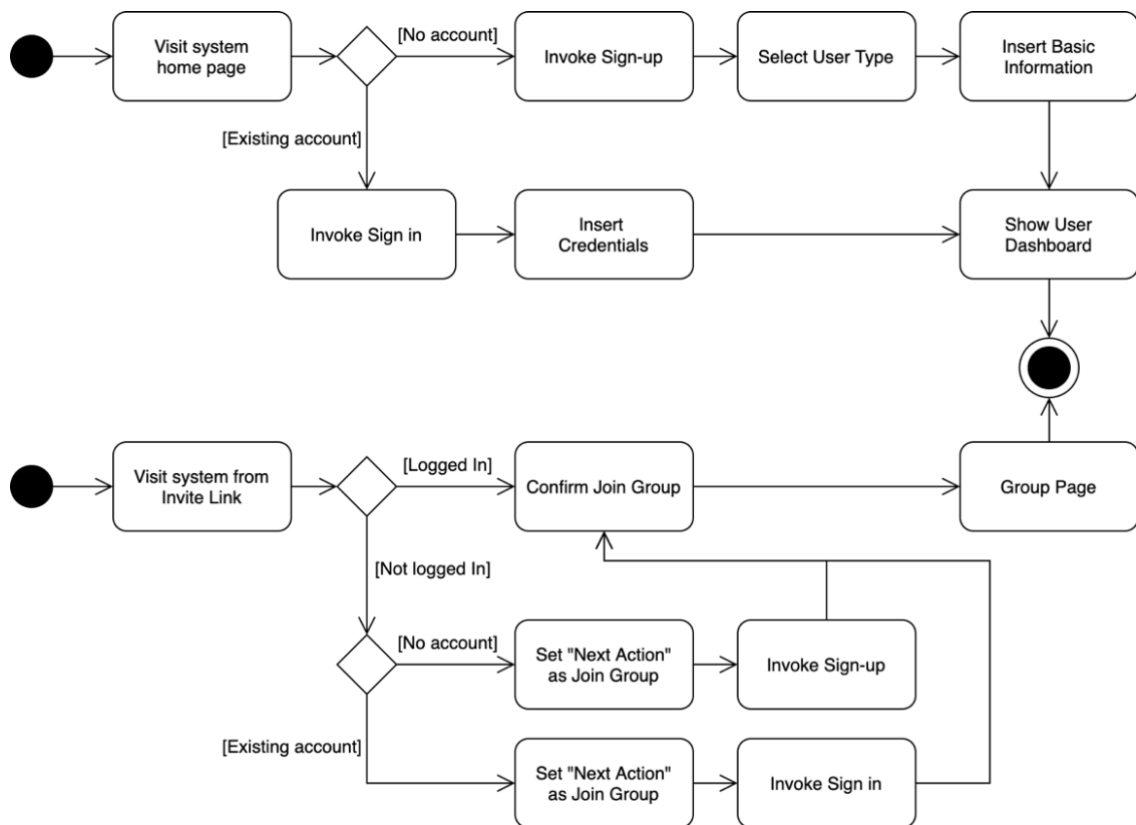


Figure 3.6 - User invite link join processes activity diagram

Figure 3.6 above shows several key activities involving onboarding users and the login process. The home page of the system allows new users to register and existing users to login, which is designed in a standard manner.

The system also allows creating invite links for new users. Invite links can be generated for a group. When a user navigates to an invite link, the system checks whether they are logged in. If so, prompts the 'join group' screen. The user can join the group immediately.

For users who are not logged in or aren't registered yet, the system will allow them to perform that action and afterwards, confirm whether to join a group. The redirection from the login page back to the join group page, is handled via a query parameter. It is set when redirecting the user to login or sign up initially.

3.5.2. Game and Game Template Creation

Since games and game templates are logically similar entities within the system, the process of creating them is also common. The transition from a template to a game, happens via a cloning process as shown in 6.1.1. of the Figure 3.7 sequence diagram.

Initially, the template selected by the user is cloned along with any assets (so they can be modified without affecting the template). The template's details are superimposed with that of the values set for the game. Finally, the new records and files are persisted and the editor acknowledges the same.

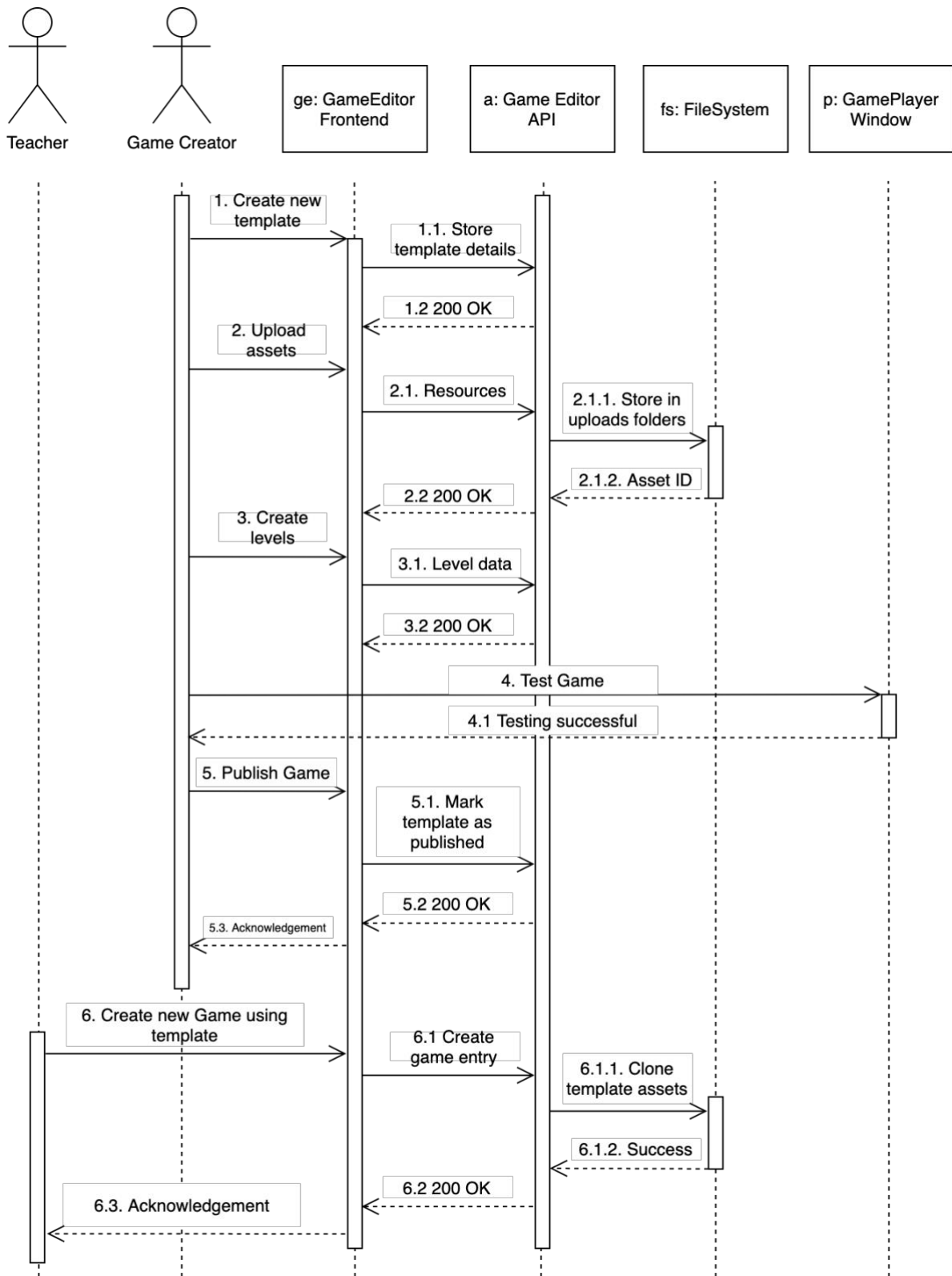


Figure 3.7 - Game and game template creation sequence diagram

3.6. Main Interfaces

User interface designs of the system were first made on pen & paper as wireframes. Once the details were fleshed out, they were prototyped using Figma (Figma Inc., n.d.). This includes the landing page, all interfaces and iconography.

3.6.1. Landing Page



Figure 3.8 - Game-based e-learning platform landing page design

The system is nicknamed, 'EDGE's and is displayed in the homepage branding Figure 3.8. The blue and white color scheme is used throughout the system.

3.6.2. Teacher Dashboard

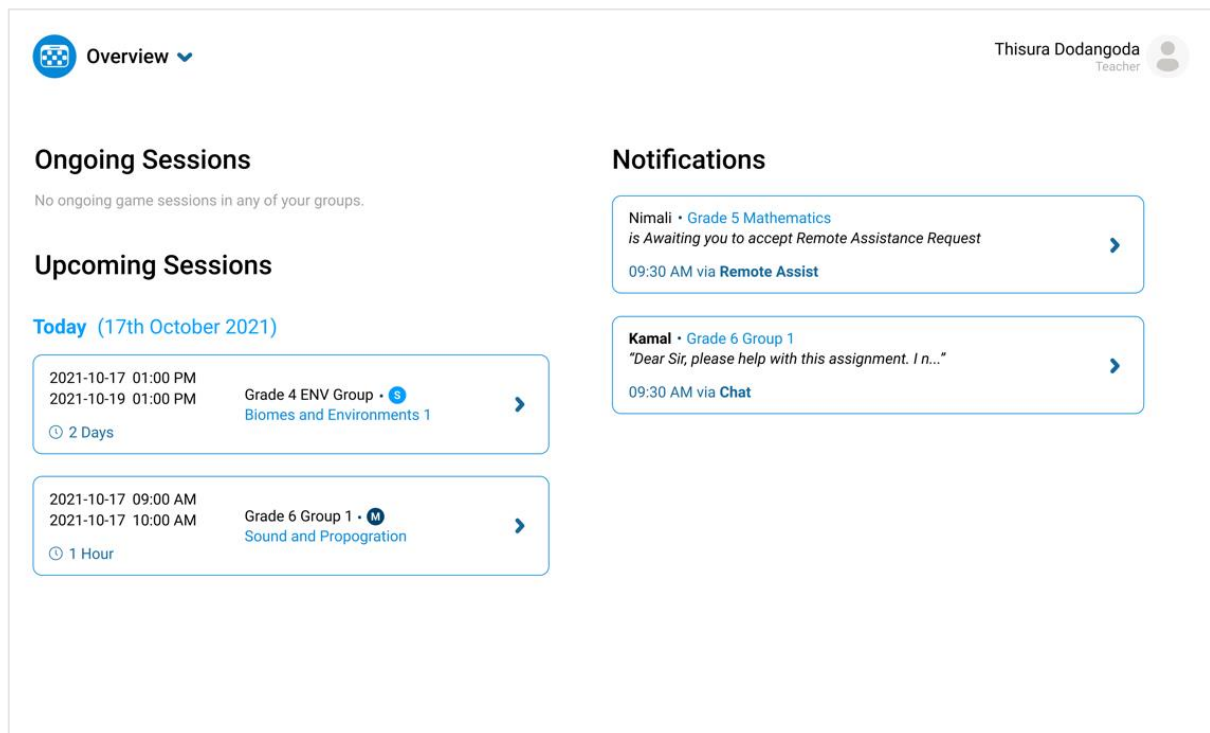


Figure 3.9 - EDGE system teacher dashboard page design

Figure 3.9 above shows the interface design of the Teacher dashboard. As mentioned on the top-left, the current page is for the overview screen. The term ‘overview’ dashboard technically refers to the ‘summary’ dashboard mentioned in the informal version of the requirements.

3.6.3. Gameplay Window

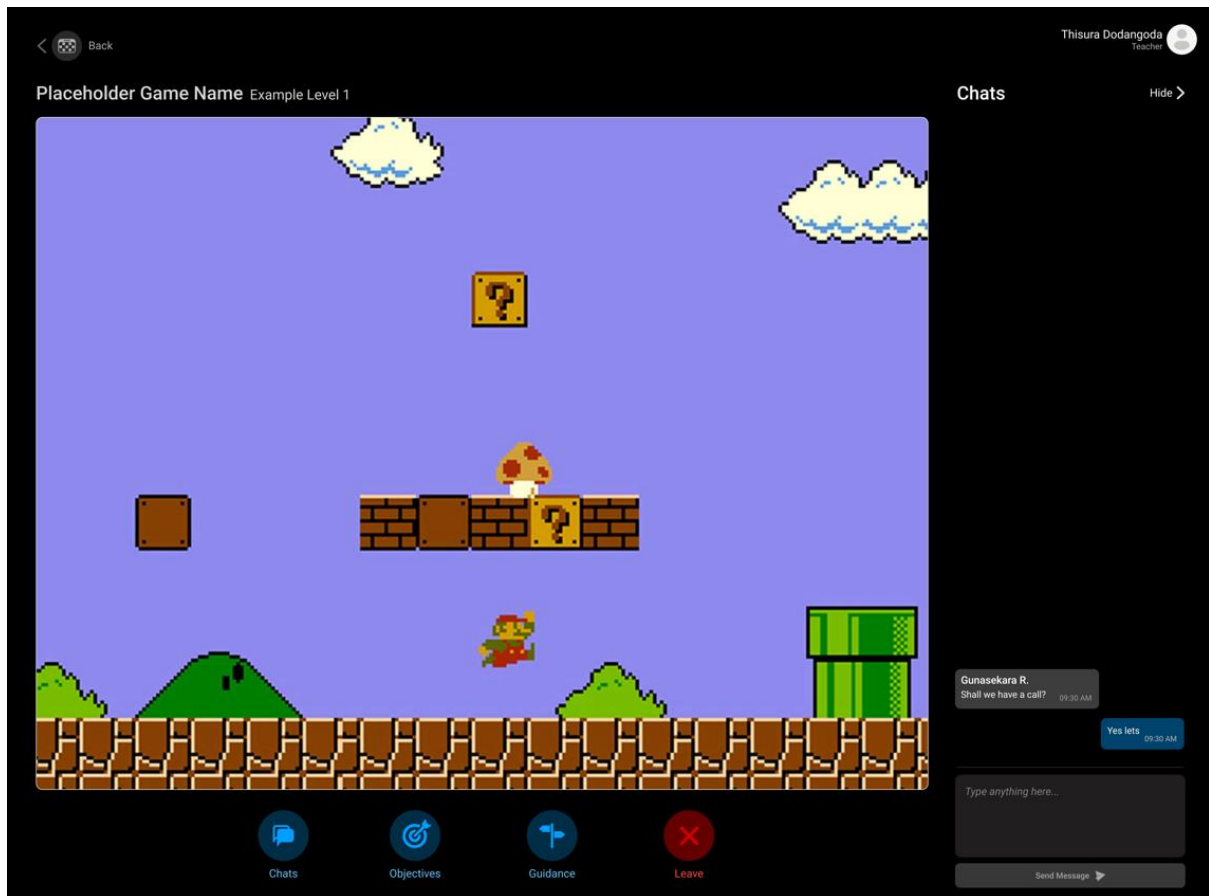


Figure 3.10 - Game play window design

Figure 3.10 above shows the planned design of the game play screen. All iconography was made as part of the project. The placeholder game is of Mario Brothers (Nintendo Co., Ltd., n.d.).

Chapter 4 - Implementation

In this chapter the implementation strategies used to develop the project are discussed. The development process, technology stack, development setup and relevant code fragments are some of the topics included. The system has not yet been fully implemented, and is in a partially developed (but functional) state.

4.1. Development Process

This project is developed using the Feature Driven Development (FDD) agile process. The rationale for its selection is that it offers the most logical organization of work items of the system. Since this is a solo project, formulating models of complex system features assisted tremendously. The customized version of the formal FDD process employed for this project is shown below in Figure 4.1 below.

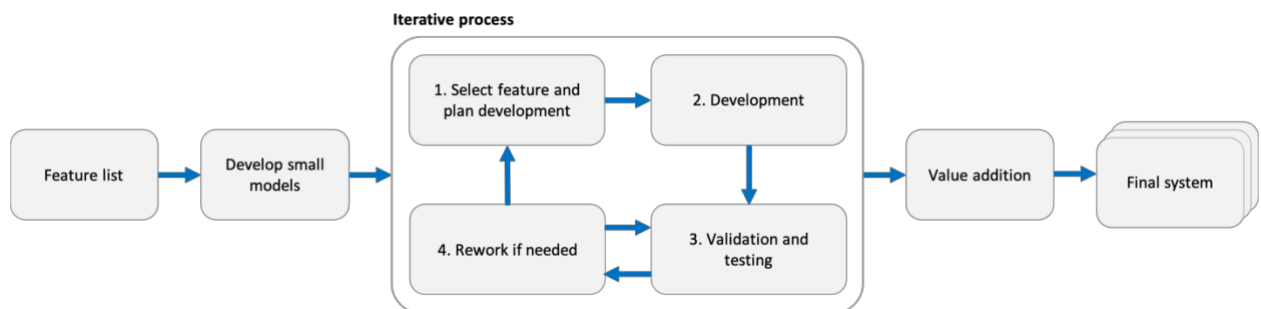


Figure 4.1 - Customized FDD development process

4.2. Technology Stack

The project uses a JavaScript technology stack for frontend and backend components. The rationale for this decision is my personal familiarity with vanilla JavaScript. Using a familiar stack reduced technical uncertainties during planning of features. Further, it reduced schedule costs of learning new technologies. The technology stack is as shown in Figure 4.2 below.

