**Battleship: Navy vs Pirates**

**Project Report**

**Team members:**

**Zehra Jabeen Mirza 22K-4781**

**M. Shehryar Zubair Khan 22K-4736**

**1. Project Overview**

*Battleship: Navy vs Pirates* is a web-based interactive game that reimagines the classic Battleship gameplay with modern enhancements. It features a themed battle between the Navy (player) and the Pirates (AI), incorporating advanced AI strategies, dynamic power-ups, responsive design, and immersive audio-visual elements. The game is designed to be both challenging and entertaining across a range of player skill levels.

**2. Development Objectives**

The primary objectives guiding the development of this project were:

1. Develop an engaging and intuitive Battleship game playable in a web browser.
2. Implement a tiered AI system with increasing strategic complexity.
3. Introduce unique gameplay features, such as the “Air Strike” power-up.
4. Craft a visually appealing, ocean-themed UI.
5. Integrate sound effects to enhance player immersion and feedback.

**3. Technical Architecture**

**Frontend Technologies**

* **HTML5/CSS3**: Structured layout with responsive design principles.
* **JavaScript**: Core gameplay logic and user interaction handling.

**Backend Technologies**

* **Python**: Game logic and AI computation.
* **Flask**: Lightweight web framework providing REST API endpoints.
* **NumPy**: AI calculations for probability-based targeting.
* **Session Management**: Persistent game state between requests.

**System Design**

* **Client-Server Model**: JavaScript frontend communicates with Flask backend.
* **REST API Endpoints**:
  + /new\_game: Initializes a new session.
  + /place\_ships: Validates and stores ship positions.
  + /player\_shoot: Processes user’s turn.
  + /player\_air\_strike: Executes special power-up.
  + /get\_game\_state: Syncs client with current game status.

**4. Game Features**

**Setup and User Interaction**

* Multiple difficulty levels: Easy, Medium, Hard, Extremely Hard.
* Ship placement via drag-and-drop or click controls.
* Ship rotation and randomized layout options.
* Real-time feedback for placement validity.

**Gameplay Mechanics**

* Alternating turns between user and AI.
* Interactive grid with hit/miss feedback.
* Ship destruction tracking and game-over states.
* Scorekeeping system for repeated sessions.

**AI Behavior by Difficulty**

|  |  |
| --- | --- |
| **Mode** | **Description** |
| **Easy** | 70% random, 30% basic follow-up on hits. Minimal logic. |
| **Medium** | Probability-based targeting with ship-size awareness. |
| **Hard** | Enhanced pattern recognition, checkerboard search, and density mapping. |
| **Extremely Hard** | Sophisticated analysis including heatmaps, center prioritization, and learned player behaviors. |

**Special Features**

* **Air Strike**: One-time special attack covering a full row or column.
* **Sound Effects**: Distinct audio for actions like hits, misses, and victory.
* **Responsive UI**: Adapts seamlessly to mobile, tablet, and desktop screens.
* **Theme Customization**: Pirate vs Navy visual elements and aesthetics.

**5. Development Challenges & Solutions**

**AI Difficulty Balancing**

* **Challenge**: Creating fair yet challenging AI opponents.
* **Solution**: Layered AI development, gradually increasing in complexity and tactical awareness.

**Game State Synchronization**

* **Challenge**: Maintaining consistent state across client-server interactions.
* **Solution**: Session-based state management and detailed state objects with error-checking logic.

**Air Strike Integration**

* **Challenge**: Implementing a balanced, impactful power-up.
* **Solution**: Carefully limited usage with intuitive UI feedback and backend validation.

**Frontend-Backend Cohesion**

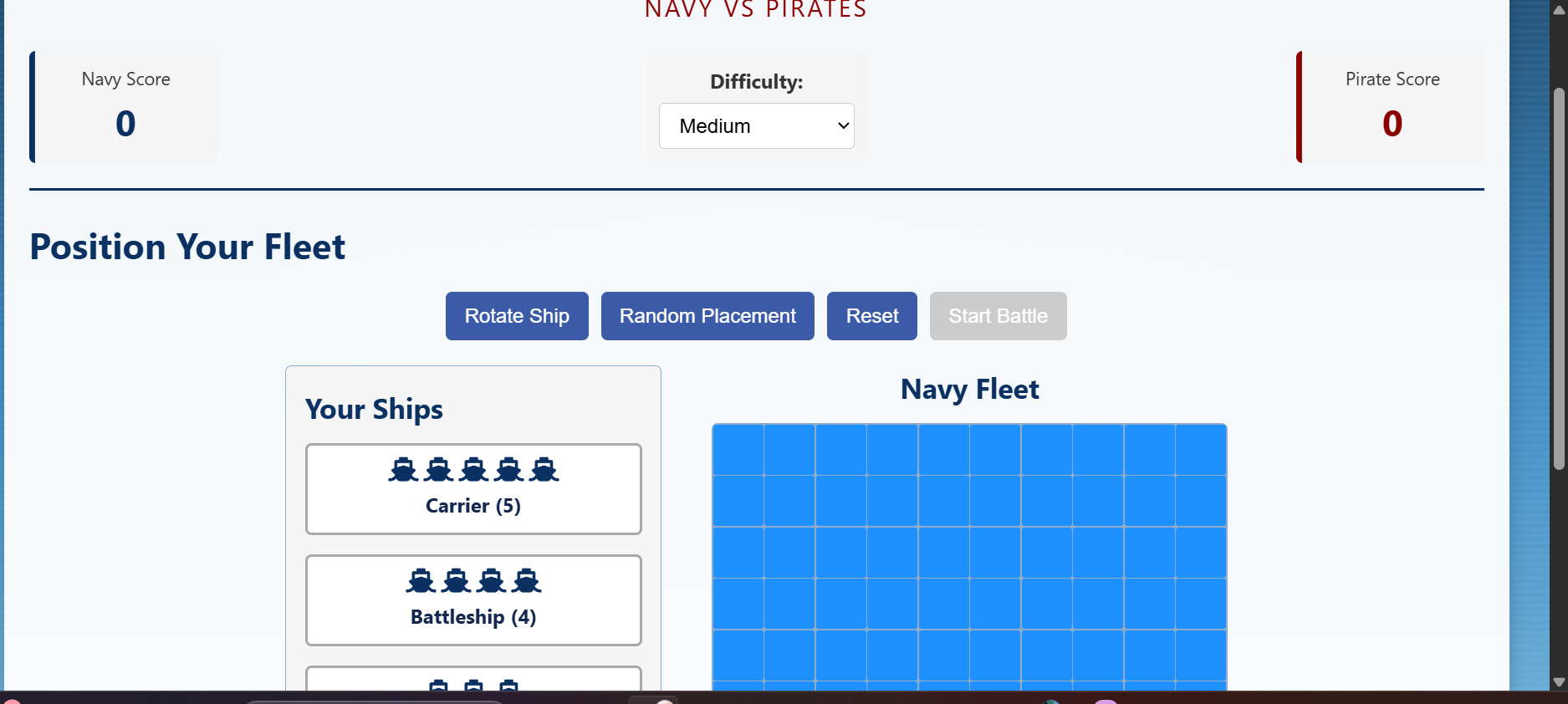
* **Challenge**: Ensuring fluid communication between JavaScript and Flask APIs.
* **Solution**: RESTful API architecture with standardized responses and robust error handling.

