Shattered Rails

Product Design Specification

Version 1.1

02/26/2024

Ankith Goutham

Mahin Haque

Jayanth Nama

Lilac Sabri

VERSION HISTORY

|  |  |  |  |
| --- | --- | --- | --- |
| **Version #** | **Implemented**  **By** | **Approval**  **Date** | **Description** |
| 1.0 | Ankith Goutham  Mahin Haque  Jayanth Nama  Lilac Sabri | 02/26/24 | First version |
| 1.1 | Ankith Goutham  Mahin Haque  Jayanth Nama  Lilac Sabri | 03/06/24 | Make corrections based on professor and GTA feedback. |
| 1.2 | Mahin Haque | 04/17/24 | Update images and descriptions |

TABLE OF CONTENTS

[1. Introduction 5](#_Toc160658117)

[2. General Overview and Design Guidelines/Approach 5](#_Toc160658118)

[2.1 Assumptions / Constraints / Standards 5](#_Toc160658119)

[2.1.1 Assumptions 5](#_Toc160658120)

[2.1.2 Technical Constraints 5](#_Toc160658121)

[2.1.3 Business Constraints 5](#_Toc160658122)

[2.1.4 Design Constraints 5](#_Toc160658123)

[3. Architecture Design 6](#_Toc160658124)

[3.1 Hardware Architecture 6](#_Toc160658125)

[3.2 Software Architecture 7](#_Toc160658126)

[3.2.1 MVT in Django 7](#_Toc160658127)

[3.3 Security Architecture 9](#_Toc160658128)

[3.4 Communication Architecture 11](#_Toc160658129)

[3.5 Performance 11](#_Toc160658130)

[4. System Design 12](#_Toc160658131)

[4.1 Use-Cases 12](#_Toc160658132)

[4.1.1 UC1: User Registration 12](#_Toc160658133)

[4.1.2 UC2: User Login 13](#_Toc160658134)

[4.1.3 UC3: Email Verification 14](#_Toc160658135)

[4.1.4 UC4: Game Selection 15](#_Toc160658136)

[4.1.5 UC5: Updating Profile 16](#_Toc160658137)

[4.1.6 UC6: Gameplay Choices 17](#_Toc160658138)

[4.1.7 UC7: Progress Through the Game 18](#_Toc160658139)

[4.1.8 UC8: User Interface Customization 19](#_Toc160658140)

[4.1.9 UC9: Viewing the Help Page and Tutorial 20](#_Toc160658141)

[4.1.10 UC10: Forgot Username 21](#_Toc160658142)

[4.1.11 UC11: Forgot Password 22](#_Toc160658143)

[4.2 Use Case Diagram 24](#_Toc160658144)

[4.3 Sequence Diagrams 24](#_Toc160658145)

[4.3.1 Sequence Diagram for User Registration 25](#_Toc160658146)

[4.3.2 Sequence Diagram for User Login 25](#_Toc160658147)

[4.3.3 Sequence Diagram For Email Verification 26](#_Toc160658148)

[4.3.4 Sequence Diagram for Game Selection 26](#_Toc160658149)

[4.3.5 Sequence Diagram for Updating Table 27](#_Toc160658150)

[4.3.6 Sequence Diagram for Gameplay Choices 28](#_Toc160658151)

[4.3.7 Sequence Diagram For Progress Through Game 29](#_Toc160658152)

[4.3.8 Sequence Diagram for User Interface Customization 30](#_Toc160658153)

[4.3.9 Sequence Diagram for Help Page 31](#_Toc160658154)

[4.3.10 Sequence Diagram for Forgot Username 32](#_Toc160658155)

[4.3.11 Sequence Diagram for Forgot Password 32](#_Toc160658156)

[4.4 Data Flow Diagram 33](#_Toc160658157)

[4.5 Database Design 33](#_Toc160658158)

[4.7 User Interface Design 33](#_Toc160658159)

[5 Appendix 38](#_Toc160658160)

[5.1 Shattered Rails 1 Branching Story Diagram 38](#_Toc160658161)

# 1. Introduction

The Product Design Specification document documents and tracks the necessary information required to effectively define architecture and system design to give the development team guidance on architecture of the system to be developed. The Product Design Specification document is created during the Planning Phase of the project. Its intended audience is the project manager, project team, and development team. Some portions of this document such as the user interface (UI) may on occasion be shared with the client/user, and other stakeholder whose input/approval into the UI is needed.

# 2. General Overview and Design Guidelines/Approach

This section describes the principles and strategies to be used as guidelines when designing and implementing the system.

## 2.1 Assumptions / Constraints / Standards

### 2.1.1 Assumptions

1. Application has a database to store information.
2. Application can be accessed through website form and it can be logged into/host an account process;
3. User has accessed the website of Shattered Rails.

