CRYPTID WEB APP

Abstract

Cryptid is a unique deduction game of honest misdirection in which players must try to uncover information about their opponents' clues while throwing them off the scent of their own. Each player holds one piece of evidence to help them find the creature, and on their turn they can try to gain more information from their opponents. Be warned; give too much away and your opponents might beat you to the mysterious animal and claim the glory for themselves! The game includes a modular board, five clue books, and a deck of set-up cards with hundreds of possible set-ups across two difficulty levels. It is also supported by an entirely optional digital companion, allowing for faster game set-up and a near-infinite range of puzzles.

2446659 – Ru'aan Maharaj
2578078 – Herry Ye
2541773 – James Stanton
2601486 – Jarren Downward

Contents

Introduction

This document provides a summary of the Scrum methodology followed throughout our project. Detailed evidence for each sprint is available in the respective folders. The screen recording of the project as well as the links to our GitHub, Web App and Taiga website.

Screen Recording video:

Here is the link to the video recording:

https://youtu.be/nb1NaSNgcY4

Publicly Hosted Application:

Here is the link to the publicly accessible web application:

https://happy-sand-063963b03.5.azurestaticapps.net/

GitHub Repository:

Here is the link to our GitHub repository:

https://github.com/2541773-wits/Cryptid-Web-App

Proof of Scrum Methodology:

- **Product Backlog**: A list of all features, enhancements, and bug fixes planned for the project. This should be a prioritized list with user stories or tasks.
- **Sprint Backlogs**: Detailed plans for each sprint, including the specific tasks and user stories selected for each sprint. Include the estimated effort (story points) for each item.
- **Sprint Burndown Charts**: Graphs showing the progress of work during each sprint. These charts should display the remaining effort over time, ideally showing a downward trend as work is completed. (Can be found in this link under Backlog: https://tree.taiga.io/project/2541773-cryptid-web-app/backlog).
- **Sprint Retrospective Reports**: Summary reports for each sprint retrospective meeting, highlighting what went well, what could be improved, and the action items for the next sprint.
- **Daily Stand-Up Summaries**: Brief notes or summaries of the daily stand-up meetings, including the main points discussed and any blockers identified.
- **Sprint Review Demonstrations**: Evidence of sprint review meetings, where the team demonstrates the work completed during the sprint. This could be in the form of meeting notes, screenshots, or video recordings.

Sprint Summaries:

Sprint 1:

1. **Objectives**: Develop the registration and login page and set up the database.
2. **Product Backlog**: Located in the Scrum meetings folder, Sprint 1 sub folder, Known as Product Backlog.
3. **Sprint Backlogs**: Located in the Scrum meetings folder, Sprint 1 sub folder, Known as Sprint Backlog.
4. **Sprint Retrospective reports**: Located in the Scrum meetings folder, Sprint 1 sub folder, Known as Sprint Retrospective.
5. **Daily Stand-up Summaries**: Located in the Scrum meetings folder, Sprint 1 sub folder, Known as Daily Scrums.
6. **Sprint Review Demonstrations**: Located in the Scrum meetings folder, Sprint 1 sub folder, Known as Sprint Review.
7. **Summary**: During Sprint 1, we successfully implemented the registration and login page and established the initial database setup. The sprint review was positive, with minor suggestions for improving the user interface.

Sprint 2:

1. **Objectives**: Set up the board configuration, develop the home page, link both with the database, and begin UML diagrams.
2. **Product Backlog**: Located in the Scrum meetings folder, Sprint 2 sub folder, Known as Product Backlog.
3. **Sprint Backlogs**: Located in the Scrum meetings folder, Sprint 2 sub folder, Known as Sprint Backlog.
4. **Sprint Retrospective reports**: Located in the Scrum meetings folder, Sprint 2 sub folder, Known as Sprint Retrospective.
5. **Daily Stand-up Summaries**: Located in the Scrum meetings folder, Sprint 2 sub folder, Known as Daily Scrums.
6. **Sprint Review Demonstrations**: Located in the Scrum meetings folder, Sprint 2 sub folder, Known as Sprint Review.
7. **Summary**: Sprint 2 focused on developing the board configuration and home page, integrating them with the database, and starting UML diagrams. The review was positive, with feedback suggesting further refinements to the board configuration.

Sprint 3:

1. **Objectives**: Make the board interactive for placing items, update UML diagrams, and enhance UI designs.
2. **Product Backlog**: Located in the Scrum meetings folder, Sprint 3 sub folder, Known as Product Backlog.
3. **Sprint Backlogs**: Located in the Scrum meetings folder, Sprint 3 sub folder, Known as Sprint Backlog.
4. **Sprint Retrospective reports**: Located in the Scrum meetings folder, Sprint 3 sub folder, Known as Sprint Retrospective.
5. **Daily Stand-up Summaries**: Located in the Scrum meetings folder, Sprint 3 sub folder, Known as Daily Scrums.
6. **Sprint Review Demonstrations**: Located in the Scrum meetings folder, Sprint 3 sub folder, Known as Sprint Review.
7. **Summary**: In Sprint 3, we made the board interactive, allowing users to place items, updated the UML diagrams, and improved the UI designs. The review highlighted the effectiveness of the interactive board and suggested minor UI adjustments.

