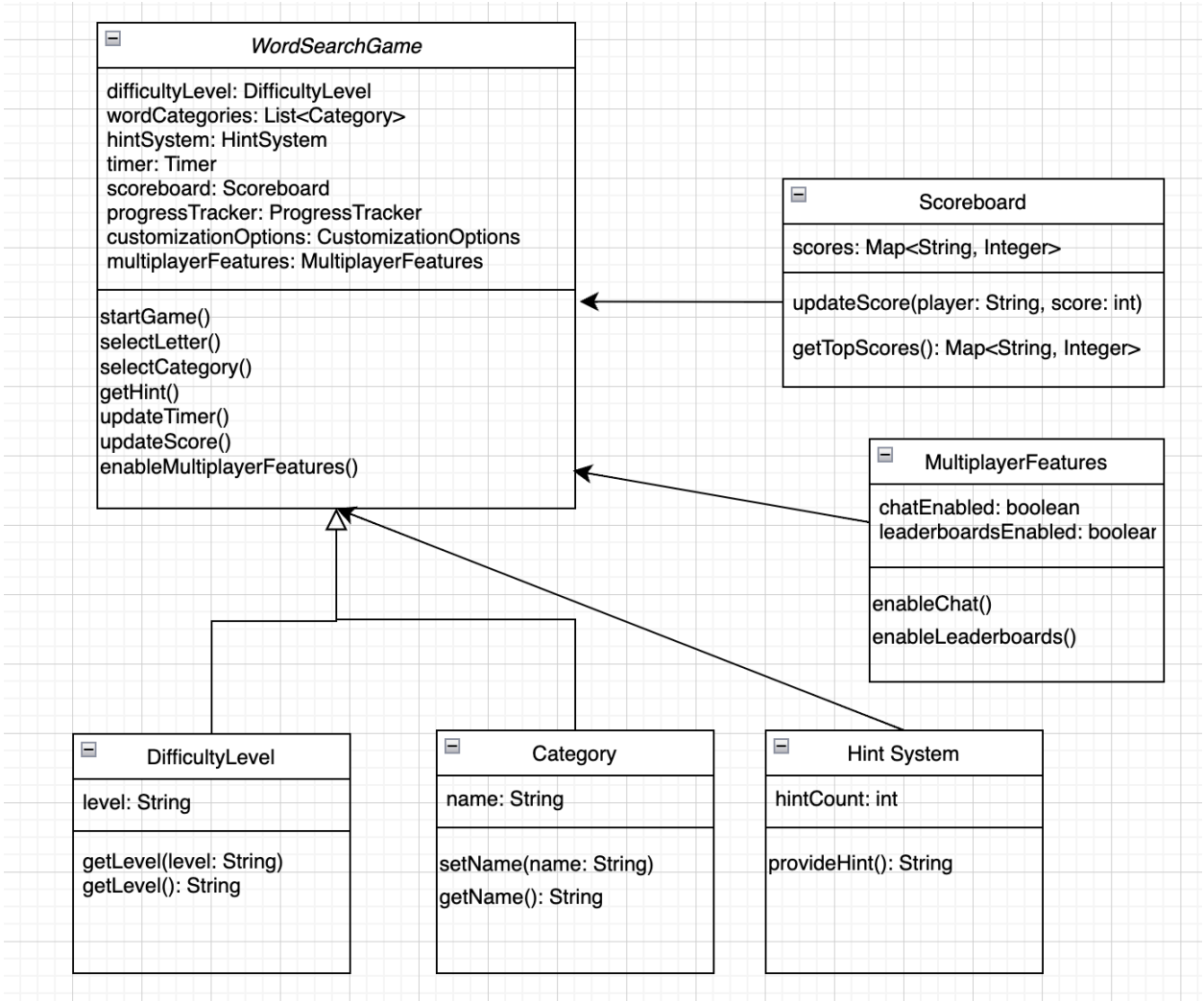
Product Design Specification (Individual Flask Project)

Project Description

Project Title: Word Search Game
GitHub Repository URL: https://github.com/daniajaison13/WordSearch

A word search game where players search for hidden words in a grid of letters. The game can include different categories and difficulty levels, making it suitable for players of all ages.



Algorithms/AI Schemes:

- Grid Generation: Generate a grid of letters with rows and columns. Randomly select letters to fill the grid.
- Word Placement Algorithm: Use techniques such as backtracking to strategically place hidden words. Ensure words intersect and do not conflict with each other or violate constraints.

Market Space and Selling Points:

- The application targets users of all ages who enjoy puzzle games.
- Selling points include multiple categories, varying difficulty levels, and an intuitive user interface.

Functional specifications

List of Product Features

- **Interactive Word Grid:**
  Display a grid of letters where players search for hidden words.
  Allow players to select letters horizontally, vertically, or diagonally to form words.
- **Multiple Difficulty Levels:**
  Offer various difficulty levels such as easy, medium, and hard.
  Adjust grid size and word complexity accordingly.
- **Word Categories:**
  Include diverse word categories like animals, fruits, countries, etc.
  Allow users to select their preferred categories for word search.
- **Hint System:**
  Provide hints for players struggling to find words.
  Limit the number of hints available to maintain challenge.
- **Timer and Scoring:**
  Incorporate a timer to track how quickly players complete the puzzle.
  Assign scores based on time taken to solve the puzzle and accuracy.
- **Progress Tracking:**
  Keep track of completed puzzles and overall progress.
  Show achievements and milestones to motivate players.
- **Customization Options:**
  Allow players to customize grid appearance and background themes.
  Enable font size adjustments for better readability.

Multi-Player Features

- **Chat Functionality:**
  Integrate a chat feature for players to communicate during gameplay.
  Enable pre-defined messages and emojis for quick interaction.

- **Leaderboards:**
  Display leaderboards showcasing top players and their achievements.
  Encourage healthy competition among participants.

Deployment

1. Prepare your Flask application by ensuring proper structure and dependencies listed in requirements.txt.

2. Sign up for a Heroku account if you haven't already done so.

3. Install the Heroku Command Line Interface (CLI) on your local machine.

4. Initialize a Git repository if your project isn't already one.

5. Use the Heroku CLI to create a new app with heroku create.

6. Deploy your Flask application by pushing your code to the Heroku remote repository with git push heroku master.

7. Set necessary environment variables for your application through the Heroku CLI or dashboard.

8. Scale your application by adjusting dynos using the Heroku CLI or dashboard.

9. Monitor your application's logs using heroku logs --tail for debugging.

10. Access your deployed Flask application through the provided Heroku URL.

Milestones with Deadlines:

Week 1: Basic grid generation and word placement algorithm implemented

Week 2: Flask integration for basic UI setup

Week 3: Category selection and difficulty levels implemented

Week 4: Integrate backend and frontend

Week 5: Final testing and bug fixing