### 2.1.2 Technical Constraints

1. The application should be experienced/accessed through a laptop/desktop computer, not any other devices such as smartphones, meaning it can’t be designed for smaller screens.
2. The application must run using JavaScript and Python for the backend, whilst HTML and CSS takes care of the front end, meaning the functions in these games can only be as complex as the upper ends of these languages and what they can create.

### 2.1.3 Business Constraints

1. Localhost Server is not responsive for website loading/access, meaning application has to be linked to server, cannot be played offline.
2. User’s internet is causing connectivity issues/is slow leading to prolongated loading times, meaning that this application can’t function on poor Wi-Fi connections for maximum performance.
3. Django’s Database isn’t connecting/working with user’s internet connection leading to not being able to log in.

### 2.1.4 Design Constraints

1. Game design should be simple enough to give a tutorial so that the user understands the way the game is played.
2. Application should have accommodations for users that are physically disabled in order to play the game, i.e. changing font size/color/family etc, so the design should revolve around catering to these needs.
3. Application must have a responsive screen size so laptops/computers of any size can play the game without compensation, front end design should be based off incorporating these accommodations.
4. UI must have buttons for the choices made in game so that the user can control the game via clicking them, putting a limit to input types to not confuse any players.
5. Game must have a difficulty feature to ease new players in whilst giving veteran players the ability to get a challenge in the game, game’s design should accommodate different difficulties.

# 3. Architecture Design

This section outlines the system and hardware architecture design of the system that is being built. It will explain the design of the hardware architecture, software architecture, security architecture, communications architecture, and the performance. Appropriate

## 3.1 Hardware Architecture

The only required hardware required for this application is a desktop/laptop with an internet connection. No additional hardware such as extra peripheral devices or sensors will be needed. One exception is that if your device doesn’t have a microphone and if you want to do voice commands in the application, then you will need an external microphone and will have to connect it to your desktop/laptop. To use the web application, the user needs to have a browser to access the website. The recommended browsers for this application will be Google Chrome for Windows and Safari for MacOS. Since our application is being hosted on the internet, we will use the hardware requirements for Google Chrome and Safari as our hardware requirements for this application.

* Hardware requirements for running Google Chrome (Preferred Browser for Windows):
  + **Windows:**
    - Windows 10 or later or Windows Server 2016 or later.
    - An Intel Pentium 4 processor or later that's SSE3 capable.
  + **Mac:**
    - macOS Catalina 10.15 or later.
  + **Linux:**
    - 64-bit Ubuntu 18.04+, Debian 10+, openSUSE 15.2+, or Fedora Linux 32+.
    - An Intel Pentium 4 processor or later that's SSE3 capable.
* Hardware requirements for running Safari (Preferred Browser for Mac):
  + Any Mac running Mac OS X Leopard 10.5.7 or Mac OS X Tiger 10.4.11 and Security Update 2009-002.
  + Mac with an Intel processor or a Power PC G5, G4, or G3 processor and built-in FireWire® 256MB of RAM.

The diagram below shows a visual representation of the hardware architecture for the whole system.

Architecture Design - Hardware ArchitectureA diagram of a computer system

Description automatically generated

## 3.2 Software Architecture

The design of this application will follow the Model View Template (MVT) software architecture similar to the Model View Controller (MVC) software architecture. The application will use Django as the web framework to implement the web interface, structure of the backend, and the database. The diagram below shows a visual representation of the software architecture.

A new Django application called main was created for this application. In Django you can create multiple applications for a single project with each performing their own functionality allowing for modularity in the code. For this project, only one application will be needed.

### 3.2.1 MVT in Django

The model component of MVT is where the data interface is located and where you can interact with the data without having to directly interact with the database. For this application, this will be handled by the file called models.py where you can declare new models for the project or modify the existing one. The models.py file was modified to create a new table in the database called Game which will extend the User table which is built into the Django framework by default. The models.py file is located within the main application. This component will be done using the Python programming language. The database diagram shown in section 4.5 provides a visual representation of the database tables.

The view component of MVT is where the business logic will be handled in the application and will act as a bridge between the model and the template. View will decide what template will be rendered and what data will be sent to the model. The main file for this component will be views.py which will handle functionalities such as rendering the templates and additional functions such as registering, logging in, logging out, editing your profile, and editing your password. Additionally, forms.py will also be used to handle user input in cases such as when the user is inputting information to register and when they are inputting information to log in. Both the views.py and forms.py will be in the main application. This component will also use Python to implement the logic.

The template component of MVT is the presentation layer of the application where the user will interact with user interface (UI). The templated folder located in the main application will store all the HTML files needed for each web application. The templates folder will contain index.html, register.html, login.html, game.html, help.html, and profile.html which will implement the user interface for the home page, login page, the game page, the additional help page, and the profile page respectively. The content of this layer will be made using HTML, CSS, and JavaScript.

A diagram of a software process

Description automatically generated

Figure 3.2.1 MVT Software Architecture

Figure 3.2.1 gives a visual representation of the MVT software architecture and how each component relates to one another. The user sends a request or receives a response from the view component. View performs create, update, and delete operations connecting to the model and displays data on the template. Additionally, view receives data input from template and gets data from the model. Template gets the data to display from the view component. The model component and the database are connected to one another and retrieves and sends data between one another.

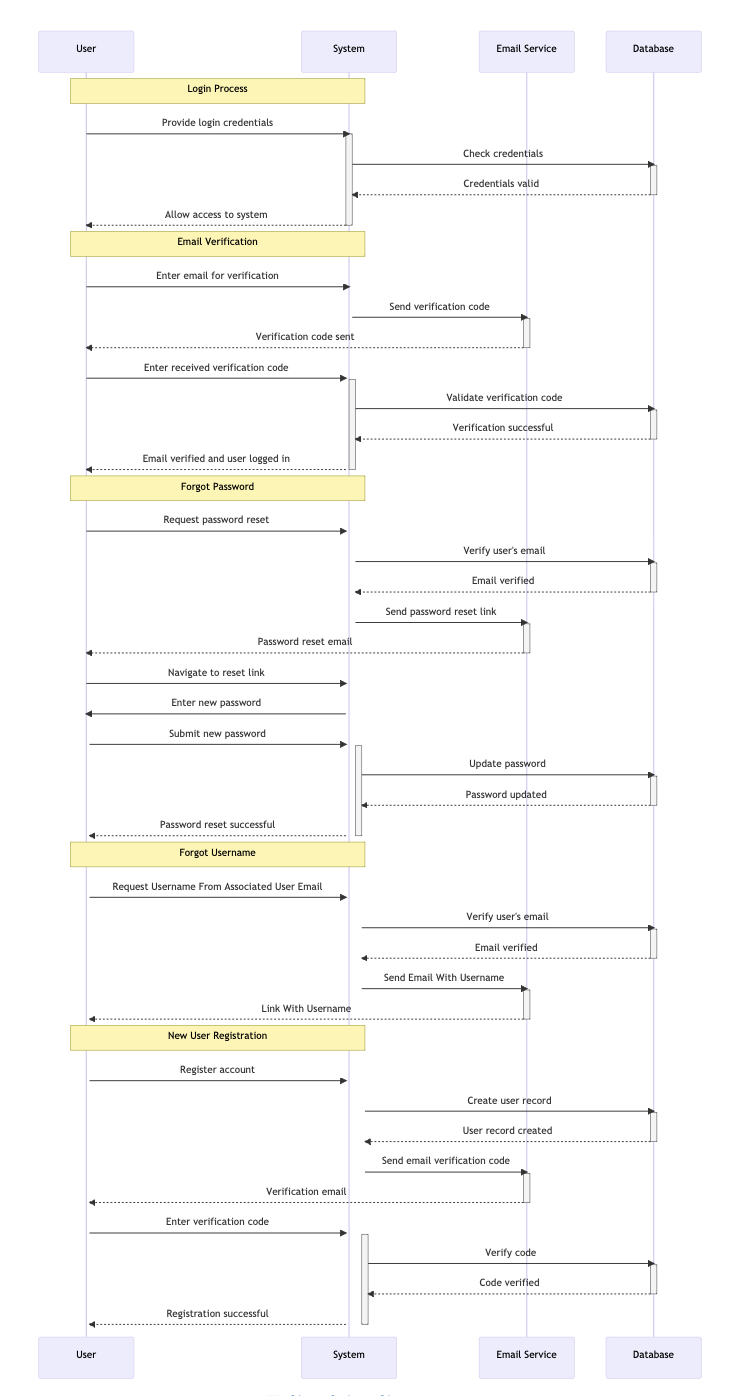
## 3.3 Security Architecture

During the account registration process, users will encounter an email verification process. This is done through receiving a four-digit code via email, which they must input correctly to proceed. This verification mechanism serves a dual purpose: firstly, it acts as a barrier against the creation of fraudulent accounts, providing a more authentic user environment within the application. Secondly, it ensures the validity of the email addresses provided during registration, enhancing overall security measures.

Moving on to the process of password or username recovery, a security protocol is implemented to safeguard user accounts. In the event of a forgotten password or username, users are directed to enter their associated email address. After an email that is stored in the database is entered, a temporary password is generated and sent to the provided email. Upon receiving this temporary password, users must input it into the application to confirm ownership of the email address and authenticate their identity. Only after successful verification will users be allowed to reset their passwords or retrieve their usernames. This verification process provides for account security, making the user less susceptible to unauthorized access or hacking attempts.

By implementing these measures, the application ensures not only the integrity of its user base but also enhances the overall trustworthiness and security of the platform. Which in turn, provides a more authentic environment for all users in the game application.

