Software Requirements and Design Document

For

Group <9>

Version 1.0

Authors:
Kaitlyn Krause
Tara Kerstetter
Megan Cole
Brandon Pina

1. Overview (5 points)

Give a general overview of the system in 1-2 paragraphs (similar to the one in the project proposal).

Our project is a web-based application where users can practice and test their typing skills. On our app, *KeyFlow,* we will have not only a standard typing test but also a variety of mini-games. In the standard type test, different time durations and difficulty levels can be selected. There will also be a mode to have personal practice based on the user's more frequently missed letters. In order to generate the actual test, we will select random words and put them in basic sentence format. Some of the statistics we will measure are words per minute, letter accuracy, and incorrect letter frequency. We also have two mini-games, *Snowfall* and *Snowslope*.

KeyFlow will also allow for the creation of accounts. With an account, users can have their stats stored and have access to modes such as personalized practice based on their stats. We will also be implementing a paid feature called the **"Battle Pass"**. With this, users will have the ability to win rewards such as exclusive profile pictures and avatars and other items for specific game modes. This can be achieved by playing games and practicing typing to gain XP.

2. Functional Requirements (10 points)

*List the **functional requirements** in sentences identified by numbers and for each requirement state if it is of high, medium, or low priority. Each functional requirement is something that the system shall do. Include all the details required such that there can be no misinterpretations of the requirements when read. Be very specific about what the system needs to do (not how, just what). You may provide a brief design rationale for any requirement which you feel requires explanation for how and/or why the requirement was derived.*

- **High:** Tracking user typing
- **High:** Sentence generation
- **High:** Statistical calculations
- **High:** User information and statistics storage
- **High:** Host multiple mini games
- **Medium:** Different practice modes in type test
- **Medium:** Battle Pass feature

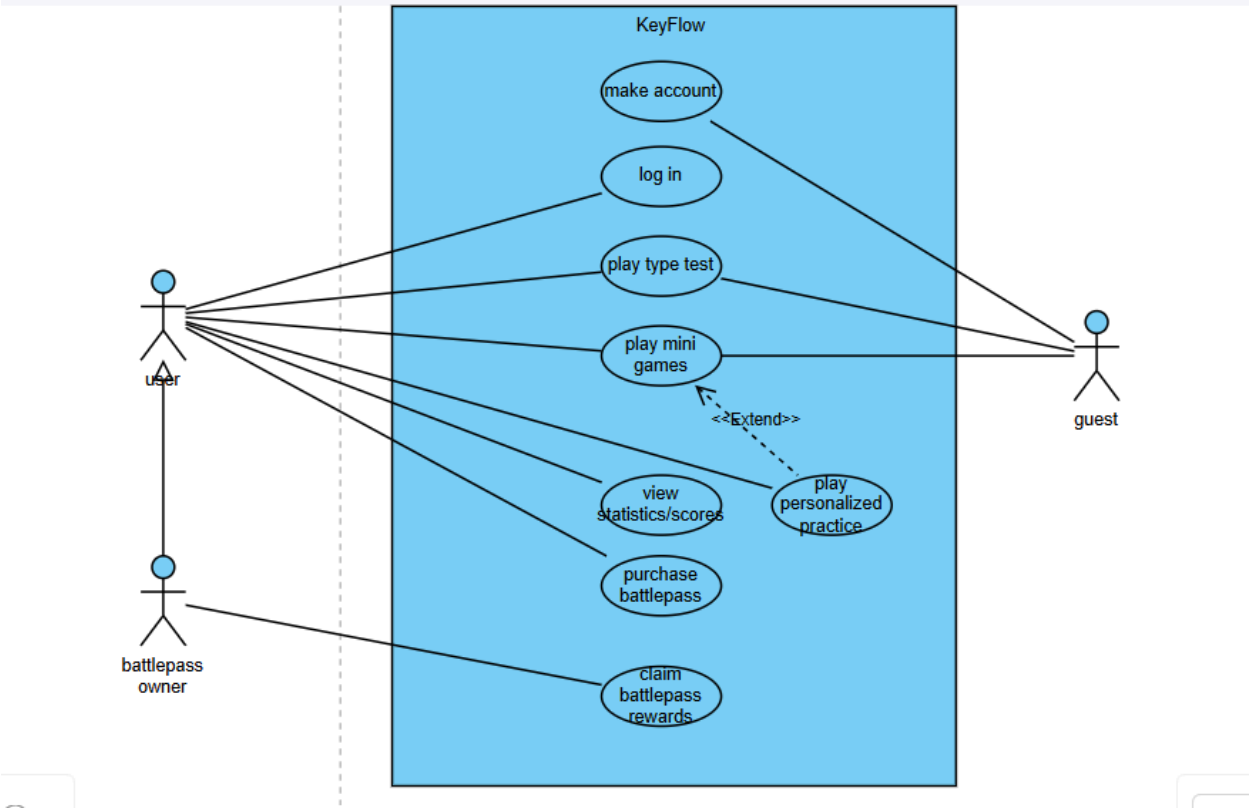
3. Non-functional Requirements (10 points)

*List the **non-functional requirements** of the system (any requirement referring to a property of the system, such as security, safety, software quality, performance, reliability, etc.) You may provide a brief rationale for any requirement which you feel requires explanation as to how and/or why the requirement was derived.*

- Password Encryption
- Ability to handle multiple users at one
- Reliability through maintaining a system uptime of 99%

4. Use Case Diagram (10 points)

*This section presents the **use case diagram** and the **textual descriptions** of the use cases for the system under development. The use case diagram should contain all the use cases and relationships between them needed to describe the functionality to be developed. If you discover new use cases between two increments, update the diagram for your future increments.*
***Textual descriptions of use cases**: For the first increment, the textual descriptions for the use cases are not required. However, the textual descriptions for all use cases discovered for your system are required for the second and third iterations.*



Make account

- **Actors:** Guest
- **Description:** Allow new actors to make an account
- **Preconditions:**
- **Postconditions:** New Account has been added to database
- **Main Flow:** Guest clicks “sign up” button -> takes them to register page -> they fill out all the necessary information -> click “sign up” -> Account is added to database
- **Alternative Flows:**
    - -> they fill out necessary information -> username or email is already in use -> reenter information
    - -> they fill out necessary information -> passwords don’t match -> reenter information
- **Exceptions:** rollback on database transaction if error with information, pre-checks username, email, and passwords before trying database transaction

Log in

- **Actors:** User
- **Description:** actors can log into their pre-existing accounts
- **Preconditions:** Account exists in the database
- **Postconditions:** User is logged in to their respective account
- **Main Flow:** click “login” button -> enter username and password -> redirected to user’s homepage
- **Alternative Flows:**

    - -> enter username and password -> credentials don't match or don't exist -> reenter credentials
- **Exceptions:** checks that username and password match

Play type test

- **Actors:** User, Guest
- **Description:** actors can play a standard type test with the option of different time durations and difficulty levels
- **Preconditions:**
- **Postconditions:** actors will play a type test. At the end, their wpm and accuracy is shown. If user is logged in then these stats are saved to the database
- **Main Flow:** click "type test" -> select duration and difficulty -> play type test -> shown stats at end
- **Alternative Flows:**
- **Exceptions:**

Play Personalized Practice

- **Actors:** User
- **Description:** actors can play a standard type test with personalization based off their most missed letters
- **Preconditions:** actor is logged in and has most missed letters stored in database
- **Postconditions:** actors will play the type test with personalization. At the end, their wpm and accuracy is shown. These stats are saved to the database
- **Main Flow:** click "type test" -> click "Personalized Practice -> play type test -> shown stats at end
- **Extends: Play typetest**

Play mini games

- **Actors:** User, Guest
- **Description:** Actors have a selection of mini games to play from
- **Preconditions:**
- **Postconditions:** Actor will play selected mini game. Their score and accuracy will be shown at the end and if they are a User their score will be saved to database
- **Main Flow:** click "minigames" button -> select game -> play game -> see score
- **Alternative Flows:**
- **Exceptions:**

View Statistics/Scores

- **Actors:** User
- **Description:** Actors can view their own statistics from type test and minigames. They can also view their friends' scores in a leaderboard format
- **Preconditions:** Actor has statistics already in database, Actor has friends with statistics in database
- **Postconditions:** Various statistics and scores are shown for the Actor and their friends

- **Main Flow:** login to account -> view statistics on homepage
- **Alternative Flows:**
    - Click “profile” -> view individual statistics
    - Click “leaderboard” -> view leaderboard with friends’ scores
    - Click friend’s profile -> view their scores

Purchase Battle Pass

- **Actors:** User
- **Description:** Actor’s can purchase the Battle Pass which will unlock the ability to gain xp to claim rewards
- **Preconditions:** Actor must have an account
- **Postconditions:** Actor will have access to Battle Pass features
- **Main Flow:** user is logged in -> click on “Battle Pass” -> enter payment information -> purchase Battle Pass
- **Alternative Flows:**
- **Exceptions:**

Claim Battle Pass Rewards

- **Actors:** Battle Pass Owner
- **Description:** Once an Actor reaches a certain level they will have the ability to claim certain rewards in the Battle Pass
- **Preconditions:** Must have the Battle Pass
- **Postconditions:** Reward gets added to Actor’s entry in the “EquippedItems” table
- **Main Flow:** user is logged in -> click on “Battle Pass” on homepage -> select reward they want to claim
- **Alternative Flows:**
    - -> select reward -> user doesn’t have enough levels -> reward doesn’t get claimed
- **Exceptions:** Actor can only claim rewards for their current level amount

5. Class Diagram and/or Sequence Diagrams (15 points)

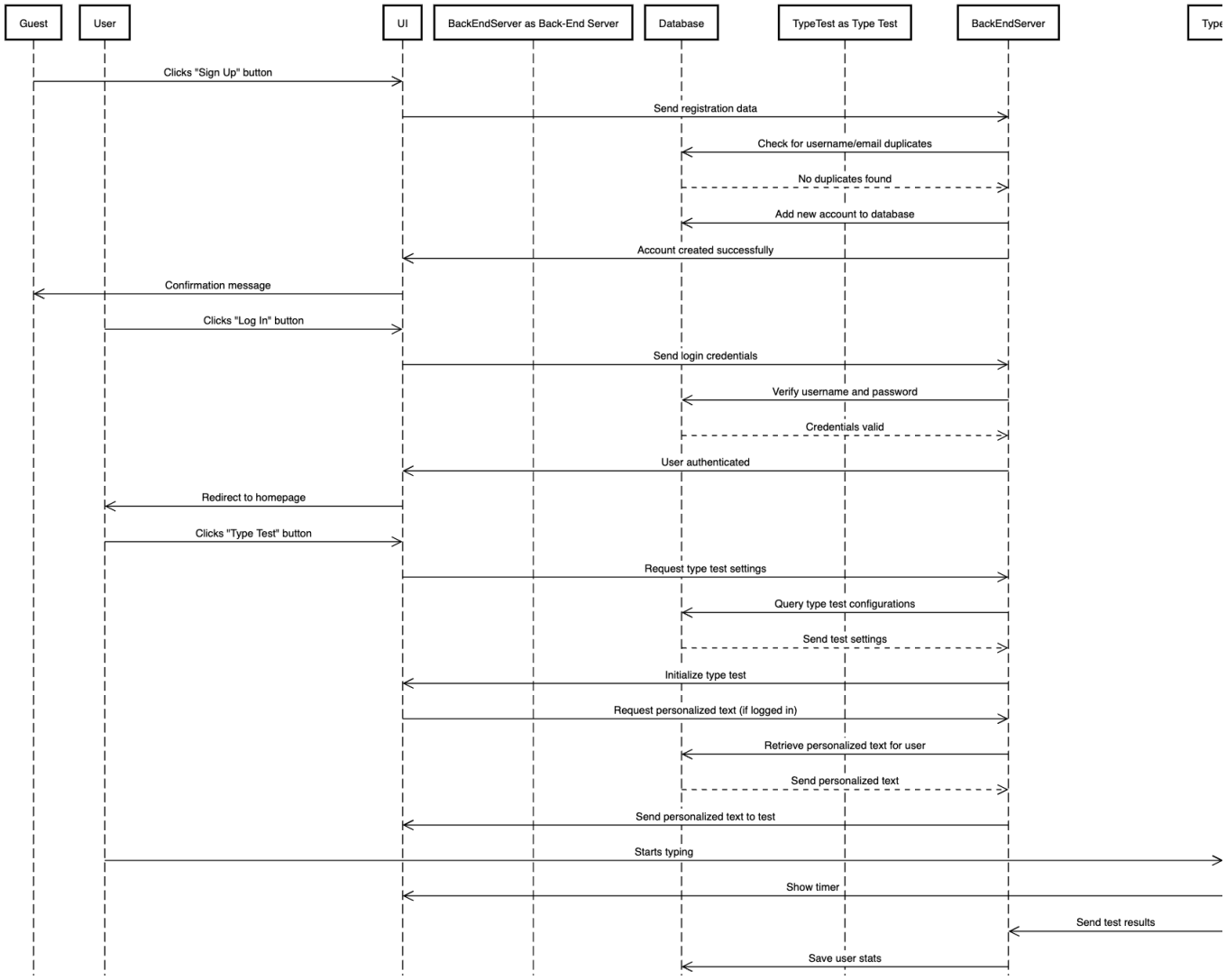
*This section presents a high-level overview of the anticipated system architecture using a **class diagram** and/or **sequence diagrams**.*