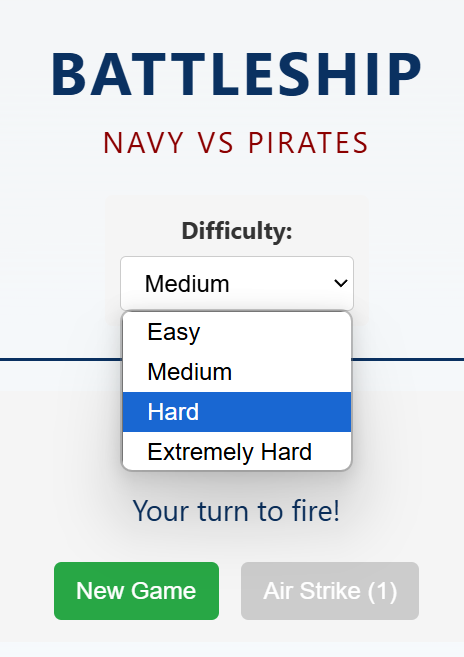
**6. Testing and Quality Assurance**

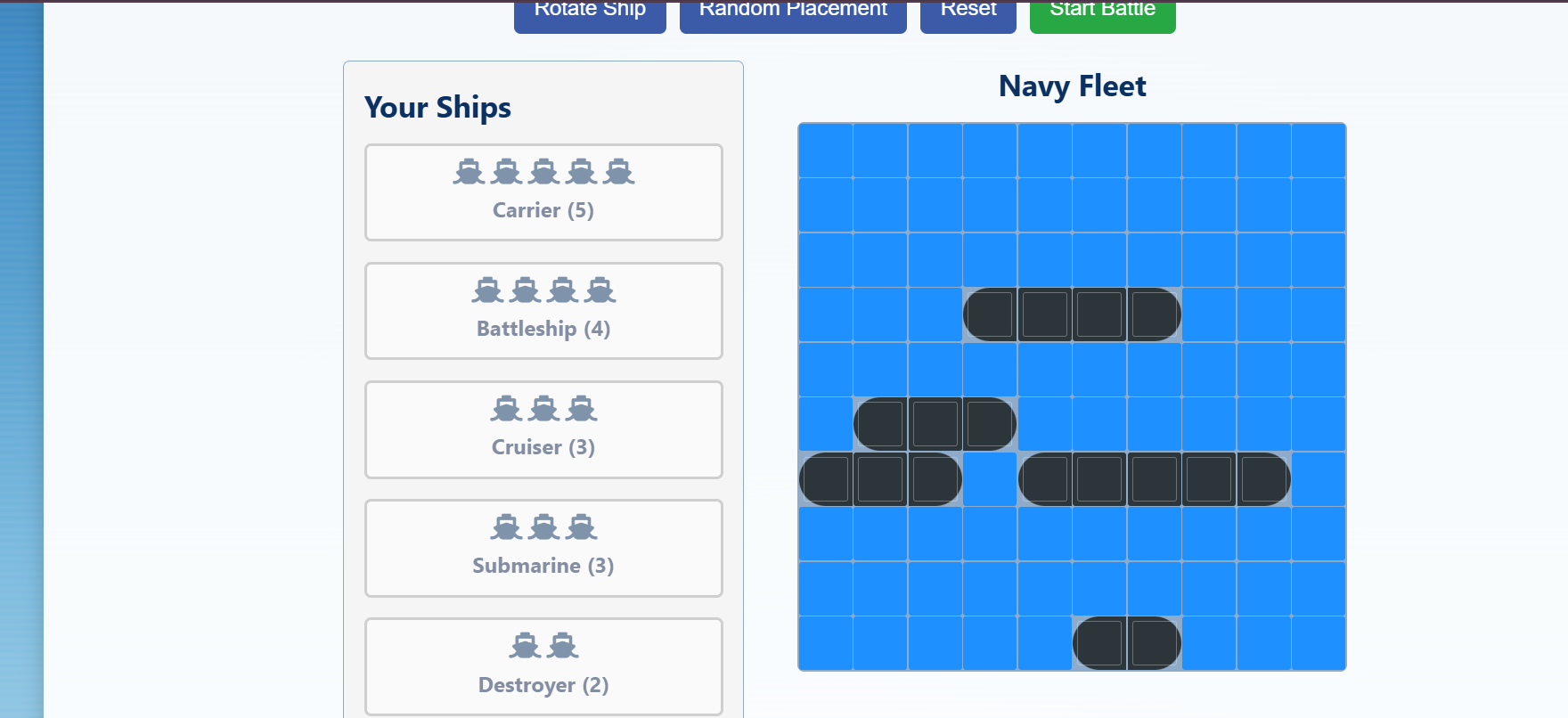
Multiple levels of testing ensured performance, stability, and usability:

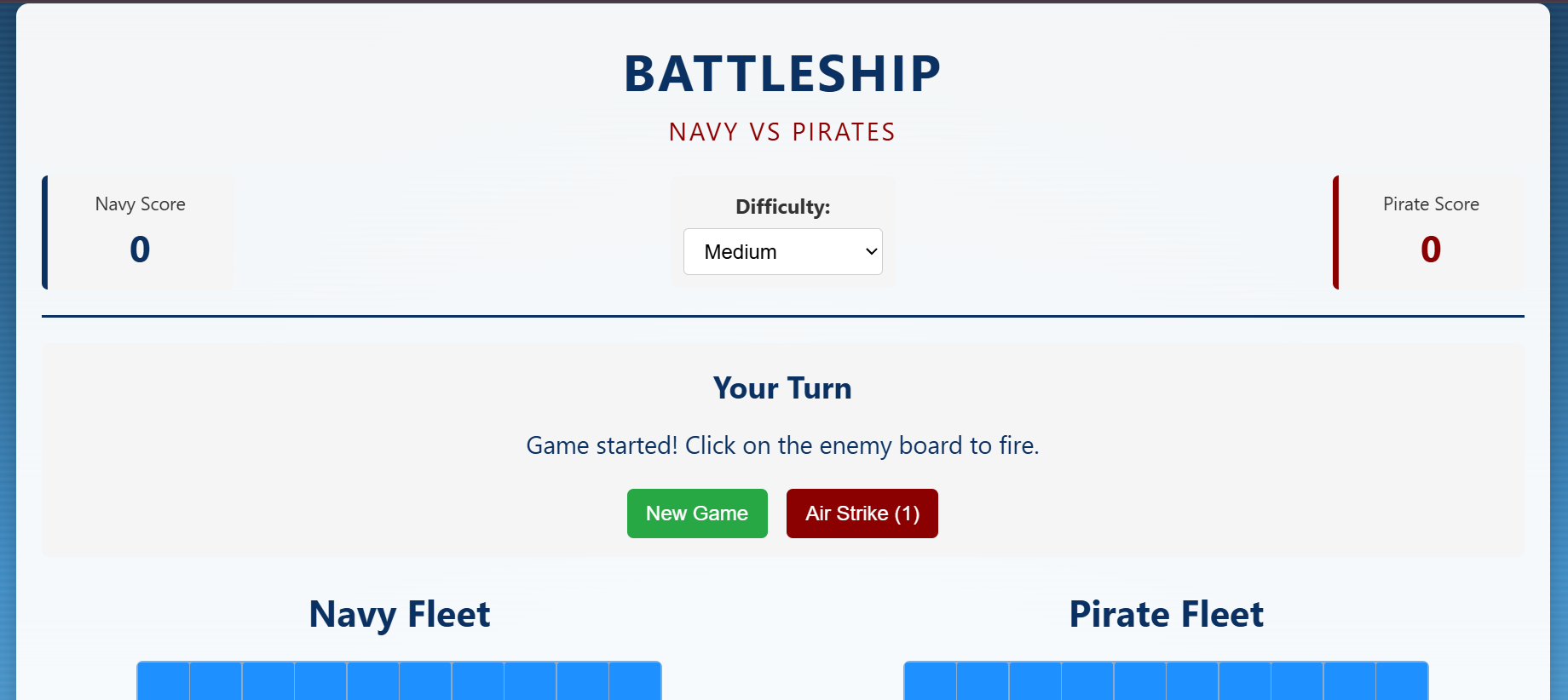
* **Unit Testing**: For AI logic, state transitions, and utilities.
* **Integration Testing**: API endpoints tested with various edge cases.
* **User Testing**: Conducted gameplay sessions for UX feedback.
* **Performance Testing**: Ensured minimal latency in user interactions and AI responses.