**Security Architecture Diagram**



## 3.4 Communication Architecture

The three main components model, view, and template communicate with one another by transferring and processing data. In the template view, POST requests will be sent during registration, login, editing your profile, and changing your password. These POST requests will carry data input by the user which will be received by the view component and processed. In response, it will send a message back to the template indicating a success message or an error message. In addition, the data will be sent to the model which will send the data to be stored in the database if needed.

## 3.5 Performance

The game application undergoes routine testing to make sure that it is running/functioning properly and address any performance issues promptly. Continuous error monitoring allows to resolve any issues that may arise. Regular maintenance keeps the software up-to-date with the latest improvements. Player feedback is valued as it helps us identify areas for enhancement, for an enjoyable gaming experience for all users.

# 4. System Design

## 4.1 Use-Cases

## 4.1.1 UC1: User Registration

|  |  |
| --- | --- |
| **ID** | UC1 |
| **Use Case Name** | User Registration |
| **Description** | Allows the user to create a new account to access the game application. |
| **Actors** | User |
| **Trigger** | User selects the register button. |
| **Preconditions** | User does not have an account. |
| **Postconditions** | The user’s account is created, allowing them to login to access the game. |
| **Normal Flow** | 1. User inputs email, username, name and password. 2. User clicks on "Register" after they have completed inputting their credentials. 3. User gets confirmation email of four digit code to verify their email. 4. User inputs the four digit code sent from the system. 5. User is now registered. 6. User can now login. 7. User can now access the game application. |
| **Alternative Flows** | There is no alternative flow for registration. |
| **Exceptions** | 1. Invalid email format. 2. password doesn’t meet credentials. |
| **Assumptions** | 1. User has a reliable internet connection.   2. The server is running properly.  3. The user has a valid email address. |

### 4.1.2 UC2: User Login

|  |  |
| --- | --- |
| **ID** | UC2 |
| **Use Case Name** | User Login |
| **Description** | User logs in to access their account and play or resume their gameplay. |
| **Actors** | User |
| **Trigger** | User selects the "Login" option. |
| **Preconditions** | User has a registered account and has went through email verification successfully. |
| **Postconditions** | Player accesses their game profile. |
| **Normal Flow** | 1. Player inputs username and password. 2. System validates credentials. 3. User is logged in. |
| **Alternative Flows** | If the user inputs an incorrect password, the system offers a "Forgot Password" and “Forgot username” option. This will allow users that forgot their credentials to login. |
| **Exceptions** | 1. Incorrect username or password entered. 2. Account locked due to multiple failed login attempts. 3. User can retry after five hours. |
| **Assumptions** | 1. User has a reliable internet connection.   2. The server is running properly.  3. The user has a registered account with a verified email. |

### 4.1.3 UC3: Email Verification

|  |  |
| --- | --- |
| **ID** | UC3 |
| **Use Case Name** | Email Verification |
| **Description** | Ensures the player's email address is verified before allowing access to the game. |
| **Actors** | User |
| **Trigger** | User attempts to log in for the first time. |
| **Preconditions** | User has registered an account. |
| **Postconditions** | User's email is verified, enabling account to be created. |
| **Normal Flow** | 1. System sends a four-digit verification code to the user's email during registration. 2. User enters the unique four digit code. 3. System confirms the email and activates the account. |
| **Alternative Flows** | User can request to resend the verification code if the original is not received. |
| **Exceptions** | Verification code expires. |
| **Assumptions** | 1. User has a reliable internet connection.   2. The server is running properly.  3. The user has access to the email address that they input in the text field for email. |

### 4.1.4 UC4: Game Selection

|  |  |
| --- | --- |
| **ID** | UC4 |
| **Use Case Name** | Game Selection |
| **Description** | Allows the player to choose which game or story to play within the text-based adventure. |
| **Actors** | User |
| **Trigger** | Player selects "Choose Game" from the menu to pick a specific game of their preferences. |
| **Preconditions** | Player is logged in with a registered account. |
| **Postconditions** | Player starts or resumes the selected game. |
| **Normal Flow** | 1. The system displays available games or stories. 2. Player selects a game. 3. If user has played and at a certain point has stopped the game and logged out. When the user logs back in, the game will resume where the user has left off. 4. If the user has played the game for the first time as a registered user, then the game will initiate. |
| **Alternative Flows** | Player can go back, while playing the game selected, and select a new game, through clicking on the go-back button. |
| **Exceptions** | N/A |
| **Assumptions** | 1. User has a reliable internet connection.   2. The server is running properly.  3. User has a registered account with a verified email  4. User has logged in.  5. Multiple games or stories are available for selection. |

### 4.1.5 UC5: Updating Profile

|  |  |
| --- | --- |
| **ID** | UC5 |
| **Use Case Name** | Updating Profile |
| **Description** | Players can edit their profile information which includes updating their username and password through inputting their current password, and the new password. |
| **Actors** | User |
| **Trigger** | Player selects “Profile” image |
| **Preconditions** | The player is logged in and registered and has verified their email. |
| **Postconditions** | Player's profile is updated. |
| **Normal Flow** | 1. Player accesses their profile settings. 2. Player edits information and submits changes. 3. System updates the profile. |
| **Alternative Flows** | Player can reset their password and username before logging in. |
| **Exceptions** | Invalid format of updated information, so profile information can’t be updated. |
| **Assumptions** | 1. User has a reliable internet connection. 2. The server is running properly. 3. User is registered and has their email verified. 4. User is logged in. 5. Player wants to keep their account information up to date. |

### 4.1.6 UC6: Gameplay Choices

|  |  |
| --- | --- |
| **ID** | UC6 |
| **Use Case Name** | Gameplay Choices |
| **Description** | During gameplay, players make choices that affect the story's outcome. |
| **Actors** | User |
| **Trigger** | A decision point is reached in the game. |
| **Preconditions** | Player is engaged in a game. |
| **Postconditions** | The game progresses based on the player's choice. |
| **Normal Flow** | 1. The game presents a situation and choices. 2. Player selects a choice. 3. The game continues based on the selected choice. 4. The game has a different ending. |
| **Alternative Flows** | Player can return to previous path, to update their choices if needed. |
| **Exceptions** | N/A |
| **Assumptions** | 1. User has a reliable internet connection.   2. The server is running properly.  3. User has a registered account with a verified email  4. User has logged in.  5. User has selected a game. |

### 4.1.7 UC7: Progress Through the Game

|  |  |
| --- | --- |
| **ID** | UC7 |
| **Use Case Name** | Progress through the game |
| **Description** | Describes the process of playing through the game, encountering various scenarios and making decisions. |
| **Actors** | User |
| **Trigger** | Player starts or resumes the game. |
| **Preconditions** | The player has selected a game to play. |
| **Postconditions** | Player progresses through the game. |
| **Normal Flow** | 1. Game presents text-based scenarios to the player. 2. Player reads and interacts with the game. 3. Game responds to player inputs, leading to new scenarios. |
| **Alternative Flows** | User can go back one path. |
| **Exceptions** | Game encounters an error and the game application needs to be refreshed. |
| **Assumptions** | 1. User has a registered account with a verified email. 2. User has logged in. 3. Player is able to view the game and interact. |

### 4.1.8 UC8: User Interface Customization

|  |  |
| --- | --- |
| **ID** | UC8 |
| **Use Case Name** | User Interface Customization |
| **Description** | Allows players to adjust the user interface settings to their preference for a better gameplay experience. |
| **Actors** | User |
| **Trigger** | 1. User can select from the buttons labeled font color, background color, terminal background color, font size dropdown menu. User can also choose from bold, italics, underline, text-speed and line-spacing. |
| **Preconditions** | 1. User has registered and verified their email. 2. User is logged in. 3. User is in the game application. |
| **Postconditions** | The user-interface is adjusted according to the player's preferences. |
| **Normal Flow** | 1. Player accesses the game. 2. Player adjusts User-interface buttons to their preferences. 3. Changes are saved and applied. |
| **Alternative Flows** | N/A |
| **Exceptions** | N/A |
| **Assumptions** | 1. User has a reliable internet connection.   2. The server is running properly.  3. User has a registered account with a verified email.  4. User has logged in.  5. UI customization will enhance player ability to read and satisfaction. |

### 4.1.9 UC9: Viewing the Help Page and Tutorial

|  |  |
| --- | --- |
| **ID** | UC9 |
| **Use Case Name** | Help Page |
| **Description** | Provides players with access to help resources and game instructions. |
| **Actors** | User |
| **Trigger** | Player/User selects on the button on the bottom right of the game application.  The user would choose see tutorial in one of the game choices. |
| **Preconditions** | N/A |
| **Postconditions** | Player views help resources. |
| **Normal Flow** | 1. Player selects "Help-page" through the game application. 2. System displays instructions on how to play the game. 3. Player navigates and reads help content. 4. Game application will prompt the user to either start the game or view the tutorial on how to play the game. 5. User will select to watch the tutorial. 6. Game application terminal will display instructions on how to play the game. |
| **Alternative Flows** | Player chooses to view the tutorial that is set in the game before starting. |
| **Exceptions** | Help content fails to load. |
| **Assumptions** | 1. User has a reliable internet connection.   2. The server is running properly.  3. User has a registered account with a verified email.  4. User has logged in.  5. Players needs assistance understanding game mechanics or solving issues. |

