

{TypeMania}

Preliminary Project Website: TypeMania.github.io

CSCI 441 – Project Report 3 Part 2

Group 2:

Shaka Elmore

Nargis Sultani

Heather Vroman

Alexandra Stevens

Nov 20, 2022

Table of Contents

Table of Contents	1
Individual Contributions	3
Summary of Changes	5
1 Customer State of Requirements	8
2 Glossary of Terms	10
3 System Requirements	13
3.1 Business Goals	13
3.2 Enumerated Functional Requirements	14
3.3 Enumerated Nonfunctional Requirements	15
Functionality	15
Usability	15
Reliability	15
Performance	15
Supportability	15
3.4 User Interface Requirements	16
4 Functional Requirements	18
4.1 Stakeholders	18
4.2 Actors and Goals	19
4.3 Use Cases	21
4.3.1 Casual Description	21
4.3.2 Use Case Diagram	23
4.3.3 Traceability Matrix	25
4.3.4 Fully-dressed Description	26
4.4 System Sequence Diagram	30
5 Effort Estimation using Use Case Points	33
6 Domain Analysis	36
6.1 Concept Definitions	36
6.2 Associate definitions	37
6.3 Attribute definitions	38
6.4 Traceability matrix	41
7 Interaction Diagrams	41

8 Class Diagram and Interface Specification	44
8.1 Class Diagram	44
8.2 Data Types and Operation Signatures	45
8.3 Traceability Matrix	50
8.4 Design Patterns	51
8.5 Object Constraint Language (OCL)	51
9 System Architecture	54
9.1 Identifying Subsystems	54
9.2 Architecture Styles	56
9.3 Mapping Subsystems to Hardware	56
9.4 Connectors and Network Protocol	56
9.5 Global Control Flow	57
9.6 Hardware Requirements	58
10 Algorithms and Data Structures	59
10.1 Data Structures	59
11. User Interface Design and Implementation	59
11.1 Preliminary UI vs. Current UI Design	59
11.2 Implementations	67
12 Design of Tests	73
12.1 Unit Testing	73
12.2 Testing Coverage	74
12.3 Integration Testing	76
13 History of Work, Current Status, & Future Work	77
13.1 History of Work	77
13.2 Current Status	78
13.3 Key Accomplishments	79
13.4 Future Work	80
14 References	81

Individual Contributions

	Shaka Elmore	Nargis Sultani	Heather Vroman	Alexandra Stevens
Discord Management	85%	5%	5%	5%
Github Management	25%	25%	25%	25%
Member Contribution	25%	25%	25%	25%
Summary of Changes	25%	25%	25%	25%
Meeting Minutes	85%	0%	15%	0%
Customer State of Requirements	0%	0%	0%	100%
Glossary of Terms	10%	0%	90%	0%
Business Goals	100%	0%	0%	0%
Enumerated Functional Requirements	0%	100%	0%	0%
Enumerated Nonfunctional Requirements	0%	100%	0%	0%
User Interface Requirements	0%	100%	0%	0%
Stakeholders	0%	0%	100%	0%
Actors and Goals	0%	0%	100%	0%
Use Cases	0%	100%	0%	0%
Casual Description	0%	100%	0%	0%
Use Case Diagram	0%	100%	0%	0%
Traceability Matrix	0%	100%	0%	0%
Fully-Dressed Description	0%	100%	0%	0%

System Sequence Diagrams	0%	0%	0%	100%
Identifying Subsystems	0%	0%	100%	0%
Architecture Styles	0%	0%	100%	0%
Mapping Subsystems to Hardware	0%	100%	0%	0%
Connectors and Network Protocol	0%	100%	0%	0%
Global Control Flow	0%	0%	0%	100%
Hardware Requirements	100%	0%	0%	0%
Effort Estimation using UCPs	100%	0%	0%	0%
History of Work	100%	0%	0%	0%
Current Status	100%	0%	0%	0%
Key Accomplishments	100%	0%	0%	0%
Future Work	100%	0%	0%	0%
Google Doc Creation	25%	25%	25%	25%
Concept Definitions	0%	0%	0%	100%
Associate Definitions	0%	0%	0%	100%
Attribute Definitions	0%	0%	0%	100%
Traceability Matrix	0%	0%	0%	100%
Interaction Diagrams	0%	0%	100%	0%
Project Management	75%	25%	0%	0%
Algorithms & Data Structures	100%	0%	0%	0%
Github Branch Maintenance	25%	25%	25%	25%
Editing Report Three	20%	40%	20%	20%
Responsive CSS Modifications	0%	0%	0%	100%

StartMenu Creation/Functionality/Implementat ion Version 1	0%	0%	0%	100%
Separation of notes into individual entities	100%	0%	0%	0%
OCL Contracts	0%	0%	0%	100%
User Interface Design & Implementation	33%	0%	34	33%
Design of Tests	0%	0%	100%	0%
Class Diagram	0%	100%	0%	0%
Design of Patterns	0%	100%	0%	0%

Summary of Changes

Section	Changes
1	<ul style="list-style-type: none"> Updated to show features that will be in future iterations, including user-specific game interface customization and user generated music upload as well as full-screen gameplay option. Updated explanation of songmap difficulty to consist of speed slider.
2	<ul style="list-style-type: none"> Added a login definition to the glossary of terms. Added a register definition to the glossary of terms.
3	<ul style="list-style-type: none"> Updated business goals diagram to show discontinuation of certain planned features. Made specifications in enumerated nonfunctional requirements. Highlighted certain use cases and requirements in red in sections 3.2 to 3.4 to indicate discontinuation of those features.
4	<ul style="list-style-type: none"> Updated System Sequence Diagram: UC-2 Sign Up (A) illustrating new user registration to accurately reflect current system processes between frontend and backend. Added specifications in user appeal and investors. Added programs being used by suppliers.

	<ul style="list-style-type: none"> Added specifications in actors and goals under sections user appeal, internal connections, and suppliers. Updated the Traceability Matrix in section 4.3.3 due to discontinuation of certain features. Highlighted in red certain use cases in section 4.3.1 that will not be part of the final product.
5	<ul style="list-style-type: none"> Updated to include the additional final demo use cases in UCP calculation. Added duration calculation.
6	<ul style="list-style-type: none"> Updated Domain Model Diagram and Attribution Definition Table to reflect changes in required responsibility attributes (changes due to alterations in system and shifting customization and music upload features to future iterations) Attribute Updates by Responsibility: <ul style="list-style-type: none"> <u>Interface</u> <ul style="list-style-type: none"> Added: viewGame, viewNav Removed: viewUserGame, viewGameGame, viewUserSettings, viewUploadMusic <u>Controller</u> <ul style="list-style-type: none"> Alteration: addUser->createUser, getLeaderboard->getLeaderbrdStats, verifyUser->verifyUserName, postScore->postStatistics Added: verifyUsername, verifyPassword, storeAccessToken, getSongSelectData, getSliderSpeed Removed: getUserSettings, postNewMusic, getUserInfo, getMusic <u>Game</u> <ul style="list-style-type: none"> Added: postStatistics, plauSongMap Removed: viewUserGame, viewScore <u>Database</u> <ul style="list-style-type: none"> Alteration: addUser->postUser, addScore->postScore Removed: getUserInfoData, getUserSettingData, addUserMusic Traceability Matrix updated to remove discontinued use cases (8, 9 10)
7	<ul style="list-style-type: none"> Fixed the login/ logout and registration process explanation as the process has developed from original plans. Added a summary of how the original plan was implemented into the design now created.
8	<ul style="list-style-type: none"> Added the changes description to the Section 8.1. Also, the parts highlighted in Yellow will not be part of the final product.
9	<ul style="list-style-type: none"> Global Control Flow updated to illustrate minute changes to frontend components and features Note about all the features originally included in the architecture of TypeMania.

10	<ul style="list-style-type: none">• N/A
11	<ul style="list-style-type: none">• Added an entirely new subsection called Preliminary UI vs Current UI. It directly compares and addresses deviations from the initial design plans & reveals motivations behind such decisions.• Updated implementation images and descriptions to better reflect the current state of the project.
12	<ul style="list-style-type: none">• Updated statistics screen design test
13	<ul style="list-style-type: none">• Converted previous Project Management sections into the new format: History of work, Current Status, Key Accomplishments, & Future Work.

1 Customer State of Requirements

As technology grows, remote activities in the workplace and in academics are growing as well. As a result, the demand for computer skills and interactive, online educational tools is increasing for both adult and youth learner populations. Typing is a foundational computer skill that promotes productivity and efficiency amongst students and those in the workforce. Furthermore, typing skills can promote learning amongst students with developmental learning disabilities by circumventing handwriting difficulties (Marom & Weintraub, 2015). However, studies have shown that many students lack proficient typing skills, which can have negative effects on academic outcomes, especially for those with learning disabilities, who rely on electronic devices. Moreover, it was found that individuals with and without learning disabilities both benefit from typing instruction equally (Marom & Weintraub, 2015). Therefore, there is a verifiable need for engaging, interactive, and online tools to promote typing skills within the adult and youth education industry. To meet this need of the education industry, the customer would like our team to develop an interactive, online education tool to build typing skills (QWERTY keyboard) of its users, which include learners of varying ages and backgrounds.

Because the key goal of this tool is to build typing skills of its users, understanding what makes up typing skill is critical to realizing a successful software. Typing on a keyboard is a neurologically complex activity including sensory-motor, linguistic, and cognitive skills. Furthermore, typing skill can be evaluated based on one's ability to type at adequate speeds with few errors and limited required attention and motor effort (Marom & Weintraub, 2015). As a result, the proposed educational tool must be able to enhance the user's typing speed and accuracy at increasing levels of difficulty. Additionally, the tool must evaluate the user's continuous progress and improvement related to his or her typing skill. Therefore, the tool must contain both an engaging, hands-on aspect to practice the motor skills related to typing as well as a user-specific evaluation aspect to track and monitor the user's progress related to typing speed and accuracy.

In developing such a typing educational tool, there is research that supports gamification instructional strategies as a means to creating engaging and interactive content for learners. Gamification encompasses using games for learning and has been shown to be effective amongst learners of varying ages, cultures, and contexts (Kim, Song, Lockee, & Burton, 2017). Moreover, educational games have been found to be compelling, motivating, and can "enhance [learner retention,] peer communication and social skills" (as cited in Cheung & Ng, 2021, p. 2). Lastly, research has shown that success in gaming can help promote self-efficacy in those with low self-esteem (Davies & Hemingway, 2014). As a result, gamification of this typing educational tool provides an excellent framework for the software based on the customer requirements and broad user audience.

With the understanding of the described industry needs and software requirements, the customer proposes that the software engineering team develop an online, web-based rhythm typing game as an educational tool to be marketed to and implemented in a variety of settings, including school-based, work-based, private, and remote-learning sectors. The customer aims to utilize this tool to meet the industry demand for interactive, online educational tools to improve typing skills amongst both adult and youth users in a broad range of learning sectors. Currently, most of the typing applications are aimed only for a specific group of learners. For example, they are specifically designed for either only kids, adults, or some other groups. This proposed application is aimed for learners/users of all ages and backgrounds. The game interface will have a simple and efficient design to meet the needs of its varied audience. Additionally, this gaming tool will be web-based and will be accessible in all settings with browser and internet access.

The customer has emphasized that game design is an important factor of this game's development as the user's perception of the game is critical to its success as an educational tool. Games, whose users perceive it as lacking entertainment qualities, are less likely to provide the desired motivational and engaging qualities (Cheung & Ng, 2021). To ensure the online, web-based rhythm typing game is perceived as entertaining, the customer has a great stake in the tool's gameplay design. Moreover, the gameplay design must address and challenge the user's ability to type keys at increasing levels of difficulty based on typing speed and key inclusion.

It can be assumed that music is popular among various ages and cultures of the world, so a rhythm-based game is a great way to engage all users in learning to meet one of the most in-demand technical skills in this technological era. The overall premise of gameplay can be summarized as having the user press a specified sequence of keys on their keyboard at timings that correspond to the beat of selected music. To prompt the user to press specified keys, graphics will move across the screen at the tempo of the music and reach a hit zone. The graphic entering the hit zone will cue the user to type the specified key at the specified beat of the music. When the song ends, the user will be given feedback on their performance in the form of a score for a particular song and accompanying difficulty level. This score can then be compared to other users' scores as well as their own previous performances to encourage improvement.

In achieving the gameplay and features previously listed, the customer wants to develop a user-specific interface, which requires unique login, as well as a guest interface option, which allows for anonymous play. The user-specific interface can be accessed through a login portal in which users can view their history, track their progress as typing skills improve, and view other useful user personalized statistics, as well as compete for high scores in a global leaderboard. . In future iterations of the software, it is planned to add additional login features, including user-specific game interface customization and user generated music uploads. Also, note that

the guest interface will not include user-specific features, like customizability and performance tracking, but will enable anonymous gameplay and as a result skill improvement. Providing a guest interface provides flexibility in the software for users who cannot or choose not to have a custom experience while still being able to improve upon typing skills.

The registered user and guest both have the option to choose to change the level of difficulty of their songs. The level of difficulty is based on the bpm, or speed of the song, which can be altered for any chosen song with a selection slider. Registered users can also choose to set the level to automatically adjust to previous performance. This helps the user to better their typing skills by always pushing the difficulty level.

While creating the interface and graphics of the game, the customer notes that the design is the first aspect which catches the eye of the user. It also must meet the needs of the broad user audience therefore is another important aspect to the game development. To meet these needs, the customer proposes that the interface be simple and efficient. The color scheme should remain simple and uniform so as to not overwhelm the adult user, but contain highlights of various colors to appeal to the younger age groups. A game menu and navigation bar will be fixed at the top of the page to promote easy, straight-forward game functionality, navigation, and setting control to enhance user experience for all ages and backgrounds. In future iterations of the software, it is planned to provide customizability of various style choices the game offers, which include changing the background, type font, font color, note colors, and note style. Lastly, future iterations will also enable a full-screen gameplay feature for users to increase visibility as well as limit distractions during completion of levels.

2 Glossary of Terms

TypeMania - Name of the musical typing game aimed towards children and adults alike.

Music - Vocal and/or instrumental sounds created in such a way that appeals to each user.

Game database - Storage for various components essential to the game such as music, information which changes level difficulty, and how each game feature works.

Game feature - Options which can be utilized for a better user experience but do not affect the quality or performance of the game.

Typing speed - How fast each user can type as in words or letters per minute.

Statistics - Stores numerical data such as a user's typing speed, leaderboards, and if they are in or out of the average typing speed for their persona.

Persona - Age groups predetermined by the creator.

Registered user - Frequent users may want to register to receive emails when an update has occurred, and can view statistics from previous gameplay sessions.

Leaderboard - Users can view top scores for each song after gameplay, within other statistics and where they stand compared to other players.

Background - The gradient or image which appears behind gameplay which can be changed by the user.

Note - The graphic labeled with a keyboard character prompting the user on which key to press.

Hit zone - The position on the scroll track that cues the user in on when the note should be pressed.

Scroll track - The ribbon of space stretching horizontally across the gameplay scene. Notes scroll across this track from right to left towards the hit zone.

Note sequence - The timeline on which keys appear during gameplay.

Songmap - The song and its associated note sequence.

Type font - How the text appears on the screen for TypeMania which can be changed by the user.

Type color - The color which the text appears on the screen which can be changed by the user.

Note font - How the notes which prompt the user which key or keys to type appear on the screen which can be changed by the user

Note color - The color the note appears on the screen which can be changed by the user.

Style choices - Various options the user can utilize to change the image and how the screen appears to the player to match their personal preferences.

Performance - How quickly the game loads and how effectively the user types in sync with the music depending on the context used.

Lag times - The delay between the user typing and the music exiting the speaker of the device the game is being played on, typically consisting of a laptop or desktop computer.

Login- Form located on TypeMania which allows the user to retrieve information from a pre-existing account.

Register- Form located on TypeMania which allows the user to create an account in order to retrieve data in the future from their account.

3 System Requirements

3.1 Business Goals

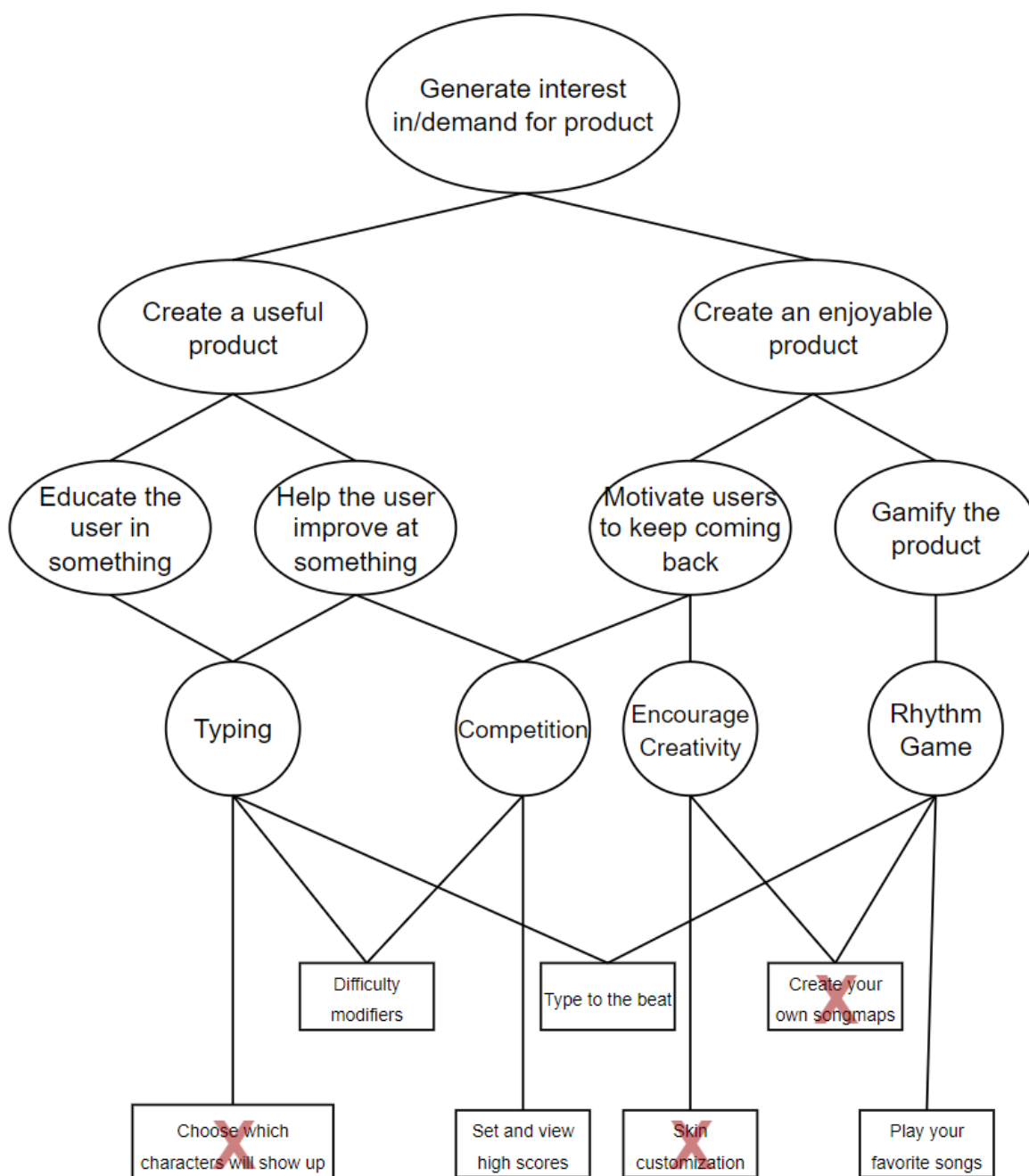


Figure 3.1.1: Business goals & their derived requirements. The red X's mark features that were cut due to time constraints.