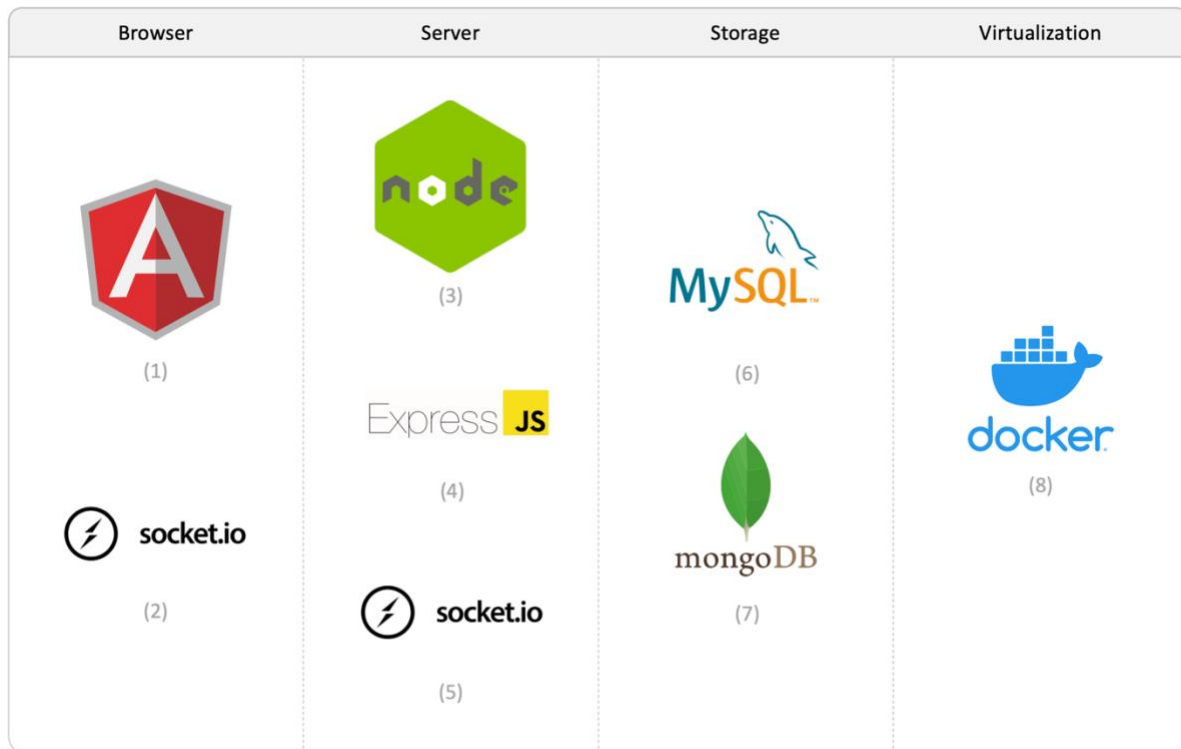


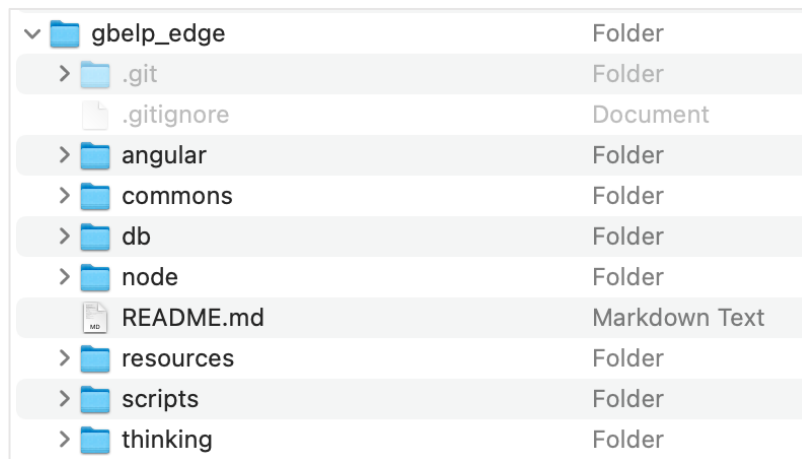
Figure 4.2 - Implementation technology stack

The formal names of the frameworks and their uses are as are,

1. Angular JS – frontend framework.
2. Socket.IO – client-side library for two-way communication.
3. Node – JavaScript runtime environment for the backend.
4. Express JS – application server framework for Node.
5. Socket.IO – server-side library for two-way communication.
6. MySQL – relational database management system.
7. MongoDB – document-oriented NoSQL database system.
8. Docker – virtualization platform used for managing storage container images.

4.3. Development Setup

The Visual Studio Code IDE was used to code the system’s frontend and backend. The project directory is organized as shown in Figure 4.3 below.



▼	gbelp_edge	Folder
>	.git	Folder
	.gitignore	Document
>	angular	Folder
>	commons	Folder
>	db	Folder
>	node	Folder
	README.md	Markdown Text
>	resources	Folder
>	scripts	Folder
>	thinking	Folder

Figure 4.3 - Development setup folder structure

4.3.1.1. Project Structure

The ‘angular’ and ‘node’ folders contain the frontend and backend code respectively. The ‘db’ folder contains SQL backups of the MySQL databases. The ‘commons’ folder contains code that is shared by both frontend and backend code.

Both frontend and backend sub-projects are configured to use TypeScript. This prevents any potential bugs caused by mismatched types or unexpected null values, since the TypeScript compiler checks this at compile time.

4.3.1.2. Code Sharing

The system architecture for this project is, client-server. Due to this reason, it is important that the client and server exchange data in a common accepted format. While using a notation such as JSON partially solves the issue, the problem of incompatible objects still remains.

A common way to address this issue is to duplicate the classes for the object in both client and server code in exactly the same manner. However, this means any change to one of the notations must be duplicated manually in the other.

To avoid this problem and save both time and energy, the ‘commons’ folder was created. It simply contains business logic and entity class definitions. Since both frontend and backend

code is written in TypeScript, the same entity definitions can be imported into both projects. Through this method, changes in one project are automatically reflected in the other.

4.3.1.3. Docker and Docker Compose

Docker Compose creates a virtual network of a predefined set of Docker containers, and allows them to act as though hosted in one logical computer. It further handles the synchronization of data to non-volatile storage. Docker was used since it minimizes the setup effort required. Further it prevented the developer's local machine from being cluttered by MongoDB and MySQL configurations and executables. A screenshot of the Docker Containers running on the local development machine is shown in Figure 4.4 below.

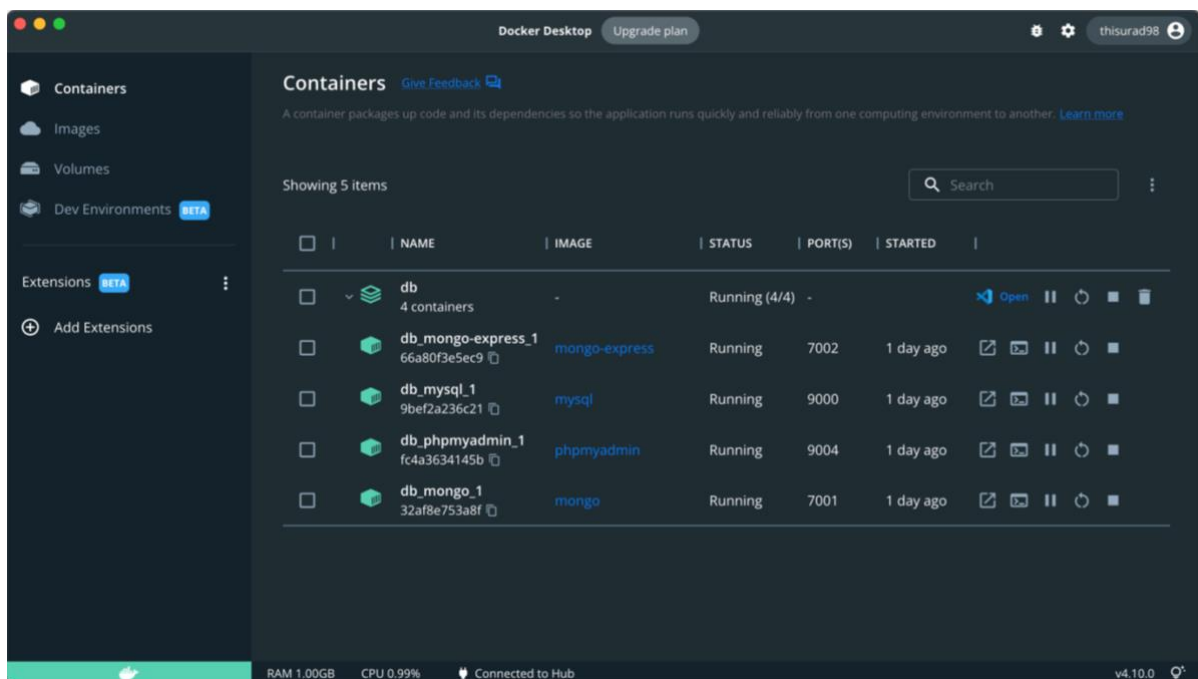


Figure 4.4 - Docker Desktop managing the MySQL and Mongo containers for the system

4.4. Relational Schema

The final MySQL database schema can be visualized through the phpMyAdmin console since it was installed at the time of setting up Docker containers. Figure 4.5 shows the relational schema of the project through its ‘Designer View’.

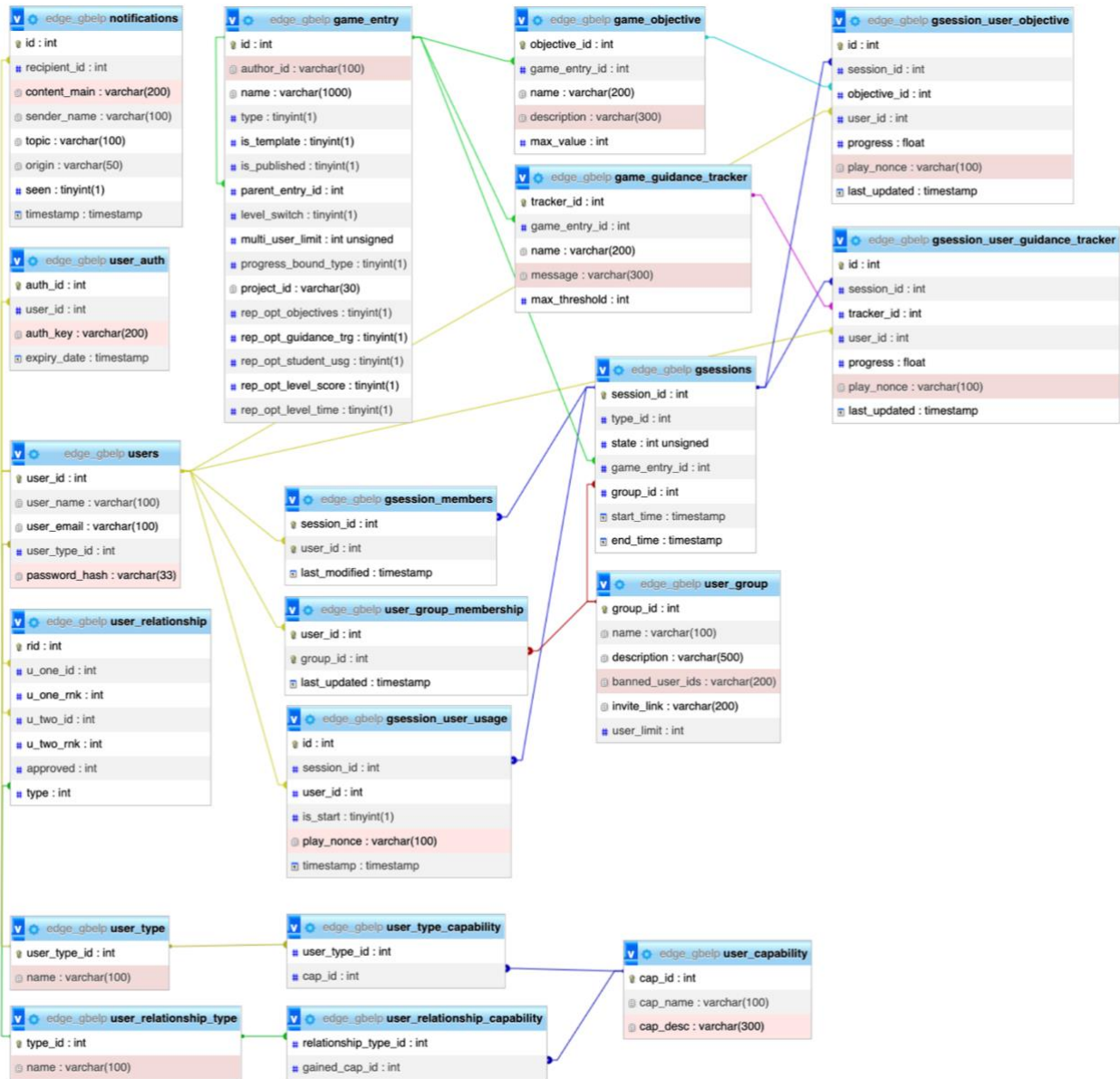


Figure 4.5 - Designer view of the EDGE system's relational database

4.5. Interfaces and Code Structures

4.5.1. Group Invite System

Once a group has been created, any member of the group can share an invite link requesting more users to join. Users can copy this link from the ‘Copy Invitation’ option in the groups page as shown in Figure 4.6.

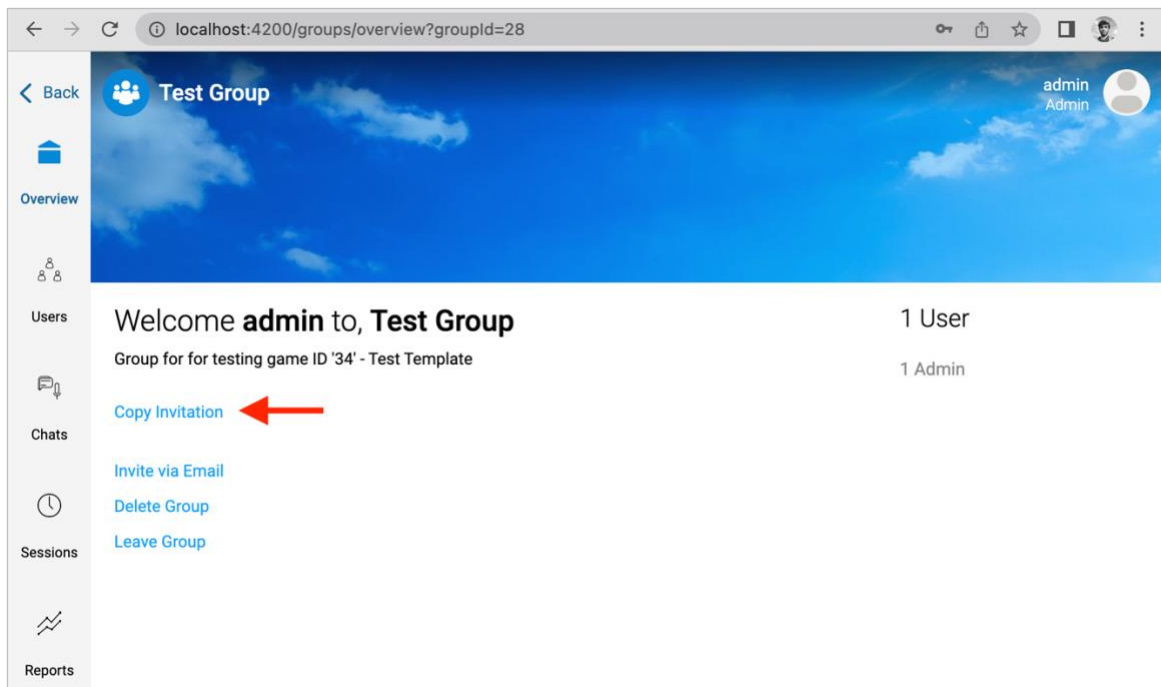


Figure 4.6 - Copy invitation link option in the group overview page

A sample group invite link is shown below.

- <http://localhost:4200/groups/join/94c18e03a1061b80e491b383cbc8c80d>

The random string of characters at the end of the link is an encrypted value representing the invited group. It is encrypted to prevent users from guessing internal group IDs and joining groups they were not invited to. The encryption uses the AES-256 cryptographic algorithm with the CBC mode of operation. The code managing cryptographic operations is shown in Figure 4.7 below.

```

import crypto from 'crypto';
import * as l from './logger';

const algo = 'aes-256-cbc';
const key = Buffer.from('***'); // 32 char key
const iv = Buffer.from('mit_3201_edge_iv');

/**
 * @param plaintext utf8 string to encrypt
 */
export function encrypt(plaintext: string): string{
  try{
    let cipher = crypto.createCipheriv(algo, key, iv);
    let encrypted = cipher.update(plaintext);
    encrypted = Buffer.concat([encrypted, cipher.final()]);
    return encrypted.toString('hex');
  }
  catch(error){
    l.logc(JSON.stringify(error), 'crypto.ts-encrypt');
    return '';
  }
}

```

Figure 4.7 - Code structure for creating the group invite link

When a user navigates to the invite link, they will see a page showing the group title and a message of the pending actions if they confirm the invite (Figure 4.8). If the user is not currently logged in, the option to login or register will be shown.

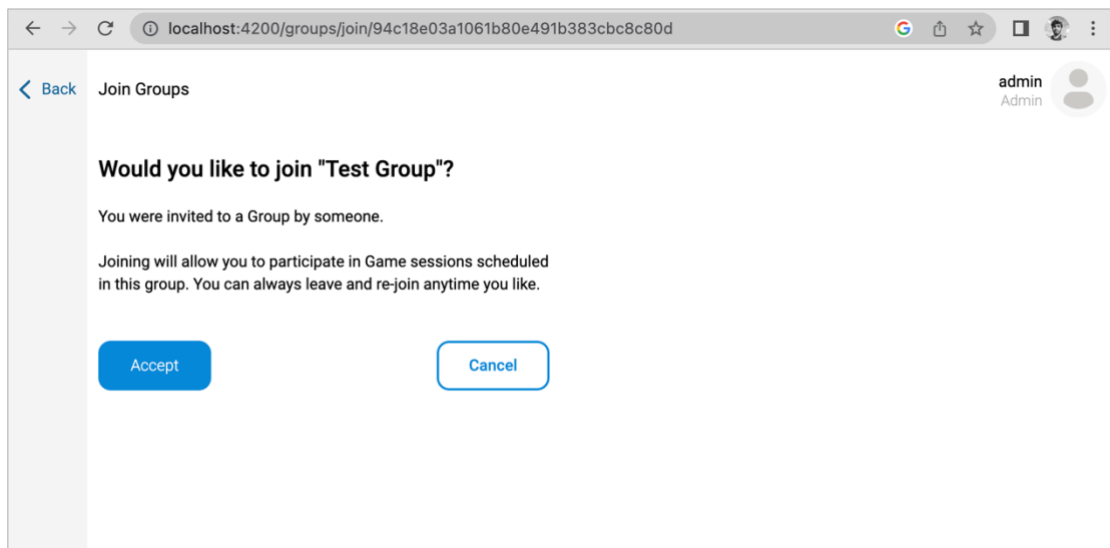


Figure 4.8 - Join group from invite link page

4.5.2. Game Editor

One of the most difficult parts to implement in the system was the game editor pages. This component is responsible for all aspects related to game template / game design. There are four main sections of the editor: overview, resources, levels and scene editor. The scene editor is further separated as, scene, properties and scripts sections.

4.5.2.1. Game Editor Overview Section

The overview section defines the metadata of the game (or game template) that's currently being edited. It also allows the user to define the main objectives and guidance trigger points for the entire game. Figure 4.9 shows the overview section of the game editor component.

The screenshot shows a web browser window at localhost:4200/template/edit?gameId=33. The page is titled 'Edit Template' and has a user profile for 'admin Admin' in the top right. A sidebar on the left contains navigation links: 'Back', 'Overview', 'Resources', 'Levels', and 'Editor'. The main content area is divided into several sections:

- Details:** Includes 'Template Name' (CoolTemplate) and 'Game Type' (Single Player).
- Game Options:** Includes 'Concurrent user limit' (0), 'Level Switching' (Time Based), and 'Objective Progress Upper Bound Type' (Limited).
- Report Options:** Includes checkboxes for 'Objective Tracking Reports', 'Guidance Tracker Reports', 'Student Usage Reports', and 'Level Score Reports'. 'Level Timing Reports' is also checked.
- Objectives:** A table with columns for Objective Name, Description, Maximum Progress, and Actions. It contains two entries: 'Objective 1: Collect 10 gold coins' and 'Objective 2: Collect 50 gold coins'.
- Guidance Trackers:** A table with columns for Tracker Name, Feedback, Feedback Threshold, and Actions. It contains two entries: 'Trigger 1: Interact with coins timeout' and 'Trigger 2: Interact with game timeout'.

Figure 4.9 - Game editor page overview section

4.5.2.2. Game Editor Resources Section

The game resources page allows users to upload image or sound assets to the game project (Figure 4.10). These assets are automatically assigned the current timestamp, and its reference is added to the project file. The operations for saving incoming files are handled by a library integrated on the backend code, named 'Multer' (OpenJS Foundation, n.d.). The configuration for file uploading is shown in Figure 4.11.

