**Software Requirements and Design Document**

**For**

**Group 9**

Version 1.0

**Authors**:

Kaitlyn Krause

Tara Kerstetter

Megan Cole

Brandon Pina

# 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 modes can be toggled on or off by the user. Some of these modes include, capital letters, punctuation, personalized practice based on most missed letters, etc. 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. In our mini games we want to implement a multi-player functionality.

*KeyFlow* will also allow for the creation of accounts. With an account, users can have their stats stored and will 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, emoji reactions, and skins for specific game modes. This can be achieved by playing games and practicing typing.

# 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.*

1. **High:** Tracking user typing
2. **High:** Sentence generation
3. **High:** Statistical calculations
4. **High:** User information and statistics storage
5. **High:** Host multiple mini games
6. **Medium:** Different practice modes in type test
7. **Medium:** Battle Pass feature
8. **Low:** Multi-player

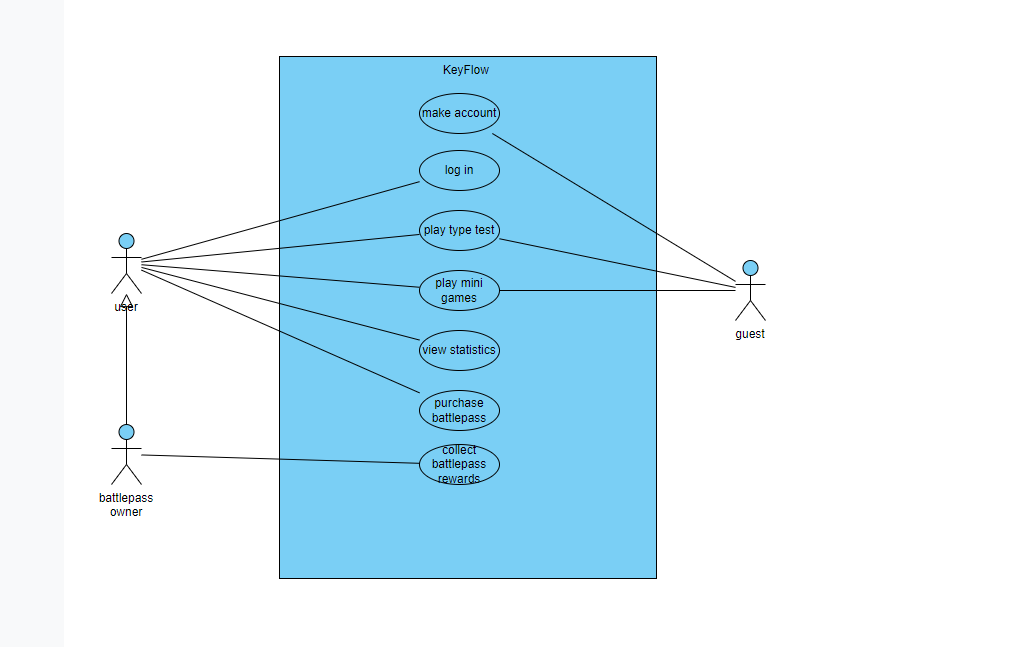
# Non-functional Requirements (10 points)

*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
* Distributed Database
* Ability to handle multiple users at one
  + especially for multiplayer implementation
* Reliability through maintaining a system uptime of 99%

# 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.*

# Class Diagram and/or Sequence Diagrams (15 points)

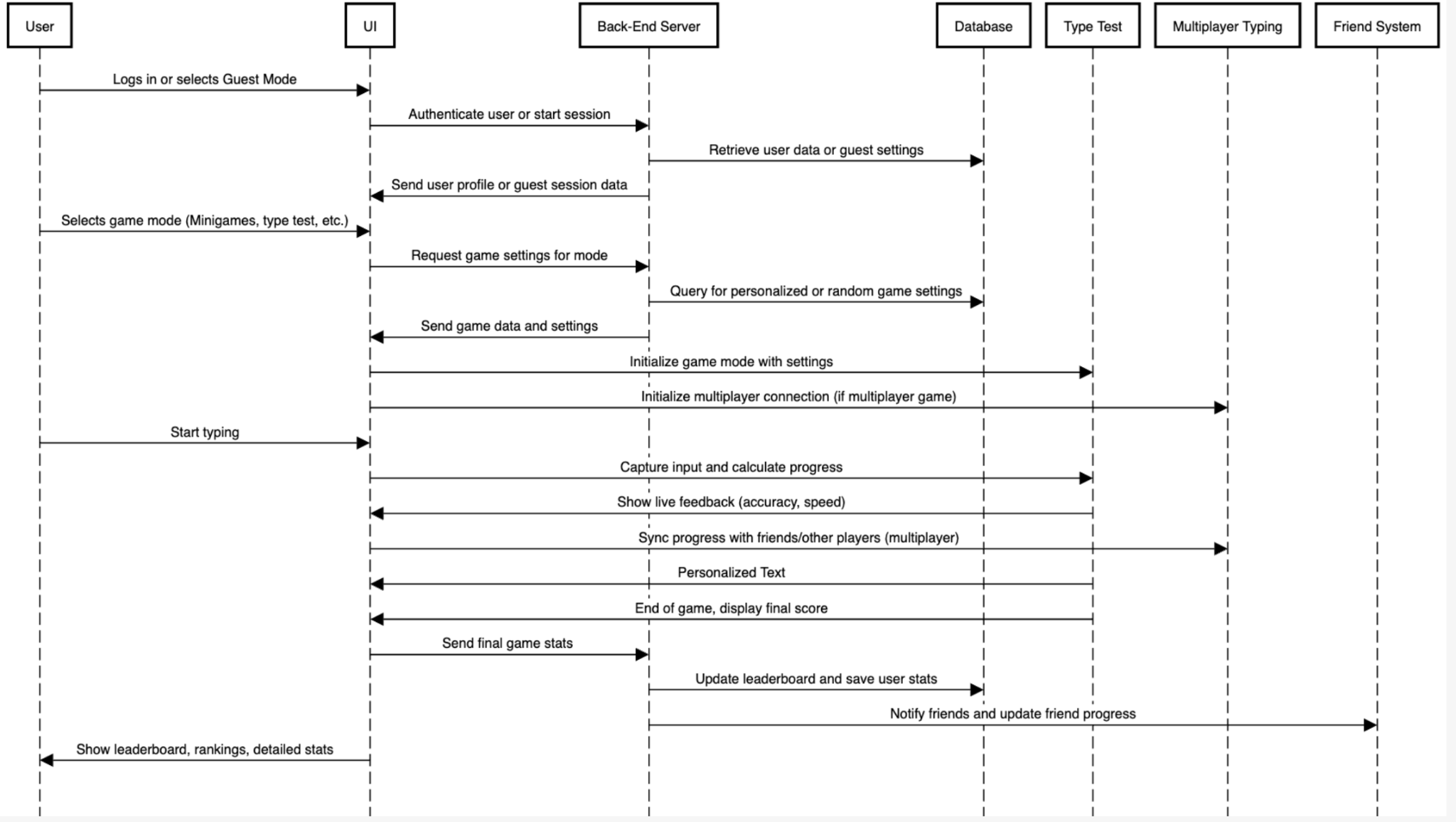
*This section presents a high-level overview of the anticipated system architecture using a* ***class******diagram*** *and/or* ***sequence diagrams****.*

*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.*



# 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.

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