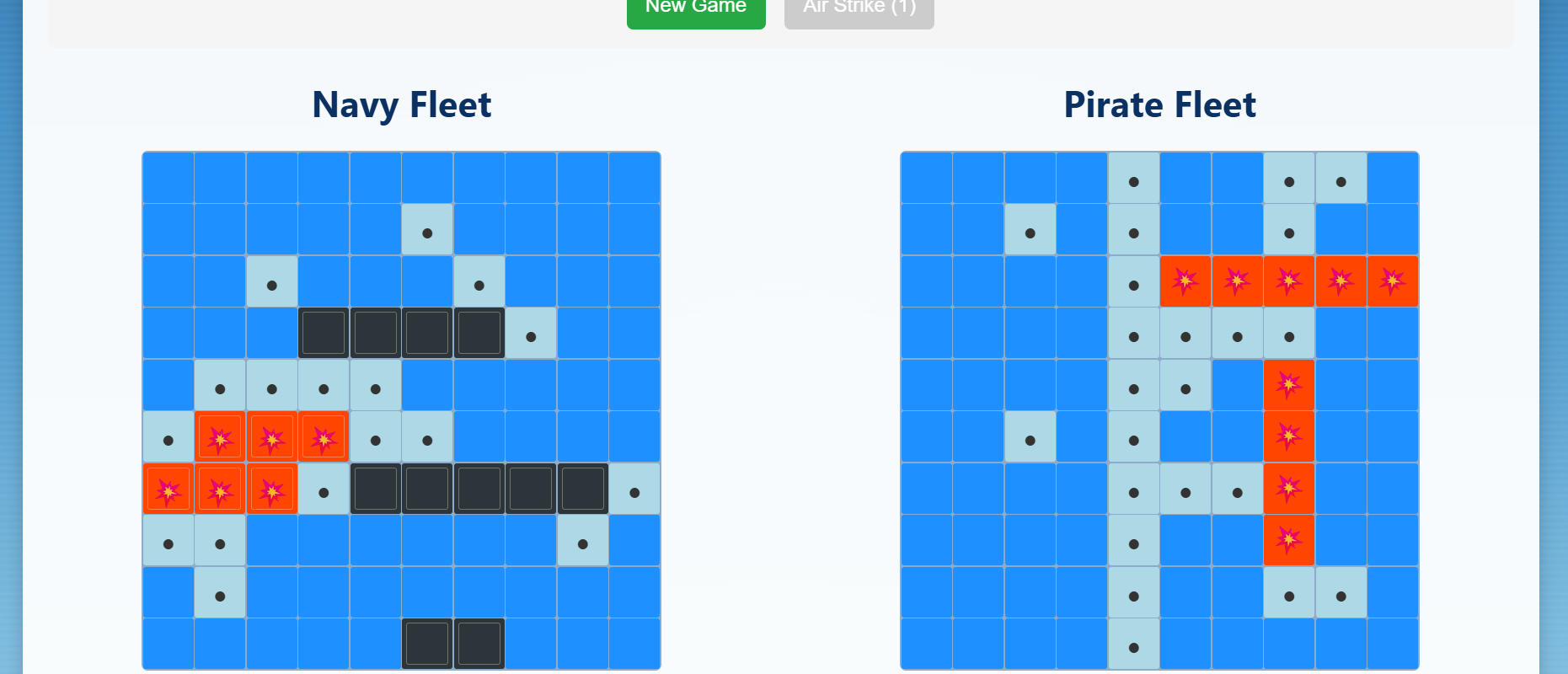
**Working of the game:**

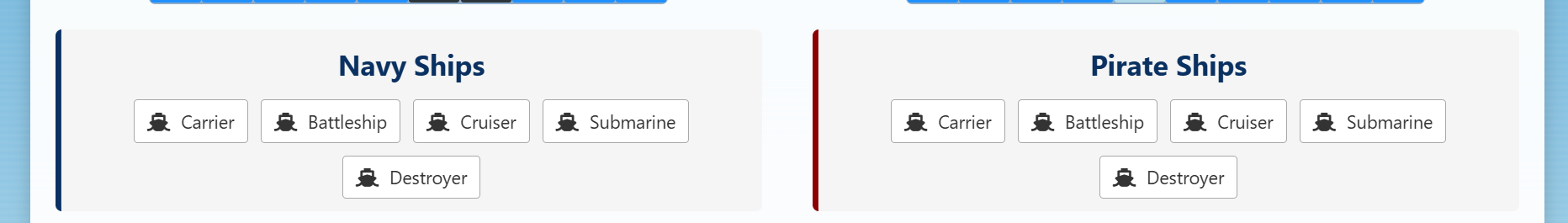


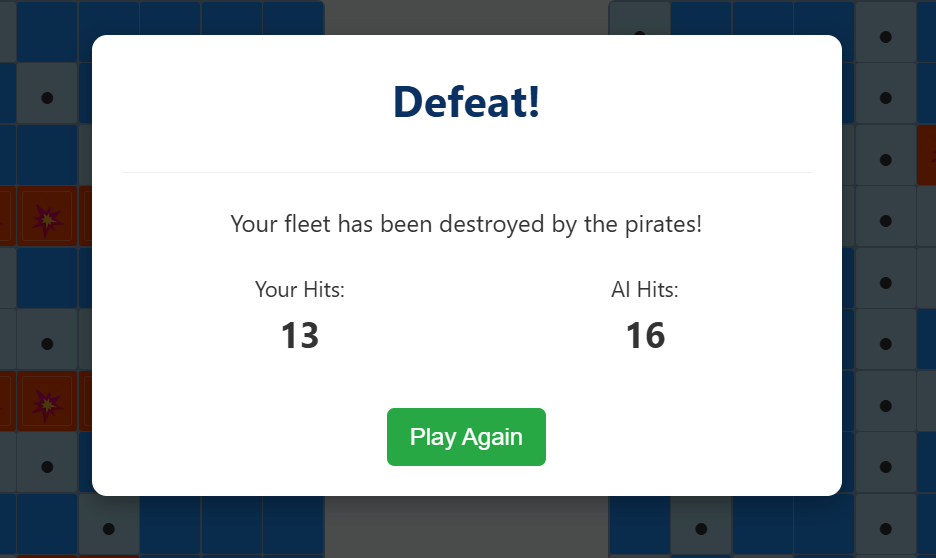












**7. Conclusion**

The successful completion of this project demonstrates the effective fusion of frontend design, backend logic, and algorithmic AI modeling. The framework laid out provides a solid base for future expansion and gameplay enhancements, ensuring lasting replay value.