Software Implementation and Testing Document

For

Group 9

Version 1.0

Authors:
Kaitlyn Krause
Tara Kerstetter
Megan Cole
Brandon Pina

1. Programming Languages (5 points)

List the programming languages use in your project, where you use them (what components of your project) and your reason for choosing them (whatever that may be).

a. Python
    i. We're using Python for the backend server and data processing
    ii. We chose Python due to its simplicity and extensive libraries. It is a good choice for rapid development.
b. JavaScript
    i. We're using javascript for game logic and frontend game development
    ii. We chose Phaser specifically for 2D game development for both of our typing minigames, Snowfall and SnowSlope
c. HTML/CSS
    i. We're using HTML/CSS for frontend design
    ii. HTML and CSS will be used to structure and style the non-game aspects of the web interface.

2. Platforms, APIs, Databases, and other technologies used (5 points)

List all the platforms, APIs, Databases, and any other technologies you use in your project and where you use them (in what components of your project).

a. PostgreSQL
    i. We are using PostgreSQL as our database for user profiles, typing statistics, game data, leaderboard stats, and to track user levels and XP for the battlepass
b. Phaser
    i. We are using Phaser for our typing games, Snowfall and Snowslope, and game logic
c. Django
    i. We are using Django as our web framework
d. EC2 (AWS):
    i. The platform that is hosting our project provides the virtual server on which all the project components are running.
e. Gunicorn
    i. It acts as the Python WSGI HTTP server that interfaces with our application. It manages all incoming HTTP requests and passes them on to the app.
f. Nginx
    i. Reverse proxy server in front of Gunicorn. It handles load balancing and the service of static files.

3. Execution-based Functional Testing (10 points)

*Describe how/if you performed functional testing for your project (i.e., tested for the **functional requirements** listed in your RD).*

High: Tracking user typing

We continued to test this feature as we added more code and functionality to our project by using the same test cases we did in increment 2 since we added no new functionality specifically to it. This was done just by conducting manual tests.

High: Sentence generation

` We tested that generated sentences with the new Personalized practice functionality generated sentences with the correct letters according to the players stats. We did this through manual tests with different test user accounts.

High: Statistical calculations

We continues to check the accuracy of statistics we calculated like WPM in new play modes like Personalized practice and Snowslope.

High: User information and statistics storage

New user information like XP and Levels were introduced in this increment as well as equipped battlepass items. We used save and retrieval operations on out database as well as using negative testing to make sure our system was reacting appropriately.

High: Host multiple mini games

We made sure that no new functionality affected this ability by reusing our old testing and testing each minigame

Medium: Different practice modes in type test

We tested our new Personalized practice feature to make sure the generated sentences were accurate and didnt experience any bugs.

Medium: Battle Pass feature

We tested the Battle Pass feature by testing it from authenticated and non authenticated accounts.

4. Execution-based Non-Functional Testing (10 points)

*Describe how/if you performed non-functional testing for your project (i.e., tested for the **non-functional requirements** listed in your RD).*

5. Password Encryption
    a. We examined our database to ensure that no passwords were stored in plain text and confirmed that our encryption algorithm is applied to both registration and login processes. We also implemented negative testing to ensure that attempting to access user accounts without correct credentials to make sure that encryption will not be bypassed.
6. Ability to handle multiple users at one
    a. We conducted load testing using tools like JMeter to simulate multiple users accessing the application simultaneously. Stress tests were performed to identify the maximum number of concurrent users the system could support without significant lag or crashes. We observed server performance, response times, and error rates during peak load times to ensure smooth operation for multiple users.
7. Reliability through maintaining a system uptime of 99%
    a. We monitored system uptime using automated monitoring tools, like AWS CloudWatch to track system availability over an extended period. We conducted failover testing to ensure the system could recover from unexpected failures.

8. Non-Execution-based Testing (10 points)

Describe how/if you performed non-execution-based testing (such as code reviews/inspections/walkthroughs).

Code Reviews:

We conducted structured code reviews within our team in order to catch bugs and improve the overall quality of our code. We did this by having each team member review code written by others focusing more on logic and readability than style guidelines. We did this during our weekly meetings. We were able to use our GitHub commits to track this.

Walkthroughs:

We conducted informal walkthroughs where we would explain our code and functionality to each other. These were focused on explaining the purpose, logic, and flow of the code and opening up discussion on possible areas for improvement. This helped the entire team with a better understanding of the codebase.