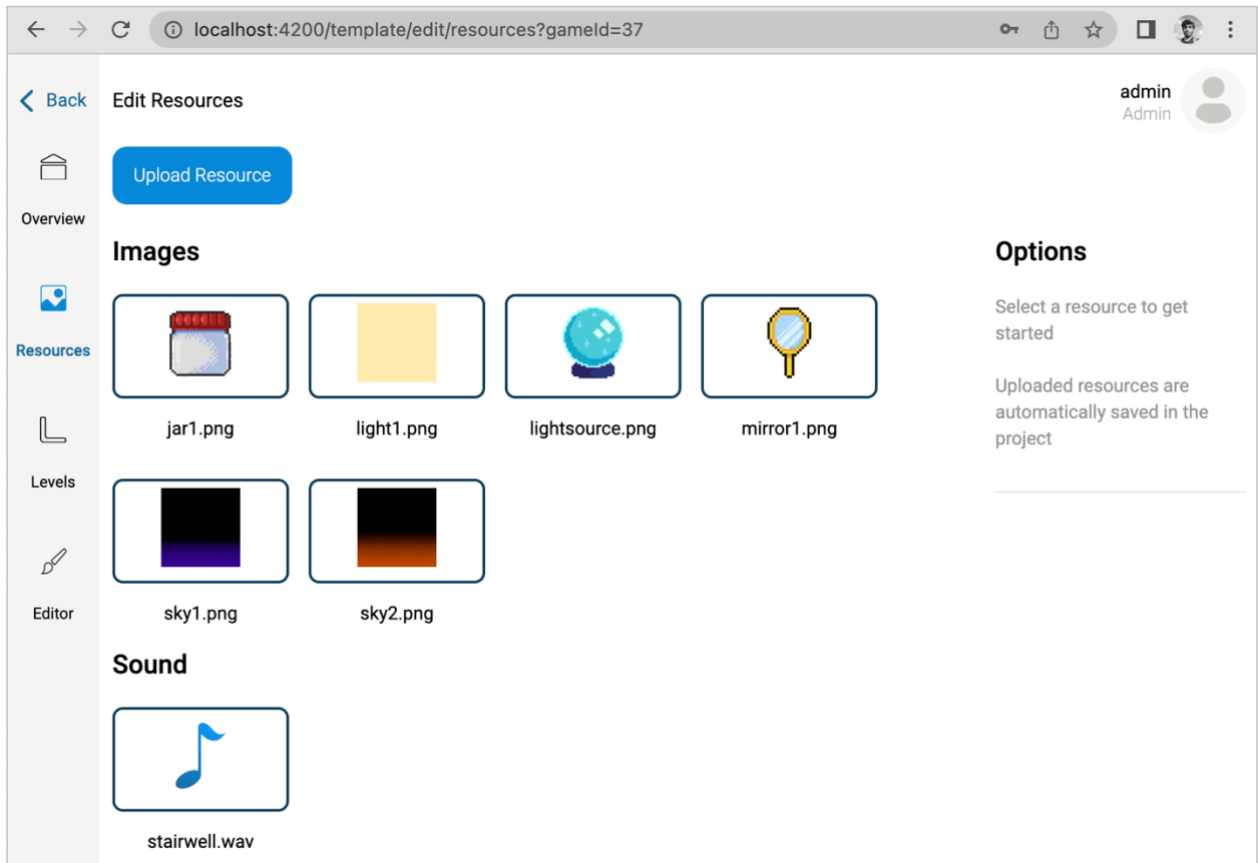


Figure 4.10 - Game editor page resources section

```

import * as pc from '../util/parseconfig';
import multer from 'multer';
import * as path from 'path';

const config = pc.parseConfig('config.json');
const multerDiskWriteConfig = multer.diskStorage({
  destination: (req, file, callback) => {
    const mimetype = file.mimetype;
    if (mimetype.includes('audio') || mimetype.includes('sound'))
      callback(null, config.fs_res_path_sound);
    else
      callback(null, config.fs_res_path_image);
  },
  filename: (req, file, callback) => {
    const filename = Date.now() + path.extname(file.originalname);
    callback(null, filename);
  }
});

```

Figure 4.11 - Multer library file upload configuration

4.5.2.3. Game Editor Levels Section

The levels section allows the user to manage existing levels and add new levels. It also allows the user to edit the title and end game screens. The page is shown in Figure 4.12.

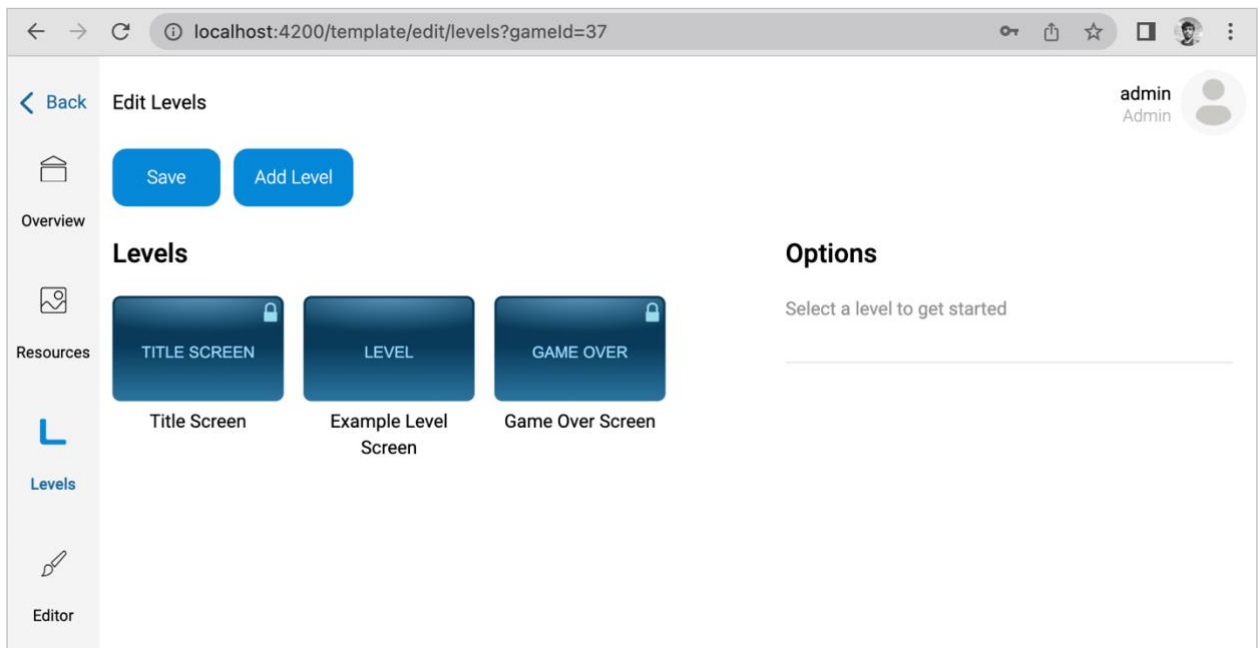


Figure 4.12 - Game editor page levels section

4.5.2.4. Game Editor Level Scene Editor Section

Figure 4.13 shows the most complex and difficult to implement section of the entire system, the level editor screen. On the left, the level editor tab expands into the scene, resources and script sections. From the top section, the user can save the current project or ‘run’ it directly in a new tab.

The ‘Hierarchy’ section provides a list of all the objects in the current scene. The center ‘Scene Map’ is an HTML canvas element built using the Fabric JS component (Zaytsev, et al., n.d.). The right section allows users to add new items from the ‘Library’ section, and change properties of existing objects from the ‘Options’ section.

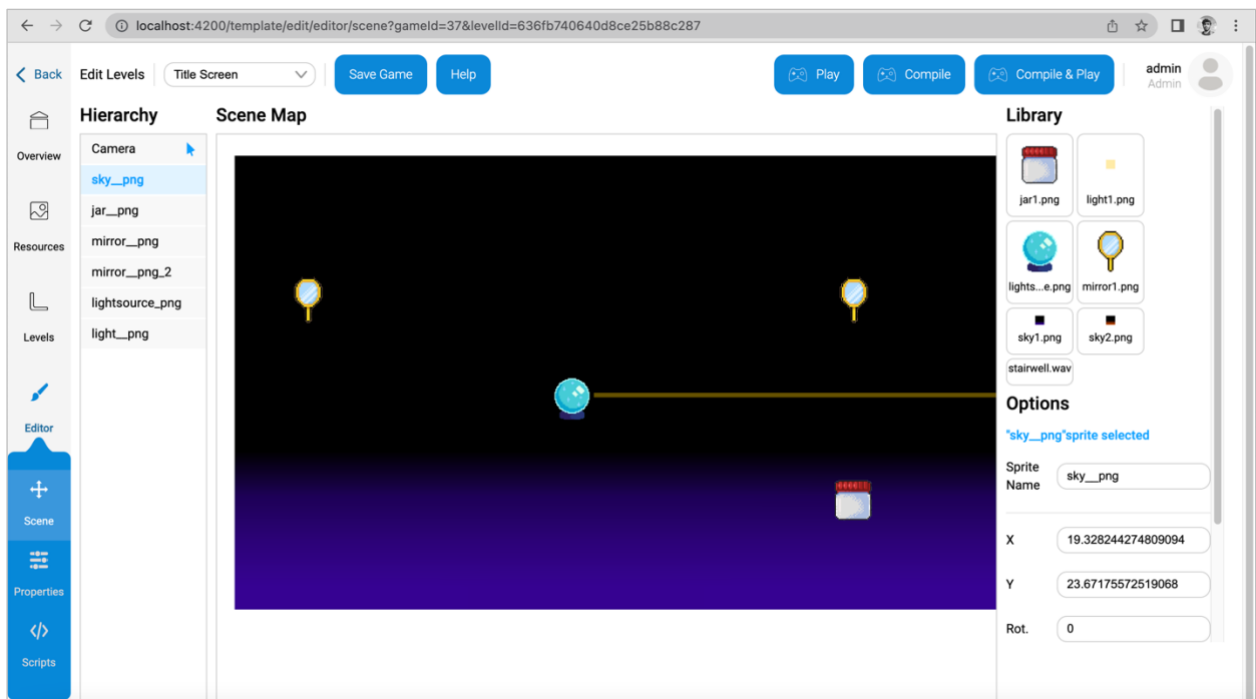


Figure 4.13 - Game editor level scene editor page

4.5.2.5. Game Editor Level Properties Editor Section

The properties editor provides property list for teachers to easily customize a level without fiddling with the level scripts. When creating game templates, this page shows two panels: the left panel shows an editor for defining the property list, and the right panel shows the actual property list generated from its definition on the left. This is demonstrated in Figure 4.14 below.

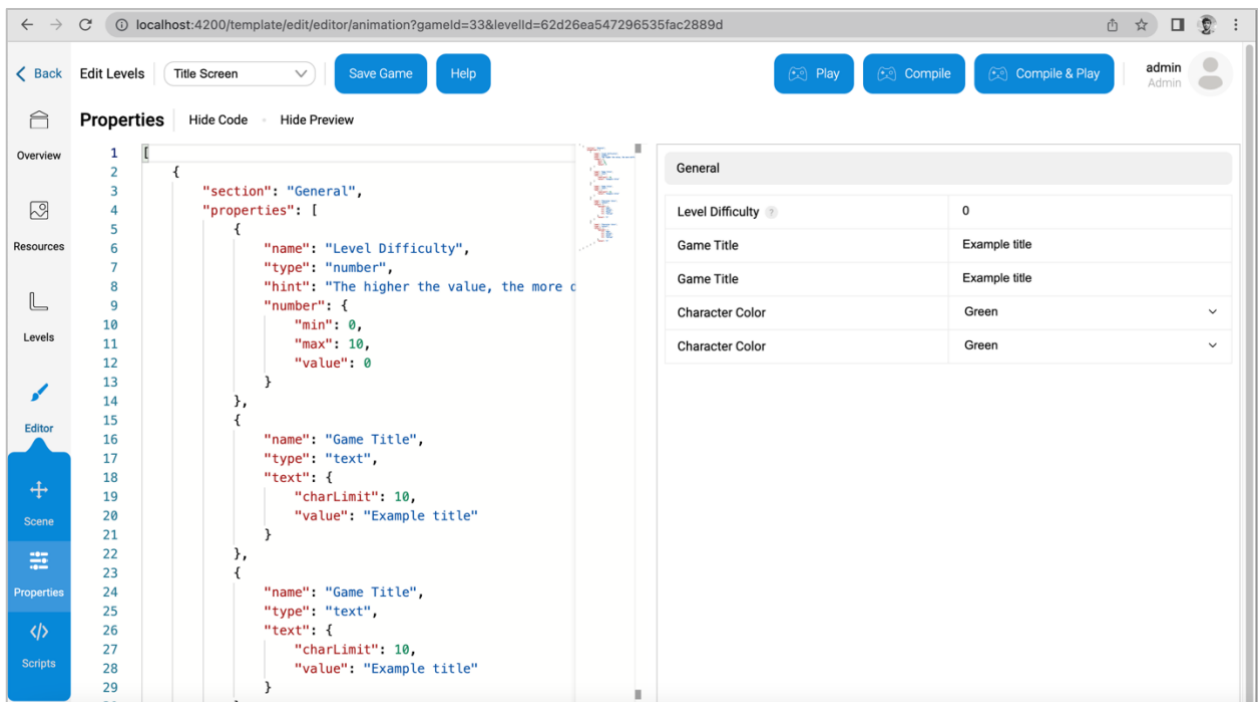


Figure 4.14 - Game editor level properties editor page

4.5.2.6. Game Editor Level Scripts Editor Section

The script editor section is intended for game creator users and not teachers. This section provides an editor interface for writing actual JavaScript code. Additionally, a custom library is loaded to interop between the user code and the system. The interface for the script editor page is shown in Figure 4.15. The interop library code is shown in Figure 4.16.

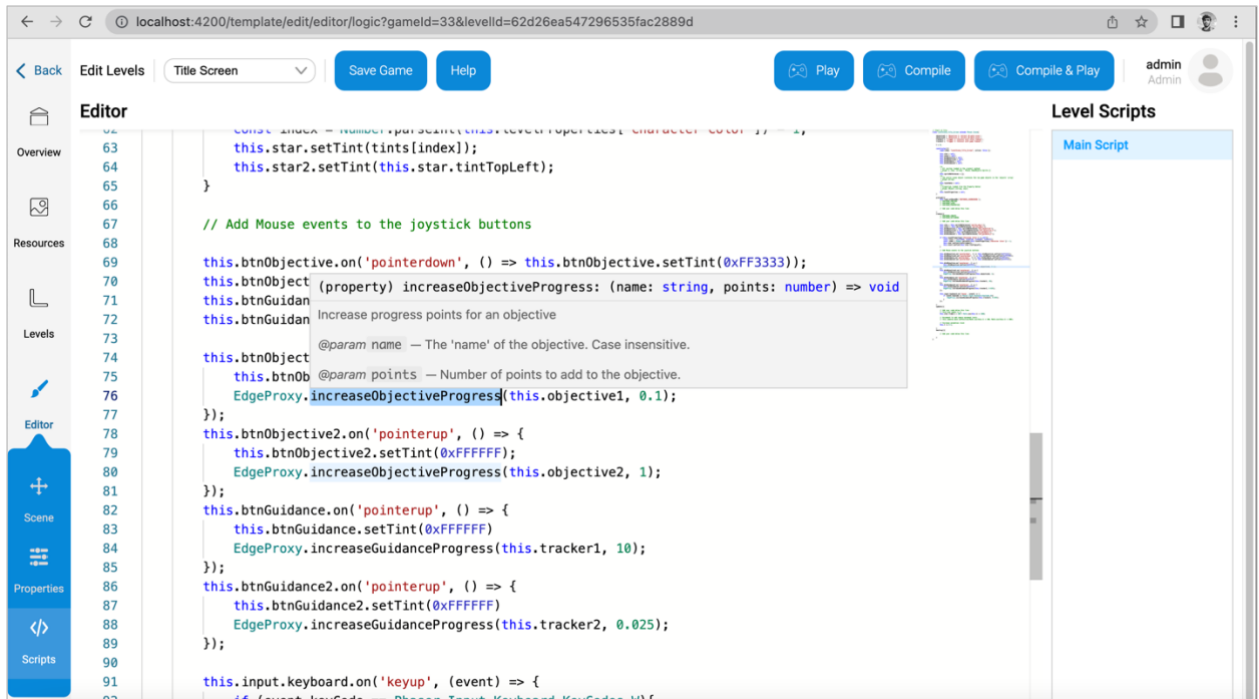


Figure 4.15 - Game editor level script editor page

```

require('../game_compiler/phaser/phaser');
require('./edgeinternals');

/**
 * Communicate with the EDGE system.
 */
const EdgeProxy = {
  /**
   * Increase progress points for an objective
   * @param {string} name The 'name' of the objective. Case insensitive.
   * @param {number} points Number of points to add to the objective.
   */
  increaseObjectiveProgress: function(name, points){
    if (window.EdgeInternals._on_updateObjective != null)
      window.EdgeInternals._on_updateObjective(name, points);
    else
      console.log("Edge Internal implementation for _on_updateObjective missing");
  },
  /**
   * Increase hitpoints for a guidance tracker
   * @param {string} name The 'name' of the guidance tracker. Case insensitive.
   * @param {number} points Number of points to add to the objective.
   */
  increaseGuidanceProgress: function(name, points){
    if (window.EdgeInternals._on_updateGuidance != null)
      window.EdgeInternals._on_updateGuidance(name, points);
    else
      console.log("Edge Internal implementation for _on_updateGuidance missing");
  }
}

```

Figure 4.16 - JavaScript interop library code

4.5.3. Performance Reports

Performance reports are generated by data collected during user play sessions. For example, if a user progresses in a certain objective for a game this change is recorded in the system. Performance reports are available in the group's reports page. Available report types are listed in the initial page (Figure 4.17). Clicking on any one of the reports will first display the report for the entire group.