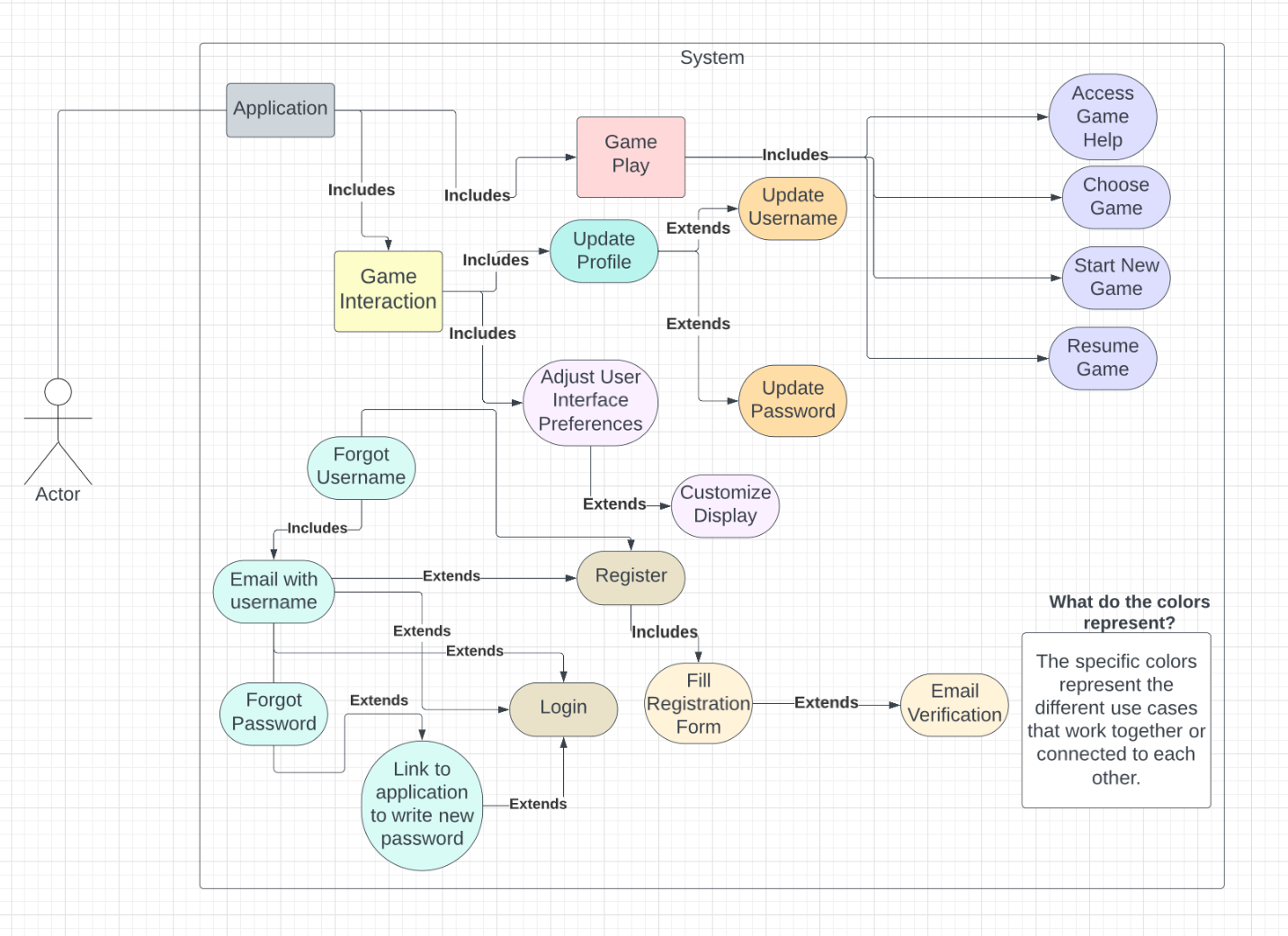
### 4.1.10 UC10: Forgot Username

|  |  |
| --- | --- |
| **ID** | UC10 |
| **Use Case Name** | This use case allows users who have forgotten their username to retrieve or create a new one. |
| **Description** | Allows users to create their username if forgotten. |
| **Actors** | User |
| **Trigger** | User clicks on link in the login page that says “Forgot Username?” |
| **Preconditions** | User is registered in the system. |
| **Postconditions** | User successfully retrieves or creates a new username. |
| **Normal Flow** | 1. User navigates to the "Forgot Username" section on the login page. 2. System prompts the user to provide information for identification, which will be their associated account email. 3. User enters the required identification information (registered email). 4. System validates the information provided. 5. If the information is valid and the email address is registered, the system sends an email for verification, which will be the user’s username. 6. User will have access to their username. |
| **Alternative Flows** | 1. If the provided identification information is invalid, the system prompts the user to verify and correct the information. 2. If the user does not receive the username retrieval email, they can request it to be resent. |
| **Exceptions** | If the user's identification information cannot be verified, the system informs the user that it cannot retrieve the username. |
| **Assumptions** | 1. User has a reliable internet connection.   2. The server is running properly.  3. The user has a registered account.  4. Users have access to the email address associated with their account for verification purposes. |

### 4.1.11 UC11: Forgot Password

|  |  |
| --- | --- |
| **ID** | UC11 |
| **Use Case Name** | Forgot Password |
| **Description** | This use case allows users who have forgotten their password to reset it and regain access to their account. |
| **Actors** | User |
| **Trigger** | User clicks on link in the login page that says “Forgot Password?” |
| **Preconditions** | User is registered in the system. |
| **Postconditions** | User successfully resets their password and gains access to their account. |
| **Normal Flow** | 1. User navigates to the "Forgot Password" section on the login page. 2. System prompts the user to provide their email for identification. 3. User enters the required identification information, which is user’s associated email. 4. System validates the information provided. 5. If the information is valid, the system sends a password reset link to the user's registered email address. 6. Users access their email, and there will be a link. 7. The user will click on that link and that will redirect them to the application page, where the user can write their new password. 8. User successfully resets their password and gains access to their account. |
| **Alternative Flows** | 1. If the provided identification information is invalid, the system prompts the user to verify and correct the information. 2. If the user does not receive the password reset link, they can request it to be resent to their associated email. |
| **Exceptions** | If the user's identification information cannot be verified, the system informs the user that the password cannot be reset. |
| **Assumptions** | Users have access to the email address associated with their account for password reset purposes. |

## 4.2 Use Case Diagram



## 4.3 Sequence Diagrams

### 4.3.1 Sequence Diagram for User Registration

### A screenshot of a computer screen Description automatically generated

### 4.3.2 Sequence Diagram for User Login

## A screenshot of a computer Description automatically generated

### 4.3.3 Sequence Diagram For Email Verification

### A screenshot of a computer screen Description automatically generated

## 

### 4.3.4 Sequence Diagram for Game Selection

A screenshot of a computer screen

Description automatically generated

### 4.3.5 Sequence Diagram for Updating Table

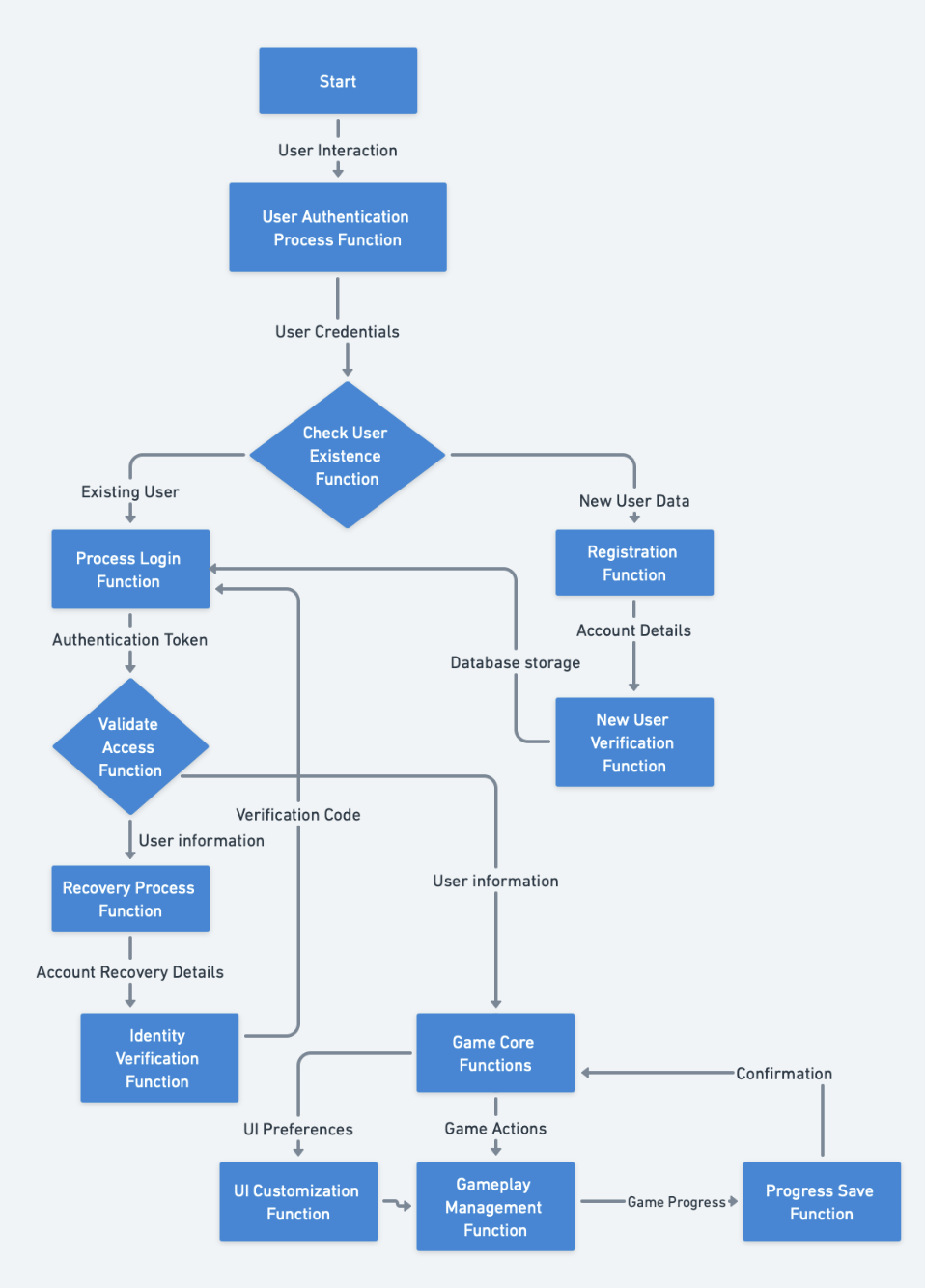
A screenshot of a computer screen

Description automatically generated

### 4.3.6 Sequence Diagram for Gameplay Choices

### A screenshot of a computer screen Description automatically generated

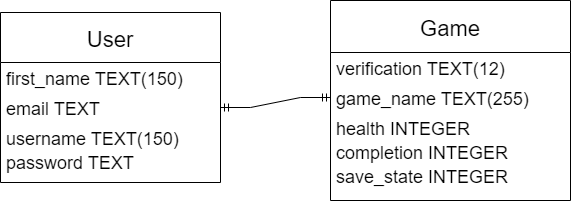
## 4.4 Data Flow Diagram



## 4.5 Database Design

SQLite will be used as the database for this application. Two tables will be used to store all the necessary information. The first table will be the User table which is built into the Django framework and allows us to store the first name, email address, username, and password in the table. One thing to note is that the raw password will not be stored, the server will encrypt the password and store the hash and then will decrypt the hash when being retrieved from the database. The second table will be Game and it will be used to store the verification status of the account, the in game name of the user, the player’s health, the percentage of the game completed, and the save state of the user. The Game table will connect to the User table in a one-to-one relationship.

