1. Introduction

- Purpose of the Documentation
  - This document is made in order to ensure the upkeep of this game is consistent. It includes the architecture overview, API documentation, testing,and maintenance.
- Audience
  - This is for the current developers, system administrators and any future members of that field in order to keep the consistency of the development and maintenance of the game.

2. API Documentation

- API Endpoints
  - "/getscores"
    - GET: Gets an object in json that contains our group name, a title, and the top 5 scores.
  - "/sendscores"
    - GET: Send an object in json that contains our group name, a title, and the top 5 scores to the public API that Professor Allgood provided.
  - "/score"
    - POST: Updates score for a certain user. Requires a username, score, level, and difficulty.
- Request and Response Formats
  - The top two API endpoints are get requests with no parameters
  - The score post request requires a name, score, level, and difficulty in a json body
    - {
      - "score": score,
      - "name":name,
      - "level":level,
      - "difficulty":difficulty
    - }
- Authentication Methods
  - There are no authentication methods

4. Testing Documentation

- Test Cases
  - Test cases created manually, and were not too extensive. We did not test for performance issues, mainly UI functionality and database compatibility.

  Examples:
  - A bug where levels were grayed out, however you could still access and play them.
  - Database scores were not being added correctly due to an issue with the level_number and level_difficulty values.
- Test Plans

    - For our testing, we mainly used Postman to test the json API, flask debugger (WSGI) for compiling issues, and the chrome dev console for further analysis.
    - Our testing strategy primarily was made up of creating a system, then testing it. For our game, we did not have too many interactions with the user which made testing simple. There were only a few edge cases to test such as what happens when a user is in a level, and then leaves.
- Quality Assurance Processes
    - While we did not follow a traditional code review process, we still looked over each other's code. For instance, if a new change was pushed, we would make sure the features worked and go through any edge cases we could think of. If none, there was no need to further analyze the code. This may not be practical for a real world environment, but for our purposes it was all we needed.
    - Furthermore, we tested our program on different systems. (Windows, Mac, and Linux)
    - No testing automation was used as that would overcomplicate our system.
    - For bug tracking, we would discuss the issue and decide whether it needed immediate attention or could be addressed in the future.

5. Integration Guides

- Libraries and Frameworks Used
    - All rendering is done with the Phaser3 framework
    - JQuery is used to send post request from the JavaScript code to update the scores and users in the databaseFlask is used to locally host the backend for all webpages and the game
    - Flask was used for the backend which hosted all webpages, including the game
- Integration Testing Guidelines
    - Each API change was tested using Postman to make sure that the correct changes were being made.
    - For the publicly provided api, we made sure that a 200 response code was returned
    - For our own api, we made sure that the correctly formatted scores were returned
    - For the score POST request, we made sure that scores were only updated if the score achieved was greater than the currently stored score for that level

6. Deployment

- Deployment Process
    - The software is deployed to a locally hosted Flask app, that is run on the host machine. To deploy the software, the user needs to run the python interpreter on the file app.py.
- URLS
    - All URLS API endpoints begin with “http://localhost:8000”
    - “/” and “/home”
        - POST: Submits form with login information, if valid or new, then moves to the level select.

        - GET: Returns the login/home page.
    - “/levels”
        - POST: Submits form with a given level based on which level was clicked. The level must be unlocked for the post request to go through.
    - GET: Returns the level select page where levels can be chosen.
    - “/leaderboard”
        - GET: Returns the leaderboard html page that shows the top 5 scores currently in the database.
- Environment Configuration
    - Our development environment involved vscode as our IDE, google chrome as the browser for testing and visualizing, JIRA for ticketing and assigning responsibilities, Github for keeping track of our code throughout development, and Postman for API testing.
- Continuous Integration/Continuous Deployment (CI/CD) Setup
    - We used JIRA and Github to allow continuous integration and deployment to occur seamlessly. We each created a separate GIT branch for each coding development and JIRA ticket and merged these branches to main when they were completed.

7. Maintenance

Bug Tracking and Resolution

- Development testing was done with the use of console.log() statements in the JavaScript. These log functions were used to constantly output expected values and actual values to determine if functions were returning the correct data.

Performance Monitoring

- All frames are calculated to line off with the timing of the music to line up each input with the beat of the song playing. This way frames are the same no matter what the refresh rate of the current monitor is.