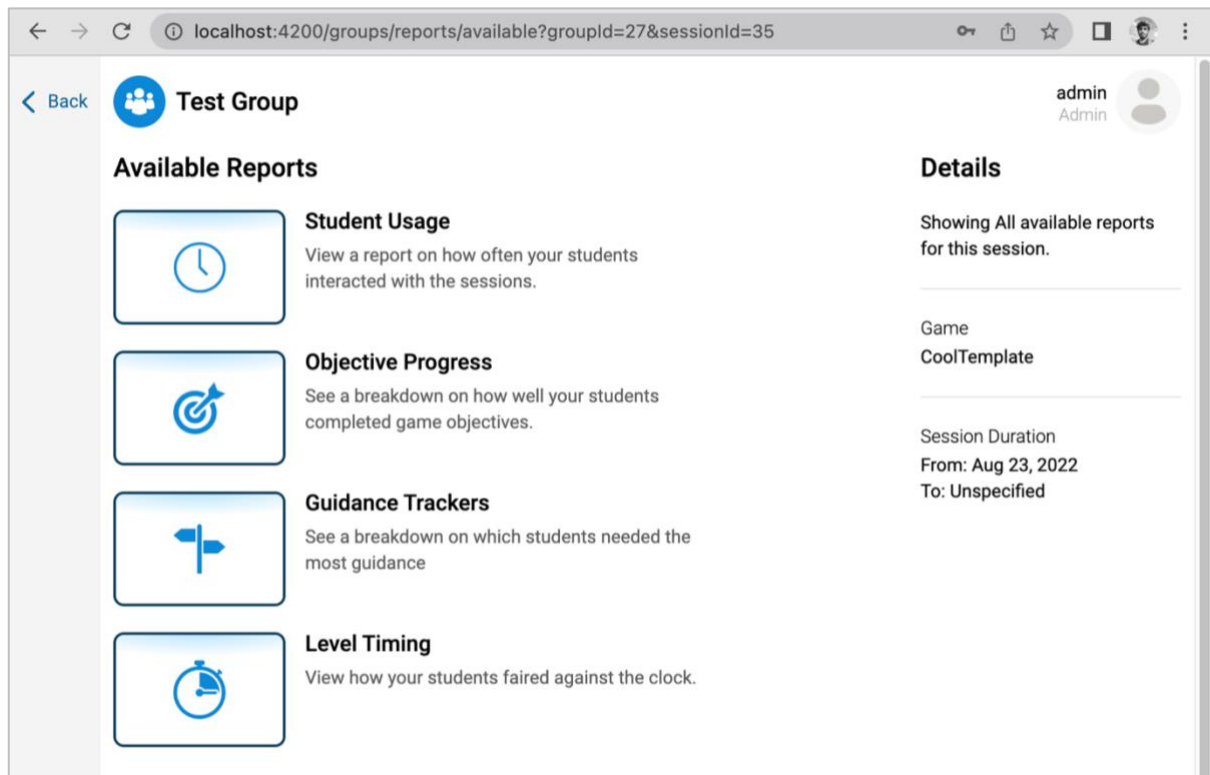


Figure 4.17 - Available report types in the group page's reports section

The process of separating performance data by each time a user plays any specific game was a challenge. This is because at the time, every user 'play session' had the same scheduled session ID. There wasn't a clear method to map which attempt the performance data of the user belonged. To resolve this issue, a 'play nonce' field was introduced. An example of objective progress data tracked by play nonce can be seen in Figure 4.18 below.

id	session_id	objective_id	user_id	progress	play_nonce <small>A Value identifying the play attempt</small>	last_updated
14	32	10	1	0.6	a846493ca5285e64bd2f76a6c83510c8fcc41e2e9eefaaf089...	2022-08-11 11:09:47
15	32	10	1	0.7	a846493ca5285e64bd2f76a6c83510c8fcc41e2e9eefaaf089...	2022-08-11 11:09:47
16	32	10	1	0.8	a846493ca5285e64bd2f76a6c83510c8fcc41e2e9eefaaf089...	2022-08-11 11:09:47
17	32	10	1	0.1	d38a6c5ec821c73dc4bf624e99c483cea25cc3c3d0c5ee7c4b...	2022-08-11 11:44:28
18	32	10	1	0.2	d38a6c5ec821c73dc4bf624e99c483cea25cc3c3d0c5ee7c4b...	2022-08-11 11:44:30
19	32	10	1	0.3	d38a6c5ec821c73dc4bf624e99c483cea25cc3c3d0c5ee7c4b...	2022-08-11 11:44:31

Figure 4.18 - Database schema for tracking game objective progress data

Using the recorded progress data chart visualization are generated and displayed to the user on-demand. The charts are interactive, so users can zoom or adjust the viewport to view more or less details. In addition to the graph visualizations, relevant breakdown of the data is also shown in tabular form below the graphs. An example from the objective progress report type is shown in Figure 4.19 below.

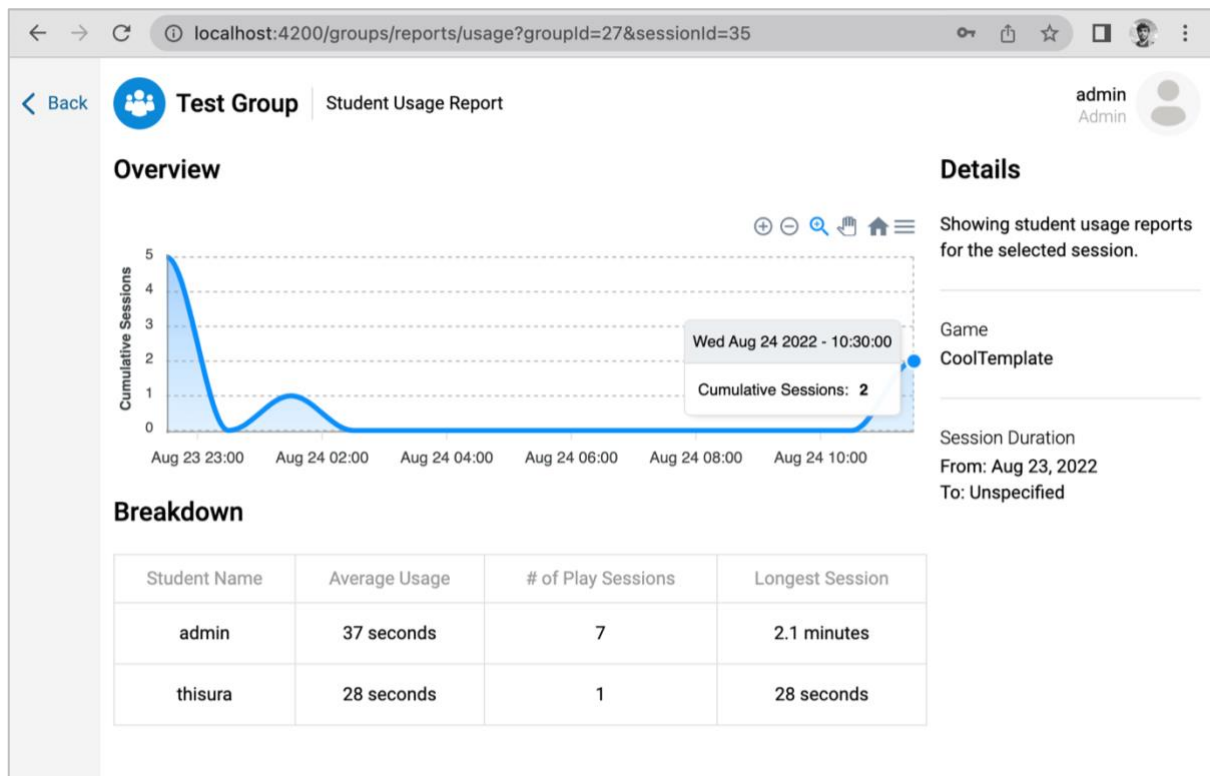


Figure 4.19 - User objective report with graph and tabular presentation

Every report type requires a different type of processing to generate the graph and tabular presentations. Some reports need to process a data using database queries prior to being retrieved. For example, the graph data for the user objective progress report processes data on the server code. An (altered) version of the code for the aforementioned example is shown in Figure 4.20 below.

```

export function processObjectivesByTime(input: GameSessionUserObjective[]){
  const yAxes = 'Objective Points';
  const xAxes = 'Seconds';
  const options = { zone: 'UTC', setZone: true }; // luxon: our timestamps are already in +0530
  let data = new ReportGraphDataUserObjectiveProgressByTime([], [], xAxes, yAxes);

  if (input.length == 0)
    return Promise.resolve(data);

  const firstSessionTime = DateTime.fromISO(isofy(input[0].last_updated), options).toMillis();
  let lastSessionTime = firstSessionTime;
  let interval = 0;
  let lastProgress: { [key: string]: number } = {};
  let totals: Map<number, number> = new Map();

  if (input.length > 1){
    lastSessionTime = DateTime.fromISO(
      isofy(input[input.length - 1].last_updated), options).toMillis();
  }

  // Determine interval
  const quantization = determineTimeQuantizationInterval(firstSessionTime, lastSessionTime)
  interval = quantization.interval;
  data.xAxesLabel = quantization.intervalName;

  for (let entry of input){
    const key = `${entry.user_id}-${entry.objective_id}`;
    const time = DateTime.fromISO(isofy(entry.last_updated), options).toMillis();
    const qt = roundedDateToIntervalMS(time, interval);
    const lp = lastProgress[key];

    let newProgress = (lp == null) ? entry.progress : (entry.progress - lp);
    newProgress = round(newProgress);

    const t = totals.get(qt);
    totals.set(qt, t == undefined ? newProgress : (t! + newProgress));
    lastProgress[key] = entry.progress;
  }

  // Convert the totalMap into readable format
  let lastTotalForQuantizedTime = 0;
  for (const [qTime, total] of totals){
    const roundedTotal = round(total);
    const targetTotal = round(total + lastTotalForQuantizedTime);
    data.labels.push(qTime);
    data.data.push(targetTotal);
    lastTotalForQuantizedTime = data.data[data.data.length - 1];
  }

  return Promise.resolve(data);
}

```

Figure 4.20 - Backend code for processing user objective progress graph data

Chapter 5 - Testing and Evaluation

The Game-based E-Learning (EDGE) system is a complex application consisting of a broad set of requirements and functionalities. Adequately testing the system was a challenge due to this complexity.

This chapter explains the strategies followed to effectively test the system from different perspectives. Some of the important test cases are document as well. In addition, the system is critically evaluated against the list of promised features form the project proposal. Finally, the chapter documents the results of the user testing process.

5.1. Test Documentation

All tests are documented using the Atlassian Jira issue tracking software (Atlassian Corporation Plc, n.d.). A custom Jira scheme was created with three issue types shown in Figure 5.1 below.













Issue Type	Description	Workflow	Field configuration	Screen
 Bug	A problem or error.	 Bug Workflow	 Default Field Configuration	 ES: Software Development Bug Screen Scheme
 Epic	A big user story that needs to be broken down. Created by Jira Software - do not edit or delete.	 Testing Workflow 2	 Default Field Configuration	 ES: Software Development Default Screen Scheme
 Test		 Testing Workflow 2	 Default Field Configuration	 ES: Software Development Default Screen Scheme

Figure 5.1 - Jira issue types used to document tests

‘Epic’ issue types define goals for different types of testing (e.g., exploratory testing, API testing, etc.). Epics allow for easy management of ‘Test’ issues, which are the actual test documents.

5.1.1. ‘Test’ Issue Types

A ‘Test’ issue can have the following fields; title, body, link to the Jira epic and links to the bug tickets. The field descriptions are provided below.

1. Test title

- This is a required field.
- Directly reference a requirement (e.g., ‘EFR-1 Home page screen’) or,
- Have a general name (e.g., ‘Game List: API response is valid’).

2. Test body

- This is a required field.
- Describe what needs to be tested and expected results. This may be in a format appropriate for each test type.

3. Link to the test 'Epic'
 - This is a required field.
4. Linked issues for any bugs discovered.
 - This is an optional field (because bugs may not be discovered during testing).

5.1.2. 'Bug' Issue Types

The 'Bug' issue types are managed independently of the tests so their progress can be tracked properly. Bug issues have the following fields; title, description, link to the Jira test issue and optional attachments. Descriptions of these fields are provided below.

1. Bug Title
 - This is a required field.
 - The title should describe the defect clearly.
2. Bug Description
 - This is a required field.
 - Pre-requisites, steps to recreate and/or notes can be mentioned.
3. Link to original Test
 - This is a required field.
 - All bugs must be referenced to the test which helped discover it.
4. Attachments
 - This is an optional field.
 - Screenshots and/or screen recordings showing the error.

5.1.3. Jira Workflows

‘Test’ and ‘Bug’ issue types each have their own Jira workflows associated with them. This prevents Jira issues from being moved to irrational states (especially if there was a large team managing the project). Figure 5.2 shows the workflow for the ‘Test’ issue type.

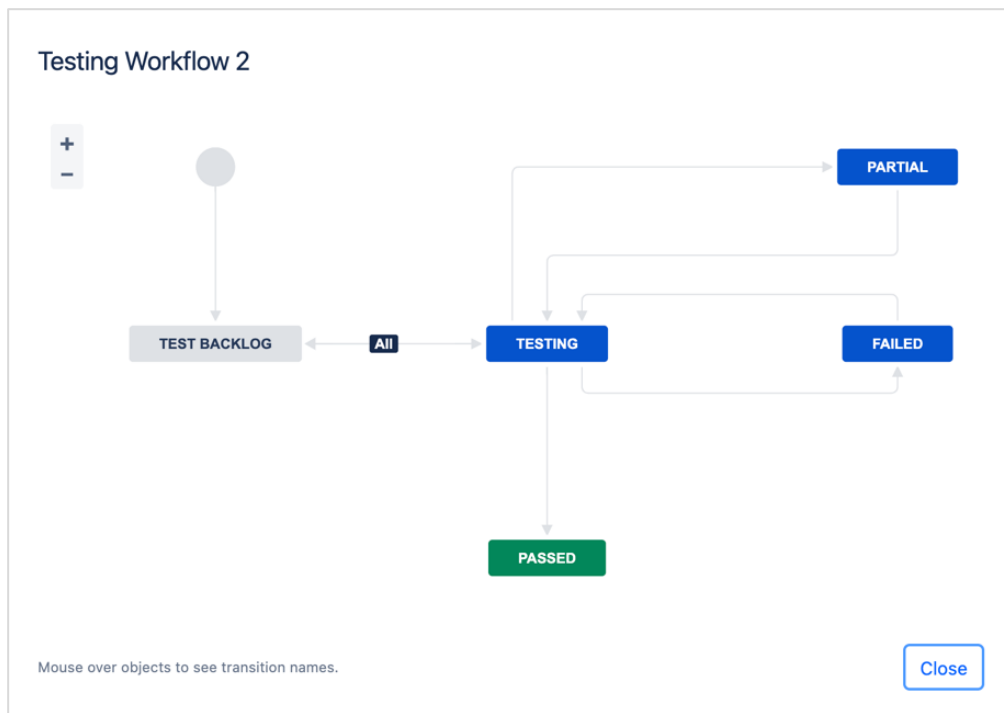


Figure 5.2 - Jira workflow for Test issue types

Bug issues are managed independently of the test document and thus have their own Jira workflow. Figure 5.3 below shows the Jira workflow for ‘Bug’ type issues.

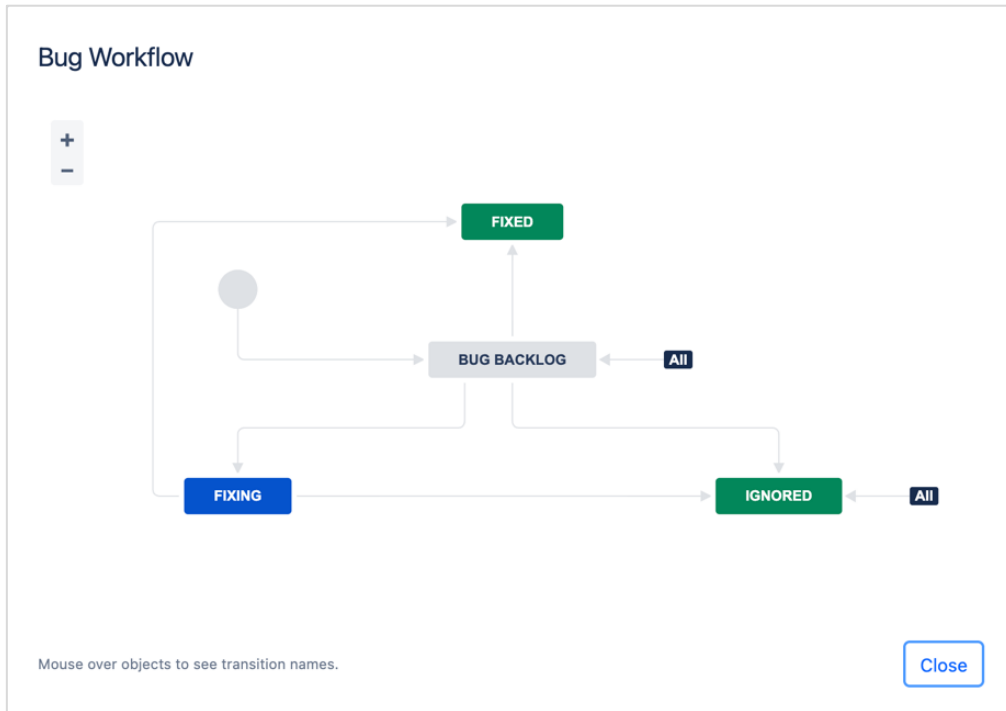


Figure 5.3 - Jira workflow for Bug issue types

5.1.4. Jira Workflow Scheme

Both test and bug issue workflows are grouped in a Jira workflow scheme as shown below in Figure 5.4 . Workflow schemes can be extended in the future to support more issue types.

Projects / EDGE System / Project settings

Workflows

EDGE System Testing Scheme

Add Workflow ▾ Switch Scheme

Workflow	Issue Types	Actions
Testing Workflow 2 (View as text / diagram)	<ul style="list-style-type: none"> Epic Test (Assign) 	
Bug Workflow (View as text / diagram)	<ul style="list-style-type: none"> Bug (Assign) 	

Figure 5.4 - Jira project workflow scheme

5.2. Testing Strategy

The testing strategy contains a mix of automated and manual testing. This approach was used since provides a balance between the cost of automation, and test coverage of the system

5.2.1. Requirements Verification

The requirements of the system are captured according to the main areas of the EDGE system. The verification process will assess the delivery of requirements using the process shown in Figure 5.5

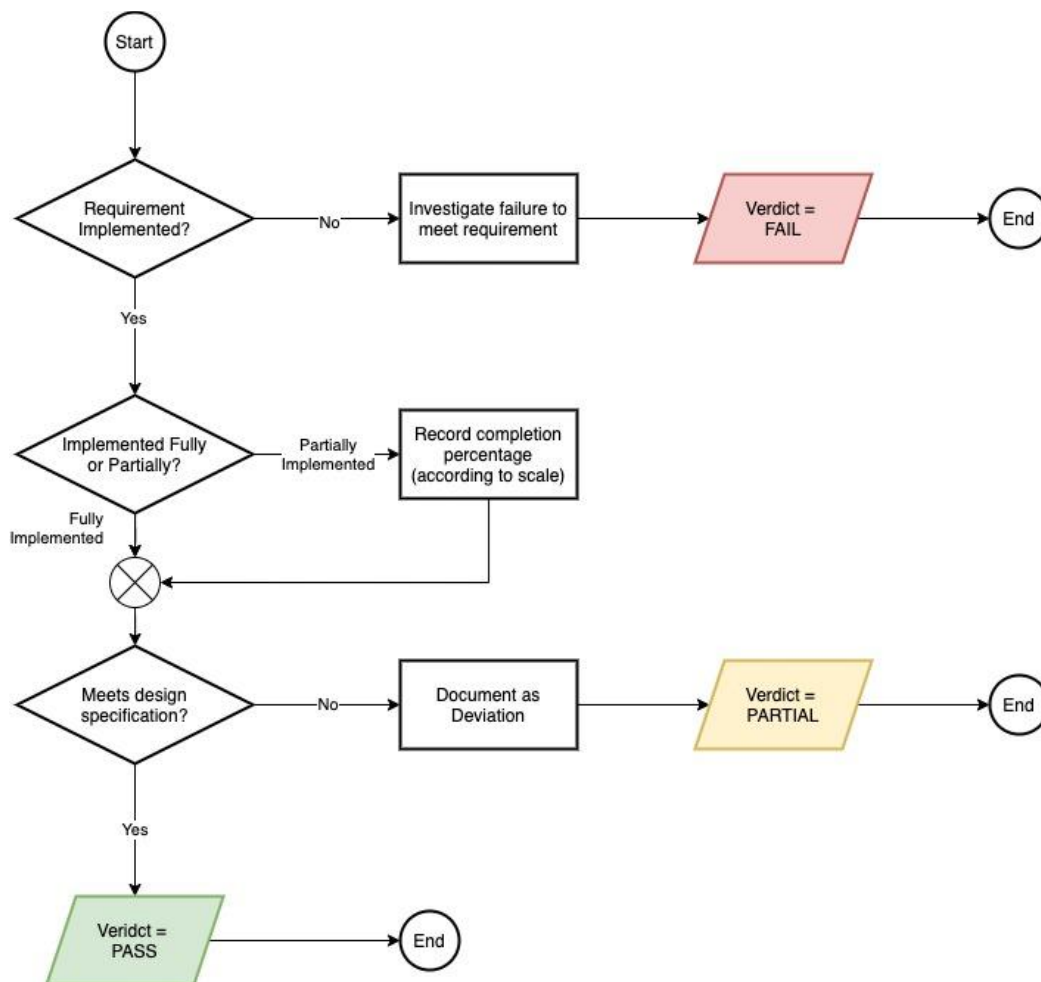


Figure 5.5 - Requirements verification process diagram

5.2.2. Development Testing

Since the system was developed using the FDD agile methodology there was always a logical feature that can be tested at each development increment. Each feature contains frontend and backend code that must be tested separately. Figure 5.6 summarizes the approach used for testing each feature increment.