3.2 Enumerated Functional Requirements

The requirements highlighted in red will not be part of the final product.

Identifier	Priority Weight (Low 1 - 5 High)	Requirement Description
REQ-1	5	The users should be able to type to the rhythm of their selected songs.
REQ-2	3	The application should allow a user-specific interface; for unique logged-in and anonymous users.
REQ-3	3	The users should be able to modify the songmap option by specifying which characters can show up.
REQ-4	4	The users should be allowed to select their favorite songs from a pre-populated list.
REQ-5	4	The users should be allowed to upload their own music to allow parents to restrict the types of music their children are listening to.
REQ-6	3	The users should be able to modify the songmap option by specifying how fast the music playback should be.
REQ-7	4	The users should receive feedback from the application on their performance.
REQ-8	3	The users should be able to customize the look and feel of the interface
REQ-9	4	The users should be able to view leaderboards to view other users' performance.
REQ-10	3	The users should be able to create a map for the songs they upload and should be able to share the map with others.
REQ-11	4	The logged-in users should be able to retrieve their data, such as their customized settings, previous scores, etc.
REQ-12	4	The user should be able to login.

Table 3.2.1: Functional Requirements

3.3 Enumerated Nonfunctional Requirements

Functionality

Authentication will be supported for the users to retrieve their data via registration and login pages. The users will have the ability to retrieve forgotten passwords. Also, the application will be supported on multiple web browsers for more ease for users on a personal computer.

Usability

The interface for this application is designed to be simple and easy to use. The ability to customize the style and the background of the interface makes it appealing to users of various backgrounds and ages.

Reliability

Redundancy and scalability of this application will help keep the failure frequency minimal. Also, the user's data will be backed up on a frequent basis. For example, when creating the footer the dimensions are in percentages compared to pixels.

Performance

The application is designed to be high performing so that the users could load the music of their choice in a timely manner and start focusing on typing. The application will offer minimum load and lag times. It is designed to serve concurrent requests by multiple users logged in. Also, it is designed to be scalable to keep performing high as the number of users increases. A continuous example of this can be seen in the leaderboard. The leaderboard consists of a consistent stream of high scores in TypeMania which is consistently updated.

Supportability

The application is designed to be easy to use and understand because the end users will be of various age groups. Since this application will be used by many, it will be made scalable to include an additional number of servers to achieve load balancing in order to meet the increasing workload and number of users. It will provide the feasibility and flexibility to update and extend any of its components or functionalities to meet the change in user's taste and preferences over time. It will offer improved versions over time that can be installed only by administrators. Also, it will be backed up to a remote server so new features and functionalities could be added and tested in a controlled environment.

3.4 User Interface Requirements

The part highlighted in red will not be part of the final interface.

Identifier	Priority Weight (Low 1 - 5 High)	Requirement Description
REQ-11	5	The interface should default to the guest page, where the user has the option to login and play as a guest. The guest only has a scoreboard and the option to select songs.
REQ-12	4	After clicking on the login button, the user should be taken to the login page, where they have the option to login, be taken to the sign up page, or retrieve password.
REQ-13	5	The logged in users should have the option to view leaderboards, their own statistics, customize UI settings, create a songmap by uploading songs

Table 3.4.1: User Interface Requirements Table

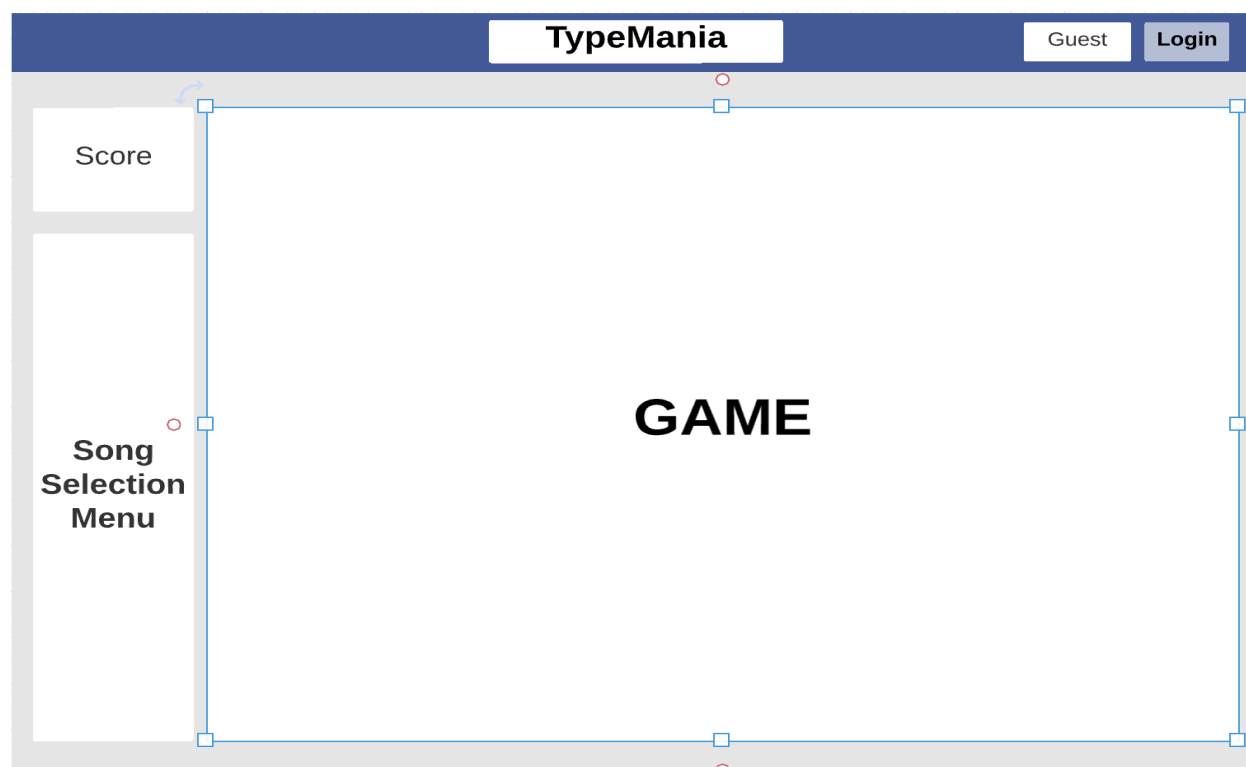
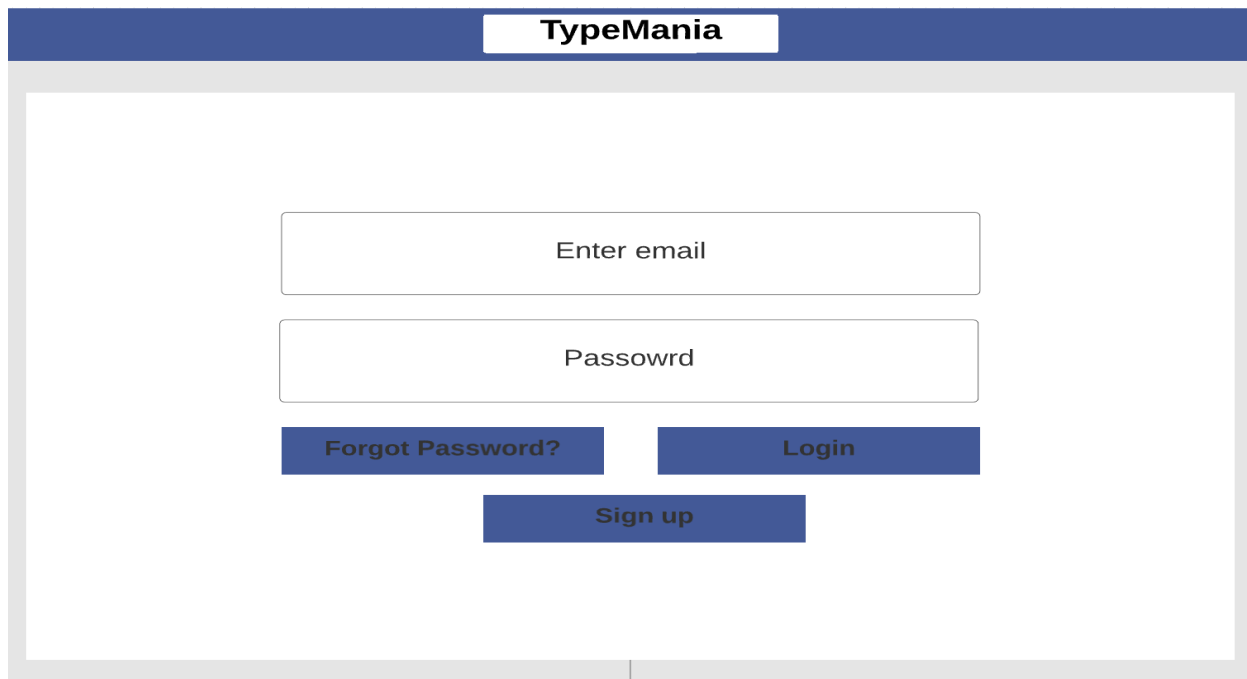
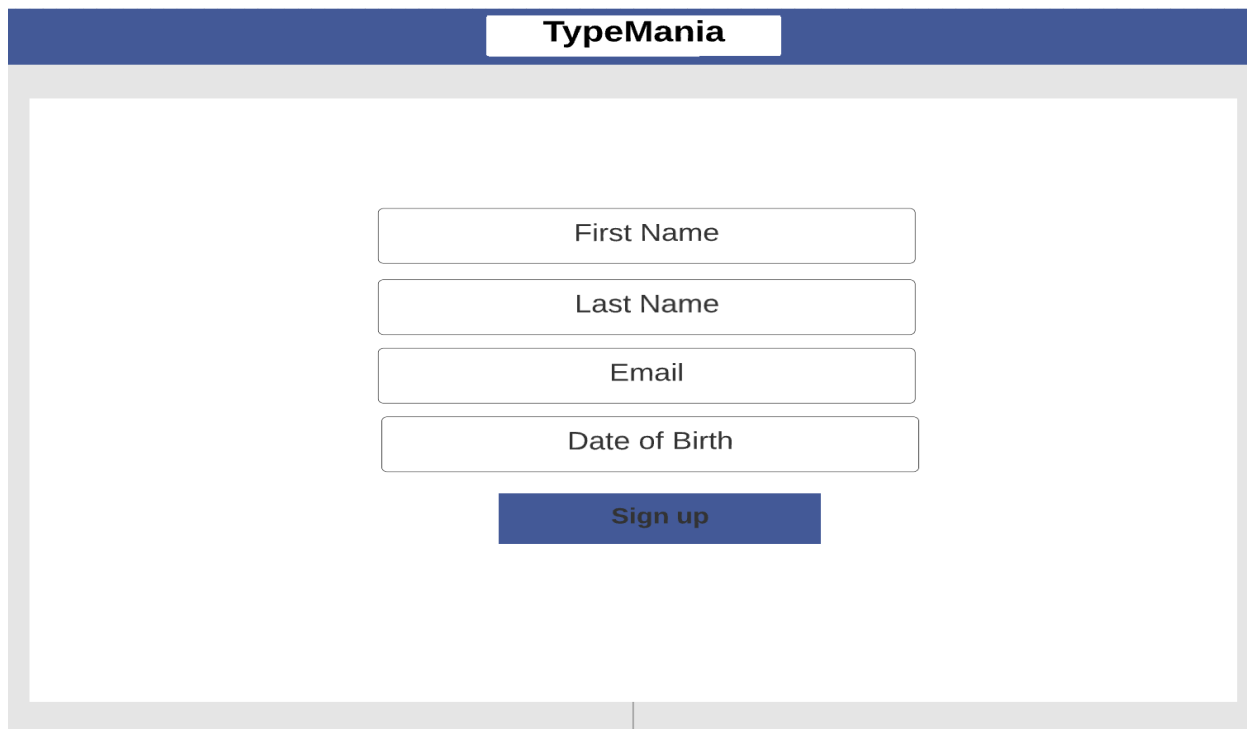


Figure 3.4.1: User Interface - Default Page



The image shows a web browser window with a dark blue header bar containing the text "TypeMania" in white. The main content area is white and contains a login form. The form consists of two text input fields: the first is labeled "Enter email" and the second is labeled "Passowrd". Below these fields are three buttons: "Forgot Password?" and "Login" are side-by-side, and "Sign up" is centered below them. All buttons are dark blue with white text.

Figure 3.4.2: User Interface - Login Page



The image shows a web browser window with a dark blue header bar containing the text "TypeMania" in white. The main content area is white and contains a sign-up form. The form consists of four text input fields: "First Name", "Last Name", "Email", and "Date of Birth". Below these fields is a single "Sign up" button. All buttons are dark blue with white text.

Figure 3.4.3: User Interface - Sign-up Page

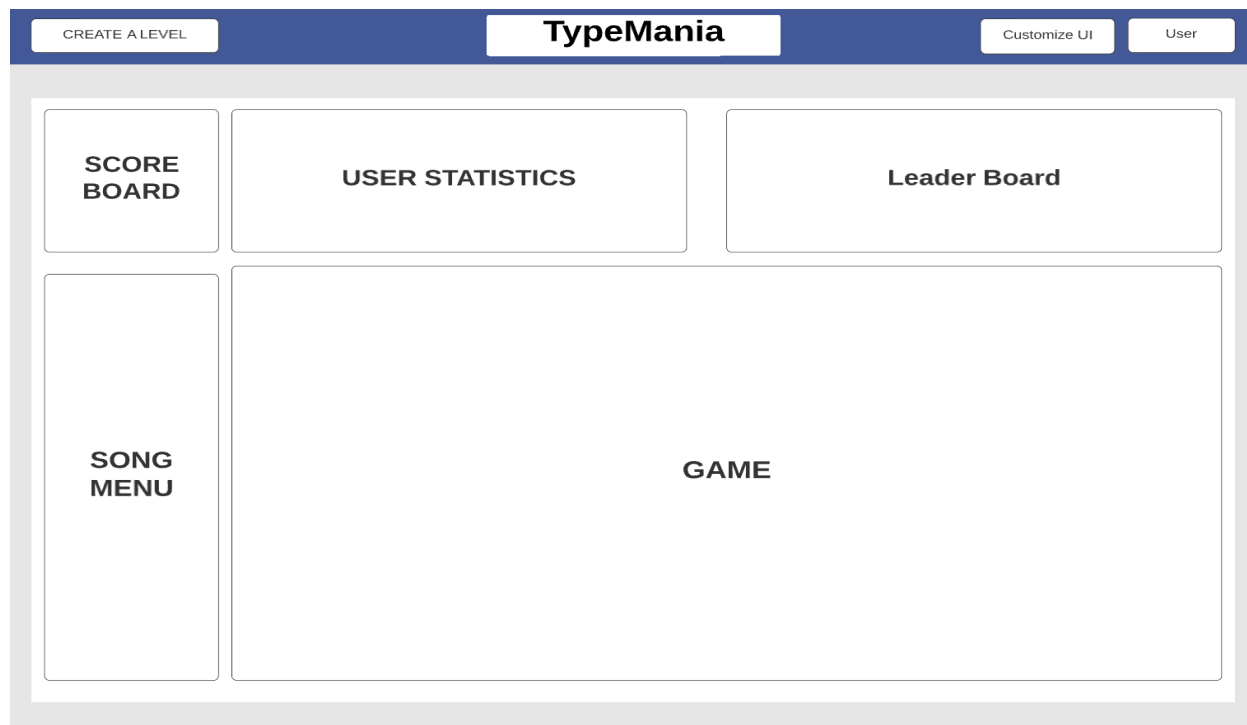


Figure 3.4.4: User Interface - Logged in User Page

4 Functional Requirements

4.1 Stakeholders

A. User Appeal

1. Externally, public educators, parents, homeschooling groups, and individuals of all ages are stakeholders due to their interest in using TypeMania.
2. It is important to conduct extensive testing resulting in smooth running software with minimal bugs. **Unit testing as an example identifies broken link and ensures gameplay is in working condition.**
3. To achieve this, identifying who will be using the software and reaching out to potential users is essential.

B. Investors

1. Obtaining investors is crucial to sustain finances such as subscriptions, payroll, business expansions, marketing
2. How many investors depends on the stage the software is at. For example, a software in the beginning stages needs less financial backing in the above

scenario than a large company such as Apple launching a new update or product.

- Any sort of growth requires financial backing, which is why we need investors. This may include upgrading software used to the run program, or investing in marketing strategies to reach potential customers.

C. Internal Connections

- Internal connections are essential in software development not only to have a team of diverse talents working together, but for support, problem solving, and everything else that goes on behind the scenes.
- Multiple teams come together to make an operation such as TypeMania come to life this includes developers, marketing, and data analysts coming together to work as a team.

D. Suppliers

- There are not many suppliers needed for TypeMania compared to opening a restaurant, but every operation needs support from established organizations.
- Suppliers would consist of github for storing code, React for deploying the game, Phaser 3 used for gameplay, and Heroku where we deploy the website on.

E. Community

- Community is important regardless of scenario, making our community an important stakeholder.
- Receiving consistent feedback from users to take into consideration, and other communication such as success stories keeps the internal teams going to improve TypeMania to its utmost potential.

4.2 Actors and Goals

Role	Type	Goals
Guest User	Initiating Actor	<ul style="list-style-type: none"> Register Select Song Select Typing Speed Play Game Receive Feedback
Registered User	Initiating Actor	<ul style="list-style-type: none"> Login Select Song Select Typing Speed Play Game Receive Feedback

Database	Participating Actor	<ul style="list-style-type: none"> • Update User Table • Authenticate User Details • Grant/Deny Access • Store Songs & Statistics
----------	---------------------	---

Table 4.2.1: Actors and Goals Table

A. User Appeal

1. The groups and individuals TypeMania appeals to each interact differently with the system depending on the goal they have set out to accomplish. **Employers may use the system for efficiency whereas schools will use the system for teaching typing skills to beginners.**
2. Potential users will test TypeMania with the goal to better the software for the company.
3. This will allow developers to work out any bugs which may appear and receive user feedback before launching the entire software to the public.
4. The potential users will gain typing skills, a free trial period to use the software after launch along with knowing they were the first to use TypeMania with the opportunity to impact the software by offering quality feedback.

B. Investors

1. Investors in the company usually do so within stock regardless if that is public or private.
2. In the long run investors expect a profit from taking the risk and backing the company from the beginning stages.
3. TypeMania needs to have a stock to offer for this to be an option.

C. Internal Connections

1. Developers need to ensure the program runs smoothly without bugs, error code, and lag time. **This can be accomplished via unit testing.**
2. The marketing team finds outside stakeholders so the operation continues to run smoothly, and data analysts study the trends on the website such as which age group visits regularly and how often individuals stay on the page.
3. In exchange for their services compensation is received for every team member along with access to TypeMania.

D. Suppliers

1. Suppliers support us by hosting our website, **hosting our database**, and providing storage for essentials such as code and design prototypes.
2. In exchange, we pay for subscriptions when payment is due.

3. Github provides the option to look at other individual and group projects allowing us inspiration and vital learning opportunities for future updates.

E. Community

1. Community may seem like a small part to a larger network of individuals but its purpose is essential.
2. Users have the opportunity to provide feedback making a difference in the TypeMania which provides them a fun way to learn to type while providing us the opportunity to continually improve the users experience
3. Community is not only about bettering the software, but each other. The opportunity to to hear success stories will encourage the teams behind the scenes and other users benefiting everyone involved.

4.3 Use Cases

The users will have access to the game as soon as they enter the site. By default, the guest page is landed where the user can play anonymously with limited functionalities. On the default page, the users will also have an option to login and play as a logged in user where they can access more functionalities of the game, such as customizing UI settings, uploading their own music and creating a songmap, and much more. The ability to customize features will attract users of various backgrounds and age groups. The core functionalities of the software will be elaborated below.

4.3.1 Casual Description

UC-1 Guest User Page - The default page allows all guest users to play with limited functionalities and gives the option to register/login with access to more functionalities.

Derivations: REQ-2

UC-2 Register - On the Register page, the user has the option to login or register so they get the benefits of accessing more functionalities.

Derivations: REQ-2

UC-3 Registered User Page - The registered users have the option to customize UI settings, view leaderboard, detailed scoreboard, upload music, create a songmap, etc.

Derivations: REQ-2

UC-4 Typing Speed Selection - All users have the ability to choose a typing speed in words or letters per minute.

Derivations: REQ-5, REQ-6

UC-5 View Score - All users are allowed to view their scores.

Derivations: REQ-7

UC-6 Music Selection - Gives all users the ability to select their favorite music to type to the rhythm of.

Derivations: REQ-4

UC-7 Play Game - Allows the users to type to the rhythm of their favorite music and typing speed that they have selected.

Derivations: REQ-1

UC-8 Creating a Songmap - Allows the registered users to create a songmap to the music that they have uploaded.

Derivations: REQ-5

UC-9 Uploading Music - Allows the registered users to upload their own music.

Derivations: REQ-5

UC-10 Style Choices Selection - Allows the registered users to change the look and feel of the interface to make it more appealing to them. They are allowed to select and change the background, type font, type color, note font, and note color.

Derivations: REQ-3, REQ-8

UC-11 Leaderboard Display - Allows the users to view top scores for each song after gameplay, within other statistics and where they stand compared to other players.

Derivations: REQ-9

UC-12 View Statistics - Allows the registered users to view their statistics.

Derivations: REQ-11

UC-13 Login - Allows the registered user to login to be able to access the system as a registered user and access more functionalities.

Derivations: REQ-12

4.3.2 Use Case Diagram

The Use Case diagram below depicts the User and System interactions. There are two types of users; guest user and registered user.

- The Guest User has the option to register.
 - The User table will either get updated in the database and the system will send confirmation back to the user; Or the system will send an error message if incorrect details were entered.
- The Guest User can choose to play the game without having to register or login.
 - The user can select the typing speed and the song which is loaded from the database to start playing the game.
 - The system generates the songmap for the selected song and starts the game.
 - The system then provides feedback/score to the user after the game ends.
- The Registered User can login to access their statistics.
 - The user initiates to login and the system will use the database to either authenticate the user and grant access if correct information is entered; Or the system will return an error message to the user.
 - After the user is granted access, the user can play game just like the Guest user and can access their statistics.

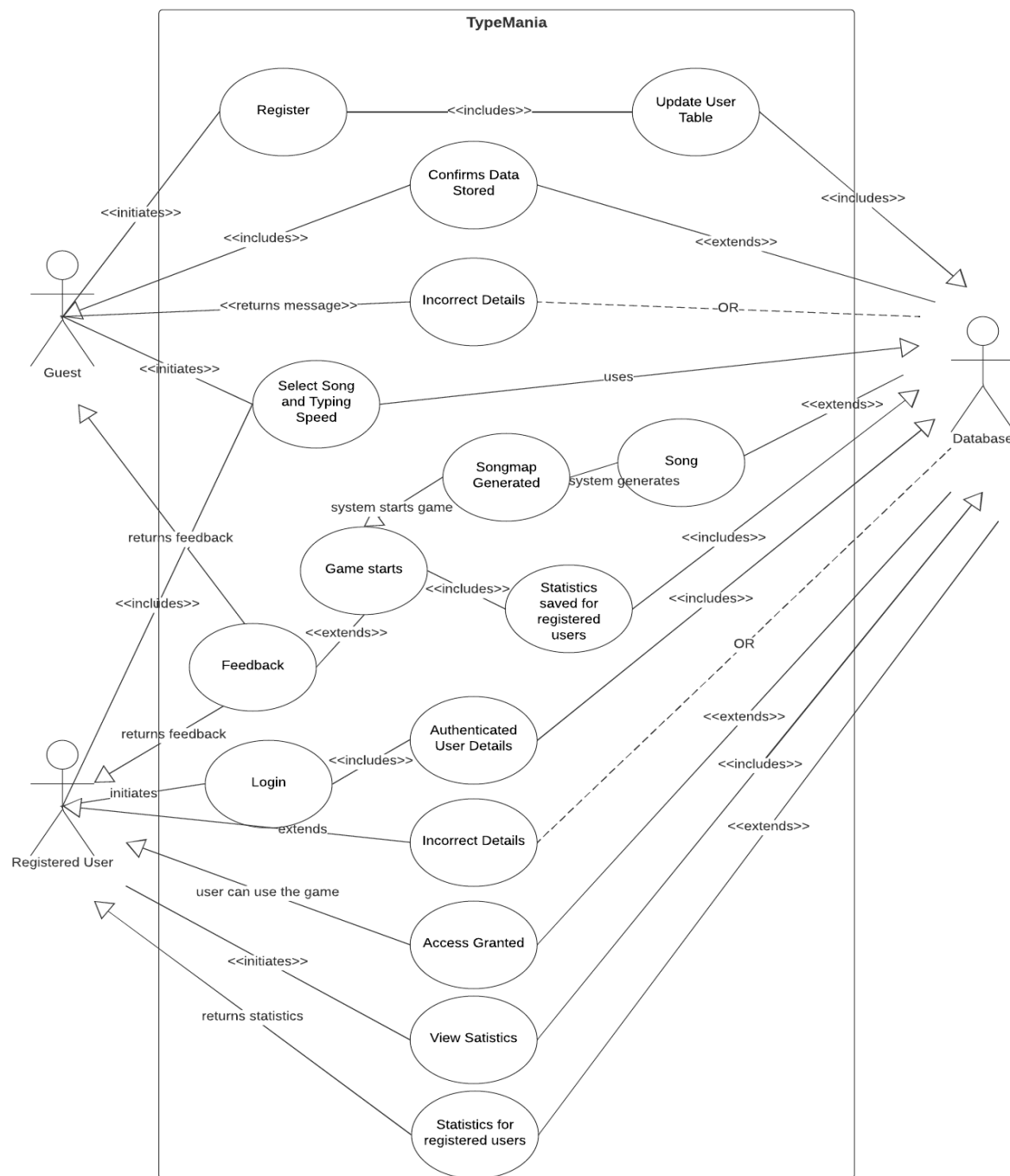


Figure 4.3.2.1: Use Case diagram depicting the user's interaction with the system.

4.3.3 Traceability Matrix

The requirements and priorities highlighted in red will not be part of the final product. Therefore, an updated Total Priority row has been added.

Requirements	Priority Weight	UC-1	UC-2	UC-3	UC-4	UC-5	UC-6	UC-7	UC-8	UC-9	UC-10	UC-11	UC-12	UC-13
REQ-1	5							X						
REQ-2	3	X	X	X										
REQ-3	3										X			
REQ-4	5						X							
REQ-5	4				X				X	X				
REQ-6	3				X									
REQ-7	3					X								
REQ-8	3										X			
REQ-9	4											X		
REQ-10	3													
REQ-11	4												X	
REQ-12	4													X
Total Priority		3	3	3	7	3	5	5	4	4	6	4	4	4
Updated Total Priority		3	3	3	3	3	3	3				4	4	4

Figure 4.3.3.1: Traceability Matrix Chart

4.3.4 Fully-dressed Description

Use Case: Register	
Related Requirements: REQ-2	
Initiating Actor: Guest User	
Actor's Goal: To be able to register	
Participating Actor: Guest user, Database	
Preconditions: Not be a registered user	
Postconditions: Get registered in the system	
Flow of Events for Main Success Scenario:	
<p>1The user selects the option to register. The user gets redirected to the register page. The user enters info the registration form and submits The system checks for any errors. The system sends the data to the user table in the database.</p>	
Flow of Events for Alternate Scenarios:	
<ol style="list-style-type: none"> 1. If there is any error, the database returns a message and asks the user to re-enter info. 2. The user re-enters info, and the user is saved into the database and the database returns confirmation. 	

Use Case: Login

Related Requirements: REQ-2

Initiating Actor: Guest User

Actor's Goal: To login and play as a registered user

Participating Actor: Registered user, Database

Preconditions: Be a registered user

Postconditions: Login and access the system as a registered user

Flow of Events for Main Success Scenario:

1. The user selects the option to login.
2. The user gets redirected to the login page.
3. The user enters credentials and submits.
4. The system authenticates the user against the database.
5. The access is granted and the user is able to use the system as a registered user.

Flow of Events for Alternate Scenarios:

1. The system returns an error message.
2. The user re-enters credentials.

Use Case: Play Game

Related Requirements: REQ-1, REQ-3, REQ-4, REQ-7, REQ-8

Initiating Actor: Guest user, Registered User

Actor's Goal: To be able to play the game to their selected song and typing speed.

Participating Actor: Guest user, Registered user, Database

Preconditions: None

Postconditions: Playing the game and viewing scores or feedback.

Flow of Events for Main Success Scenario:

1. The user selects music and typing speed and the play button.
2. The system loads the song from the database and generates the songmap.
3. The user can start playing.
4. The system returns feedback.
5. The registered user's statistics are saved in the database.

Use Case: View Statistics

Related Requirements: REQ-2

Initiating Actor: Registered User, database

Actor's Goal: To be able to view statistics.

Participating Actor: Registered user

Preconditions: Must be a registered user and logged in.

Postconditions: User can view statistics

Flow of Events for Main Success Scenario:

1. The user initiates selects to view statistics.
2. The user is redirected to the statistics page.
3. The user's statistics are loaded from the database into the page.
4. The user can view statistics.

Flow of Events for Alternate Scenarios:

1. If the system runs into any issue, an error message is returned.

4.4 System Sequence Diagram

https://www.canva.com/design/DAFMaPutao8/lq9tL-EeQ-NJic_gmN-WQQ/view?utm_content=DAFMaPutao8&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

System Sequence Diagram: UC-2 Sign Up (A).

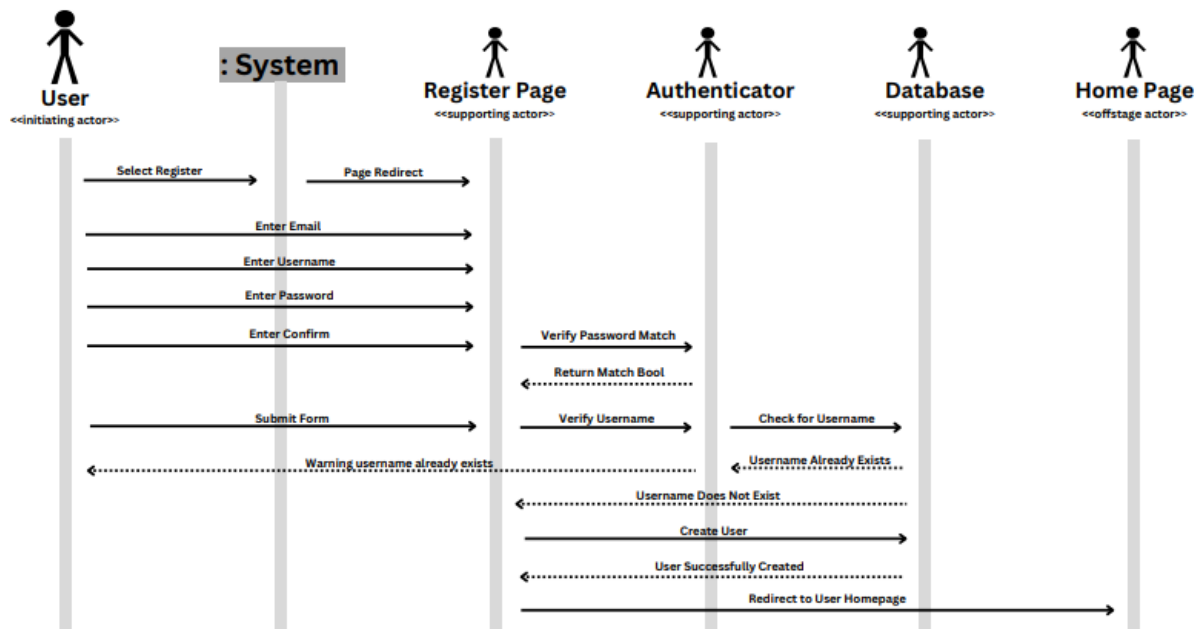


Figure 4.4.1: System Sequence Diagram – UC-2 Sign Up (A): Register a new user, main success scenario

System Sequence Diagram: UC-2 Sign Up (B).

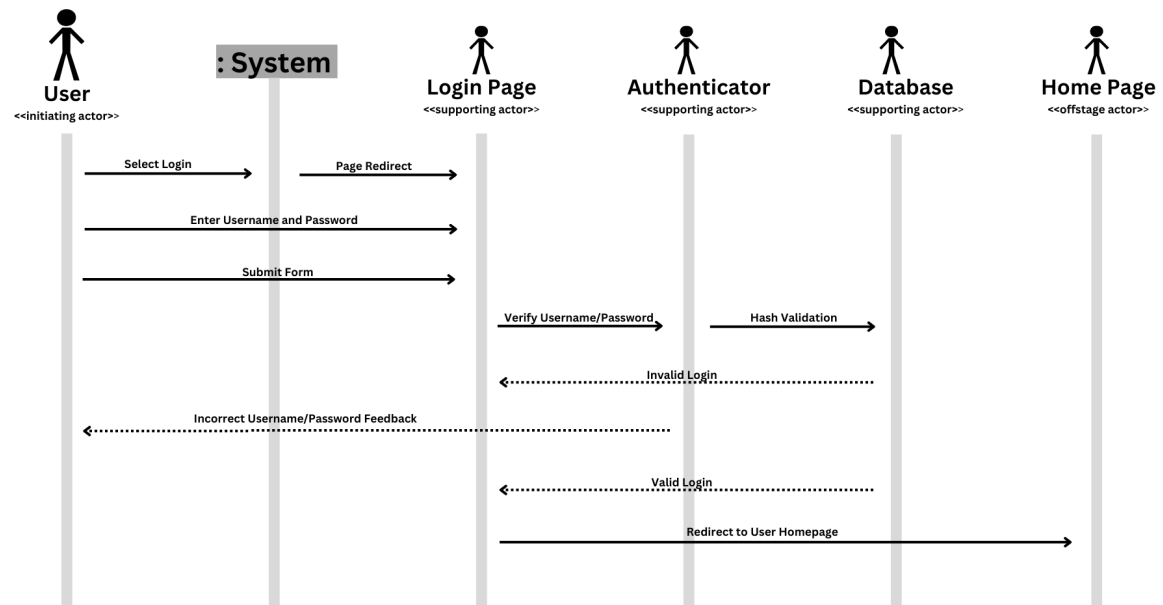


Figure 4.4.2: System Sequence Diagram – UC-2 Sign Up (B): Login Existing User, main success scenario

System Sequence Diagram: UC-7 Play Game

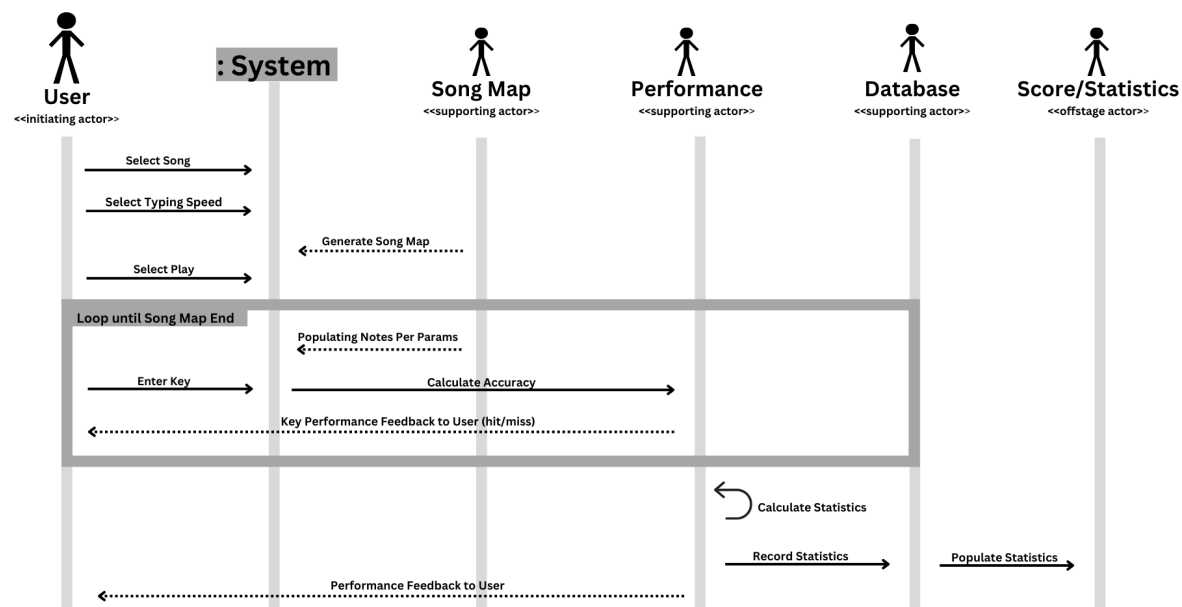


Figure 4.4.3: System Sequence Diagram – UC-7 Play Game, main success scenario

System Sequence Diagram: UC-12 View Statistics

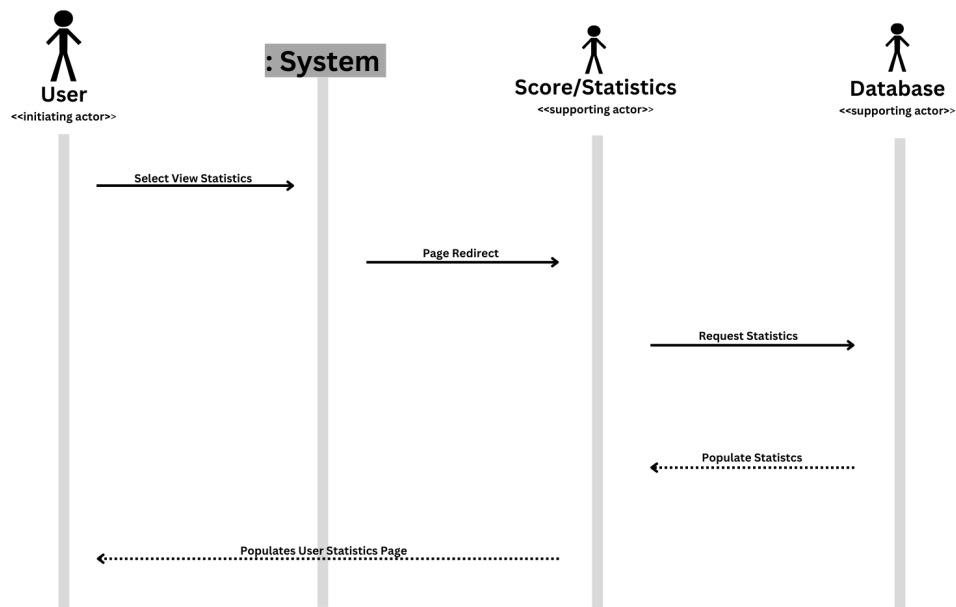


Figure 4.4.4: System Sequence Diagram – UC-12 View Statistics, main success scenario

5 Effort Estimation using Use Case Points

Unadjusted Actor Weight (UAW)

Actor	Description of relevant characteristics	Complexity	Weight
Database (Participating)	Another system that interacts with our system through a well defined API.	Simple	1
Guest User (Initiating)	A person interacting with our system via GUI.	Complex	3
Registered User (Initiating)	A person interacting with our system via GUI.	Complex	3
UAW Total:			7

Table 5.1: Unadjusted Actor Weight calculations.