Sprint 4:

1. **Objectives**: Add the recording session feature, show users every move they played using a JSON file, and finalize UI developments.
2. **Product Backlog**: Located in the Scrum meetings folder, Sprint 4 sub folder, Known as Product Backlog.
3. **Sprint Backlogs**: Located in the Scrum meetings folder, Sprint 4 sub folder, Known as Sprint Backlog.
4. **Sprint Retrospective reports**: Located in the Scrum meetings folder, Sprint 4 sub folder, Known as Sprint Retrospective.
5. **Daily Stand-up Summaries**: Located in the Scrum meetings folder, Sprint 4 sub folder, Known as Daily Scrums.
6. **Sprint Review Demonstrations**: Located in the Scrum meetings folder, Sprint 4 sub folder, Known as Sprint Review.
7. **Summary**: Sprint 4 concentrated on adding the recording session feature, showing users their moves through a JSON file, and finalizing UI developments. The review showed that the product was ready for submission, and the team felt confident about the results.

Overall Sprint Summary:

Overall, our project progressed smoothly through the four sprints. We consistently improved our planning, task delegation, and communication. Each sprint review and retrospective provided valuable insights that we incorporated into subsequent sprints, leading to a successful project completion. The use of Scrum methodology ensured that we remained focused, adaptable, and collaborative throughout the project.

Overview of Additional Artifacts:

- **Project Plan**: A high-level document outlining the overall plan for the project, including major milestones and deadlines.
- **Architecture Diagram**: A visual representation of the software architecture, showing the main components and their interactions.
- **Design Documents**: Any additional design documentation, such as class diagrams, sequence diagrams, or wireframes, that support the development of your project.
- **Test Plan and Results**: A document detailing your testing strategy, test cases, and the results of your testing efforts. Screenshots or logs of your automated testing will suffice.

Project Plan:

Overview

We used Taiga for documenting everything with tasks and keeping track of progress. The project plan outlines the overall schedule, major milestones, and deadlines.

Major Milestones

1. Sprint 1: Setup and implement registration and login page, initial database setup.
2. Sprint 2: Develop board configuration, home page, link to database, begin UML diagrams.
3. Sprint 3: Make the board interactive, update UML diagrams, enhance UI designs.
4. Sprint 4: Add recording session feature, display moves via JSON, finalize UI developments.

Timeline

1. Sprint 1: [10/04/2024] - [21/04/2024]
2. Sprint 2: [21/04/2024] - [28/04/2024]
3. Sprint 3: [28/04/2024] - [06/05/2024]
4. Sprint 4: [06/05/2024] - [20/05/2024]

Detailed Plan

Refer to the Taiga Project Plan for detailed task breakdowns and timelines.

Here is the link to the Taiga Project:

https://tree.taiga.io/project/2541773-cryptid-web-app/timeline

Architecture Diagram

Overview

The architecture diagram provides a visual representation of the software's structure, highlighting the main components and their interactions.

Diagrams

Overall Architecture Diagram: Shows the main components and their interactions.

Detailed Diagrams:

Refer to the Architecture Diagrams in the folder that says All UML Diagrams, which contains 3 subfolders, which are the sprint 2,3 and 4 UML diagrams.

Design Documents

Overview

Design documents include various UML diagrams that were updated throughout the sprints to reflect the evolving design of the project.

Documents

- Sprint 2 UML Diagrams: Initial class diagrams and sequence diagrams.
    - Class Diagram
    - Sequence Diagram
    - Deployment Diagram
    - Use case Diagram
    - State Diagram
    - Component Diagram
    - Activity Diagram

- Sprint 3 UML Diagrams: Updated class diagrams and sequence diagrams.
    - Class Diagram
    - Sequence Diagram
    - Deployment Diagram
    - Use case Diagram
    - State Diagram
    - Component Diagram
    - Activity Diagram

- Sprint 4 UML Diagrams: Finalized class diagrams and sequence diagrams.
    - Class Diagram
    - Sequence Diagram
    - Deployment Diagram
    - Use case Diagram
    - State Diagram
    - Component Diagram
    - Activity Diagram

Detailed Diagrams:

Refer to the Architecture Diagrams in the folder that says All UML Diagrams, which contains 3 subfolders, which are the sprint 2,3 and 4 UML diagrams.

Test Plan and Result

Introduction

This section outlines the testing strategy, test cases and results for the testing of the JavaScript modules: “script.js”, “login-validation.js”, “game.js”, “runGame.js”.

Testing strategies

- Unit Test: Individual functions and methods are tested in isolation to make sure they behave as expected.
- Integration Test: Modules are tested together to ensure their integration and compatibility.