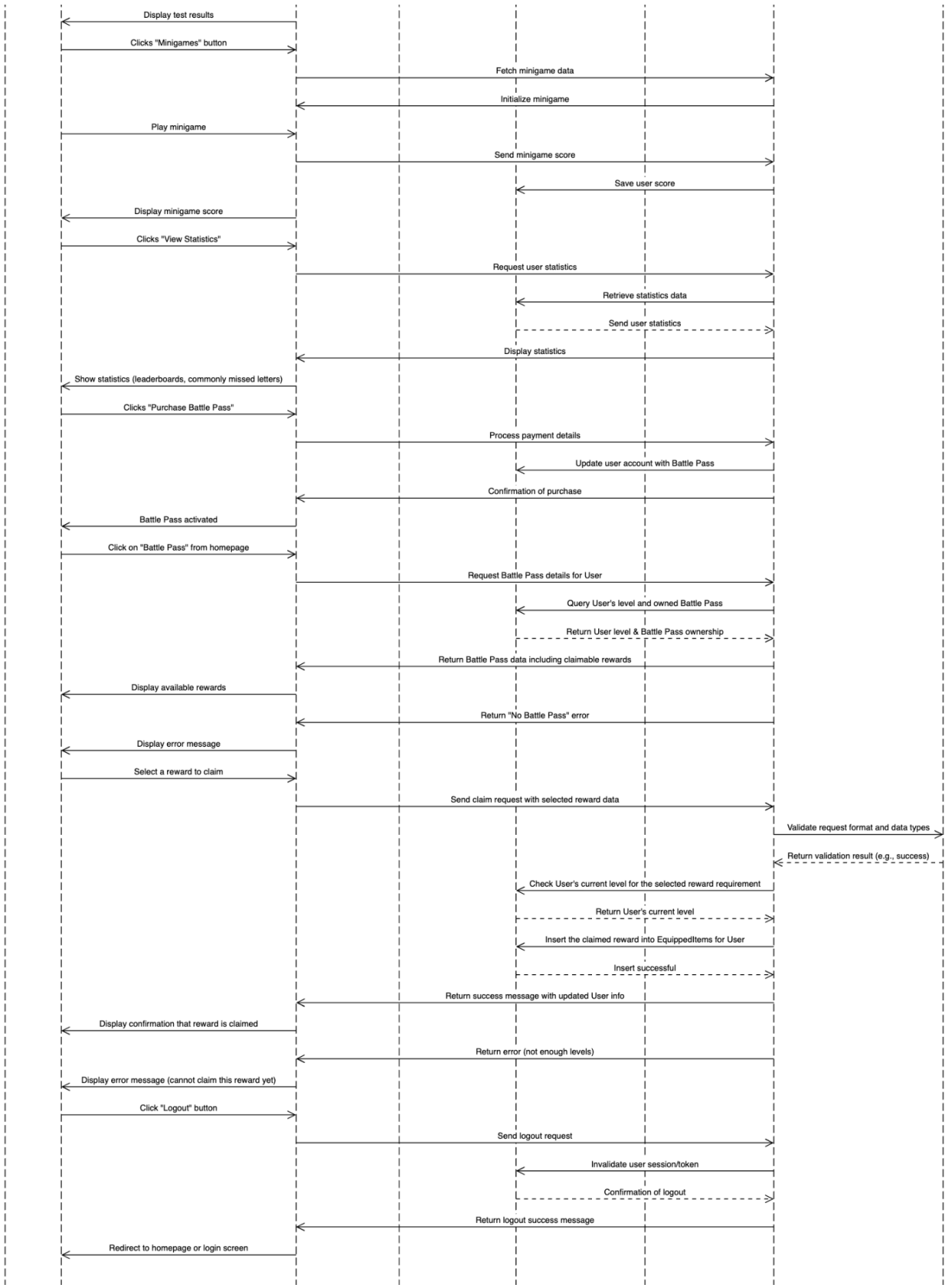
*If the main **paradigm** used in your project is **Object Oriented** (i.e., you have classes or something that acts similar to classes in your system), then draw the **Class Diagram of the entire system and Sequence Diagrams for the three (3) most important use cases in your system.***

