Manual Test Cases:

1. **Usability Testing:**

   Interface Navigation:

   - Action: Navigate through different sections of the game interface (homepage, betting area, account settings).
   - Expected Result: Smooth navigation with clear visual cues and intuitive layout.

   Instruction Clarity:

   - Action: Read and follow game instructions for placing bets and spinning the wheel.
   - Expected Result: Clear and concise instructions that guide users through gameplay without confusion.

   Consistency Check:

   - Action: Compare design elements (colour scheme, font style) across various screens and sections.
   - Expected Result: Consistency in design elements, ensuring a cohesive user experience throughout the game.

   Button Responsiveness:

   - Action: Click on different buttons (bet placement, spin) and observe response time.
   - Expected Result: Buttons respond promptly to user interactions, providing immediate feedback.

   Overall User Experience:

   - Action: Engage in gameplay from start to finish, including placing bets, spinning the wheel, and viewing outcomes.
   - Expected Result: Smooth and enjoyable user experience with no major usability issues encountered.

2. **Security Testing:**

   Data Handling Review:

   - Action: Review data handling practices to ensure encryption of sensitive information (user credentials, payment details).
   - Expected Result: User data is securely encrypted both at rest and in transit, mitigating the risk of unauthorized access.

   Authentication Mechanism Testing:

- Action: Attempt to bypass authentication mechanisms and access restricted areas of the game.
- Expected Result: Authentication mechanisms effectively restrict access to authorized users only, preventing unauthorized access.

Compliance Assessment:

- Action: Evaluate compliance with security best practices and industry standards.
- Expected Result: The game adheres to established security standards, minimizing the risk of security breaches and data breaches.

3. **Localization Testing:**

Translation Accuracy:

- Action: Review translated text for accuracy and consistency across different languages supported by the game. (DeepL Translate)
- Expected Result: Translations accurately convey the intended meaning without grammatical errors or mistranslations.

Cultural Appropriateness:

- Action: Assess game content and imagery for cultural sensitivity and appropriateness across diverse user demographics.
- Expected Result: Game content is culturally sensitive and respectful, avoiding stereotypes or offensive imagery.

Text Display Validation:

- Action: Verify that localized text fits within designated UI elements without truncation or overflow.
- Expected Result: Text is displayed correctly within UI elements, ensuring readability and aesthetic appeal.

Date, Time, and Currency Formats:

- Action: Change locale settings to different regions and verify the correct display of date, time, and currency formats.
- Expected Result: Date, time, and currency formats adjust according to locale settings, ensuring consistency with user preferences.

Terminology Consistency:

- Action: Review the consistency of terminology and language usage throughout the game for each supported language.
- Expected Result: Consistent terminology and language usage enhance user understanding and readability across different languages.

4. **Accessibility Testing:**

Screen Reader Compatibility:

- Action: Test the game interface using screen reader software to ensure compatibility and accessibility.
- Expected Result: Screen reader software accurately reads aloud all important game elements, providing accessibility for visually impaired users.

Keyboard Navigation:

- Action: Navigate through the game interface using only keyboard controls (tab, arrow keys).
- Expected Result: All game features and functionalities are accessible and operable via keyboard navigation, ensuring accessibility for users with motor disabilities.

Colour Contrast Ratios:

- Action: Evaluate colour contrast ratios to ensure readability for users with visual impairments.
- Expected Result: Text and graphical elements exhibit sufficient colour contrast ratios, facilitating readability and usability for users with low vision.

Alternative Text for Non-Textual Elements:

- Action: Verify the availability of alternative text descriptions for non-textual elements such as images and icons.
- Expected Result: Alternative text descriptions provide meaningful information for users who rely on screen readers or have images disabled.

5. **Regression Testing:**

Bug Retesting:

- Action: Retest previously identified bugs to ensure they have been effectively resolved in the current version of the game.
- Expected Result: Previously identified bugs are successfully resolved, with no recurrence of issues in the current version.

Regression Check for New Issues:

- Action: Verify that fixes for previous bugs have not introduced new issues or regressions in the game functionality.
- Expected Result: Fixes for previous bugs do not cause unintended side effects or introduce new issues, maintaining the overall stability and functionality of the game.

Integration Testing:

- Action: Test integrations with other platform features (user authentication, payment processing) to ensure compatibility and stability.

- Expected Result: Integration with other platform features is seamless, with no disruptions or conflicts affecting game functionality.

Performance Impact Assessment:

- Action: Validate that performance improvements have not negatively impacted other aspects of the game, such as stability or user experience.
- Expected Result: Performance improvements enhance overall gameplay experience without compromising stability or introducing new issues.

Cross-Environment Validation:

- Action: Verify the correct functioning of critical features across different environments and configurations (browsers, devices).
- Expected Result: Critical features perform consistently across different environments, ensuring a seamless experience for users regardless of their setup.

Reasons why the above are manual tests:

Manual testing requires subjective evaluation and human judgement, especially when assessing usability, security, and accessibility. For a thorough analysis of complex scenarios such as security vulnerabilities and cultural sensitivities, human expertise is required. Manual regression testing ensures that fixes are thoroughly validated and that no new issues are introduced due to human verification.

Automated Test Cases:

1. **Roulette Wheel Spin:**
    - **Description:** Automated test to simulate the spinning of the roulette wheel and validate its behaviour.
    - **Actions:**
        - Open the roulette game interface.
        - Initiate the spin action programmatically.
        - Capture the outcome of the wheel spin.
    - **Expected Result:** The roulette wheel spins smoothly and stops at a random position, with the outcome accurately captured for further validation.
2. **Bet Placement:**
    - **Description:** Automated test to systematically place bets on different bet types and amounts.
    - **Actions:**
        - Navigate to the bet placement section of the game.
        - Programmatically place bets on various bet types (colour, number) with different amounts.
        - Verify that bets are successfully placed and reflected in the game interface.
    - **Expected Result:** Bets are accurately placed according to specified parameters, and the game interface displays the placed bets correctly.
3. **Winning Bets and Payout Calculation:**
    - **Description:** Automated test to validate winning bets and calculate payouts accurately.
    - **Actions:**
        - Place bets on winning outcomes programmatically.
        - Simulate the roulette wheel spin to determine the winning number/colour.
        - Verify that winning bets are correctly identified and payouts are calculated accurately.
    - **Expected Result:** Winning bets are accurately identified, and payouts are calculated correctly based on the game outcome.
4. **Stability under Load:**
    - **Description:** Automated test to simulate multiple concurrent users placing bets to assess game stability.
    - **Actions:**

        - Simulate multiple virtual users accessing the game simultaneously.
        - Programmatically perform actions such as placing bets and spinning the wheel for each user.
        - Monitor for any performance degradation or system crashes.
    - **Expected Result:** The game remains stable under load conditions, with no performance degradation or system crashes observed.

5. **Consistency of Outcomes:**
    - **Description:** Automated test to verify the consistency of game outcomes across multiple sessions.
    - **Actions:**
        - Programmatically initiate multiple game sessions with the same initial conditions.
        - Capture and compare game outcomes (winning numbers/colours) across sessions.
    - **Expected Result:** Game outcomes remain consistent across multiple sessions, ensuring reliability and fairness.
6. **Response Time:**
    - **Description:** