**Designing a Software Solution for Battleship**

**Battleship Software Design**

**Overview**

This project aims to develop a software version of the classic two-player game **Battleship**. The software will replicate the core gameplay mechanics of the physical board game while ensuring a user-friendly, rule-enforced digital experience. The first phase of this project involves gathering requirements through a simulated interview and converting them into user stories with clear acceptance criteria.

**Part 1: Requirements Gathering via an Interview**

**Interview Report**

**Interviewee:** Simulated Battleship Expert  
**Interviewer:** Poojitha Reddy Challa

**Q1: What is the goal of Battleship?**  
**A:** To sink all of the opponent’s ships by guessing their positions on the grid.

**Q2: How many players play the game?**  
**A:** Two players.

**Q3: What items are used in the game?**  
**A:** Each player has two grids, five ships (carrier, battleship, cruiser, submarine, destroyer), and red and white pegs.

**Q4: How do players play the game?**  
**A:** Players take turns calling out grid points to hit the other player’s ships. The game continues until one player's fleet is completely destroyed.

**Q5: How are ships placed?**  
**A:** Vertically or horizontally, without overlapping or going off the grid.

**Q6: How do you know when a ship sinks?**  
**A:** When all its grid positions have been hit.

**Q7: What aspects are challenging for players?**  
**A:** Remembering previous guesses, tracking which ships are still in play, and recalling the rules.

**Part 2: User Stories and Acceptance Criteria**

**1. Ship Placement**

**User Story:**  
As a player, I want to place my ships on a grid so I can prepare for the game.

**Acceptance Criteria:**

* Ships must be placed vertically or horizontally (not diagonally).
* Ships must remain within the grid boundaries.
* Ships cannot overlap.
* Players can confirm placement before starting the game.

**2. Turn-Based Attacks**

**User Story:**  
As a player, I want to call out a grid coordinate to try and hit my opponent’s ship.

**Acceptance Criteria:**

* Coordinates use a letter (A–J) and a number (1–10).
* The game notifies the player with “Hit” or “Miss.”
* The UI clearly reflects successful hits and misses.

**3. Guess Tracking**

**User Story:**  
As a player, I want to mark my guesses, so I don’t repeat them.

**Acceptance Criteria:**

* Red pegs indicate hits; white pegs indicate misses.
* Previous guesses are visually marked on the top grid.
* Players cannot repeat the previous coordination.

**4. Ship Sinking Notification**

**User Story:**  
As a player, I want to know when one of my ships is sunk.

**Acceptance Criteria:**

* When all grid spots of a ship are hit, a message appears (e.g., “You sunk my cruiser!”).
* A visual update reflects the ship's status.

**5. Rule Enforcement**

**User Story:**  
As a player, I want the game to prevent invalid actions like illegal ship placements.

**Acceptance Criteria:**

* The game checks for valid ship positions during placement.
* Error messages guide players to correct mistakes.

**6. Fair Turn Order**

**User Story:**  
As a player, I want the game to randomly select the starting player to ensure fairness.

**Acceptance Criteria:**

* A system randomly determines the starting player.
* A message announces who begins the game.

**7. Game End Condition**

**User Story:**  
As a player, I want the game to notify me when I win or lose.

**Acceptance Criteria:**

* The game monitors the status of all ships.
* A message is displayed when a player wins.

**8. Progress Visualization**

**User Story:**  
As a player, I want to see which enemy ships I’ve sunk to feel my progress.

**Acceptance Criteria:**

* The interface shows a list or icon of sunken ships.
* Each sunk ship is labeled (e.g., “Cruiser,” “Destroyer”).