Unadjusted Use Case Points (UUCP)

Use Case	Description of relevant characteristics	Category	Weight
UC-1 Guest User Page	Simple user interface interaction of 0 steps (landing page) for main success scenario. No Participating Actor.	Simple	5
UC-2 Register	Moderate user interface interaction of 6 steps for main success scenario. One Participating Actor (Database).	Average	10
UC-3 Registered User Page	Simple user interface interaction of 0 steps (landing page) for main success scenario. One Participating Actor (Database).	Simple	5
UC-4 Typing Speed Selection	Simple user interface interaction of 2 steps for main success scenario. No Participating Actor.	Simple	5
UC-5 View Score	Simple user interface interaction of 3 steps for main success scenario. One Participating Actor (Database).	Simple	5
UC-6 Music Selection	Simple user interface interaction of 2 steps for main success scenario. One Participating Actor (Database).	Simple	5
UC-7 Play Game	Simple user interface interaction of 3 steps for main success scenario. One Participating Actor (Database).	Simple	5
UC-11 Leaderboard Display	Simple user interface interaction of 2 steps for main success scenario. One Participating Actor (Database).	Simple	5
UC-12 View Statistics	Simple user interface interaction of 1 step for main success scenario. One Participating Actor (Database).	Simple	5
UC-13 Login	Moderate user interface interaction of 4 steps for main success scenario. One Participating Actor (Database).	Average	10
UUCP Total:			60

Table 5.2: Unadjusted use case points calculations.

Technical Complexity Factors (TCF)

Technical	Description	Weight	Perceived	Calculated
-----------	-------------	--------	-----------	------------

Factor			Complexity	Factor
T1	Distributed system.	2	3	$2 * 3 = 6$
T2	Performance objective.	1	5	$1 * 5 = 5$
T3	End-user efficiency.	1	2	$1 * 2 = 2$
T4	Complex internal processing.	1	1	$1 * 1 = 1$
T5	Reusable design or code.	1	4	$1 * 4 = 4$
T6	Easy to install.	0.5	0	$0.5 * 0 = 0$
T7	Easy to use.	0.5	3	$0.5 * 3 = 1.5$
T8	Portable.	2	0	$2 * 0 = 0$
T9	Easy to change.	1	3	$1 * 3 = 3$
T10	Concurrent use.	1	3	$1 * 3 = 3$
T11	Special security features.	1	1	$1 * 1 = 1$
T12	Direct access for third parties.	1	0	$1 * 0 = 0$
T13	Special user training.	1	0	$1 * 0 = 0$
Technical Factor Total:				26.5
$TCF = 0.6 + 0.01 * 26.5 = 0.865$				

Table 5.3: Technical Complexity Factor calculations.

UCP = UUCP * TCF
 = $60 * 0.865$
 = **51.9** or **52** use case points

Duration = UCP * PF
 = $51.9 * 28$
 = **1453.2** total hours

6 Domain Analysis

6.1 Concept Definitions

The domain model concepts and corresponding responsibilities were derived from the thirteen previously defined system use cases defined in Report 1. Based on these use cases, the domain model is made up of the interface, game, controller, validator, and database concepts. See Figure 1 below illustrating the full domain model diagram. The detailed domain model concepts and responsibilities are summarized in Table 1.

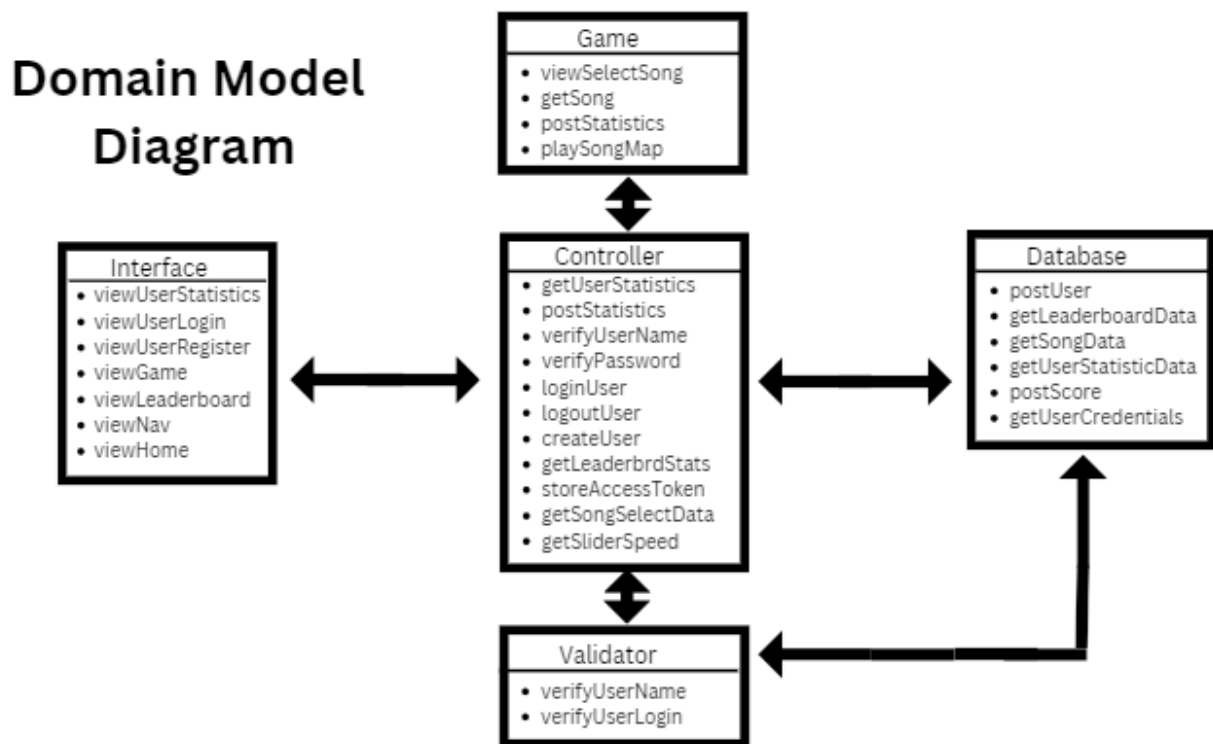


Figure 6.1.1: The system's domain model diagram

https://www.canva.com/design/DAFNxvymw8Q/W2q7h9VMPLQ- M67ITduGA/edit?utm_content=DAFNxvymw8Q&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

Responsibility Description	Type	Concept Name
----------------------------	------	--------------

Rs1. React Website that features user statistics, user login, user registration, game, and leaderboard; UI with game component with database connection and get and post communication	K	Interface
Rs2. Coordinate the actions of all concepts, including the interface, game, validator, and database; coordinates all get and post actions to and from the database; provides data from database to create user experience, game play scores, and leaderboards	K, D	Controller
Rs3. Validates posted user credentials with stored user credentials in the database; verifies new user is not a duplicate user in the database	K	Validator
Rs4. Contains all game functionality; views game UI using getSongSelectData for song select menu and getSliderSpeed for song difficulty; plays user chosen song for gameplay; views player score during game play, posts user scores to database for statistics and leaderboard	K, D	Game
Rs5. Stores all user data and song data and completes get and post requests for data; requests come through validator and controller from interface and game concepts	K, D	Database

Table 6.1.1: Domain model concepts and responsibilities

6.2 Associate definitions

In order to achieve the defined responsibilities shown in Table 1, all of the listed domain model concepts have association with other concepts throughout the model. These concept association descriptions are listed below in Table 2.

Concept Pair	Association Description	Association Name
--------------	-------------------------	------------------

Interface«Controller	Interface makes get requests to Controller, Controller processes request communicating with the database and sends back data to Interface to display pages; Interface makes post requests to Controller and Controller processes the posts requests	Generates request, Conveys result,
Game«Controller	Game makes get requests to Controller, Controller processes request and sends back data to Game display on game interface; Game makes posts to Controller and Controller processes the posts requests	Generates request, Conveys result
Controller«Validator	Controller makes requests to Validator to verify user credentials, Validator processes request by communicating with database and sends back data to Controller to move to next step of current process	Generates request, Conveys result
Validator«Database	Validator makes requests to Database to verify user credentials, Database processes request and sends back data to Validator to return a true or false verification	Generates request, Prepares request, Conveys result
Controller«Database	Controller makes get and post requests to Database, Database processes request; Controller then moves to next step in current process	Generates request, Prepares request

Table 6.2.1: Domain model concepts associations

6.3 Attribute definitions

When examining the above listed concepts and their relationships, we must also detail the various attributes that accompany each concept. The attributes have various functions that contribute to the overall actions or goals of the concepts. The system's concept attributes are described below in Table 3.

Concept	Attribute	Attribute Description
Interface	viewUserStatistics	Displays user statistics page and data of logged in registered user; sends request to Controller via getUserStatistics
	viewUserLogin	Displays user login form; accepts inputs; sends request to Controller viewUserLogin
	viewUserRegister	Displays user register form; accepts inputs and sends to Controller addUser
	viewGame	Displays home game page with user data; sends request to Controller to getUserInfo
	viewLeaderboard	Displays updated leaderboard on home game page; sends request to Controller to getLeaderboard
	viewNav	Displays the the guest navigation bar for guest users and the user navigation bar for registered users
	viewHome	Displays the the guest home page for guest users and the user home page for registered users
Controller	getUserStatistics	Requests user statistics from database and returns results
	postStatistics	Posts user statistics to database
	verifyUserName	Sends request to Validator to verifyUserLogin for username; returns result
	verifyPassword	Sends request to Validator to verifyUserLogin for password; returns result
	loginUser	Sends request to validator to verifyUserLogin to get access token to create login cookie
	logoutUser	Clears login cookie
	createUser	Adds new user to database
	getLeaderbrdStats	Requests leaderboard data from database and returns results

	storeAccessToken	Stores data for user session following verifyUserLogin
	getSongSelectData	Sends post request to database to get song data
	getSliderSpeed	Gets slider speed selected by user
Validator	verifyUser	Requests if user credentials already exists; processes request and returns result
	verifyUserLogin	Requests if user credentials are correct from database; processes and returns results
Database	postUser	Processes postUser and adds user to database
	getLeaderboardData	Processes getLeaderboardData and returns leaderboard data
	getSongData	Processes getSongData and return info data matching requested song
	getUserStatisticData	Processes getUserStatisticData and return statistic data matching requested user
	postScore	Stores score data for matching user
	getUserCredentials	Returns the data for requested user login info
Game	viewSelectSong	Displays select song game menu; sends request to get song data to populate dropdown to Controller to getSongData
	getSong	Requests song data and songmap to Controller to getSongData
	postStatistics	Sends post request to Controller to postScore
	playSongMap	Processes playSongMap and plays songmap correlated to selected song and slider

Table 6.3.1: Domain model attribute definitions

6.4 Traceability matrix

Below is Table 4, which illustrates the traceability matrix for the described domain model. This table shows the relationship between the various domain model concepts and the system use cases described in report 1.

Domain Model	UC-1	UC-2	UC-3	UC-4	UC-5	UC-6	UC-7	UC-11	UC-12	UC-13
Interface	X	X	X		X	X	X	X	X	X
Controller	X	X	X	X	X	X	X	X	X	X
Validator		X								X
Database		X	X	X	X	X	X	X	X	X
Game	X		X	X	X	X	X			
Total Priority	3	4	4	3	4	4	4	3	3	4

Table 6.4.1: Domain model traceability matrix

7 Interaction Diagrams

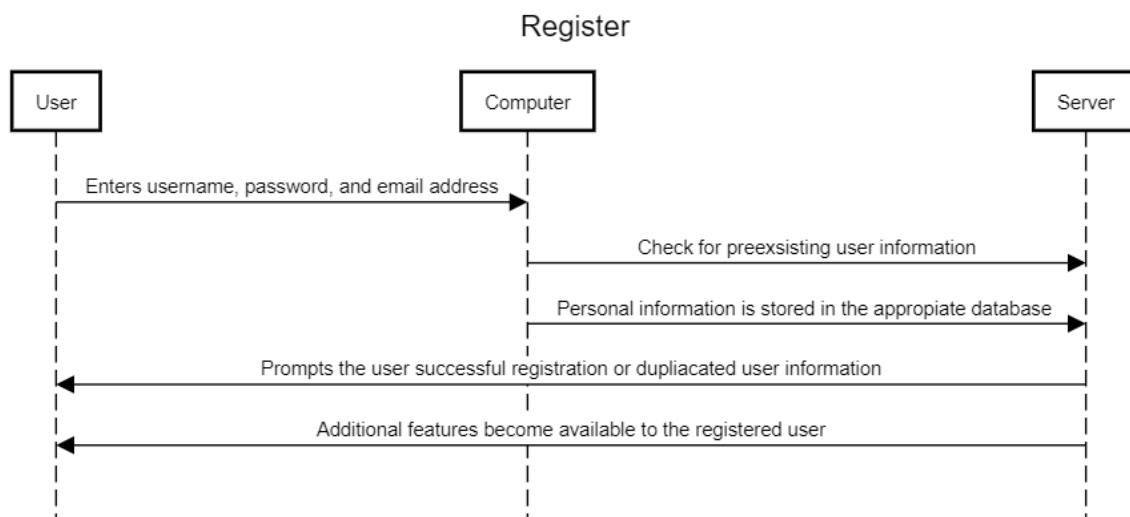


Figure 7.1: Register Interaction

The user can use the home screen to login or logout, or visit a separate tab in the main menu to create an account. This makes registering as a user independent, making the code much easier to read. Design for flexibility is applied when the user is registering because it allows

additional features for only users to be displayed after the user logs in compared to set features. Also, the user can change their personal information which is stored in the database as long as said information does not match pre-existing information being stored, which serves as another flexible option.

The above diagram displaying the Registration design was implemented as originally planned. The only difference was instead of the original plan to stay on one page during the entire process, there are multiple pages, therefore registration will bring you to a page separate from the homescreen. However, the number of pages is limited to prevent the user from becoming lost on the application.

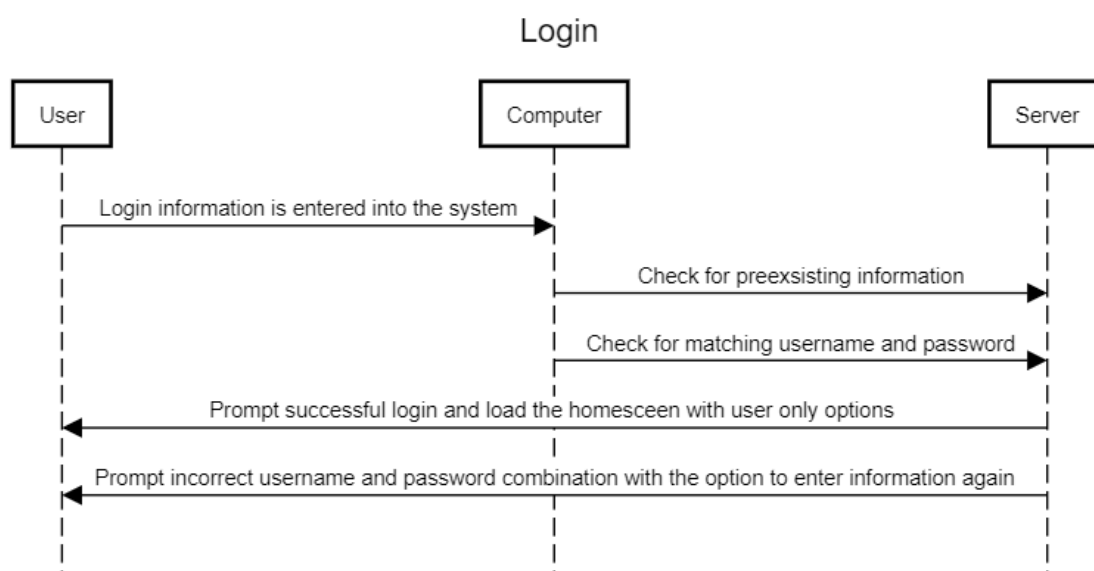


Figure 7.2: Login Interaction

Logging in or out is independent when making the code much easier to read. Increased reusability is applied making it easier to login if the user inputs the wrong information due to a typing error or other mistake. Instead of having to find the login link again, the application will prompt the user that there has been an error with the information entered and bring the screen to try logging in again to the user's screen.

The above diagram displaying the Login design was implemented as originally planned. The only difference was instead of the original plan to stay on one page during the entire process, there are multiple pages, therefore logging in will bring you to a page separate from the homescreen. However, the number of pages is limited to prevent the user from becoming lost on the application.

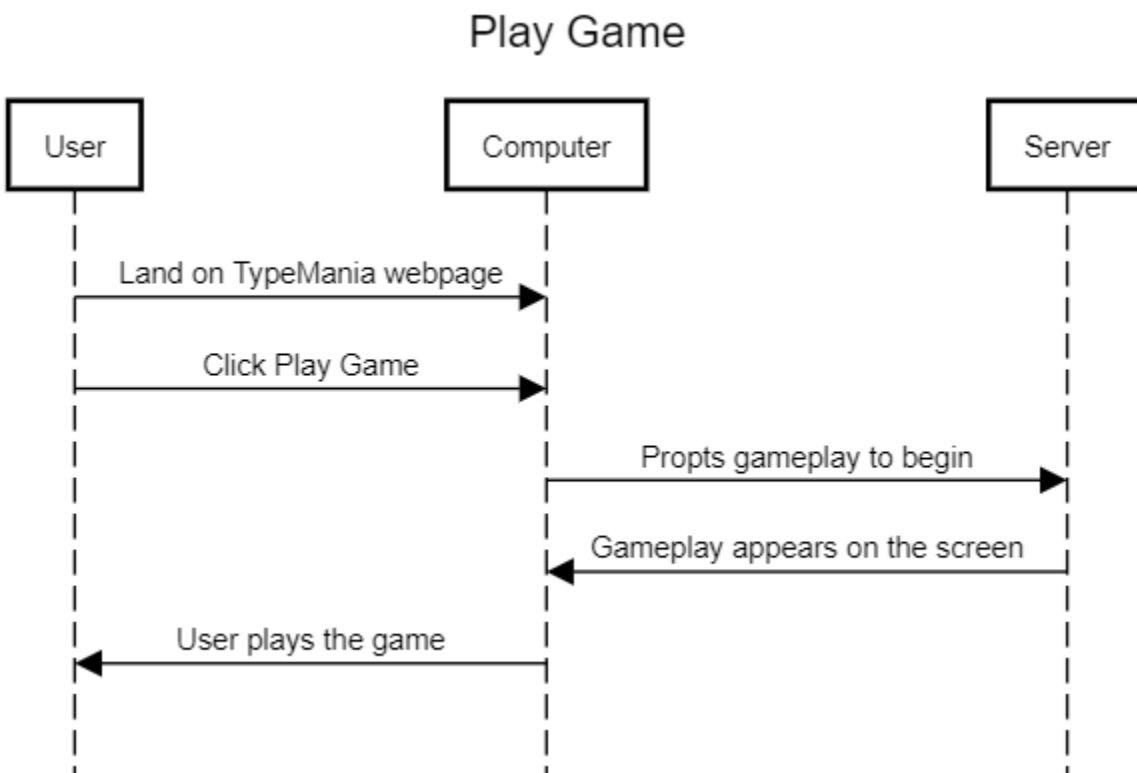


Figure 7.3: Play Game Interaction

Design for flexibility is applied to the gameplay because as the basic prompts will remain the same, the functions within are subject to change as time progresses. Abstraction will also be applied to game play. This allows the code to be duplicated for gameplay purposes rather than writing new code for every scenario the user may encounter.

The above diagram displaying the gameplay has not changed from the original plan. The user can play the game after landing on the webpage and choosing to play said game.