8. Appendix

- References and External Documentation
    - Phaser3: https://phaser.io/
    - Flask: https://flask.palletsprojects.com/en/3.0.x/
    - Postman: https://www.postman.com/
    - JQuery: https://jquery.com/
    - JIRA: https://www.atlassian.com/software/jira
    - Github: https://github.com/
- Change Log

1. Tue Apr 30 17:31:14 2024 -0400: edit
2. Tue Apr 30 17:28:07 2024 -0400: leaderboard return button added
3. Tue Apr 30 17:20:34 2024 -0400: Leaderboard update
4. Sun Apr 28 16:02:11 2024 -0400: Update README.md
5. Sun Apr 28 16:01:39 2024 -0400: Update README.md

6. Sun Apr 28 16:00:47 2024 -0400: Update README.md
7. Sun Apr 28 16:00:36 2024 -0400: Update README.md
8. Sun Apr 28 16:00:24 2024 -0400: Update README.md
9. Sun Apr 28 15:53:45 2024 -0400: added public api call
10. Sun Apr 28 15:42:31 2024 -0400: Merge pull request #9 from jsidle1/multiplier
11. Sun Apr 28 15:42:13 2024 -0400: added multiplier
12. Sun Apr 28 15:24:21 2024 -0400: Merge pull request #8 from jsidle1/nightmare-fix
13. Sun Apr 28 15:23:52 2024 -0400: db changes
14. Sun Apr 28 15:20:43 2024 -0400: nightmare locked until completion and logout
15. Tue Apr 9 22:11:28 2024 -0400: fixed nightmare leveling visual
16. Tue Apr 9 22:02:22 2024 -0400: added api
17. Tue Apr 9 21:23:41 2024 -0400: Update index.html
18. Tue Apr 9 20:06:53 2024 -0400: Merge branch 'main' of https://github.com/jsidle1/Sphynx-Project
19. Tue Apr 9 20:04:38 2024 -0400: Added completed_difficulty functionality, alongside total_score in the users table.
20. Tue Apr 9 19:06:10 2024 -0400: fix broken index.html
21. Tue Apr 9 18:48:38 2024 -0400: Merge pull request #7 from jsidle1/level-select-gating
22. Tue Apr 9 18:48:22 2024 -0400: Merge branch 'main' into level-select-gating
23. Tue Apr 9 18:38:44 2024 -0400: remove db
24. Tue Apr 9 18:35:38 2024 -0400: add dynamic level select
25. Tue Apr 9 18:09:47 2024 -0400: DB update ignore
26. Tue Apr 9 18:07:48 2024 -0400: Updated DB to support nightmare levels.
27. Tue Apr 9 14:48:01 2024 -0400: Update README.md
28. Tue Apr 9 14:47:04 2024 -0400: Update README.md
29. Tue Apr 9 14:46:43 2024 -0400: Merge pull request #5 from jsidle1/game-bg
30. Tue Apr 9 14:45:57 2024 -0400: added multiple bg images
31. Tue Apr 9 14:31:18 2024 -0400: added game background and level text
32. Tue Apr 9 13:58:05 2024 -0400: added invalid password prompt
33. Tue Apr 9 13:49:51 2024 -0400: nightmare mode fixes
34. Tue Apr 9 03:54:14 2024 -0400: Update README.md
35. Tue Apr 9 03:53:45 2024 -0400: Merge pull request #4 from jsidle1/login-page-style
36. Tue Apr 9 03:53:26 2024 -0400: changed level select border
37. Tue Apr 9 03:50:56 2024 -0400: added background images and changed styles
38. Tue Apr 9 01:42:55 2024 -0400: Merge pull request #3 from jsidle1/scoring-changes
39. Tue Apr 9 00:04:50 2024 -0400: miss, good, great, and perfect added with pop-up
40. Mon Apr 8 23:04:00 2024 -0400: Merge pull request #2 from jsidle1/level-design
41. Mon Apr 8 23:03:48 2024 -0400: Merge branch 'main' into level-design
42. Mon Apr 8 23:00:16 2024 -0400: colors change with levels
43. Mon Apr 8 22:50:26 2024 -0400: levels added
44. Mon Apr 8 21:40:02 2024 -0400: Added instructions screen on the side of the game window.
45. Mon Apr 8 21:23:39 2024 -0400: Fixed DB not working (user_username for score table was spelled as just username)

46. Mon Apr 8 10:53:53 2024 -0400: Merge pull request #1 from jsidle1/timing
47. Mon Apr 8 10:51:38 2024 -0400: timing for audio
48. Sun Apr 7 20:11:05 2024 -0400: Added database functionality
49. Sat Mar 30 01:15:17 2024 -0400: Update index.js
50. Sat Mar 30 01:14:54 2024 -0400: Update index.html
51. Sat Mar 30 00:55:29 2024 -0400: communication
52. Wed Mar 27 14:51:09 2024 -0400: Update README.md
53. Wed Mar 27 14:49:42 2024 -0400: Update README.md
54. Wed Mar 27 14:49:22 2024 -0400: flask implementation
55. Wed Mar 27 14:48:39 2024 -0400: Delete phaser.min.js
56. Wed Mar 27 14:48:30 2024 -0400: Delete index.js
57. Wed Mar 27 14:48:20 2024 -0400: Delete index.html
58. Wed Mar 27 14:48:04 2024 -0400: Delete assets directory
59. Tue Mar 26 18:38:41 2024 -0400: Create README.md
60. Tue Mar 26 18:37:10 2024 -0400: Add files via upload