Test Cases

- script
  1 registration_validation
    - return true for valid input.
    - returns false and sets error message for username less than 3 characters.
    - returns false and sets error message for invalid email format.
    - returns false and sets error message for password less than 8 characters.
    - returns false and sets error message for passwords not matching.

  2 create users
    - should call fetch with the correct endpoint and data.
    - should return the response value.
- login-validation
  1 list
    - fetches user data from API endpoint.
    - handles API error.

  2 findUser
    - find user by username and password.
    - find user by email and password.
    - return undefined if user is not found.

  3 login-validation
    - show user not found popup when user is not found.
    - show success popup when user is found.

- game.js
  1 get_board_configuration
    - Get board configurations from mapcode.
    - Get board configurations from hard mapcode.

  2 get_game_config
    - return correct game configuration.

  2 get_destination
    - return correct destination coordinates.

  3 get_clues

- return correct clues.

4 get_hints
- return correct hint.

5 getSubmatrix
- return correct submatrix.

6 rotate
- rotates a 2x2 matrix.
- rotates a 3x3 matrix.

7 mergeMatricesByRows
- merges two 2x2 matrices.
- throws an error if matrices have different number of rows.

8 board_layout
- generates the correct board layout.

9 isBear function
- return true for valid combinations.
- should return false for invalid combinations.

10 isCougar function
- return true for valid combinations.
- return false for invalid combinations.

* runGame.js

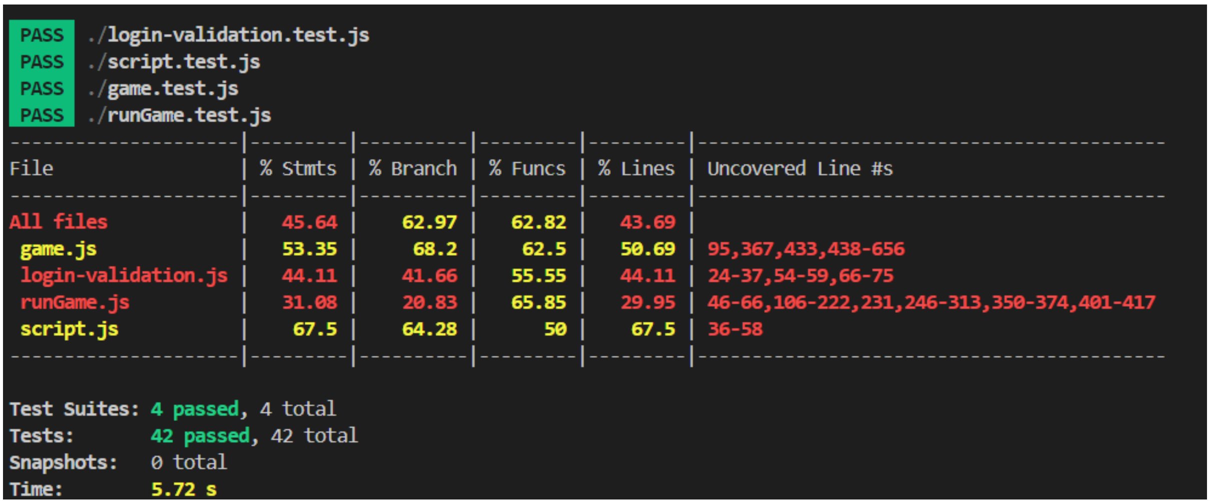
1 game class
- get number of players.
- get current player.
- move to the next player.
- change the current player.

2 boardInfo class
- listen for player selection.
- listen for yes/no button clicks.
- place a disc on the board.
- place a cube on the board.
- place a pawn on the board.
- populate player select options correctly.

3 GameRecord class
- record a move.
- convert moves to JSON.

Testing Result

Code passes all the test cases. Below is a screenshot of running test locally.



```
 PASS  ./login-validation.test.js
 PASS  ./script.test.js
 PASS  ./game.test.js
 PASS  ./runGame.test.js
---------------------|---------|----------|---------|---------|-------------------------------------------
File                 | % Stmts | % Branch | % Funcs | % Lines | Uncovered Line #s
---------------------|---------|----------|---------|---------|-------------------------------------------
All files            |   45.64 |    62.97 |   62.82 |   43.69 |
 game.js             |   53.35 |     68.2 |    62.5 |   50.69 | 95,367,433,438-656
 login-validation.js |   44.11 |    41.66 |   55.55 |   44.11 | 24-37,54-59,66-75
 runGame.js          |   31.08 |    20.83 |   65.85 |   29.95 | 46-66,106-222,231,246-313,350-374,401-417
 script.js           |    67.5 |    64.28 |      50 |    67.5 | 36-58
---------------------|---------|----------|---------|---------|-------------------------------------------

Test Suites: 4 passed, 4 total
Tests:       42 passed, 42 total
Snapshots:   0 total
Time:        5.72 s
```

Conclusion

This document provides a comprehensive overview of the additional artifacts required for our project. Each section includes detailed references to the respective documents and diagrams, which are essential for understanding the planning and the design of the project.