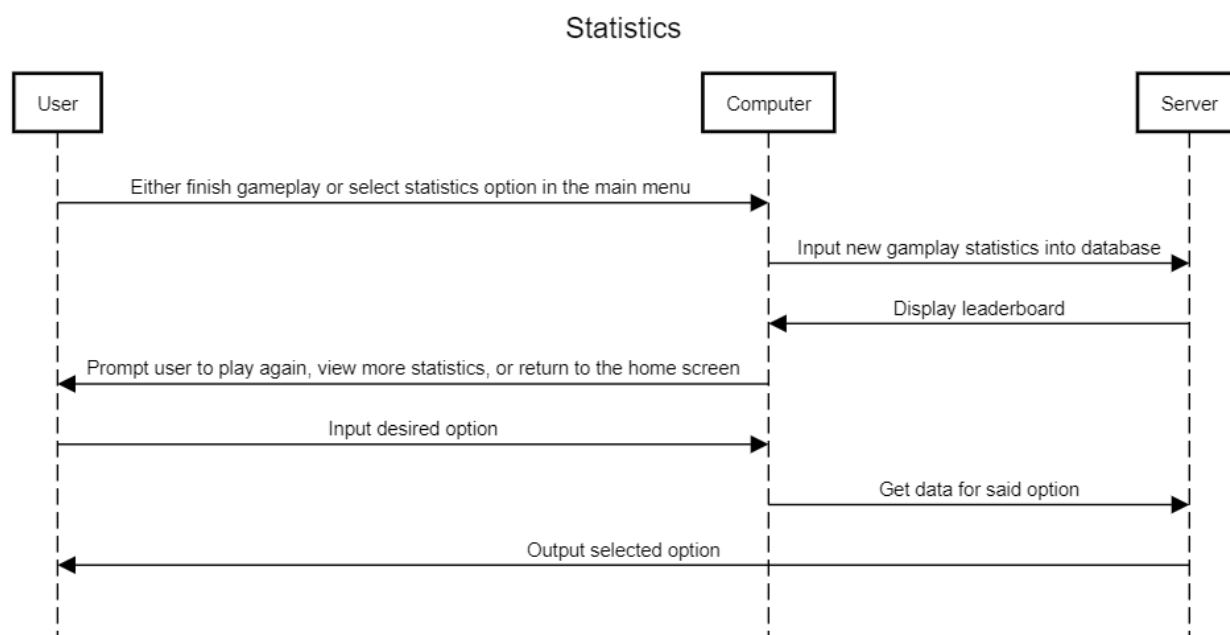


Figure 7.4: Statistics Interaction

Due to the complexity of the statistics page the divide and conquer design principle is applied. The larger system is divided into smaller systems while the diagram shows the user landing on the leaderboard after game play the smaller systems the user can select. These consist of play again which leads to gameplay, returning to the homescreen, or viewing more statistics. Viewing more statistics allows the user to access their full history, past scores, and typing efficiency progress hence the divide and conquer strategy.

The above diagram displaying the Statistics screen has not changed within the design aspect. As some statistics have changed and been removed due to time constraints, the design and how the statistics page functions has remained the same from the beginning.

8 Class Diagram and Interface Specification

8.1 Class Diagram

As illustrated in the diagram, there will be a total of five classes for this application: User, Song, Game, Statistics, and Leaderboard.

The relationships between the classes are described below:

Each User can play multiple Games at different times, however one Game can be played by only one User at a time. Each User can upload multiple Songs and play only one song at a time. Only one Song can be played during each Game. The Statistics will have multiple Games for

each user. The LeaderBoard will have a list of Users and Games, it has a many-to-many relationship between both the User and Game.

Since the Upload functionality is discontinued and will not be part of the final product, there will be no relationship between the User and the Song table. Also, there will not be any customizing UI Settings functionality.

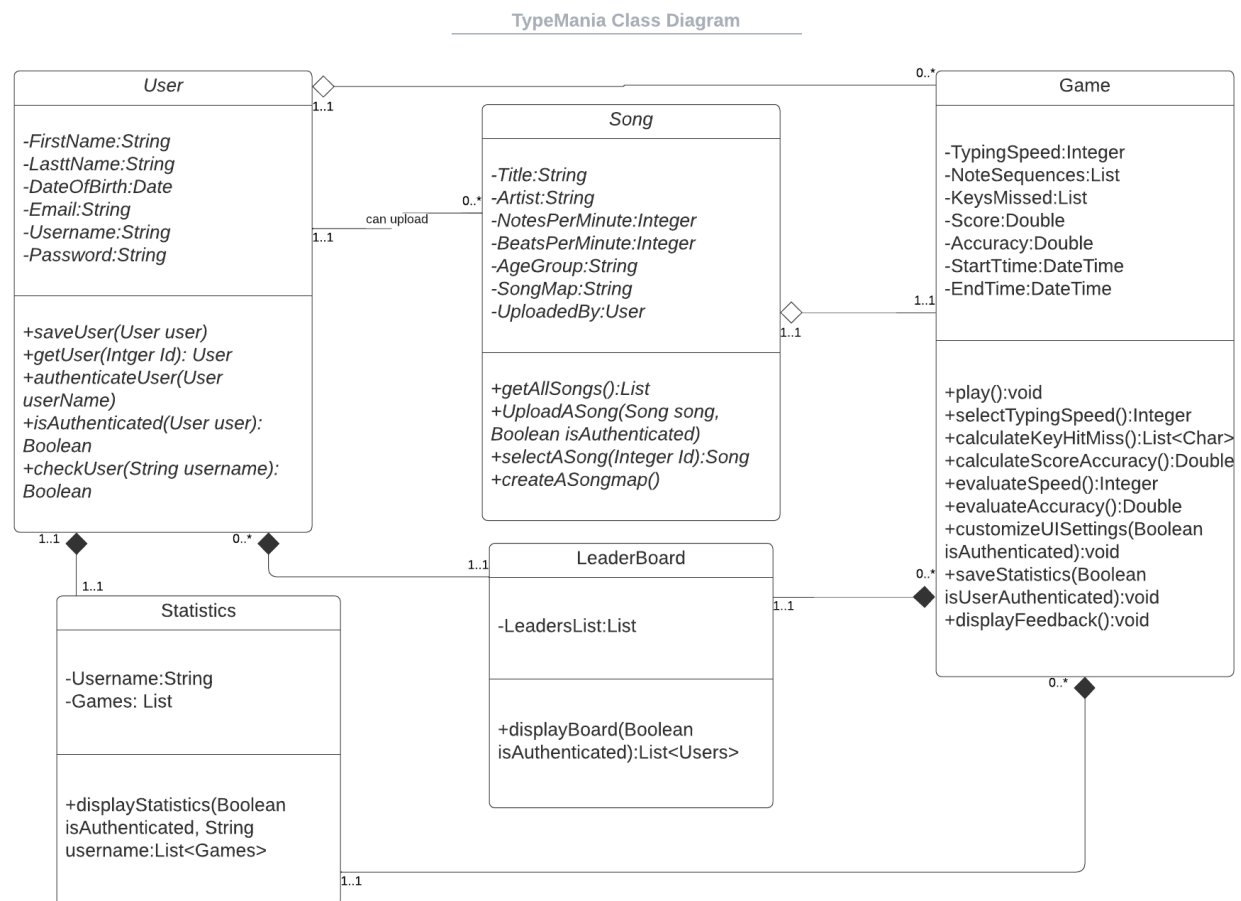


Figure 8.1.1: TypeMania Class Diagram

https://lucid.app/lucidchart/ca78cfcf-e53f-4b87-be80-d883e64634bd/edit?viewport_loc=-255%2C61%2C2603%2C1245%2C0_0&invitationId=inv_e56a1c43-0ee5-4824-9e0a-5d2f24c2957e#

8.2 Data Types and Operation Signatures

I. User Class:

User Class	
The user class performs all functions related to user information including saving a new user for registration, getting user information, and authenticating the user.	
Attributes	
Data Type	Description
-FirstName: String	This stores the user information: first name
-LastName: String	This stores the user information: last name
-DateOfBirth: Date	This stores the user information: date of birth
-Email: String	This stores the user information: email
-Username: String	This stores the user information: username
-Password: String	This stores the user information: password
Operations	
Signature	Description
-saveUser(User user)	This adds the user information to the database as a new user
+getUser(User username): User	This returns the user information data matching the user id parameter
-authenticateUser(User username User password)	This authenticates the user information based on username and password
-isAuthenticated(User user): Boolean	This returns the boolean value for if the user is authenticated
-checkUser(User username): Boolean	This returns the boolean value for if the username parameter matches an existing user

Table 8.21. User Class Data Type and Operation Signature Table

II. Statistic Class:

Statistic Class

The statistic class returns all user statistic data and displays user statistic component to the user interface

Attributes	
Data Type	Description
-Username: String	This stores the username data that will display on the statistics page
-Games: List	This stores the game history of the user that will display on the statistics page
Operations	
Signature	Description
+displayStatistics(Boolean isAuthenticated, String username):List<Games>	This displays the user statistic data on the user interface for the statistics page

Table 8.2.3. Statistic Class Data Type and Operation Signature Table

III. Song Class:

Song Class	
The song class stores all song data and performs all functions to interact with and select the song data	
Attributes	
Data Type	Description
-Title: String	This stores the song information for the title of a song
-Artist: String	This stores the song information for the artist of a song
-NotesPerMinute: Int	This stores the int value for notes per minute
-BeatsPerMinute: Int	This stores the int value for beats per minute
-AgeGroup: String	This stores the string value for age group of a song
-SongMap: String	This stores the string value for the song map that accompanies a specific song

-UploadedBy: User	This stores the user data of the user that uploaded a specific song
Operations	
Signature	Description
+getAllSong():List	This returns the list of all song titles
+uploadASong(Song song, Boolean isAuthenticated)	This performs the action of a user adding new song data to the song database
+selectASong(Int Id): Song	This action selects a specific song and returns the respective songmap
+createASongmap()	This action creates and adds a songmap for the specific song and speed attributes

Table 8.2.4. Song Class Data Type and Operation Signature Table

IV. Leaderboard Class:

Leaderboard Class	
The leaderboard class returns leaderboard data and displays the leaderboard data to the user interface for the leaderboard component	
Attributes	
Data Type	Description
+LeadersList:List	This stores the leaderboard user and score information
Operations	
Signature	Description
+displayBoard():List<Users>	This displays the leaderboard data and react component

Table 8.2.5. Leaderboard Class Data Type and Operation Signature Table

V. Game Class

Game Class

The game class stores all game data and game functionality including game user interface as well as game events and actions

Attributes	
DataType	Description
-TypingSpeed: Int	This stores the user user typing speed for game session
-NoteSequences: List	This stores the note sequences for the selected songmap
-KeysMissed: List	This stores the user data for keys missed and key names for a particular game session and song
-Score: Double	This stores the user data for the score of a particular game session and song
-Accuracy: Double	This stores the user data for the accuracy of a particular game session and song
-StartTime: DateTime	This stores the start time of a particular song
-EndTime: DateTime	This stores the end time of a particular song
Operations	
Signature	Description
+play():void	This begins the songmap for a game session and start time
+selectTypingSpeed(): Integer	This instantiates the selected typing speed for a game session to build the songmap
+calculateKeyHitMiss(): List<Char>	This performs the calculation and returns the data for key accuracy
+calculateScoreAccuracy(): Double	This performs the calculation and returns the data for game session accuracy
+evaluateSpeed():Integer	This evaluates the user typing speed during the game session
+evaluateAccuracy()Double	This evaluates the user accuracy

+customizeUISettings(Boolean isAuthenticated):void	This returns user game interface settings
+saveStatistics(Boolean isAuthenticated):void	This saves user game statistics to database
+displayFeedback():void	This displays user feedback during game session

Table 8.2.6. Game Class Data Type and Operation Signature Table

8.3 Traceability Matrix

The traceability matrix below highlights how our classes relate to our domain concepts. The purpose is to document the motivations behind the creation of such classes. The relationships between classes and concepts tend to be many-to-many. That is, one concept may be connected to multiple classes and one class may be derived from multiple concepts.

Domain Concept	Derived Classes	Description
Controller	User, Game, Song, Statistics, Leaderboard	The controller receives and manipulates data from throughout the entirety of the system. It is concerned with the coordination of all concepts and their respective classes. For example, the systemization of the Game class requesting a Song object from the database.
Interface	User, Game, Song, Statistics, Leaderboard	The interface must interact with every element of the system to display pertinent information to the user. This differs from the controller in that it doesn't work behind the scenes and instead operates in full view of the user to make them a part of the system.
Validator	User	The validator is purely concerned with authentication during login and registration events. Information entered is compared against or stored into the User class's credentials.
Game	Game, Song, Statistics	The game concept emphasizes gameplay and functionality. On top of the gameplay UI, it also handles the Song Select UI and is responsible for posting scores when the user finishes playing a songmap.
Database	User, Game, Song, Statistics,	The database stores data. Classes are inherently a factory for producing organized sets of data so this

	Leaderboard	concept interacts directly with each one of them. Our system will need someplace to hold the objects output by such factories and that is where the database comes in.
--	-------------	--

Table 8.3.1: Traceability matrix linking classes to domain concepts.

8.4 Design Patterns

The TypeMania game is a web-based rhythmic typing game, where the user can access the Interface, select a song and typing speed, and start playing the game. It could have been better if the user had the option to select multiple songs and set the timer on how long they want to play. Since this game is a learning game, it would've been better if the users could decide how long they wanted to practice for, instead of having to manually select a new song.

8.5 Object Constraint Language (OCL)

Object constraint language contracts, found in Table 8.5.1 below, illustrate the invariant, pre-conditions, and post-conditions for the operations related to each class for the TypeMania software application.

Class	Invariant	Operation	Pre-Condition	Post-Condition
context User	self.numUsers >= 0	context User::createUser(user)	pre : !checkUser(username) && containsUserInput(inputEmail, inputUsername, inputPassword)	post : createUser(user) == "created"
		context User::getUser(username)	pre : containsUsername(username)	post : get(username) = "username"
		context User::authenticateUser(username, password)	pre : containsUser(username, password) !containsUser(username, password)	post : authenticateUser(username, password) == "accessToken" != "accessToken"

		context User::isAuthenticated(user)	pre : containsAcessToken() !containsAcessToken()	post : isAuthenticated(user) == True isAuthenticated(user) == False
		context User::checkUser(username)	pre : containsUsername(username) !containsUsername(username)	post : checkUser(username) == True checkUser(username) == False
context Statistic	self.numStatistics > 0	context Statistic::displayStatistics(isAuthenticated, username)	pre : isAuthenticated == True && containsUsername(username)	post : displayStatistics(isAuthenticated, username) == "stats"
context Song	self.numSongs > 0	context Song::getAllSong()	pre : None	post : getAllSong() == "songs"
		context Song::selectASong(Id)	pre : containsSongID(Id)	post : getAllSong() == "songmap"
context Leaderboard	self.numUsers > 0	context Leaderboard::displayBoard(LeaderStats)	pre : containsLeaderStats(LeaderStats)	post : displayBoard(LeaderStats) == LeaderboardComponent
context Game	self.game > 0	context Game::play(songmap, sliderValue)	pre : containsSongMap(songmap, sliderValue)	post : play(songmap, sliderValue) == GameStarts
		context Game::selectTypingSpeed(sliderValue)	pre : containsSliderValue(sliderValue)	post : selectTypingSpeed(sliderValue) = "bpmSpeed"

		context Game::calculateKeyHitMiss (key[], keyAccuracy[])	pre : containsKeys(keys[]) && containsKeyAccuracy (keyAccuracy[])	post : calculateKeyHitMiss(key[], keyAccuracy[]) == KeysMissed[]
		context Game::calculateScoreAccur acy()	pre : GameStarts == True	post : calculateScoreAccur acy() == "gameStats"
		context Game::postStatistics(usern ame, isAuthenticated, gameStats)	pre : isAuthenticated == True && containsUsername(u sername) && GameStarts == True && containsGameStats(gameStats)	post : postStatistics(userna me, isAuthenticated, gameStats) == "posted"
		context Game::evaluateAccuracy(u sername, isAuthenticated, gameStats)	pre : isAuthenticated == True && containsUsername(u sername) && GameStarts == True && containsGameStats(gameStats)	post : evaluateAccuracy(us ername, isAuthenticated, gameStats) == "userAccuracy"
		context Game::displayFeedback(ga meStats)	pre: containsGameStats(gameStats)	post: displayFeedback(ga meStats) = "stats"

Table 8.5.1: OCL Contract Table for TypeMania Classes

(Bruegge & Dutoit, 1999; Object Management Group Standards Development Organization, 2006)

9 System Architecture

9.1 Identifying Subsystems



Figure 9.1.1: User Information, Register, Login, & Gameplay subsystems.

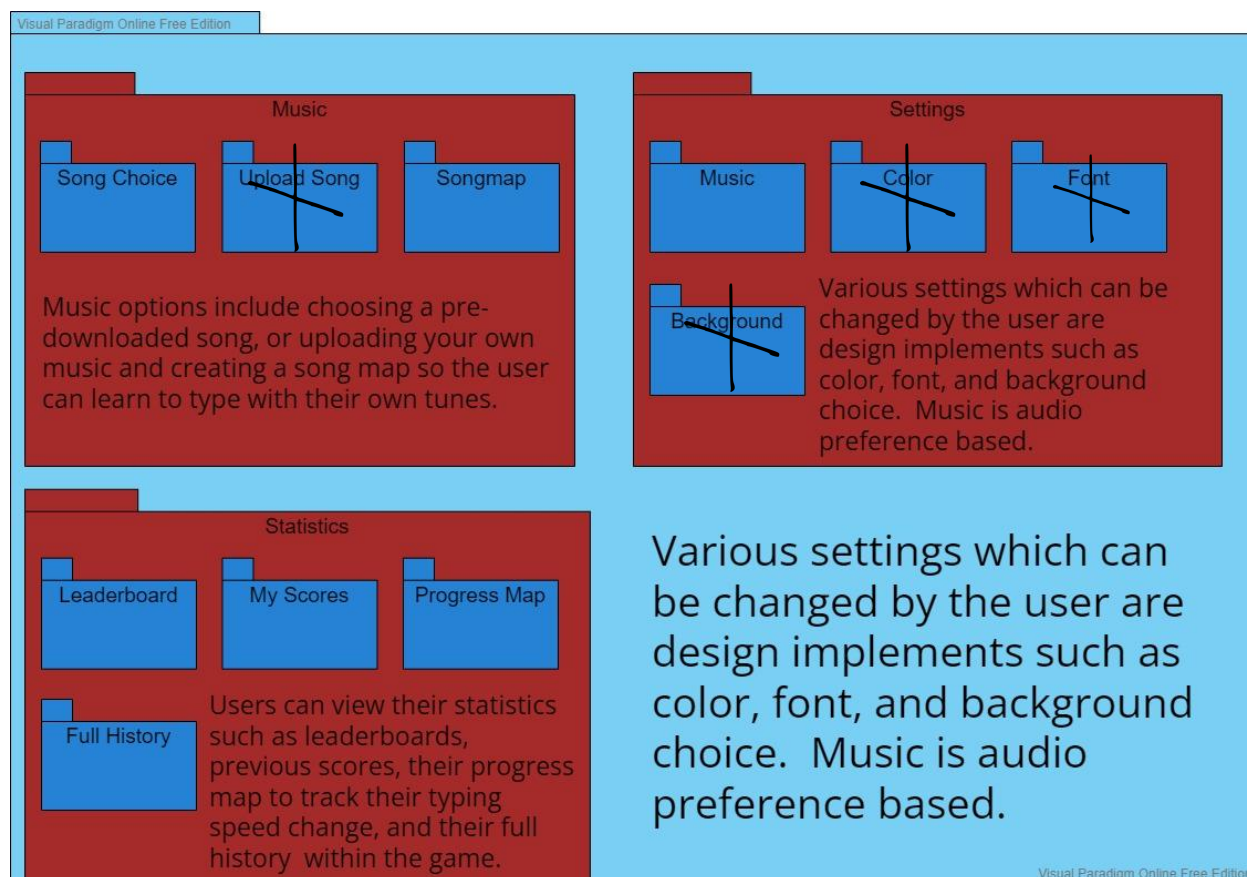


Figure 9.1.2: Music, Settings, & Statistics subsystems.

Each image above is categorized in folders displaying the subsystems of TypeMania. The primary subsystems are user information, registering, login, play game, music, settings, and statistics. Each subsystem contains components that will be found within that area of TypeMania from a development standpoint. The team has decided to use one a one webpage layout minimizing the mapping. However an order pertains to everything being that first the user will either choose to play as a guest, login, or create an account. Then the user will edit their settings and choose their song based on their music preferences. Finally, it is time to play the game. After gameplay the user can view their statistics, play again, or return to the home screen.

As time progresses into the semester, the architecture remains the same however some settings which do not affect gameplay may be put on hold for future releases.

9.2 Architecture Styles

TypeMania displays data-centered architecture when the user first enters the homepage. The center of this is user information stored within a database such as their username, email address, password, security questions, and gameplay statistics. Various features revolve around said data such as logging in, viewing various statistics, and creating a user account.

Subprogram architecture can be found within the data-centered architecture style as well. TypeMania is layered. For example if you go into the various settings the player can change before game play the music option is not just choosing a song. Music consists of choosing or uploading your own song. Then, if the user uploads their own song they can create a songmap to apply to the gameplay in correlation with the song they uploaded. Another example is the statistics menu after gameplay. The user can glance at their score and replay the game or return to the home screen. However the various layers exist so the user can view how they rank compared to the leaderboards or view their history in order to track their progress.

This layered architecture is created so the data is entered on the top level and as more information is gathered, the program works its way down to the core level which is usually the database. In TypeMania this is where the statistics and other user information is stored. This specific architecture is most common due to the fact that it is maintainable, testable, simple enough to divide amongst group members, and easy to update. When updating each layer can be updated separately, making it easier to identify various bugs and errors which may appear.

9.3 Mapping Subsystems to Hardware

The system will be deployed on two separate virtual computers on Heroku. On one machine, a PostgreSQL database will be deployed. On the second machine, the application itself will be deployed. The database will act as a microservice to request queries and respond with results over HTTP. On the second machine, the javascript and static contents that will be rendered on the client side by the user's web browser, will be served. The application will send the queries over HTTP to the PostgreSQL server and respond to the client side with the query result. Also, as the number of users grows, the number of machines could be scaled up to meet higher demand and more traffic.

9.4 Connectors and Network Protocol

The system will use the standard Hypertext Transfer Protocol, HTTP. HTTP is an important protocol between the client and application interface. The users will be able to access the contents of the application via HTTP. They can also send request queries, which are sent to PostgreSQL server by the application, and receive query results all over HTTP.

9.5 Global Control Flow

Execution Orderliness

Overall, our system is event-driven with various procedure driven aspects. When first navigating to the website, the system is event-driven, in which the user has multiple procedure-driven actions they can generate. These actions include guest play, register, and login. If a player chooses to login, this triggers a larger event-driven experience, including more procedure-driven actions, like song select and your stats.

Each procedure-driven action has linear steps the user must follow to complete. For example, if the user chooses to register, they click the register option, enter in the required text inputs, click submit, then login at the login page. Or if the user chooses to play game, they must choose difficulty with the speed slider, click song select and click the chosen song they want to play to begin the game.

The actual game play is also procedure-driven as the screen shows various notes the user must select as the song plays. Once the song ends, the game ends, and the user is shown their score. Then, the system is event-driven as the user can decide to play again, exit to the home page, or view statistics, all of which have varying subsequent event-driven and procedure-driven actions to be performed.

Time Dependency

There is event-response time dependency for two aspects in this system with no concern for real-time: user login and game play. Because this is a web-based game with a user login feature, there will be an event-response time dependent aspect regarding the login of a user. For example, once the user logs in, after a period of inactivity, the user will be automatically logged out. For example, if the user does not manually logout, the user will be automatically logged out after a period of 24 hours of inactivity through the expiration of session cookies in the user's browser.

The game play also has an event-response time dependency. For example, each game play session will last the duration of the selected song. Therefore, the game play is time dependent on song length. Once the song completes, the game will end. As a result, each game play session is dependent upon the selected song.

9.6 Hardware Requirements

Our system is a web application that will be hosted via Heroku. That is, the server-side equipment is outsourced by Heroku. The client-side hardware can be found in most modern laptops and desktops and refers to the hardware requirements for users.

Server-side hardware requirements

Database storage space

The hosting database will need enough long term storage space to hold several songs and quality graphics. The amount of songs is expected to grow continually over time, so the expected hard disk space required for the term is approximately 512MB.

Internet connection

The database server must be able to communicate with its clients from a remote location. To communicate in this way the database server will make use of the internet. To serve multiple clients simultaneously, it is recommended that the database's network bandwidth be able to transfer at a rate of at least 512kbps.

RAM / Main Memory

The database server must communicate with only a few clients at one time with a relatively small database. The database server will utilize 512MB of RAM to handle these interactions.

Client-side hardware requirements

Color display

The user should be able to see the user interface in order to interact with it properly. The user's computer system should have access to a monitor to view the graphics. The minimum recommended resolution for such a monitor is 640 x 480 pixels.

Physical keyboard

The user must be able to see most of their display whilst they type, so on-screen keyboards aren't recommended. The minimum character set makes use of the alpha characters, so the user should have access to a physical keyboard that can emulate such keys.

Internet connection

The user must access the database to request the code to be loaded to their computer. To make this request, internet access through a modern web browser is required. Because the game makes use of graphics, a minimum bandwidth of 128kbps is recommended.

RAM / Main Memory

The user must use a modern web browser for the game to load properly. Between the user's OS, web browser, and web application, a RAM of at least 2GB is necessary.

10 Algorithms and Data Structures

10.1 Data Structures

Our software must store sequences of integers. The integers are not unique and may be repeated multiple times within the structure. Each integer represents a character that will appear on a note. The notes will appear one after another in the order that the sequence presents them. The initial sequence of integers is linear and requires shuffling. To shuffle effectively, the structure should be able to read/write to an element without having to traverse the entire sequence to reach said element. From what has been stated above we can derive these requirements for our data structure:

- Sequentiality - The structure must support chronology.
- Duplicability - The structure must accept repeated elements.
- Mutability - The structure must allow data within to be modified.
- Lookup - The structure should implement quick access to elements.

Sequentiality is not the strong suit of hash tables and sets void duplicates. Our need for a quick lookup renders linked lists ineffective and mutability is disallowed in tuples. Arrays support chronology, accept duplicate elements, allow data manipulation, and have an effective lookup via indices. Arrays are the necessary blend of flexibility and performance and are our project's data structure of choice for these reasons.

11. User Interface Design and Implementation

11.1 Preliminary UI vs. Current UI Design

The purpose of this section is to highlight major deviations from the initial design proposals found in report #1. Motivations & reasons for not strictly adhering to the plan will be provided below each pair of images.

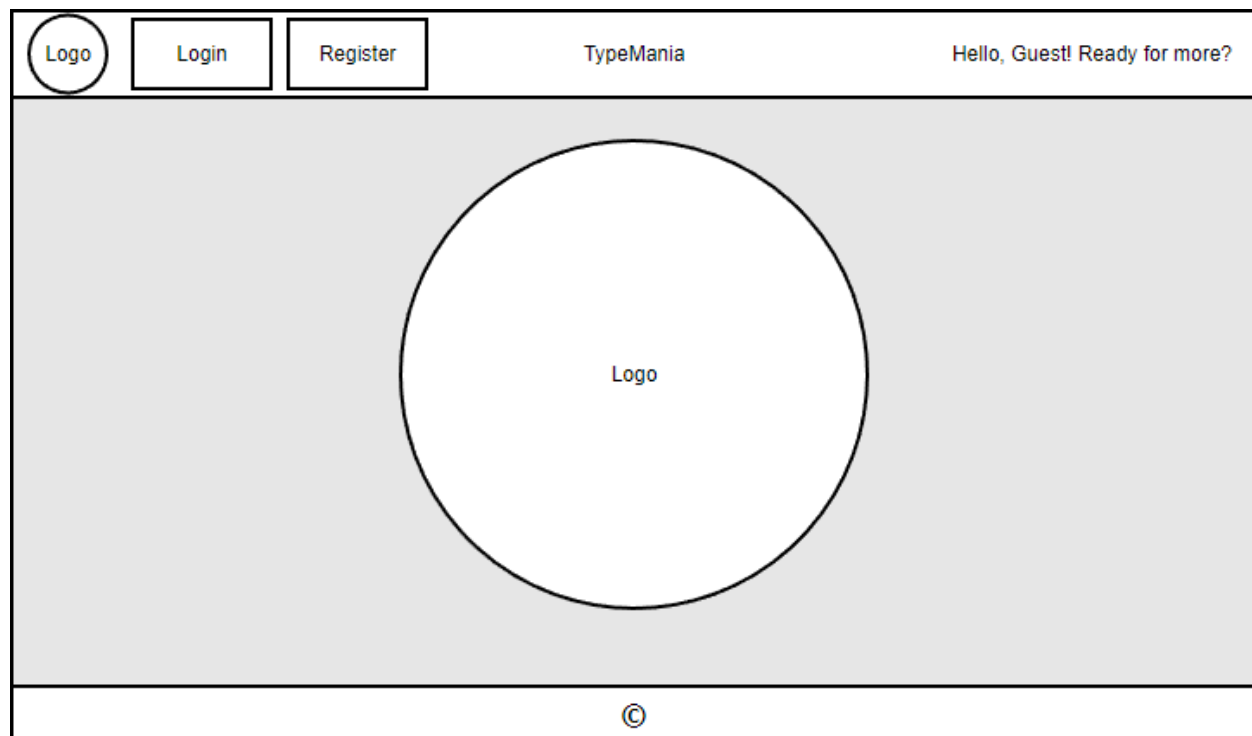


Figure 11.1.1: Preliminary design of guest landing page as of Report #1.



Figure 11.1.2: Current design (11/20) of guest landing page as of Report #3.

Implementing a main menu in Phaser has proven extremely difficult. We only have a rudimentary “Click to Play” thanks to the power of React which allows us to effortlessly cover our other web page elements. The first major deviation that sticks out here is the center of the screen. Instead of a fancy logo we simply have a big black drape. Another deviation is in how the navigation is laid out. Instead of a thin top layer of options, we have opted to make our main header “TypeMania” larger and more eye-catching.

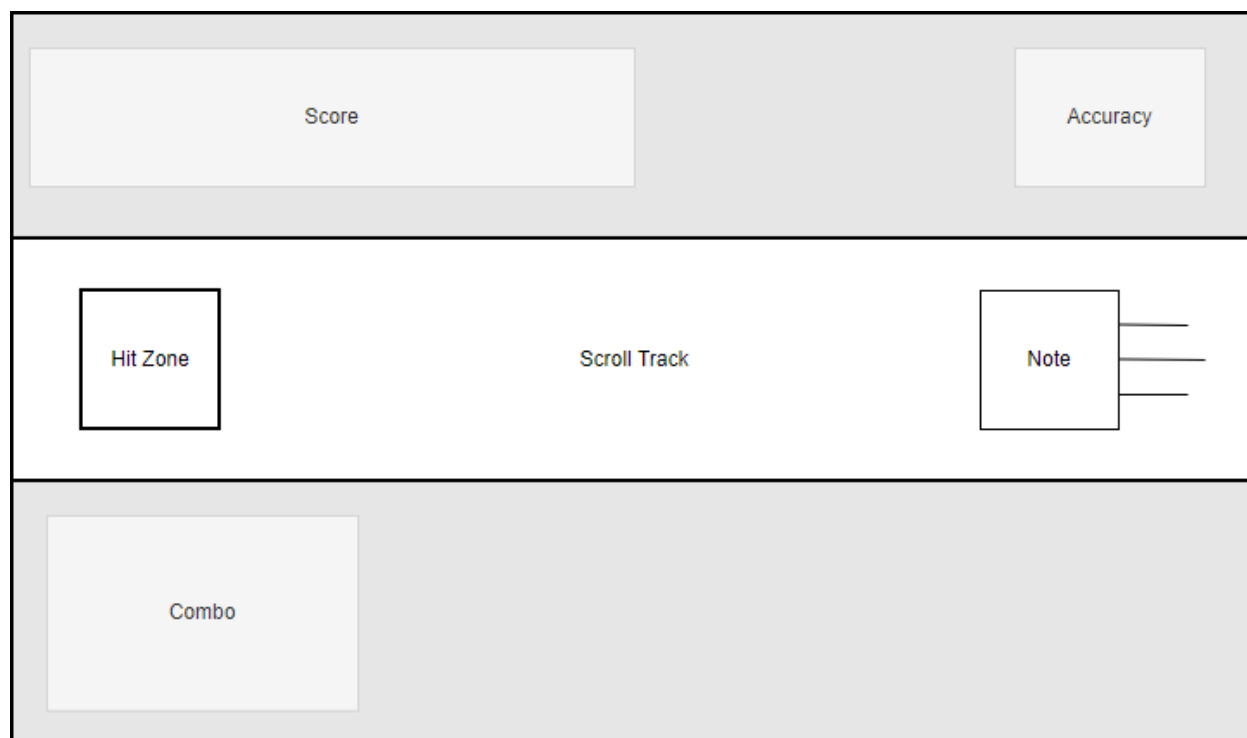


Figure 11.1.3: Preliminary design of gameplay scene as of Report #1

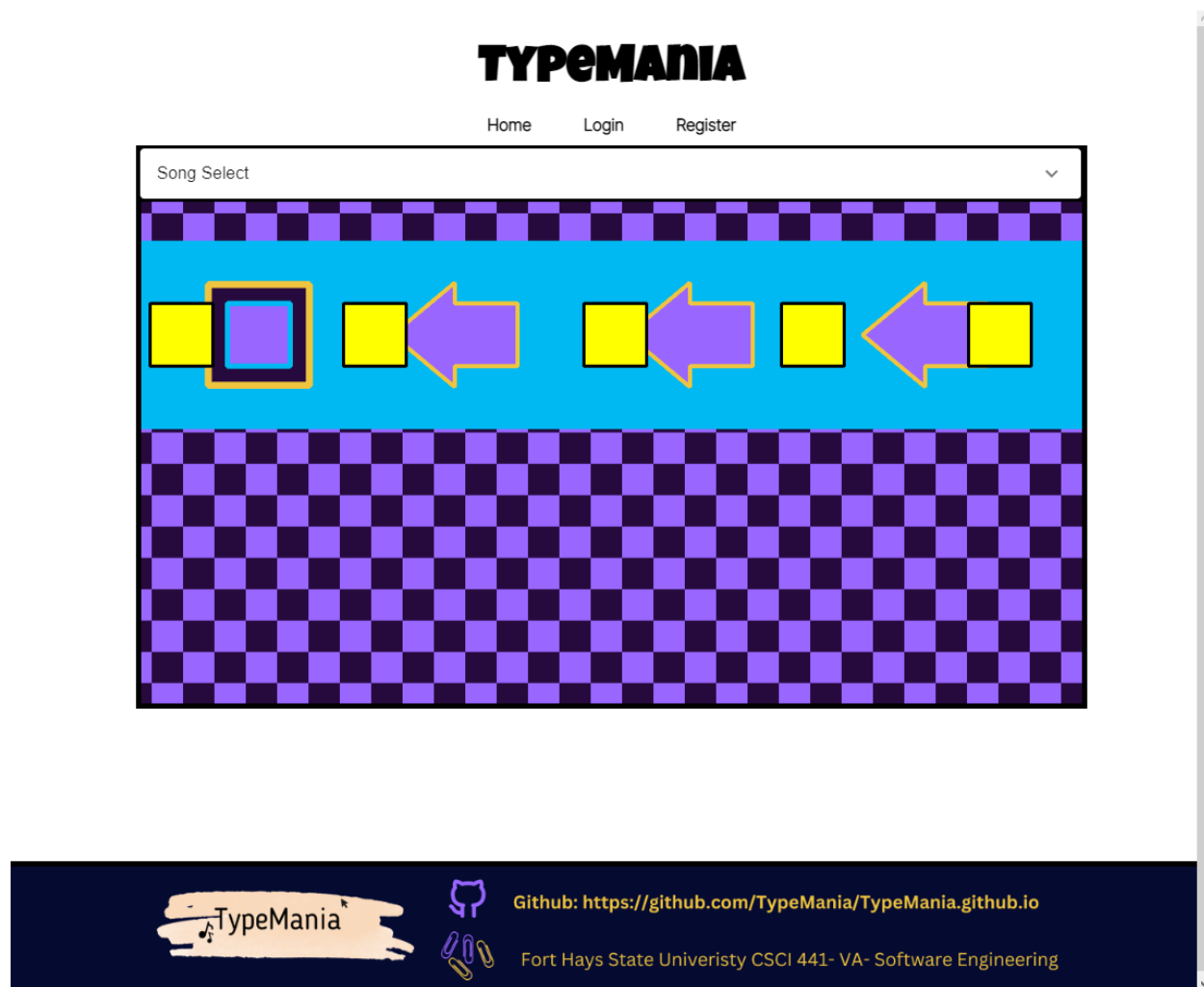


Figure 11.1.4: Current design (11/20) of gameplay scene as of Report #3.

The preliminary design has been respected for the most part except for one thing. The song select was supposed to be its own menu option but because there is no menu (figure 11.1.2) it has become a part of the gameplay scene. Since the upper quarter of the gameplay scene is now occupied, we must rethink the positions of the planned score & accuracy components.

<div>Logo</div> <div>Login</div> <div>Register</div>		TypeMania		Hello, Guest! Ready for more?	
Leaderboard for Selected Song		Song		Selected Song Information	
Player : Score		Song		Song Name	
Player : Score		Song		Artist Name	
Player : Score		Selected Song		Difficulty Rating: Notes Per Minute	
Player : Score				Beats Per Minute	
Song					
Modifiers	<div>Playback Speed</div> <div> <input type="range"/> </div>	<div>Scroll Speed</div> <div> <input type="range"/> </div>	Character Options	Settings	
©					

Figure 11.1.5: Preliminary design of song select screen as of Report #1.

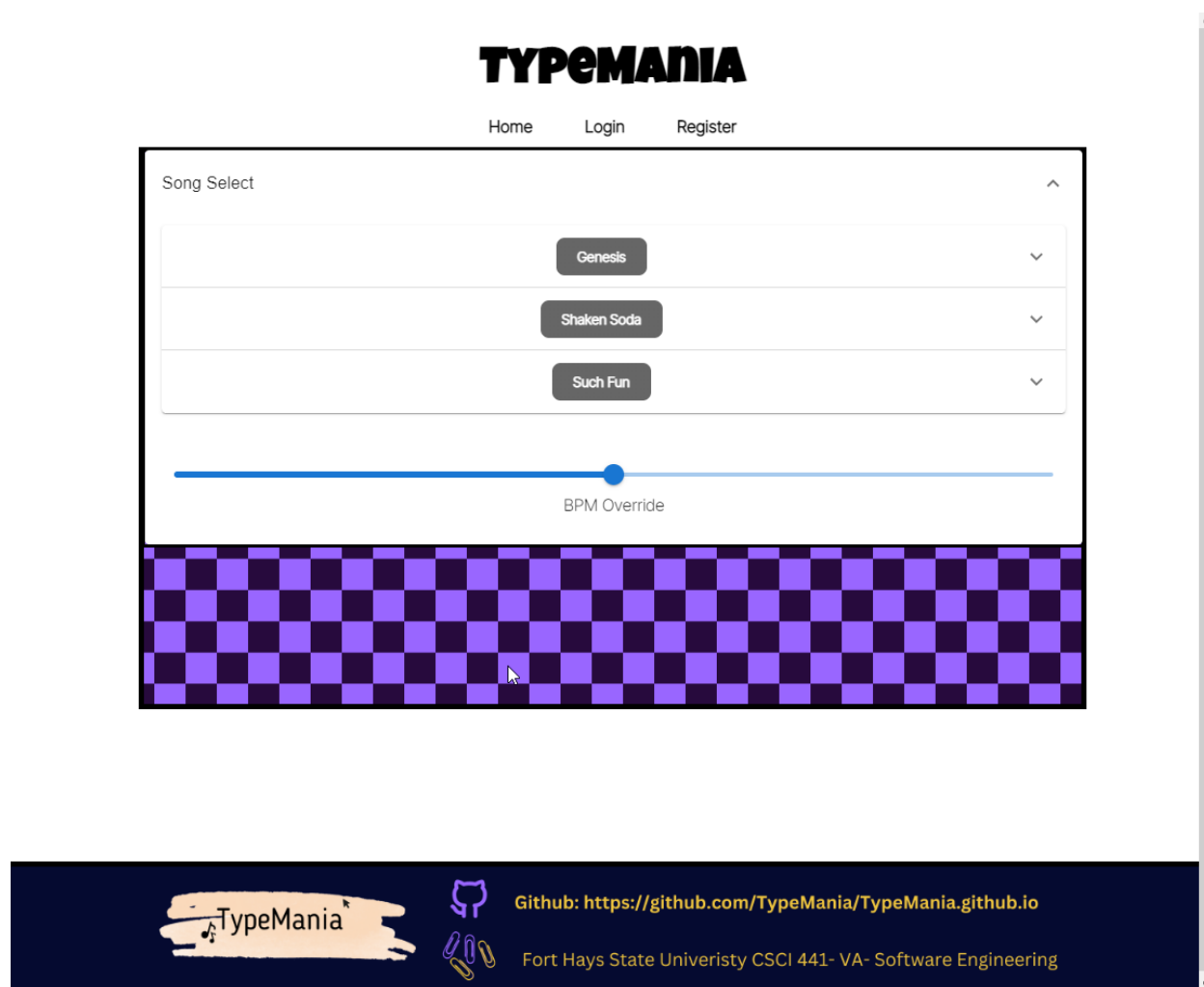


Figure 11.1.6: Current design (11/20) of song select screen as of Report #3.

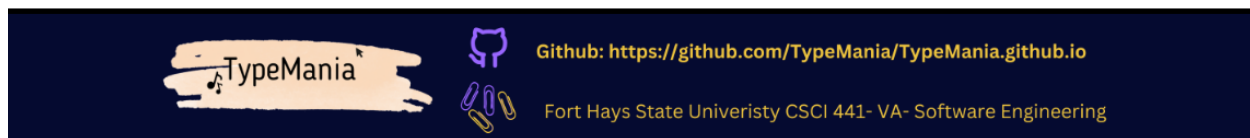
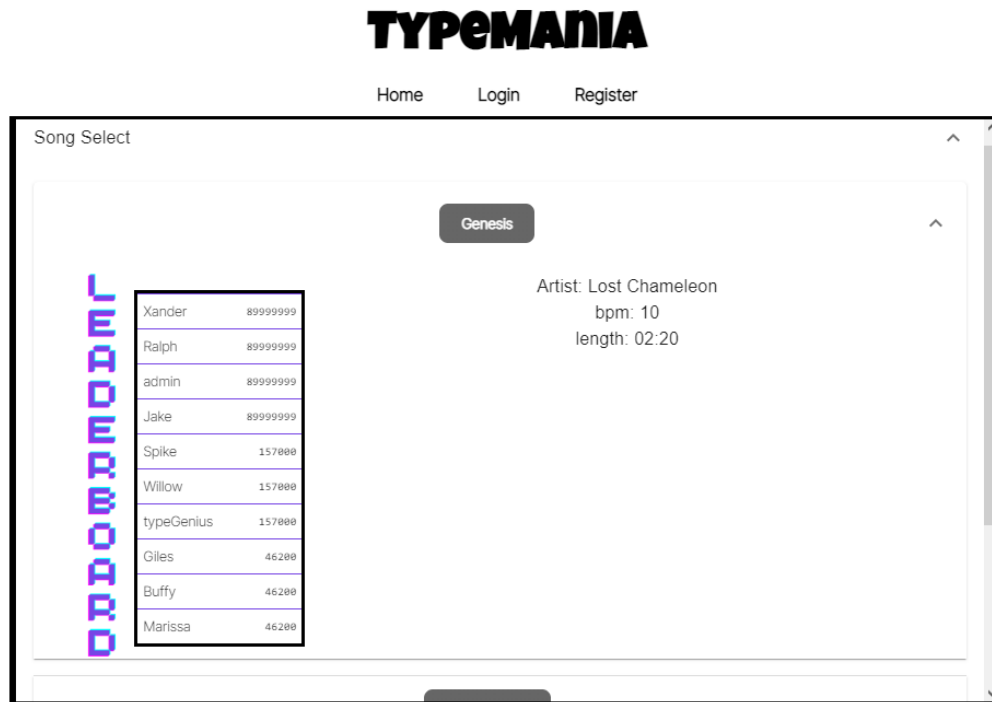


Figure 11.1.7: Current design (11/20) of song select screen with a song selected as of Report #3.

The wheel/carousel was traded out for a less graphically demanding accordion. A replacement that still serves our purpose nonetheless.. The major deviation here is the lack of difficulty modifiers as seen at the bottom of figure 11.1.7. Due to time constraints, we will not see much more in the area of difficulty modifiers than the BPM override that was seen in figure 11.1.6.

These are of course not the only alterations but they are the most notable. The next section details feature by feature what was implemented and gives some more reasons on why it was updated if it had a previously functional version.

11.2 Implementations

Based on the development of TypeMania conducted thus far, there have been some modifications to the design throughout the software's planning and implementation process. More modifications will likely be made as the development of the web app continues. This may be related to design/ease-of-use improvements as well as to improve development and implementation processes.

Firstly, the navigation bar design has been modified and implemented to list a larger variety of options than what was proposed in the first draft. The first draft contained only login and register links in the navigation bar, whereas the currently implemented version contains home, login, statistics, upload song, and register links, see Figure 11.2.1. Additionally, the logo is clickable to act as a second homepage navigation link. Having more options in the navigation bar increases the ease-of-use of registered users, specifically related to the statistics and upload song items. With this change, these items are easier to navigate to by having a constant presence on the webpage. This strengthens our design's user control and freedom as well as consistency, which are two key components of successful user interface design (Nielson, 2020). ~~Lastly, if a guest user selects a link that requires a registered account, the guest is directed to the login screen, where they can login or navigate to the register page.~~

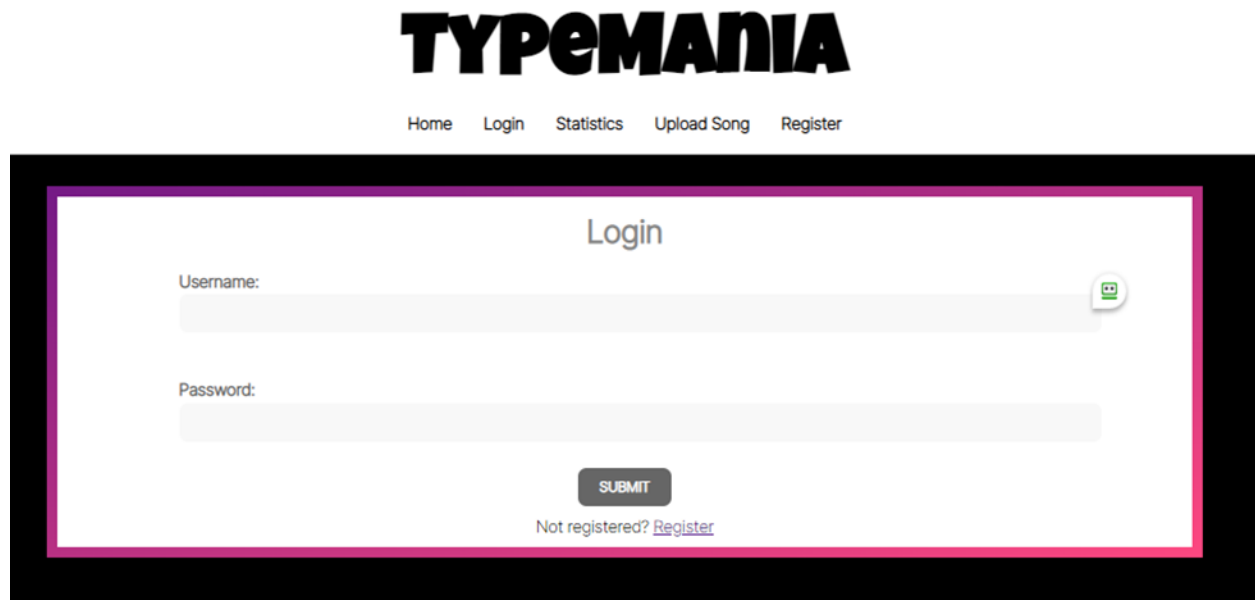


Figure 11.2.1: Discontinued TypeMania navigation bar for guests.

Navigation links that require a registered account are now hidden from guests. While this certainly conceals functionality from the average first-time guest user, we've found better ways to generate interest in registered features. That being by allowing guests to play the game first and showing that their scores can be ranked on leaderboard.

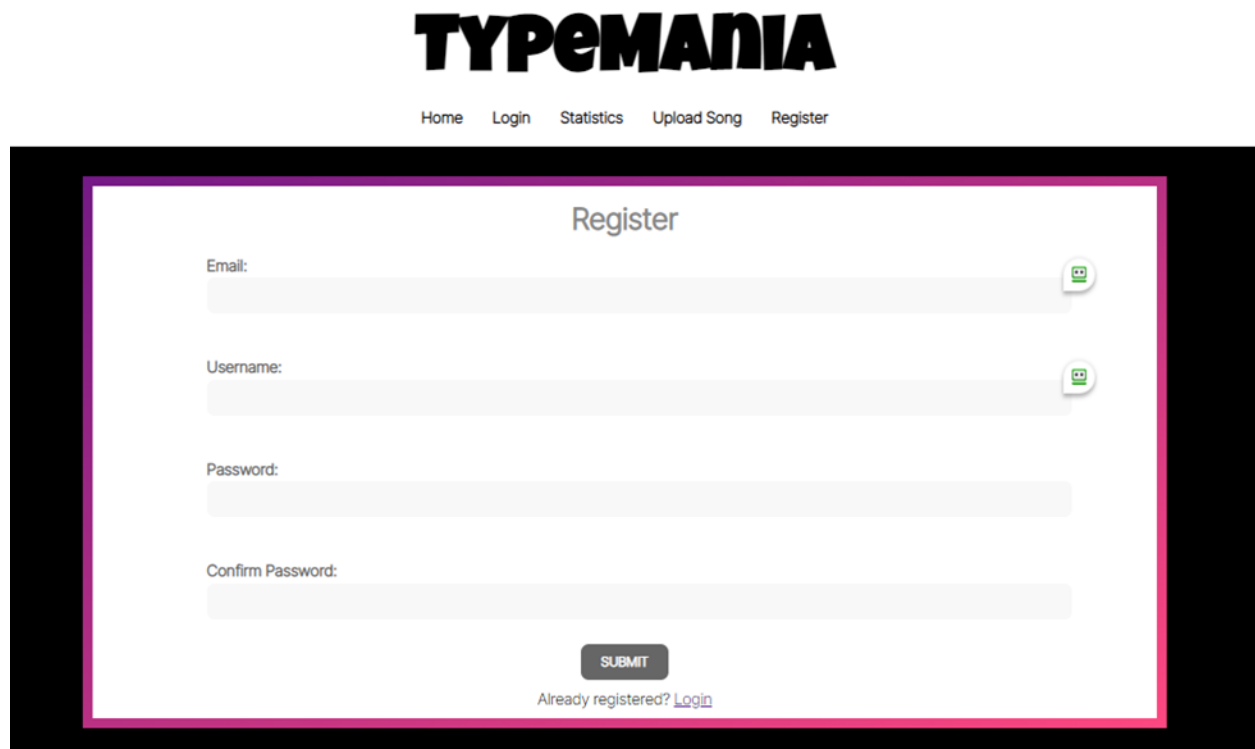
The login page is very similar to the previously proposed interface design; however, it also contains an additional link to navigate to the registration page. Below the submit button sits the text, "Not registered," followed by a Register link, see Figure 11.2.3. This strengthens user control and freedom as well as consistency and help/communication (Nielson, 2020). Moreover, the registration page contains the text "Already registered?" followed by a login link underneath its submit button, see Figure 11.2.4. These two additional messages and navigation links also help to improve user ease-of-use, reducing their memory load, following logical communication, and enabling easy, seamless navigation (Nielson, 2020). The similarity between the login and

registration form makes it easier for the user to transition between steps. Moreover, each form will display information messages and error messages to guide the user to enter the correct, required input to successfully process the form. This helps to meet successful interface design requirements, including error prevention, diagnosis and recovery (2020).



The image shows a web browser window with the TypeMania logo at the top. Below the logo is a navigation bar with links: Home, Login, Statistics, Upload Song, and Register. The main content area is a login form titled "Login". It contains two input fields: "Username:" and "Password:". The "Username:" field has a small green icon with a speech bubble on the right. Below the "Password:" field is a "SUBMIT" button. At the bottom of the form, there is a link: "Not registered? [Register](#)".

Figure 11.2.2: TypeMania login form implementation



The image shows a web browser window with the TypeMania logo at the top. Below the logo is a navigation bar with links: Home, Login, Statistics, Upload Song, and Register. The main content area is a registration form titled "Register". The form contains four input fields: Email, Username, Password, and Confirm Password. Each input field has a small green icon with a white 'x' in the top right corner. Below the input fields is a "SUBMIT" button. At the bottom of the form, there is a link: "Already registered? [Login](#)".

Figure 11.2.3: TypeMania registration form implementation

The landing page, see Figure 11.2.4, has had some design alterations related to implementation of the various development components and will still be undergoing design improvements. The current in-progress landing page features the navigation bar, leaderboard component, and gaming component. The leaderboard component, magnified in Figure 11.2.6, is an in-progress design and will be tweaked and improved in future development. This component helps to promote visibility of system status, which is a key element of good interface design (Nielson, 2020). It also helps to entice users to play TypeMania more and compete with friends and other leaders. The leaderboard component will be visible to both guests and registered users.

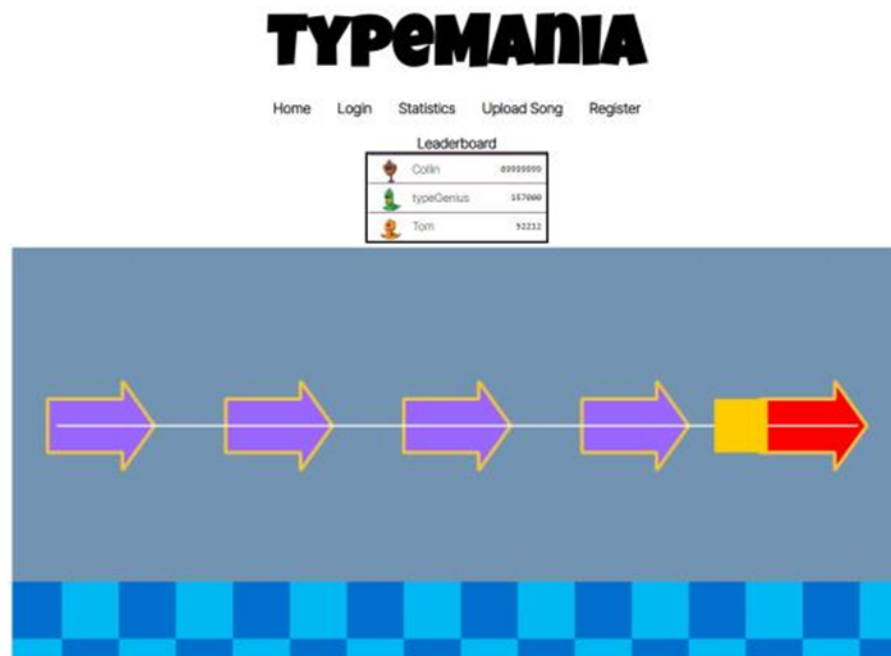


Figure 11.2.4: TypeMania registered landing page as of early Report #2.

We have since added to the design since report #2. Our latest gameplay scene is as imaged below in figure 11.2.5. A song selection menu has been added to the user interface in time for the first demo. On top of that, a slider that allows the user to control the speed of the animations and music has been added (see 11.1.6). The functionality is limited to animations for now but future iterations intend to override the playback speed of songs.

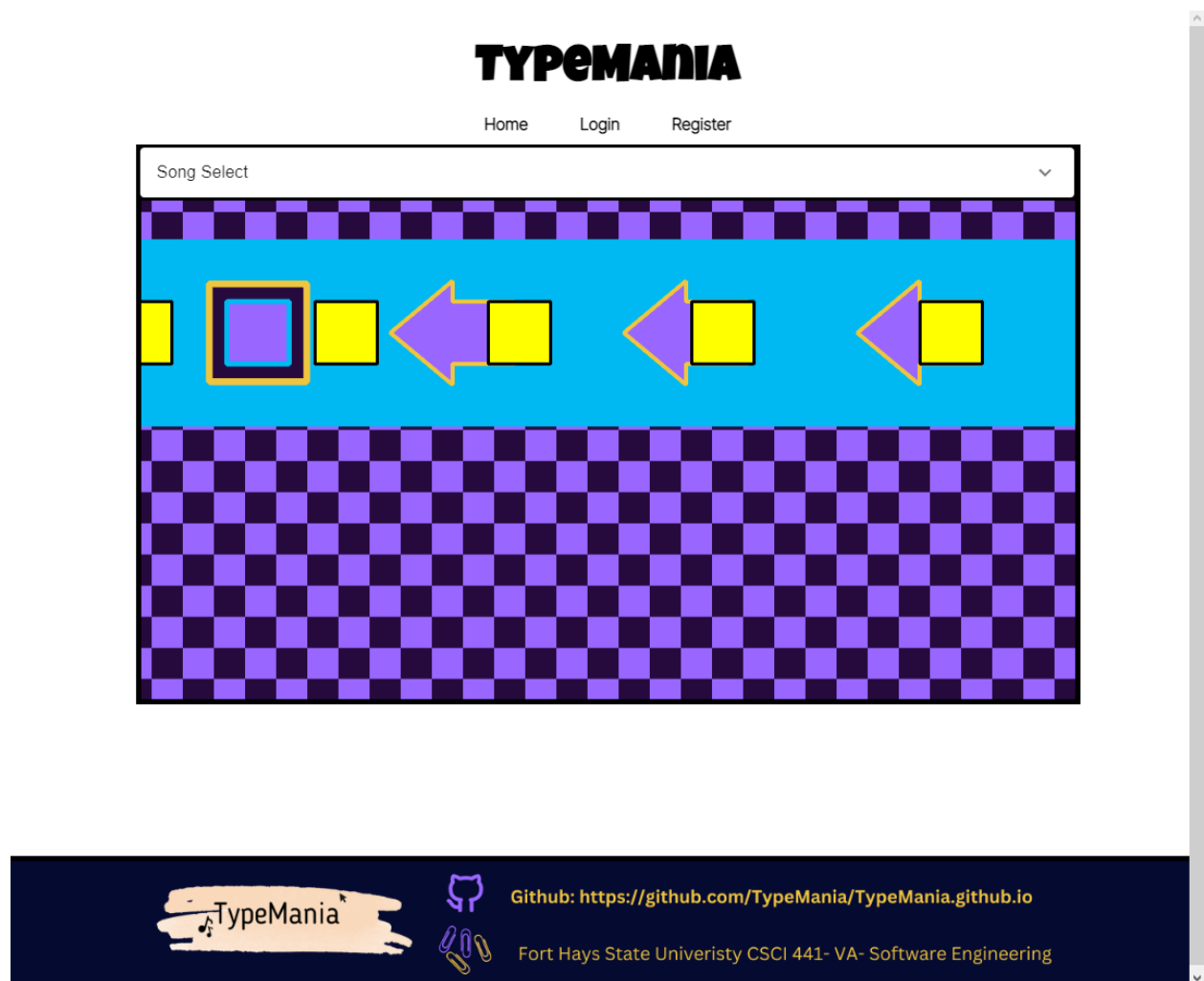


Figure 11.2.5: TypeMania Landing Page as of Report #3.