*If the main **paradigm** in your system is **not Object Oriented** (i.e., you **do not** have classes or anything similar to classes in your system) then only draw **Sequence Diagrams**, **but for all the use cases of your system.** In this case, we will use a modified version of Sequence Diagrams, where instead of objects, the lifelines will represent the functions in the system involved in the action sequence.*

***Class Diagrams** show the **fundamental objects/classes** that must be modeled with the system to satisfy its requirements and **the relationships** between them. Each class rectangle on the diagram **must also include the attributes and the methods of the class** (they can be refined between increments). All the **relationships between classes and their multiplicity** must be shown on the class diagram.*

*A **Sequence Diagram** simply depicts **interaction between objects** (or **functions -** in our case - for non-OOP systems) in a sequential order, i.e. the order in which these interactions take place. Sequence diagrams describe how and in what order the objects in a system function.*




Display test results
Clicks "Minigames" button
Fetch minigame data
Initialize minigame
Play minigame
Send minigame score
Save user score
Display minigame score
Clicks "View Statistics"
Request user statistics
Retrieve statistics data
Send user statistics
Display statistics
Show statistics (leaderboards, commonly missed letters)
Clicks "Purchase Battle Pass"
Process payment details
Update user account with Battle Pass
Confirmation of purchase
Battle Pass activated
Click on "Battle Pass" from homepage
Request Battle Pass details for User
Query User's level and owned Battle Pass
Return User level & Battle Pass ownership
Return Battle Pass data including claimable rewards
Display available rewards
Return "No Battle Pass" error
Display error message
Select a reward to claim
Send claim request with selected reward data
Validate request format and data types
Return validation result (e.g., success)
Check User's current level for the selected reward requirement
Return User's current level
Insert the claimed reward into EquippedItems for User
Insert successful
Return success message with updated User info
Display confirmation that reward is claimed
Return error (not enough levels)
Display error message (cannot claim this reward yet)
Click "Logout" button
Send logout request
Invalidate user session/token
Confirmation of logout
Return logout success message
Redirect to homepage or login screen

6. Operating Environment (5 points)

Describe the environment in which the software will operate, including the hardware platform, operating system and versions, and any other software components or applications with which it must peacefully coexist.

Hardware platform - Cloud server
Operating system versions - Linux, Windows, macOS
Software components -
Django 3.2 or higher
Python 3.8 or higher
PostgreSQL 12 or higher
Phaser.js
Modern Browsers (Safari, Firefox, etc.)

7. Assumptions and Dependencies (5 points)

List any assumed factors (as opposed to known facts) that could affect the requirements stated in this document. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project.

Assumed stable internet connection since it is a website.
Assumes that all the frameworks will remain compatible and active in the future
Works on most browsers.
Scalability with simultaneous usage.
Assumes that website service is functioning
Third party Hosting Services - Depends on the availability and performance of these services as they are beyond our reach.
If a payed service is created, it depends on the safety of them like Paypal.
Potentially User authentication - ex. Google, facebook for login, depends on stability and availability of service