The following diagram shows the different tables used in the application and how those tables are related to one another.



## 4.7 User Interface Design

This is the home page. It will be the first page users visit when first visiting the website.

A screenshot of a computer

Description automatically generated

This is the register page. The user will register for an account on this page and then will be prompted to verify their email.

A screenshot of a computer

Description automatically generated

This is the login page where the user will login with the credentials they made during registration and verified their email address. After logging in, they will be taken to the game selection page.

A screenshot of a computer login

Description automatically generated

This is the page after registering for an account. It prompts you to verify your email by checking the email sent to the email address.

A screen shot of a computer

Description automatically generated

This is the game selection page where the user will pick which game they want to play. The left card will lead to Shattered Rails 1 and the right card will lead to Shattered Rails 2.

Screens screenshot of a computer

Description automatically generated

This is the start page before Shattered Rails 1. Clicking on the left card on the game selection page will lead to this page.

A screen shot of a computer screen

Description automatically generated

This is the start page before Shattered Rails 2. Clicking on the right card on the game selection page will lead to this page.

A screenshot of a computer

Description automatically generated

This is the help page. This page will explain the mechanics of the game and offer tips for the user.

A screenshot of a computer

Description automatically generated

This is the profile page on the “Edit Profile” section where you can edit the information about your profile.

A screenshot of a computer

Description automatically generated

This is the profile page on the “Edit Password” section where the user can change their password.

# A screenshot of a computer login Description automatically generated

# This is the page where you will play Shattered Rails 1. Many tools for user interface customization can be found under the settings button.

# A screenshot of a computer Description automatically generated

# This is the page where you will play Shattered Rails 2. Many tools for user interface customization can be found under the settings button.

# A screenshot of a computer Description automatically generated

A screenshot of a computer

Description automatically generated

# 5 Appendix

## 5.1 Shattered Rails 1 Branching Story Diagram

A diagram of a network

Description automatically generated

Figure 5.1.1 Branching Story Diagram for Shattered Rails 1

Figure 5.1.1 shows a diagram showing all the possible paths in the Shattered Rails 1 game. It shows what decisions the user has to make to reach one of the specific types of endings. There are four types of endings being good ending, bad ending, secret ending, and true ending. The end of each path is denoted by a black circle with a red outline. The green circles is a checkpoint where the game will autosave. If the user logs out or visits another page and then returns back to the game page, they will start at their last checkpoint.