Leaderboard




	Collin	89999999
	typeGenius	164584
	Tom	46288

Figure 11.2.6: TypeMania In-Progress Leaderboard Component

The game component shown on the landing page is also an in-progress element. The presence of the game component will be consistent for both guest and registered users, but registered users will have a more customized game user interface, which will promote the key design principle of flexibility and efficiency of user by enabling customizable experiences for expert, or registered, users (Nielson, 2020).

The gameplay scene is not yet implemented in the gameplay component. We still cannot be considered as playing the game until there are characters on the notes and they disappear on the corresponding keypresses. Slight changes will be made in the gameplay scene, see Figure 11.2.8 that will improve the user experience and ease-of-use once implemented into the game component. First the combo section from the first design is removed from the bottom left corner to create a more simplistic component that will not be as overwhelming for the user. Additionally, the flow of notes to the hit zone have been flipped to follow a left to right sequence, which will be more helpful to those of languages that read left to right. This flow of notes may be a customizable option in future installments. Notice the target key notes are shown in the blue squares, which will move to the hit zone, where the user must type at that time. The scene also features the score and accuracy similar to the prior design. All of these aspects of the gameplay component improve user experience providing real-time statistics during gameplay.

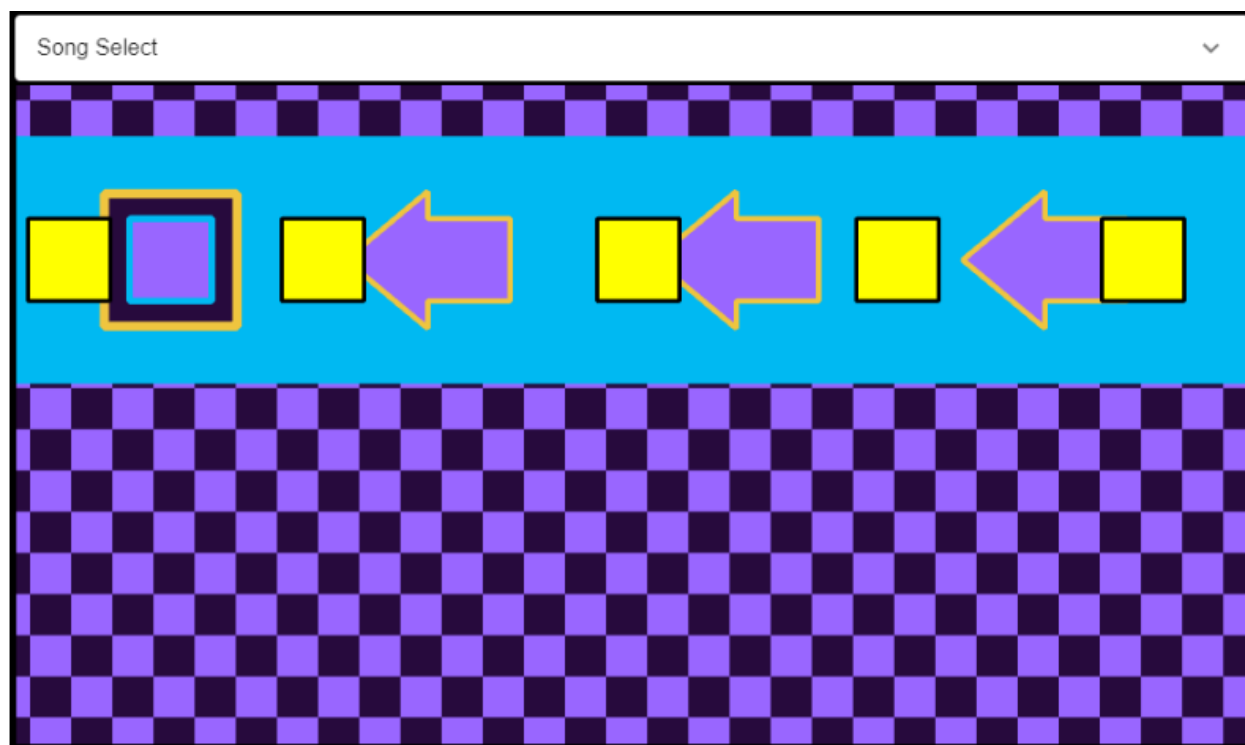


Figure 11.2.7 Current gameplay scene (11/20).



Figure 11.2.8: Gameplay scene to aim for, but reversed.

The implementation of the gameplay interface, statistics page, and upload song page are still in development. Currently, each of these elements will still follow the proposed design; however, they will likely be changed as development continues and implementation is underway. During this process, component usage flow is better understood and tweaked for efficiency and ease-of-use. Additionally, implementation barriers can also cause adjustments in design to ease the stages of development.

12 Design of Tests

12.1 Unit Testing

Test Case	Description
Deadlink Checker	This code will check for links that either do not work or lead to nowhere.
Home page loads	Code prompts for the homepage to output the user's screen.

Login works for existing users	Code attempts the login process of a user successfully in the system and they are successfully logged in.
Additional features are available to registered users.	Code checks that additional features are available to the user after login is successful.
Login error message prompts for invalid information entered	Code entered invalid user information into the system while logging in to ensure the error message prompt appears.
Statistics screen displays up to date information	Code checks the last time the statistics screen was updated. The last update should be the last time a gameplay was successful.
Statistics screen appears to the left of gameplay	Code checks the statics screen is visible to the left of the gameplaly screen.
Statistics information changes depending if the the user is a guest or a registered user	Code checks the information available to registered users only appears after a user has successfully logged in.
Music functions work during gameplay	Code checks the music features played during gameplay.
Registration creates a new user	Code attempts registering as a new user with randomized information to ensure a new user can be created. However after the unit testing is complete the information will be deleted.
Registration prompts an error message if the user is pre-existing	Code attempts registering as a new user using random information.
Gameplay function displays working background	Code attempts to enter gameplay as a user ensuring basic gameplay functions are working.

Table 12.1.1: Test Cases and Descriptions

12.2 Testing Coverage

Test Case	Testing Coverage
-----------	------------------

Deadlink Checker	This will check every link in the program to ensure there are not any deadlinks. The deadlink checker will be automated, with written code to check all the links. As the amount of individual links is hard to approximate before the completed project, the number will undoubtedly be large between 50-100 making it very time consuming to perform manually.
Home page loads	This will check to make sure the home page loads properly for the user. The code will be directed to the home page link and ensure no error codes are returned.;
Login works for existing users	Testing will check the database for pre-existing information and ensure the inputted information matches the information in the database. Only then will the user access additional features of TypeMania.
Additional features are available to registered users.	Testing will ensure the appropriate links are successfully loaded onto appropriate menus and work for the user after successfully logging in.
Login error message prompts for invalid information entered	This will check the error message after a user inputs the wrong login information. The system will be looking for typical error codes from the system, and ensure the pop up works correctly via hyperlink.
Statistics screen displays up to date information	Testing checks and outputs to the programmer the last time successful gameplay was completed and the last time the statistics were updated for the individual user and all users of TypeMania.
Statistics screen appears to the left of the gameplay screen	The system will ensure, via hyperlink, the statistics screen automatically displays to the left of gameplay. This primarily features the leaderboard for said song the user finished with.
Statistics information changes depending if the the user is a guest or a registered user	After a user has successfully logged in, the system will check that the appropriate links are activated so the user can access those

	features. The testing will also check those features are not available to a guest user because there would not be enough information to make said links useful to the player.
Music functions work during gameplay	Testing will play music so the user can hear and utilize the song they chose to play with. This includes volume at an appropriate level and the correct song being played.
Registration creates a new user	Testing ensures duplicate information is not accepted into the system and stores successful registration information in the appropriate database.
Registration prompts an error message if the user is pre-existing	When a user attempts to register with information that is already in use an error message is prompted. This will be tested via activating the correct hyperlink.
Gameplay function displays working background	Testing ensures the appropriate background is displayed for the user. This includes loading the correct screen with the correct colors, framework, and display options.

Table 12.2.1: Test Case and Testing Coverage

12.3 Integration Testing

Bottom up integration will be applied to TypeMania, we will test the smaller functions and proceed to testing the bigger picture progressively. First, all of the unit tests will be run, therefore if there are any bugs they will be easier to localize. The unit tests were designed to focus on the main functions of the application and if any tests need to be added as testing begins that can be done. After unit testing is successfully completed, the tests will become more broad. Being that TypeMania is a smaller application compared to major corporations there are few steps to working up the ladder. The second level of testing will include: registration, logging in or out, song upload, testing each individual statistic as a whole function, and testing each individual setting as a whole function. The last level of testing will include: statistics, settings, login, registration, and gameplay as those are the main headings in the menu. Running the entire program as a whole at one time and experimenting during gameplay will be the final test to ensure the user experience is of utmost quality.

13 History of Work, Current Status, & Future Work

13.1 History of Work

The first milestone planned for our project was to achieve a minimally functional prototype. This was meant to establish the core gameplay and to set a foundation for future work. We initially planned to hammer this out by early October. When October arrived, we reassessed and planned for early November. When November arrived, we reassessed and planned for December. With December being the end of the semester and leaving no more room for postponement, it was decided that our later milestone plans were not realistic. Below is a showing of our previous planning efforts.

Report #1 Product Roadmap

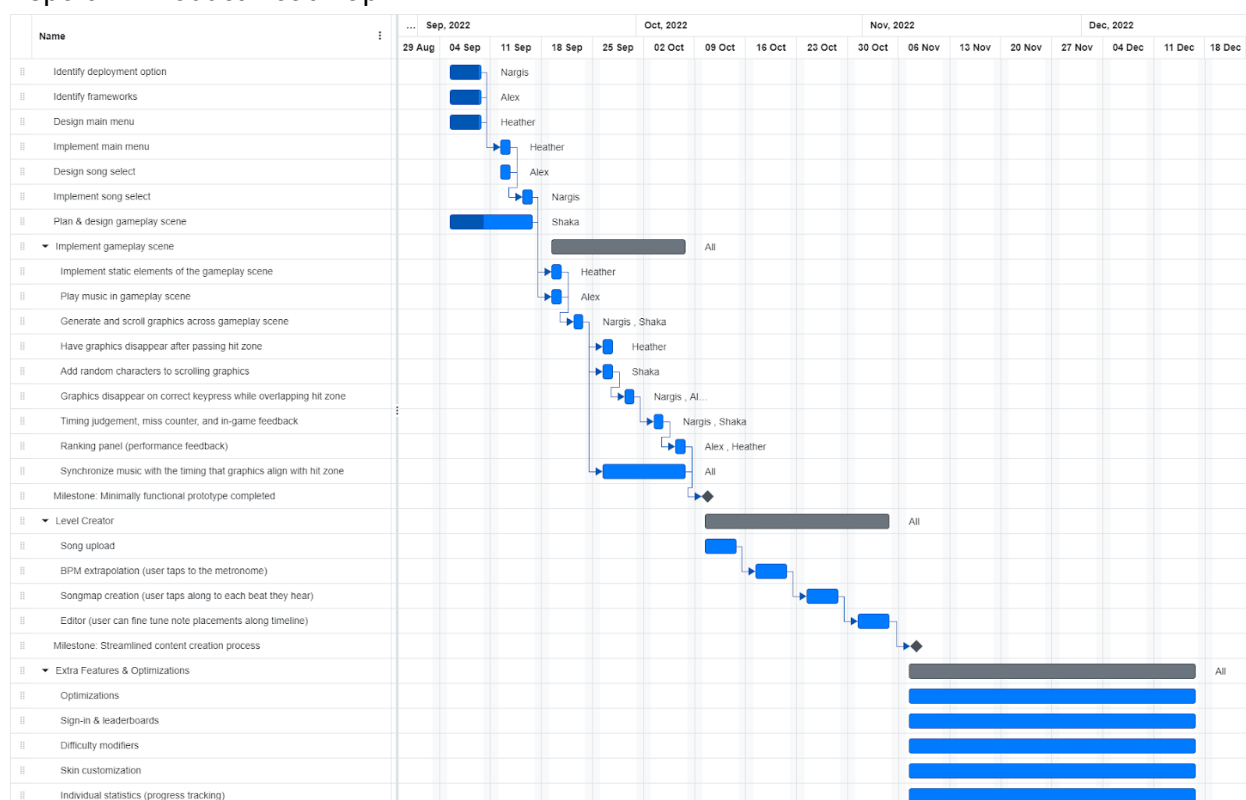


Figure 13.1.1: The earliest version of our product roadmap outlining milestones & deadlines. [Full size image here.](#)

Report #2 Product Roadmap

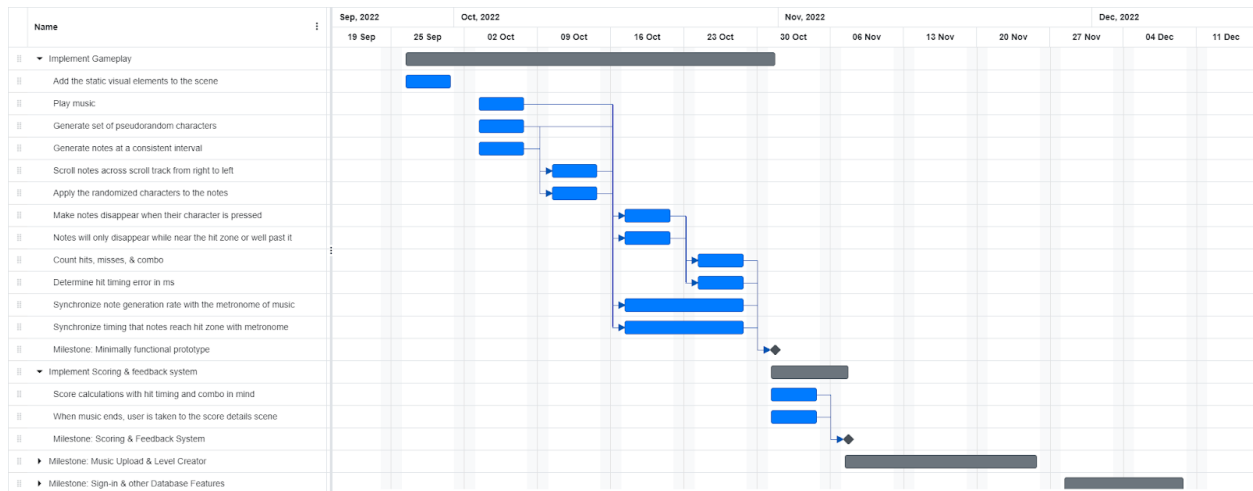


Figure 13.1.2: Version 2 of our product roadmap outlining milestones & deadlines. [Full size image here](#)

Report #2 Projected milestones

Milestone	Projected Accomplishment Date
Minimally functional prototype (deployment & gameplay)	Oct 30th, 2022
Scoring & feedback system	Nov 5th, 2022
Music upload with level creator	Nov 25th, 2022
Sign-in & other database features	Dec 8th, 2022

Table 13.1.1: Projected milestones as of Report #2

13.2 Current Status

Use Case	Status
UC-1 Guest User Page	Completed
UC-2 Register	Completed
UC-3 Registered User Page	Completed
UC-4 Typing Speed Selection	Functional
UC-5 View Score	In Progress
UC-6 Music Selection	Functional

UC-7 Play Game	In Progress
UC-8 Create a Songmap	Discontinued
UC-9 Uploading Music	Discontinued
UC-10 Style Choices Selection	Discontinued
UC-11 Leaderboard Display	Functional
UC-12 View Statistics	In Progress
UC-13 Login	Completed

Table 13.2.1: Use case & functional feature implementation status

Legend:

Completed - It's working and no further changes will be made.

Functional - It works but more work will be done to improve it.

In Progress - Little to no functional work has been contributed but will be contributed soon.

Discontinued - No work has or will be contributed.

13.3 Key Accomplishments

There were a number of important additions and improvements that took the project a step further along than where it was prior. These all added up and made the project into what you see now.

- Landing page & navigation - Late September
- Moving graphic in gameplay scene - Mid October.
- Database & Backend established for project - Mid October.
- Functional registration & login - Mid October.
- Song selection menu - Late October.
- Typing speed selector - Early November.

13.4 Future Work

There are a great many features our team discussed that will not be seen in our final product. These features include but are not limited to:

- Users may choose which keyboard characters to include/exclude from appearing on the notes.
- Users may modify the look and feel of the graphical elements of the game.
- Users may upload their favorite music.
- Users may create their own songmaps for the music they've uploaded.

14 References

- Bruegge, B., & Dutoit, A. H. (1999). *Object-Oriented Software Engineering: Conquering Complex and Changing Systems* (1st ed.). Pearson College Div.
- Cheung, S. Y., & Ng, K. Y. (2021). Application of the Educational Game to Enhance Student Learning. *Frontiers in Education*, 6, 1–10. <https://doi.org/10.3389/feduc.2021.623793>
- Davies, J. J., & Hemingway, T. J. (2014). Guitar Hero or Zero? *Journal of Media Psychology*, 26(4), 189–201. <https://doi.org/10.1027/1864-1105/a000125>
- Kim, S., Song, K., Lockee, B., & Burton, J. (2017). *Gamification in Learning and Education: Enjoy Learning Like Gaming (Advances in Game-Based Learning)* (1st ed.). Springer.
- Marom, H. W., & Weintraub, N. (2015). The effect of a touch-typing program on keyboarding skills of higher education students with and without learning disabilities. *Research in Developmental Disabilities*, 47, 208–217. <https://doi.org/10.1016/j.ridd.2015.09.014>
- Fountain, T. (2020, July 13). *How Many Bars is a 3-Minute Song? + 9 Three-Minute Song Videos - OpenMic*. Open Mic UK. Retrieved September 17, 2022, from <https://www.openmicuk.co.uk/advice/how-many-bars-is-a-3-minute-song/#:%7E:text=Taking%20into%20account%20all%20types,the%20song%20divided%20by%20four.>
- Object Management Group Standards Development Organization. (2006, May). About the Object Constraint Language Specification Version 2.0. OMG. <https://www.omg.org/spec/OCL/2.0/>
- osu! (2022, April 24). Osu! Retrieved September 17, 2022, from <https://osu.ppy.sh/home>
- [(osu!, 2022) used as inspiration for the design of the TypeMania user interface illustrated in section 4.1 Preliminary Design.]*
- Jha, R. (2021, January 4). *GRASP Design Principles*. mySoftKey. Retrieved October 9, 2022, from <https://www.mysoftkey.com/architecture/grasp-design-principles/>
- Above resource was used in coordination with the textbook.

Nielsen, J. (2020, November 15). 10 Usability Heuristics for User Interface Design. Nielsen Norman Group. Retrieved October 16, 2022, from <https://www.nngroup.com/articles/ten-usability-heuristics/>