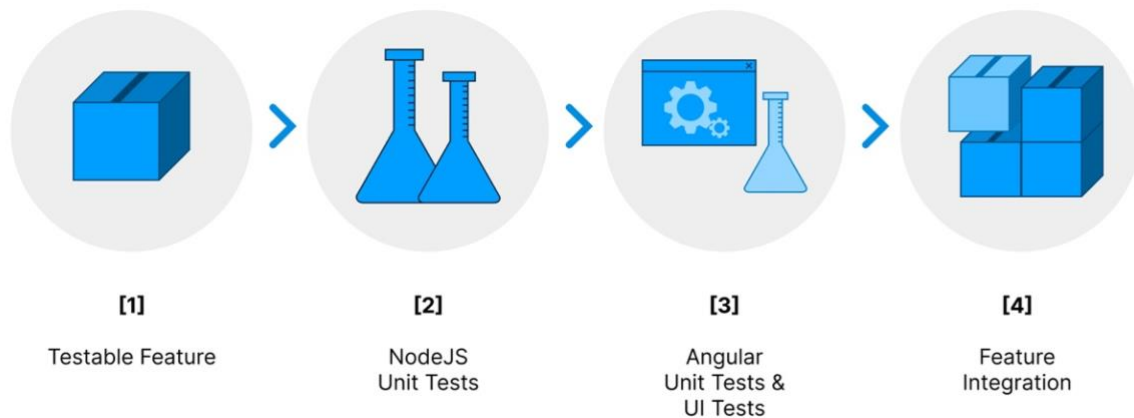


Figure 5.6 - Development testing process

Details of the process summarized above (Figure 5.6) are as follows.

1. **Testable Feature** – A feature with testable functionality is developed.
2. **NodeJS Tests** – Server logic eligible for testing is identified (for example, calculation logic for the performance reports). Test plans are created, executed and identified defects are fixed. Finally, manual inspection of the API may take place using Postman.
3. **Angular Tests** – The Angular JS code is examined and the user flows that must be tested (as well as any dependencies that must be mocked) are identified. Unit tests are written to exercise the UI and any other Angular code related to the feature. The tests are executed and identified defects are fixed.
4. **Feature Integration** – After a final inspection the feature is considered integrated.

5.2.3. User Testing

Initially, user testing was to be carried out at key development milestones with real users. However, due to development tasks rising in priority user testing was scheduled for much later in the development life cycle.

A group of five volunteers were rounded up. All users were assigned roles in the system (which represents an imaginary school classroom use case). Each user received a chance to evaluate the system during a scheduled online video conference. Feedback during the online session was collected, followed by a post-evaluation online survey.

5.2.3.1. Limitations in Volunteer Selection

The final list of volunteers does not reflect the actual target user base of the system. For example, it was not possible to round up students studying between school grades four and six.

Further, the system was not yet available as a hosted web application. Therefore, volunteers were provided with a remote desktop session into the local development environment for evaluation. Presenting the system for real users with a hosted version of the system was planned for a beta-testing phase, but this did not materialize due to various reasons.

5.2.3.2. Volunteer Selection Criteria

The five volunteers were round up according to the following criterion.

- I. A volunteer must,
 1. Be able to communicate in English.
 2. Have general IT literacy.
 3. Have access to a computer with an internet connection.
 4. Have a suitable computer setup for video conferencing.
 5. Be able to attend the video conference scheduled for system evaluation on a pre-determined date.
 6. Consent to having the entire testing session recorded.

- II. A volunteer must be able to commit,
 - 1. 2 hours per week for the pre-planning session.
 - 2. 4 hours per week for the evaluation session.
 - 3. 1 hour per week for the post evaluation survey.

5.2.4. Automated Testing Tools

The Jasmine Framework (Jasmine Project, n.d.) was used for writing automation test scripts. These scripts were written in the same language as the rest of the system. Technical issues of automated testing were greatly minimized due to the use of a common programming language throughout.

The test scripts are placed in the same directory structure as the sub-project it belongs to. For example, Figure 5.7 highlights two such scripts which test the service classes in the Angular sub-project.

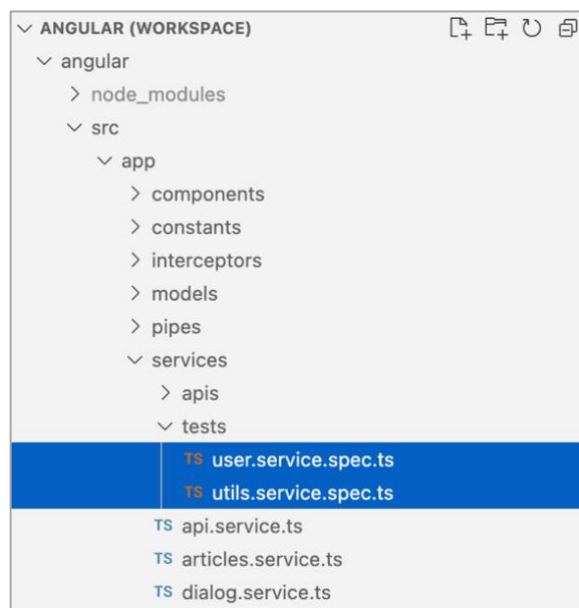


Figure 5.7 - Jasmine test scripts in the Angular sub project

The complete list of tools / software used for automated testing is list below in Table 5.1.

Category	Software
Test Framework	Jasmine Test Framework (Jasmine Project, n.d.)
Test Scripting IDE	Visual Studio Code (Microsoft Corporation, n.d.)
UI Test Execution	Google Chrome (Google LLC, n.d.)
Unit & Integration Test Execution	System Console

Table 5.1 - Automated testing tools

5.2.5. Manual Testing Methods

Manual testing was carried out in forms the forms of user acceptance testing, API testing and exploratory testing. These tests were performed at different points in the project life cycle as explained below.

- **User testing** – Performed after the system had most of its functionality implemented.
- **API testing** – Performed at the last stage of every NodeJS unit test.
- **Exploratory testing** – Performed at random intervals during the project life cycle.

It should be noted that the goal of exploratory testing was to ensure continuous development hasn't introduced regression issues. These tests also ensured that features were observed from a fresh perspective, rather than just after they were developed (during developer testing). Table 5.2 lists the software used for the above manual testing methods.

Category	Software
Volunteer test administration	Google Calendar (Google LLC, n.d.)
	Google Meet Video Conferencing (Google LLC, n.d.)
	AnyDesk Remote Desktop Application (AnyDesk Software GmbH, n.d.)
API testing	Postman API Platform (Postman, Inc., n.d.)
Exploratory testing	Google Chrome Browser (Google LLC, n.d.)

Table 5.2 - Manual testing software

5.3. Test Cases

This section lists a few of the test cases from this project. The test cases were selected to represent different types of manual and automated tests carried out, on both frontend and backend components. Relevant screenshots are attached to showcase the test documents and test results.

5.3.1. Exploratory Testing

5.3.1.1. Testing the Game Canvas

The game canvas is the area of the game editor in which teachers and game creators design levels (Figure 5.8). There are several requirements for the game canvas, and the first is to ensure that the user can manipulate objects in the game canvas (EFR-1, Table C.1).

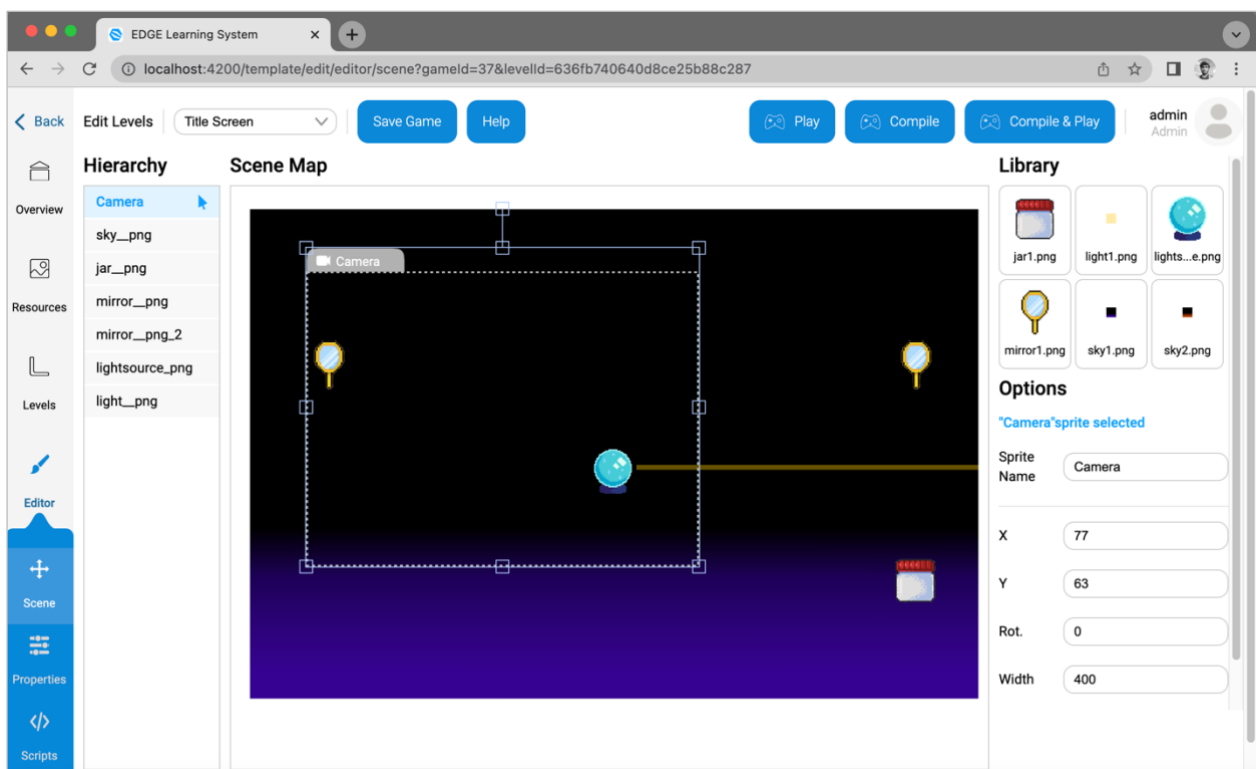


Figure 5.8 - Game editor page of the EDGE system

The exploratory test plan for this requirement interacted with the game canvas as a typical user with three key behavior expectations. These behaviors are: uninterrupted usage, window resizing and retaining performance. The test partially passed, with only one expectation being met properly.

Three defects were identified during the exploratory test. First, the game editor UI is not responsive to window resizing events. A browser refresh is required if the user needs to resize page. Secondly, repeated changes to the camera bounding box will eventually result in it being unresponsive. Finally, a defect exists in the selected item highlighting of the game editor scene hierarchy. The second and third issues does not occur instantly, but only after continuous use of the game editor. The Jira tickets for the test case and logged bugs are shown in Figure 5.9.

Projects / [EDGE System](#) / [ES-1](#) / [ES-8](#)

EFR-1 - Game canvas operates as expected

[Attach](#) [Link issue](#) [...](#)

Description

Expectation	Status
Canvas reacts to window resizing events	Failed
Canvas doesn't stop responding within the duration of testing	Partial
Canvas performance is retained as new objects are added	Pass

Linked issues [+](#)

is blocked by

- [ES-10](#) Game canvas doesn't react to window resize events [BUG BACKLOG](#)
- [ES-11](#) The 'camera' bounding box in the Game Canvas becomes un... [BUG BACKLOG](#)
- [ES-12](#) Editor Hierarchy not selecting items on the canvas [BUG BACKLOG](#)

Activity

Show: [All](#) [Comments](#) [History](#) [Work log](#) Newest first [↓](#)

Details [^](#)

Assignee
Thisura Dodangoda

Reporter
Thisura Dodangoda

Labels
None

Time tracking
10m logged

Epic Link
[Exploratory Testing](#)

Priority
Medium

More fields [^](#)

Original estimate
0m

Components

Figure 5.9 - Jira ticket documenting a game canvas exploratory test

Problems discovered during the exploratory tests are recorded as 'bug' issues in the Jira project. As an example, Figure 5.10 shows the bug related to the camera bounding box. It is linked to the 'exploratory testing' epic and original test case. It states the pre-conditions required for the defect to occur, as well as the exact steps that can be used to reproduce the issue. The defect is reproduced seemingly at random and is clearly stated in the defect description.

The screenshot shows a Jira issue page for a bug titled "The 'camera' bounding box in the Game Canvas becomes unselectable". The breadcrumb trail is "Projects / EDGE System / ES-1 / ES-11". The issue has a "Bug Backlog" label and a "Details" sidebar on the right. The main content area includes sections for "Description", "Pre Conditions", "Steps to reproduce", "Expectation", "Observation", "Environment", and "Attachments (1)".

Description

Pre Conditions

1. Be logged in.
2. Have a game/template scene created.
3. Be in the Scene Editor for that game.

Steps to reproduce

1. Start moving objects around.
2. Move the canvas viewport by middle-clicking.
3. Move the camera around and scale it.
4. Repeat steps 1-2-3 in random order for about 3 times.
5. Try to select the camera bounding box.

Expectation

The camera mouse-target box aligns with the displayed camera bounding box after moving and/or scaling the camera.

Observation

Camera mouse-target box is offset from the displayed bounding box.

Environment

None

Attachments (1)

Name	Size	Date added
Screen Recording 2022-09-30 at 11_...mov	1.4 MB	30 Sep 2022 11:49am

Details sidebar:

- Assignee: Unassigned
- Reporter: Thisura Dodangoda
- Labels: None
- Epic Link: Exploratory Testing
- Priority: Medium
- Created: 14 hours ago
- Updated: 14 hours ago
- Configure

Figure 5.10 - Jira bug ticket related to the game canvas camera bounding box

5.3.2. Backend Unit Testing

NodeJS unit tests were written in the TypeScript language for the Jasmine test framework. They exercise logic contained in the code unit without the requirement for a frontend connection or other dependencies (such as a database connection or libraries). Since a significant portion of the backend code deals with data access, I have elaborated below a complex unit test dealing with group member data.

5.3.2.1. Testing Data Access

Unit testing backend data access proposed a unique challenge: it is not possible to test with user data. Jasmine test scripts can mock dependencies in order to test a particular code unit. However, in the case of DAO classes, providing mock database connections defeats the purpose. To resolve this problem, DAO tests needed to be run in isolation. The database initialization logic was built to respect a 'test mode' flag in support this requirement.

If the test mode is enabled, the system would connect to a test database instead of the production database. The test mode is turned off by default, but can be switched on by invoking the 'setTestMode' method at code level. The unit test helper code written to perform this action is shown in Figure 5.11 below. The test method call can be seen in line number eight.

```
1  import * as testDAO from '../src/model/dao/test';
2  import * as sql from '../src/util/connections/sql/sql_connection';
3  import * as pc from '../src/util/parseconfig';
4  import * as utils from '../src/util/utils';
5
6  export async function initializeTestDB(){
7      const config = pc.parseConfig('config.json');
8      utils.setTestMode(true);
9      sql.initialize(config);
10
11     const clearStatus = await testDAO.clearTestDatabase();
12     expect(clearStatus).toBeTrue();
13 }
```

Figure 5.11 - Backend unit test helper code

5.3.2.2. Testing Group Member Data Access

This unit test ensures that the associations between members of a group are correctly returned by the backend server. Figure 5.12 shows the frontend screen related to this backend test and the setup between databases.

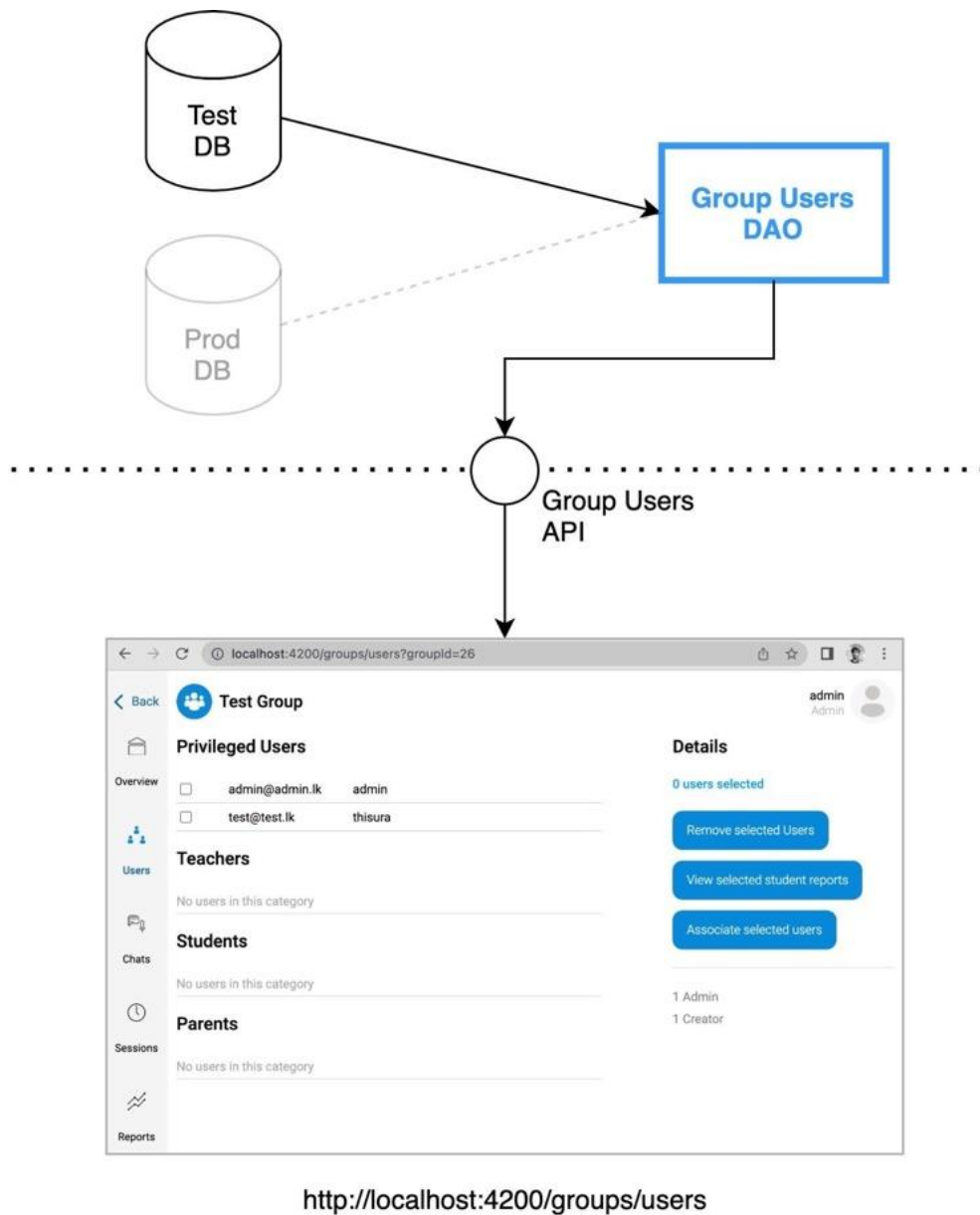


Figure 5.12 - Overview of the group members DAO test

Testing group member data access is complex since the system tracks user relationships¹. For example, a student member of a group can be associated with their parent. In the list displaying group members, the association between these two users must be correctly shown. An example of this is shown in Figure 5.13 below, where the row for Nithika shows relationship to their parents, and the parent Tissa shows their relationship to the student.

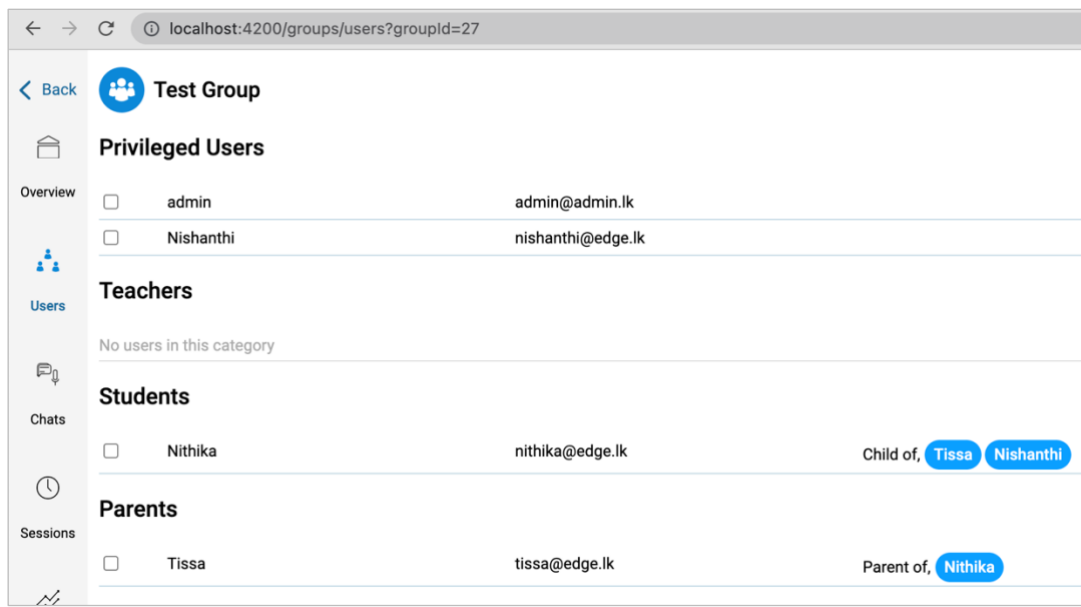


Figure 5.13 - Group members page showing user associations

User relationship tracking gets complicated when the number of users increase, and there are many relationships among users. Since its infeasible to test this manually, the group member test sets up two scenarios of group member combinations.

- **Scenario 1** – A group consisting of three users; a teacher, student and parent.
- **Scenario 2** – A group consisting of four users; a teacher, 2 students and a parent.

¹ The term ‘relationship’ is used interchangeably with ‘association’. The meaning of these terms are exactly the same within the context of this project.

Note that in scenario 2, additionally, the two students have the same parent (i.e., the students are siblings). The test script pre-calculates the number of associations for each association type, per each user. These values are asserted with the data returned from the group members DAO. Further, it also performs cross validation of user IDs returned for each relationship. For example, in the case of a parent-child relationship, it is possible to assert that the student's relationship returns the parent ID, and the parents relationship returns the student ID. The complete code for this unit test script can be found in Appendix D. The Jira ticket created for this unit test is shown below in Figure 5.14.

The screenshot shows a Jira ticket interface. The left pane contains the ticket details, and the right pane shows a summary of the test results.

Projects / ● EDGE System / ES-5 / ■ ES-20

Description
Test Scenario: Single Parent

Data:

```

1 privileged = 0
2 teachers = 1 (t1)
3 students = 2 (s1, s2)
4 parents = 1 (p1)
5
6 p1 and s1 are related as parent-child.

```

Test Cases:

Test #	Test Category	Expected Values
1	Composition	<ul style="list-style-type: none"> privileged = 0 teachers = 1 students = 2 parents = 1
2	Association Count	<ul style="list-style-type: none"> s1 should have only 1 association type s2 should have only 1 association type p1 should have only 1 association type p1 should have only 2 users in the first association type
3	Association Composition	<ul style="list-style-type: none"> Student 1 <ul style="list-style-type: none"> s1.associations[0].user[0].user_id must equal p1.user_id s1.associations[0].user[0].user_name must equal p1.user_name Student 2 <ul style="list-style-type: none"> s2.associations[0].user[0].user_id must equal p1.user_id s2.associations[0].user[0].user_name must equal p1.user_name Parent 1 <ul style="list-style-type: none"> p1.associations[0].user[0].user_id must equal s1.user_id p1.associations[0].user[0].user_name must equal s1.user_name p1.associations[0].user[1].user_id must equal s2.user_id p1.associations[0].user[1].user_name must equal s2.user_name

Details

Passed ▼

Assignee
 Unassigned
[Assign to me](#)

Reporter
 Thisura Dodangoda

Labels
 None

Epic Link
[Node JS Unit Tests](#)

Priority
 Medium

More fields Original estimate, Tim... ▼

Created last week ⚙️ Configure
 Updated last week

Figure 5.14 - Jira ticket related to the group members unit test

The initial round of testing the group members DAO did not succeed. A bug was discovered where the backend was returning user ID and email in the wrong fields, as shown in Figure 5.15 below. This was rectified, and the second round of unit testing was successful as shown in the updated Jira ticket shown in Figure 5.16 and unit test terminal output shown in Figure 5.17.

```
{
  "user_id": 1,
  "user_email": "admin",
  "user_name": "admin@admin.lk",
  "associations": []
},
{
  "user_id": 13,
  "user_email": "thisura",
  "user_name": "test@test.lk",
  "associations": []
}
```

Figure 5.15 - Group members DAO response highlighting incorrect fields

The screenshot shows the 'Linked issues' section of a Jira ticket. It is titled 'is blocked by' and contains three items:

- ES-21** Group Members DAO - user ID & email fields swapped (FIXED)
- ES-49** Groups Members DAO - Three Users Test (PASSED)
- ES-20** Groups Members DAO - Single Parent Test (PASSED)

Figure 5.16 - Final state of the Jira ticket for the group members test

```
ES-18: Group Members DAO tests
✓ ES-19 (1): Three Users - Composition
✓ ES-19 (2): Three Users - Association Names
✓ ES-19 (3): Three Users - Association Composition
✓ ES-20 (1): Single Parent - Composition
✓ ES-20 (2): Single Parent - Association Count
✓ ES-20 (3): Single Parent - Association Composition
```

Figure 5.17 - Terminal output of the group members unit test after correction

5.3.3. Angular UI Testing

All Angular projects come pre-configured with the Jasmine and Karma frameworks out of the box. Tests can be written using these frameworks to exercise logic in the project files or run automated tests. UI tests run in a special browser window and therefore can be used to simulate real user behavior.

5.3.3.1. Testing the Game Resources View

This page allows users to upload image and sound assets to the game project. Assets are organized into a grid with a preview icon. The UI test for this page presents three expectations: the page must load without issues for a project with no assets, existing assets must appear correctly and the upload process must work as expected.

Spy objects provided by the Jasmine framework were used to mock server responses with both empty and existing game resources, as well as other dependencies of the page. A snippet of the test setup code can be found in Figure 5.18 below.

```
await TestBed.configureTestingModule({

  declarations: [ ResourceUrlTransformPipe, GameEditResourcesComponent ],
  providers: [
    ResourceUrlTransformPipe,
    { provide: ActivatedRoute, useValue: activatedRoute },
    { provide: Router, useValue: jasmine.createSpyObj('Router', ['navigate']) },
    { provide: ApiService, useClass: MockAPIService },
    { provide: DialogService, dialogServiceSpy },
    { provide: MatDialog, useValue: jasmine.createSpyObj('MatDialog', ['open']) },
    { provide: ToastrService, useValue: toastrServiceSpy }
  ],
})
.compileComponents();
```

Figure 5.18 - Code snippet for setting up the Angular UI test for the game resources page

The Jira ticket for the game resources UI test was set up with the three expectations, and mapped to requirement ID EFR-7 (Table C.1) as shown in Figure 5.19. The Karma framework carries out test execution and the results are displayed in a separate browser tab. As shown in Figure 5.20, the UI test for this page has passed without errors.

Projects / ● EDGE System / ES-6 / ■ ES-22

EFR-7 - Add image and sound assets to Game Resources

📎 Attach
🔗 Link issue
⌵
⋮

Description

Test #	Expectation	Result
1	Page must load without errors	Pass
2	If API provides 2 image and 3 sound assets, page should display them accordingly	Pass
3	When the user adds a file to upload, the upload path should execute and finally a toast message should be displayed	Pass

Figure 5.19 - Jira ticket for the game resources UI test

Chrome is being controlled by automated test software.

Karma v 6.3.2 - connected; test: complete; DEBUG

Chrome 105.0.0.0 (Mac OS 10.15.7) is idle

Jasmine 3.7.1 Options

3 specs, 0 failures finished in 0.434s

ES-22: GameEditResourcesComponent

- ES-22 (1): should create
- ES-22 (2): should load 2 images and 3 sounds
- ES-22 (3): should upload data

Figure 5.20 - Karma UI test results for the UI tests

5.4. Test Result Summary

Continuous testing was done throughout the development process in various methods. While most tests passed successfully, there were instances where some issues were not fixed due to schedule constraints.

Due to the complexity of the system, it is not feasible to document all test cases and their outcomes in this document. However, by factoring in all types of tests as well as their current statuses I have created a test summary figure. This figure is shown in Table 5.3 below. The overall pass rate is **75%**. The pass rate for each test epic is also documented.

Test Epic	No. of tests	Passed test count	No. of bugs	Fixed bug count	Pass rate
Exploratory Testing	12	2	15	4	17%
Manual Testing	10	10	0	0	100%
Node JS Unit Tests	15	12	20	16	80%
Angular Unit Tests	12	10	18	12	84%
Angular UI Tests	15	14	8	7	94%
Total	64	48	61	49	75%

Table 5.3 - Test result summary table

5.5. Requirements Verification

All requirements are verified against the requirement specification documentation according to the process mentioned under the ‘Testing Strategy’ section.

5.5.1. Requirements Verification Summary

Table 5.4 shows the high-level summary of requirements verification. A total of 76 requirements were planned to be developed in the system. The final percentage of the system completion (in terms of requirement implementation) is 80%.

Feature	Partial Requirements	Failed Requirements	Fully Implemented Requirements	Total Requirements
Game Editor	1	0	16	17
Game Play	1	1	10	12
Game Session	1	3	1	5
Groups	3	1	8	12
Performance	2	0	9	11
Home Page	0	0	4	4
Dashboard	1	2	12	15
Total	9	7	60	76
%	11.8 %	9.2 %	80 %	

Table 5.4 - Requirements verification summary table

5.5.2. Requirement Verification Breakdown

Table 5.5 – Table 5.10 document the requirements which have not been fully implemented across different system features. Their completion is measured on a scale of 1 through 5, where a measure of 1 represents ‘poorly implemented’ and 5 represents ‘implemented to specification’. Any deviations or comments are documented.

5.5.2.1. Game Editor Requirements

#	Requirement	Verdict	Completion	Deviations / Comments
2	Add behavior to objects	Partial	3	Behavior can only be added through code

Table 5.5 - Unfulfilled game editor requirements

5.5.2.2. Gameplay Requirements

#	Requirement	Verdict	Completion	Deviations / Comments
19	Get notified about network errors as soon as they occur	Fail	0	Network error not handled properly
25	Prompt user with hints to progress in the game, if they are struggling	Partial	2	Guidance triggers are tracked, but UI for showing hints is not fully implemented

Table 5.6 - Unfulfilled game play requirements

5.5.2.3. Game session Requirements

#	Requirement	Verdict	Completion	Deviations / Comments
30	Schedule sessions for a group	Partial	2	The page is present in the frontend, but the rest of its UI is not implemented. Backend support is available.
31	When scheduling a session, specify the game, time and users	Fail	0	-
32	Add and remove users to/from a session that is already scheduled	Fail	0	-
33	Get an email notification when a game is scheduled	Fail	0	SMTP Email service is not implemented

Table 5.7 - Unfulfilled game session requirements

5.5.2.4. Groups Requirements

#	Requirement	Verdict	Completion	Deviations / Comments
40	Only users of the group must have access to its views & sessions	Partial	3	Some views do not validate user membership.
45	Delete groups they are members of	Partial	2	APIs are available to delete a group, but the consequence of deleting a group is not handled properly (e.g., removing user sessions).
46	Associate student and parents	Fail	0	The backend support for this requirement is present, but it is not implemented in the frontend.
47	Remove users in a group	Partial	1	Frontend implements the necessary controls, but the functionality is not present.

Table 5.8 - Unfulfilled groups feature requirements

5.5.2.5. Performance Reports Requirements

#	Requirement	Verdict	Completion	Deviations / Comments
52	If a game was time based, view report on how well the user faired against the clock	Partial	1	UI is partially implemented, but report is not implemented.
53	If a game was score based, view report on the overall user scores	Partial	1	UI is partially implemented, but report is not implemented.

Table 5.9 - Unfulfilled performance report requirements

5.5.2.6. Dashboard Requirements

#	Requirement	Verdict	Completion	Deviations / Comments
62	View the latest 5 new chat notifications across all groups where user is member of	Fail	0	Chat notification schema is not present, but the UI is implemented
68	Preview an existing game	Fail	0	Preview icon is present but its action is not implemented
72	Know the number of members in each group at a glance	Partial	3	UI is present. Backend simply needs to return the member count.

Table 5.10 - Unfulfilled dashboard feature requirements

5.6. User Testing

Since the system was still only available through the development computer, the only feasible method to provide users with a version of the system was through a remote desktop software. With this limitation in mind and a list of volunteers finalized, the plan for volunteer evaluation was as shown below in

Stage	Action Items
Setup	<ul style="list-style-type: none"> • Create and configure new databases in the EDGE system for testing. • Clear any test project / resource files from the EDGE file system. • Setup a mobile device for video conferencing. • Setup development environment with remote desktop software.
Pre-planning	<ul style="list-style-type: none"> • Finalize a date and time for hosting a three-hour evaluation session. • Assist all volunteers set up the remote desktop client software. • Host a pre-planning meeting to summarize the evaluation process and pre-assign roles in the system.
Evaluation	<ul style="list-style-type: none"> • Host online video conference and start remote desktop session. • Let each volunteer evaluate the system while providing support for any queries by the volunteer. Document any important points. • After all volunteers have evaluated the system discuss findings. • Wrap up session.
Survey	<ul style="list-style-type: none"> • Send out the online link to re-watch the recording of the user evaluation process. • Send out an online survey form to collect feedback.
Feedback compiling	<ul style="list-style-type: none"> • Send out a ‘contribution acknowledgement’ mail to volunteers. • Collect responses of the online surveys and combine with documentation from the evaluation session itself, to produce a compiled list of findings.

Table 5.11 - User testing action plan

5.6.1. Finalized User Roles

The five volunteers selected were assigned five different roles in the EDGE system to represent a hypothetical school classroom setup. The volunteer's names will remain anonymous and instead be referred to as volunteers A through E. The roles are assigned as shown in Table 5.12 below.

Volunteer	Role	Association
A	Teacher	-
B	Student 1	Child of Parent 1
C	Student 2	Child of Parent 2
D	Parent 1	Parent of Student 1
E	Parent 2	Parent of Student 2

Table 5.12 - User testing volunteer user roles

5.6.2. Informal User Feedback

After users evaluated the system for the first time, a discussion was initiated to understand their impression and collect feedback regarding the EDGE system. This discussion was informal and precedes the online survey that would take place afterwards. The collected user feedback is documented below in Table 5.13.

User	Impression	Informal Feedback
Volunteer A	Refreshing	<p>“I like the interface of the system. It is user friendly and neatly organized.</p> <p>After registration I was taken to a blank screen so it took a few seconds to realize what I should do next. After that however the system became intuitive to use.</p> <p>I am impressed by the ability to literally edit a game inside of my web browser. There was only one template to try out but I believe more can be added in the future.</p> <p>Since it wasn’t possible to schedule a session, I used the session automatically created by the system for my game and created a group around it.</p> <p>Adding users was as simple as sharing a Zoom meeting link. It’s really nice.</p> <p>It was a little confusing that in order to access the session chat, I needed to play a game but didn’t need to actually do any actions in the game. Maybe these can be separated.</p> <p>Finally, the reports were not extremely detailed but I can imagine how it can be improved further. I’m not sure if a non-techy teacher can feel at home with this system, but it’s a really refreshing attempt at teaching in schools.”</p>
Volunteer B	Interesting	<p>“Students should definitely get an email or some form of notification when a session is scheduled. Not having this is a major drawback.</p> <p>Once I started the game it just feels like any generic mini game. The system’s full potential is not realized because the demo game is not very fun to play.</p> <p>The objectives and guidance systems were interesting to see. I noticed it updated when I made progress in the game. It also reflected on the overall session report.”</p>
Volunteer C	Needs improvement	<p>“The concept is very novel. I was excited to see what this new system was like, especially since I enjoy computer games. While progress tracking, chats and reports are nice I feel the actual gameplay aspect needs improvement. I also didn’t feel like I learned anything.”</p>

Volunteer D	Good	<p>“If I were a real parent and I could get insights into my child’s school work using online reports – I could take informed decisions on how I can help them.</p> <p>Today’s education systems really cut parents out of the equation. I feel like the EDGE project has a lot of potential.”</p>
Volunteer E	Okay	<p>“Parents don’t have a lot to do in this system other than view reports. I wish they could observe their child’s performance in-game. But other than that it looks like the system works as described”</p>

Table 5.13 - Informal feedback collected during user testing

5.6.3. Online Survey

The pool of users was not large enough to conduct a comprehensive online survey about the system. However, a survey was still used to quantify user feedback after the evaluation process. The questions contained in the online survey are shown in Table 5.14 below. All questions have a linear scale, where low and high values represent negative and positive responses respectively.

Category	#	Question
Impression and usability	1	Are you satisfied about the current methods of delivering knowledge in Sri Lankan Schools?
	2	Do you agree that an E-Learning system like 'EDGE' can address issues in learning in Sri Lanka?
	3	Do you think 'EDGE' is too complicated for the Sri Lankan education system?
	4	Do you think the current set of features address the needs of you, as a user?
Performance	5	Was the user interface of the system responsive?
	6	Are you satisfied with how the game window is displayed?
Features	7	Do you like how game progress is tracked during games?
	8	Do you like how the system guides users when they get stuck?
	9	Do you agree the generated reports convey useful information?
Future work	10	Finally, do you think the 'EDGE' system has room to improve?

Table 5.14 - User testing online survey questions

Figure 5.21 shows a preview of the Google Form created containing the online survey questions mentioned in Table 5.14 above. They survey can be accessed using the web link (Dodangoda, 2022)

EDGE System Post Evaluation Survey

Hello! Thank you for participating in the user evaluation for the EDGE Game-based E-Learning system as part of my Master's Thesis project.

This is the final step in the user evaluation of the EDGE system - the online survey. Kindly fill out this short survey about your experience with the system. It will help me quantify various aspects about the system.

[Sign in to Google](#) to save your progress . [Learn more](#)

*** Required**

Are you satisfied about the current methods of delivering knowledge in Sri Lankan Schools? *

1 2 3 4 5

Not Satisfied Highly Satisfied

Do you agree that an E-Learning system like 'EDGE' can address issues in learning in Sri Lanka? *

1 2 3 4 5

Highly Disagree Highly Agree

Do you think 'EDGE' is too complicated for the Sri Lankan education system? *

1 2 3 4 5

Very Complicated Very Intuitive

Figure 5.21 - post evaluation survey form preview

5.6.4. Survey Results

5.6.4.1. Impression in Context

Question 1 and 2 in the survey measure whether users believe the system’s use case is justified in the context of the Sri Lankan education system. The user’s responses to the survey are documented in Table 5.15 and graphically shown in Figure E.1. Users seem to agree that there indeed is a problem the method of learning at Sri Lankan schools, and that a system such as EDGE could address those problems.

#	Question	Low Scale	High Scale	Responses to each scale				
				1	2	3	4	5
1	Are you satisfied about the current methods of delivering knowledge in Sri Lankan Schools?	Not Satisfied	Highly Satisfied	4	1			
2	Do you agree that an E-Learning system like 'EDGE' can address issues in learning in Sri Lanka?	Highly Disagree	Highly Agree				2	3

Table 5.15 - Survey responses for system impression

5.6.4.2. Usability

Questions 3 & 4 address the user’s opinion about the system usability. While complexity is subjective, the purpose of the user testing process is to understand the user’s perspective and thus I believe it is a suitable question for this survey.

The responses shown in Table 5.16 and Figure E.2 are spread throughout the scale and don’t show a clear pattern with regards to system usability. It can be concluded that users feel conflicted regarding the systems usability – which may be reflecting of the unfinished nature of the project.

#	Question	Low Scale	High Scale	Responses to each scale				
				1	2	3	4	5
3	Do you think 'EDGE' is too complicated for the Sri Lankan education system?	Very Complicated	Very Intuitive	1	2	1	1	
4	Do you think the current set of features address the needs of you, as a user?	Not at all	Yes, it does!		1	1	1	2

Table 5.16 - Survey responses for system usability

5.6.4.3. Performance

Questions 5 and 6 represent the user's perception regarding the system performance in two unrelated questions. They are documented in Table 5.17 and Figure E.3. While users believe the system is highly responsive (even when evaluated through a remote desktop connection), they have reserved opinions regarding how the game window is presented.

The game player component uses a HTML canvas that may not render the game at smooth frame rate. Further, it could be that this was an effect perceived when viewed through the remote connection. As a conclusion regarding performance, while other parts of the system can be left as is, the game window's performance needs a tune up.

#	Question	Low Scale	High Scale	Responses to each scale				
				1	2	3	4	5
5	Was the user interface of the system responsive?	Very Sluggish	Very Responsive					5
6	Are you satisfied with how the game window is displayed?	Not Satisfied	Highly Satisfied		1	2	2	

Table 5.17 - Survey responses for system performance

5.6.4.4. Feature Set

Questions 7, 8 and 9 evaluate the user’s opinion regarding the features offered in the EDGE system. They are documented in Table 5.18 and Figure E.5. According to the responses users are generally satisfied with the objectives tracking system rather than the guiding system. This could be due to how its worded throughout the system.

Specifically, the term ‘guidance trigger’ is less understandable than the term ‘objective’. The performance reports feature has been perceived as very useful by some users and moderately useful by one user.

#	Question	Low Scale	High Scale	Responses to each scale				
				1	2	3	4	5
7	Do you like how game progress is tracked during games?	Not Satisfied	Highly Satisfied				4	1
8	Do you like how the system guides users when they get stuck?	Not Satisfied	Highly Satisfied			4	1	
9	Do you agree the generated reports convey useful information	Highly Disagree	Highly Agree			1	3	1

Table 5.18 - Survey responses for system feature set

5.6.4.5. Feature Expansion

The final question asks users whether they believe the system can be improved in the future (i.e., whether the system has potential to grow). This question is documented in Table 5.19 and Figure E.4. All users unanimously agree that the system can be improved. This could be due to various reasons but it is reassuring to know that users feel the project is deserving of future improvement.

#	Question	Low Scale	High Scale	Responses to each scale				
				1	2	3	4	5
10	Finally, do you think the 'EDGE' system has room to improve?	Highly Disagree	Highly Agree					5

Table 5.19 - Survey responses for future expansion

Chapter 6 - Conclusion

Referring back to the original objectives of the system, the goal was to create a hosted version of the system and demonstrate its applicability with real users. Whilst these objectives were not completely met, the resulting implementation of the system provides a sufficient solution to address the original problem: making game-based e-learning more accessible to the Sri Lankan education system.

6.1. Retrospective

The main driving factor and its most critical weakness is the scale of the project. Some of the problems encountered during development are:

- Initial difficulties in adopting agile methodologies to a one-person project.
- Underestimating the nuance complexity of the initially planned features.
- Poorly planned user testing sessions.
- Focusing effort on both planning ahead and development simultaneously.

However, it was possible to address some of these problems creatively and continue developing the system to its current state. Some of the lessons learnt are:

- Agile development allows new constraints to be discovered as the project is developed. It is important to incorporate these constraints into future planning instead of trying to push against them.
- Organizing code bugs with systems such as Jira improves developer productivity.
- Real volunteer test users should have been finalized at the start of the project instead of very late into development.
- Prototyping user interfaces and code with pen & paper puts problems into perspective. It is easier to think with the problem in front of you, rather than in your mind.
- Systems incorporating many different technologies is exciting to propose and start working on. However, it is important to always put the problem at the center of thinking and not the number of technologies used.

6.2. Future Work

Stating that this project was a challenge to develop is a severe understatement. After two years in development, it is still not complete. After the conclusion of the MIT individual thesis, I will continue to fix defects and work towards releasing a hosted version for testing with real users.

References

BERG, B. M., 2015. *UNPACKING DIGITAL GAME- BASED LEARNING The complexities of developing and using educational games*, Skövde: University of Skövde.

Kapuler, D., 2020. *50 Sites & Apps for K-12 Education Games*. [Online]

Available at: <https://www.techlearning.com/tl-advisor-blog/4684>

[Accessed 23 August 2020].

eduLanka, n.d. *Grade 6 School syllabus and Teacher Instruction Meterials*. [Online]

Available at: <https://www.edulanka.lk/syllabus/grade6>

Prodigy Education Inc, n.d. *Prodigy Education*. [Online]

Available at: <https://www.prodigygame.com/main-en/>

[Accessed 10 November 2020].

JumpStart, n.d. *Hyper Blast 2 HD – Math App for iPhone, iPad, iPod Touch, & Nook*.

[Online]

Available at: <https://www.jumpstart.com/mathblaster/mobile-apps/math-blaster-hyperblast-2>

Oodlü Ltd., n.d. *The science behind Oodlu*. [Online]

Available at: <https://oodlu.org/the-science-behind-oodlu>

Educandy, n.d. *Educandy – Making learning sweeter!*. [Online]

Available at: <https://www.educandy.com/>

[Accessed 10 November 2020].

Kahoot! Corporation, n.d. *What is Kahoot! | How to play Kahoot!*. [Online]

Available at: <https://kahoot.com/what-is-kahoot/>

Kahoot! Corporation, n.d. *Business pricing - Kahoot!*. [Online]

Available at: <https://kahoot.com/business/pricing/>

Dalsgaard, C., n.d. *Social software: E-learning beyond learning management systems*.

European Journal of Open, Distance and E-Learning.

Astutik, S. & Prahani, B. K., 2018. The Practicality and Effectiveness of Collaborative Creativity Learning (CCL) Model by Using PhET Simulation to Increase Students' Scientific Creativity. *International Journal of Instruction*, 11(4), pp. 409-424.

University of Colorado, n.d. *Interactive Simulations for Science and Math*. [Online]
Available at: <https://phet.colorado.edu/>
[Accessed 10 November 2020].

Anon., n.d. *Can I use... Support tables for HTML5, CSS3, etc.* [Online]
Available at: <https://caniuse.com/?search=HTML5>

Martínez-Monés, A., Reffay, C., Hoyos-Torío, J. E. & Muñoz-Cristóbal, J. A., 2017. *Learning Analytics with Google Classroom: Exploring the possibilities*. s.l., s.n.

Google, n.d. *Run an originality report on your work - Classroom Help*. [Online]
Available at: <https://bit.ly/2N5dmgv>

Cavus, N. & Zabadi, T., 2014. A Comparison of Open Source Learning Management Systems. *Procedia - Social and Behavioral Sciences*.

Anon., n.d. *Moodle in English: Disadvantages to using Moodle?*. [Online]
Available at: <https://moodle.org/mod/forum/discuss.php?d=231989>

Blender Foundation, n.d. *Node Editor Introduction — Blender Manual*. [Online]
Available at: <https://bit.ly/3aHaK0I>
[Accessed 10 November 2020].

BuildBox, n.d. *BuildBox*. [Online]
Available at: <https://www.buildbox.com/help/buildbox-3-manual/ad-monetization/>
[Accessed 10 November 2020].

Anon., n.d. *BuildBox Review*. [Online]
Available at: <https://www.slant.co/options/1098/~buildbox-review>

Anon., n.d. *An Honest Comparative Review Of BuildBix & Unity From A Complete Novice*. [Online]
Available at: <https://www.buildbox.com/forum/index.php?threads/an-honest-comparative-review-of-bb-unity-from-a-complete-novice.17162/>

Gorynych, M., n.d. *Meet ct.js - your new 2D game editor*. [Online]

Available at: <https://ctjs.rocks/>

Giera, J., 2014. *The Costs And Risks Of Open Source*, s.l.: FORRESTER.

Davoudian, A., Chen, L. & Liu, M., 2018. A survey on NoSQL stores. *ACM Computing Surveys*, Volume 51, pp. 1-43.

Redis.IO, n.d. Redis FAQ. <https://redis.io/topics/faq>.

LevelDB, n.d. *LevelDB Wiki*. [Online]

Available at: <https://en.wikipedia.org/wiki/LevelDB>

[Accessed 10 January 2021].

RocksDB, n.d. *RocksDB FAQ*. [Online]

Available at: <https://rocksdb.org/docs/support/faq>

[Accessed 10 January 2021].

LMDB, n.d. *LMDB Release Notes*. [Online]

Available at: <https://lmdb.readthedocs.io/en/release/>

[Accessed 10 January 2021].

StackOverflow, 2020. *Stack Overflow Developer Survey 2020*. [Online]

Available at: <https://bit.ly/3cND5F7>

[Accessed 10 January 2021].

LeFever, L., n.d. *The Art of Explanation*. s.l.:s.n.

Griss, M., 2000. *Implementing Product-Line Features with Component Reuse*. Vienna, s.n.

Figma Inc., n.d. *Figma user interface design tool*. [Online]

Available at: <https://www.figma.com/>

Nintendo Co., Ltd., n.d. *Super Mario Bros*. [Online]

Available at: <https://www.nintendo.co.uk/Games/NES/Super-Mario-Bros-803853.html>

Atlassian Corporation Plc, n.d. *Jira | Issue & Project Tracking Software*. [Online]

Available at: <https://www.atlassian.com/software/jira>

Jasmine Project, n.d. *Jasmine Test Framework*. [Online]

Available at: <https://jasmine.github.io/>

Microsoft Corporation, n.d. *Visual Studio Code Product Page*. [Online]

Available at: <https://code.visualstudio.com/>

Google LLC, n.d. *Google Chrome Browser*. [Online]

Available at: <https://www.google.com/chrome/>

Google LLC, n.d. *Google Calendar*. [Online]

Available at: <https://www.google.com/calendar/about/>

Google LLC, n.d. *Google Meet*. [Online]

Available at: <https://apps.google.com/meet/>

AnyDesk Software GmbH, n.d. *AnyDesk Remote Desktop Software*. [Online]

Available at: <https://anydesk.com/en/solutions/remote-desktop>

Postman, Inc., n.d. *Postman API Platform*. [Online]

Available at: <https://www.postman.com/>

Dodangoda, T., 2022. *EDGE System Post Evaluation Survey*. [Online]

Available at: <https://forms.gle/ZztjBwhRw8FRAEup7>

OpenJS Foundation, n.d. *GitHub page for the Multer file-handling middleware for Node JS*. [Online]

Available at: <https://github.com/expressjs/multer>

[Accessed 2021].

Zaytsev, J., Kienzle, S. & Bogazzi, A., n.d. *Fabric JS Home Page*. [Online]

Available at: <http://fabricjs.com/>

[Accessed 2021].

Appendix A – Prodigy System Teacher Reports

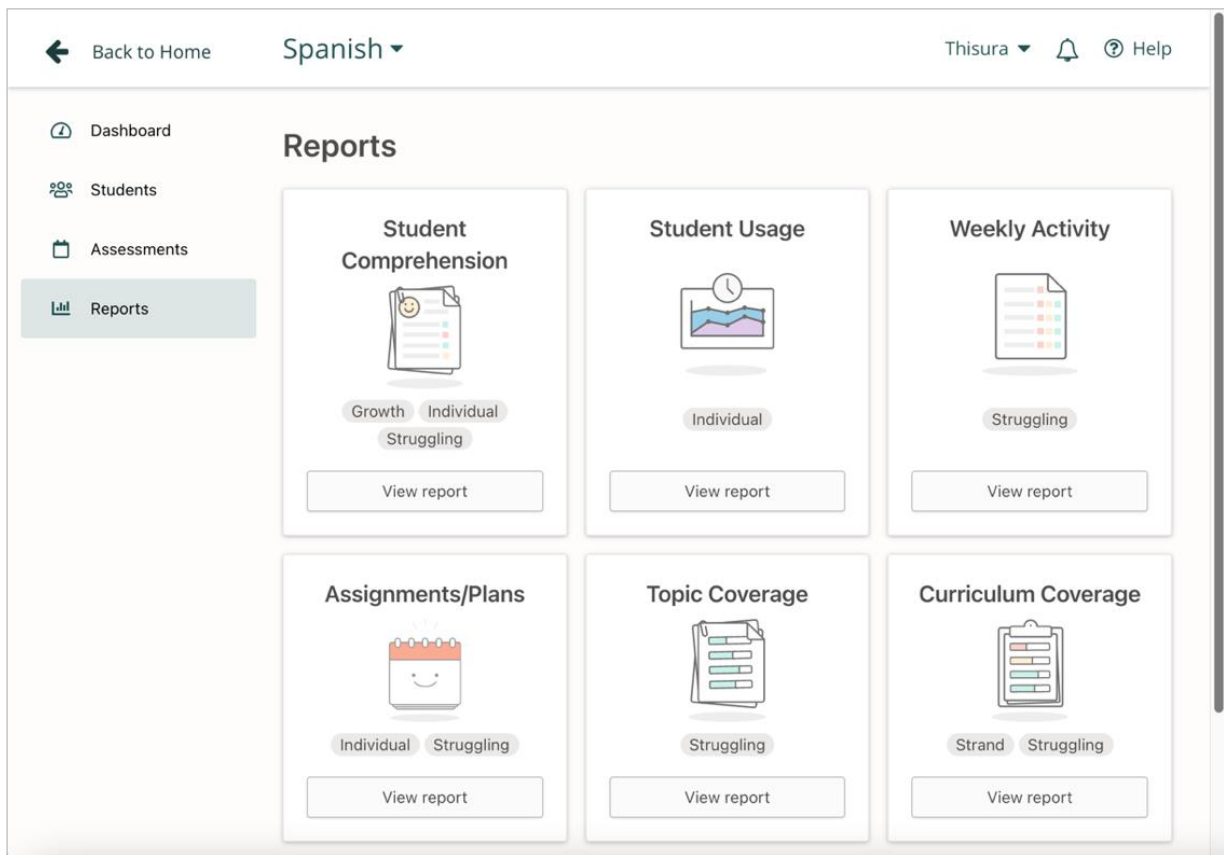


Figure A.1 - Prodigy mathematics game teacher reports

Appendix B – Oodlu Platform Student Analytics

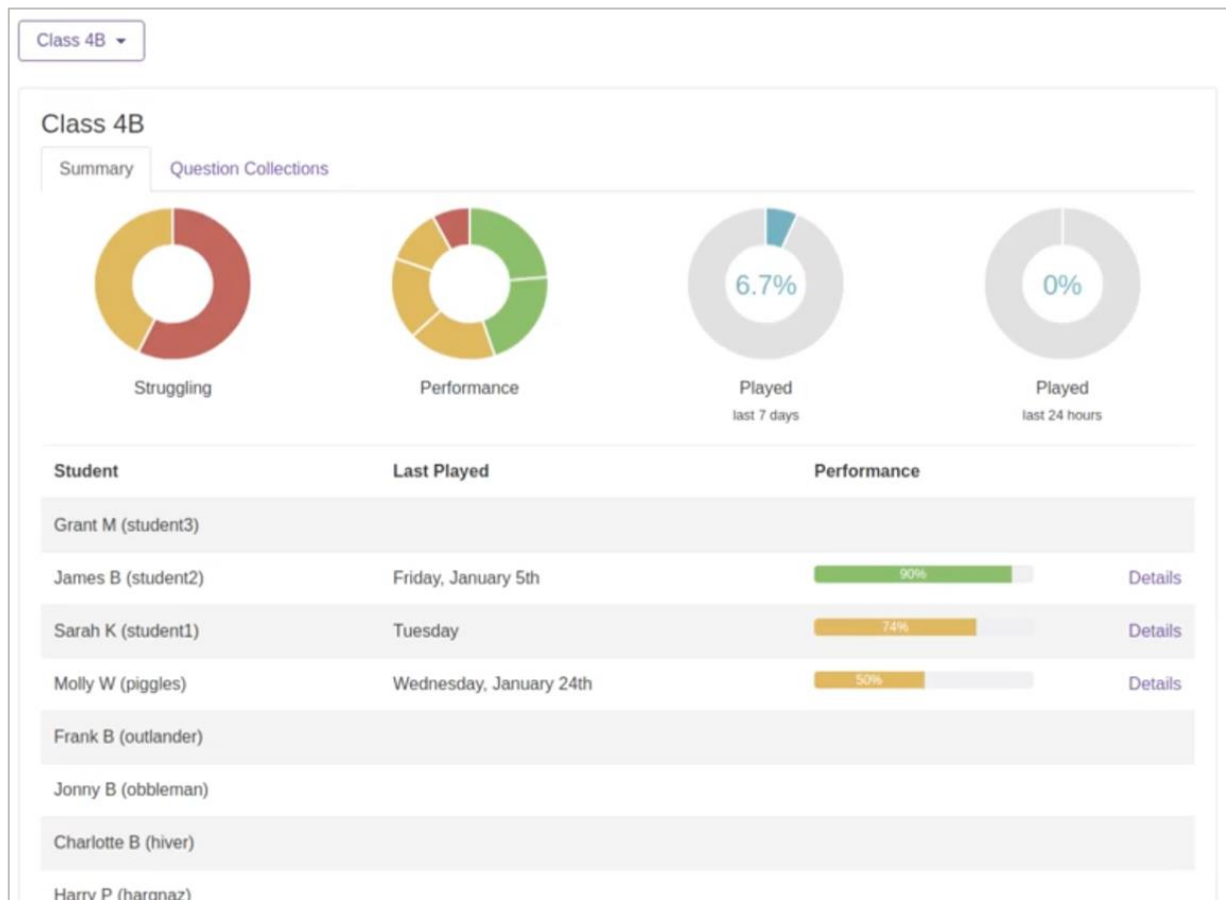


Figure B.1 - Oodlu platform analytics screen

Appendix C – System Requirement Specification

C.1. Game Editor Requirements

Area	User	Requirement	Req. ID
Level Editor	Game creator, Teacher	Manipulate objects in a level scene	EFR-1
		Add behavior to objects	EFR-2
		Customize a level using a friendly UI without coding	EFR-3
	Game creator	Define UI for customizing level behavior	EFR-4
		Set default values for level customizations	EFR-5
		Add behavior to objects via code	EFR-6
Asset Manager	Game creator, Teacher	Add image and sound assets to the game from user's device	EFR-7
		Replace existing assets in game templates	EFR-8
Level Mgmt.	Game creator, Teacher	Create levels	EFR-9
		Duplicate existing levels	EFR-10
		Delete levels	EFR-11
Game meta-data mgmt..	Game creator, Teacher	Set game title and description	EFR-12
		Select a template for a game	EFR-13
		Template must not be modifiable after creating games	EFR-14
		Set objectives for game	EFR-15
		Set guidance trigger points for the game	EFR-16
		Specify report types for a game	EFR-17

Table C.1 - Project game editor requirements

C.2. Game Play Requirements

Area	User	Requirement	Req. ID
Entire module	All except parents	Participate in game play sessions	EFR-18
		Get notified about network errors as soon as they occur	EFR-19
Game canvas	All except parents	View the game scene in acceptable visual fidelity	EFR-20
		Interact with game using a Keyboard and Mouse connected to the host device	EFR-21
		Pause game if user backgrounds the session	EFR-22
		Track user game progress and events for reports	EFR-23
Guidance system	All except parents	Record user's progress in the game for each session	EFR-24
		Prompt user with hints to progress in the game, if they are struggling	EFR-25
Objectives system	All except parents	Record user's objective completion in the game for each session	EFR-26
		Provide indicator about the progress of each objective in the game	EFR-27
Communication	All except parents	Facilitate a group chat for each game play session	EFR-28
		Group chat must persist across play sessions for each session	EFR-29

Table C.2 - Project game play requirements

C.3. Game Sessions Requirements

Area	User	Requirement	Req. ID
Entire Module	Game creator, Teacher	Schedule sessions for a group	EFR-30
		When scheduling a session, specify the game, time and users	EFR-31
		Add and remove users to/from a session that is already scheduled	EFR-32
		Get an email notification when a game is scheduled	EFR-33
		Participate in a game session	EFR-34
	Student	Participate in a game session	EFR-35
Get an email notification when a game is scheduled		EFR-36	

Table C.3 - Project game sessions requirements

C.4. Groups Requirements

Area	User	Requirement	Req. ID
Membership	All users	Join a group using an invite link	EFR-37
		When a logged out user joins via a group invite link, require them to login / registration first. Afterwards, they must be redirected to the group page.	EFR-38
		Leave a group the user was added to	EFR-39
		Only users of the group must have access to its views & sessions	EFR-40
Views	All users	View group summary	EFR-41
		View sessions in the group	EFR-42
		View users in the group	EFR-43
		View reports in a group	EFR-44
Management	Game creator, Teacher	Delete groups they are members of	EFR-45
		Create new groups	EFR-79
		Associate student and parents	EFR-46
		Remove users in a group	EFR-47
		Edit sessions in a group	EFR-48

Table C.4 - Project groups requirements

C.5. Performance Reports Requirements

Area	User	Requirement	Req. ID
Report Types	All Users	View report on how often a user interacted with the system.	EFR-49
		View report on how well the user completed the game objectives.	EFR-50
		View report on how and what guidance points (hints) were triggered.	EFR-51
		If a game was time based, view report on how well the user faired against the clock	EFR-52
		If a game was score based, view report on the overall user scores	EFR-53
Data Collection	All except parents	Collect performance metrics on users participating in game sessions	EFR-54
		Only collect metrics marked for collection in the game metadata	EFR-55
		Refrain from collecting or processing PII of users, other than names	EFR-56
Report Visualization	All Users	Reports should be made available for each session in a group	EFR-57
		Overall reports calculated for all users must be visible to all users	EFR-58
		Reports be visualized as graphs and tabular data where possible	EFR-59

Table C.5 - Project performance reports requirements

C.6. Dashboard Requirements

Area	User	Requirement	Req. ID
Summary	All Users	View list of scheduled ongoing sessions across all groups for the logged user	EFR-60
		View list of scheduled upcoming sessions across all groups for the logged user	EFR-61
		View the latest 5 new chat notifications across all groups where user is member of	EFR-62
		When clicked on any of the list items navigate to its corresponding view in the correct group	EFR-63
Games & templates	Game creators, Teacher	Ability to start creating a new game	EFR-64
		Search list of games	EFR-65
		Delete an existing game	EFR-66
		Edit an existing game	EFR-67
		Preview an existing game	EFR-68
		List games created by the user in a tabular form	EFR-69
Groups	Game creator, Teacher	Ability to start creating a new group	EFR-70
	All Users	List groups the user is member of in a tabular form	EFR-71
		Know the number of members in each group at a glance	EFR-72
		Copy the invite link to each group	EFR-73
	View the group	EFR-74	

Table C.6 - Project dashboard requirements

C.7. User Requirements

Area	User	Requirement	Req. ID
Login	All Users	Login using user only email & password	EFR-75
		Logout of the system once logged in	EFR-76
Registration	All Users	Any user must be able to register in the system	EFR-77
		Users cannot enter emails of existing user accounts	EFR-78

Table C.7 - Project user requirements

Appendix D – Group Members Test Script

```
import * as usersDAO from '../src/model/dao/users';
import * as userRelationshipsDAO from '../src/model/dao/users/relationships';
import * as groupUsersDAO from '../src/model/dao/group/users';
import * as groupsDAO from '../src/model/dao/group';
import { UserRelationshipType, UserType } from '../../commons/src/models/user';
import { initializeTestDB } from './utils/utils';

const kPrivileged = 'privileged';
const kTeachers = 'teachers';
const kStudents = 'students';
const kParents = 'parents';
const kRStudent = 'relStudent';
const kRParent = 'relParent';

let groupWithThreeUsers = '';
let groupWithOneParentTwoChildren = '';

function createUser(name: string, type: string): Promise<string>{
    return new Promise<string>((resolve, reject) => {
        usersDAO.createUser(name, name + '@test.lk', type, '', (status, msg, result) => {
            if (status){
                if (result == null)
                    reject('Create user returned null response');
                else
                    resolve(result.user_id);
            }
            else
                reject(msg);
        });
    })
}

function createGroup(tag: string, users: string[]): Promise<string>{
    return groupsDAO.createGroup(tag, 'Test Group', '', 'testkey', '10', users);
}

/**
 * @param parentId Parent
 * @param childId Child
 * @returns
 */
function createParentChildAssociation(parentId: string, childId: string): Promise<boolean>{
    return userRelationshipsDAO.createRelationship(
        parentId, childId, UserRelationshipType.guardianAndChild
    );
}

async function setupData(){
    let t1 = await createUser('teacher', UserType.teacher);
}
```

```

let p1 = await createUser('parent1', UserType.parent);
let s1 = await createUser('student1', UserType.student);
let r1 = await createParentChildAssociation(p1, s1);

let p2 = await createUser('parent2', UserType.parent);
let s2 = await createUser('student2', UserType.student);
let r2 = await createParentChildAssociation(p2, s2);

let p3 = await createUser('parent3', UserType.parent);
let s3 = await createUser('student3', UserType.student);
let s4 = await createUser('student4', UserType.student);
let r3 = await createParentChildAssociation(p3, s3);
let r4 = await createParentChildAssociation(p3, s4);

expect(r1).toBeTrue();
expect(r2).toBeTrue();
expect(r3).toBeTrue();
expect(r4).toBeTrue();

groupWithThreeUsers = await createGroup('Three Users', [t1, p1, s1]);
groupWithOneParentTwoChildren = await createGroup('Single Parent', [t1, p3, s3, s4]);
}

describe('ES-18: Group Members DAO tests', () => {

  beforeAll(async () => {
    try{
      await initializeTestDB();
      await setupData();
    }
    catch(error){
      console.log("Database Error while setting up data...");
      fail(error);
    }
  });

  // MARK Test: Three Users

  it('ES-19 (1): Three Users - Composition', async () => {
    const result = await groupUsersDAO.getGroupUsers(groupWithThreeUsers);

    expect(result.privileged.length).withContext(kPrivileged).toBe(0);
    expect(result.teachers.length).withContext(kTeachers).toBe(1);
    expect(result.students.length).withContext(kStudents).toBe(1);
    expect(result.parents.length).withContext(kParents).toBe(1);
  });

  it('ES-19 (2): Three Users - Association Names', async () => {
    const result = await groupUsersDAO.getGroupUsers(groupWithThreeUsers);
    const reLOfStudent = result.students[0].associations[0];
    const reLOfParent = result.parents[0].associations[0];
  });
}

```



```

    expect(relofStudent.relationshipName).withContext(kRStudent).toBe('Child of');
    expect(relofParent.relationshipName).withContext(kRParent).toBe('Parent of');
  });

  it('ES-19 (3): Three Users - Association Composition', async () => {
    const result = await groupUsersDAO.getGroupUsers(groupWithThreeUsers);

    const relofStudent = result.students[0].associations[0];
    const studentRelUsers = relofStudent.users!;

    const relofParent = result.parents[0].associations[0];
    const parentRelUsers = relofParent.users!;

    expect(studentRelUsers.length).withContext('sRel1').toEqual(1);
    expect(studentRelUsers[0].user_id).withContext('sRel2').toBe(result.parents[0].user_id);
    expect(studentRelUsers[0].user_name).withContext('sRel3').toBe(result.parents[0].user_name);

    expect(parentRelUsers.length).withContext('pRel1').toEqual(1);
    expect(parentRelUsers[0].user_id).withContext('pRel2').toBe(result.students[0].user_id);
    expect(parentRelUsers[0].user_name).withContext('pRel3').toBe(result.students[0].user_name);
  });

  // MARK Test: Single Parent

  it('ES-20 (1): Single Parent - Composition', async () => {
    const result = await groupUsersDAO.getGroupUsers(groupWithOneParentTwoChildren);

    expect(result.privileged.length).withContext(kPrivileged).toBe(0);
    expect(result.teachers.length).withContext(kTeachers).toBe(1);
    expect(result.students.length).withContext(kStudents).toBe(2);
    expect(result.parents.length).withContext(kParents).toBe(1);
  });

  it('ES-20 (2): Single Parent - Association Count', async () => {
    const result = await groupUsersDAO.getGroupUsers(groupWithOneParentTwoChildren);

    const student1 = result.students[0];
    const student2 = result.students[1];
    const parent = result.parents[0];

    expect(student1.associations.length).withContext('s1ReIs').toEqual(1);
    expect(student2.associations.length).withContext('s2ReIs').toEqual(1);
    expect(parent.associations.length).withContext('pReIs').toEqual(1);
    expect(parent.associations[0].users!.length).withContext('pRelUsers').toEqual(2);
  });

  it('ES-20 (3): Single Parent - Association Composition', async () => {
    const result = await groupUsersDAO.getGroupUsers(groupWithOneParentTwoChildren);

    const student1 = result.students[0];
    const student2 = result.students[1];

```

```
const parent = result.parents[0];

const student1Rel = result.students[0].associations[0].users!;
const student2Rel = result.students[1].associations[0].users!;
const parentRel1 = result.parents[0].associations[0].users!;

expect(student1Rel[0].user_id).withContext('s1ParentId').toBe(parent.user_id);
expect(student2Rel[0].user_id).withContext('s2ParentId').toBe(parent.user_id);

expect(student1Rel[0].user_name).withContext('s1ParentName').toBe(parent.user_name);
expect(student2Rel[0].user_name).withContext('s2ParentName').toBe(parent.user_name);

expect(parentRel1[0].user_id).withContext('pRel1Id').toBe(student1.user_id);
expect(parentRel1[1].user_id).withContext('pRel2Id').toBe(student2.user_id);

expect(parentRel1[0].user_name).withContext('pRel1Name').toBe(student1.user_name);
expect(parentRel1[1].user_name).withContext('pRel2Name').toBe(student2.user_name);
});
});
```

Appendix E – Online Survey Results

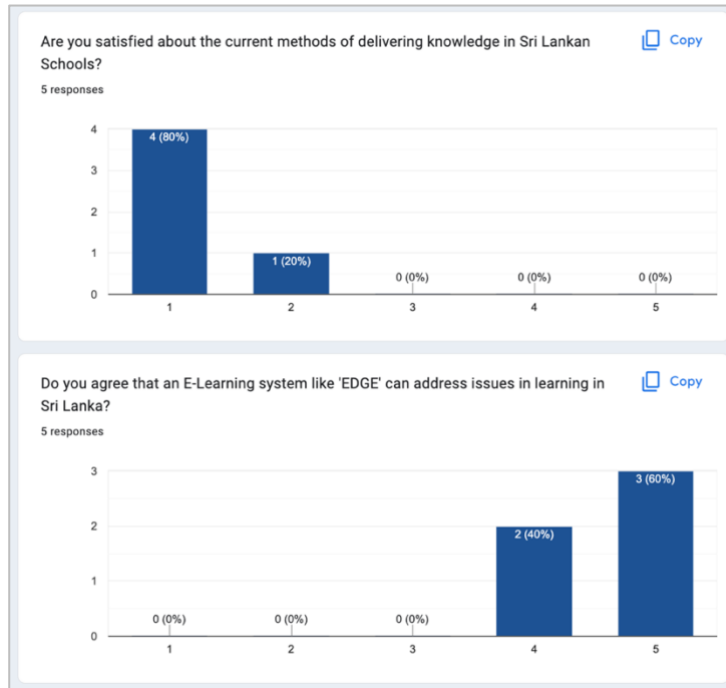


Figure E.1 - Online survey responses for system impression

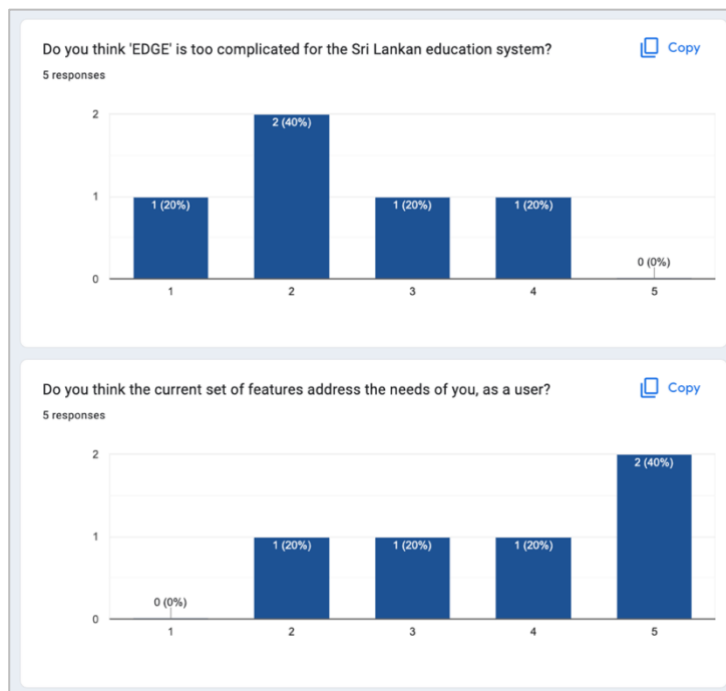


Figure E.2 - Online survey responses for system usability

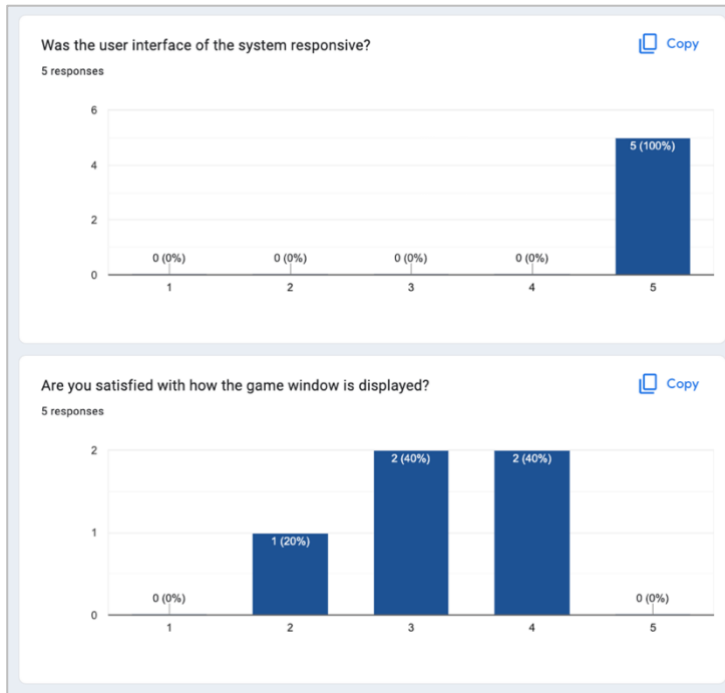


Figure E.3 - Online survey responses for system performance

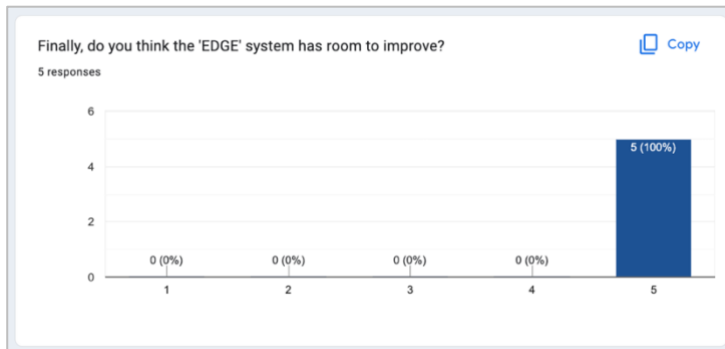


Figure E.4 - Online survey response for future expansion

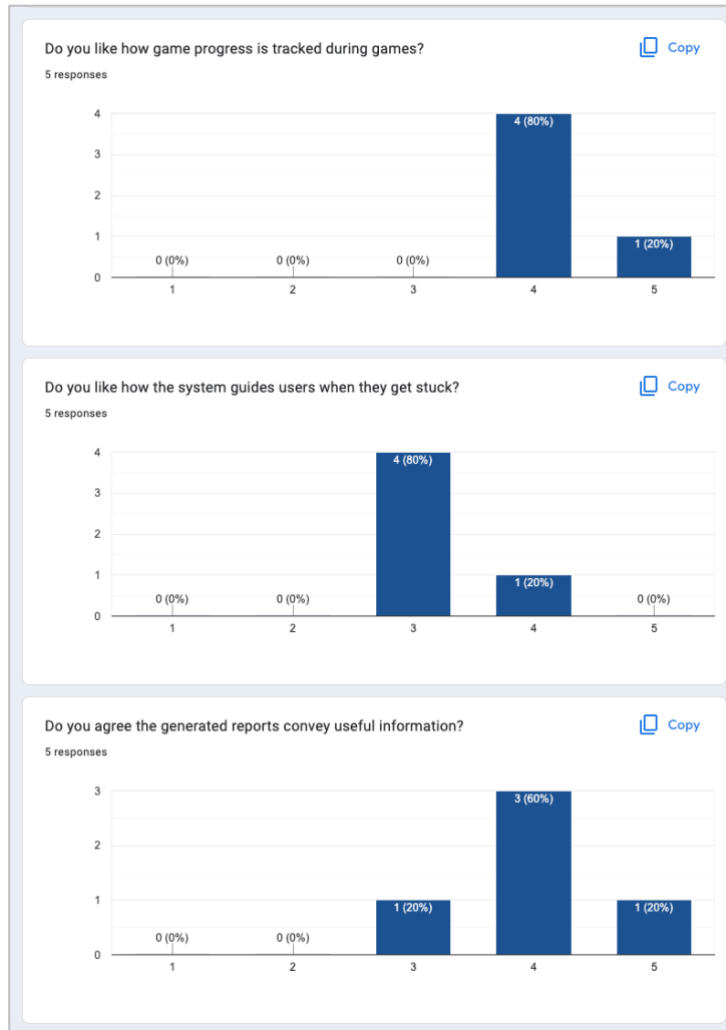


Figure E.5 - Online survey responses for feature set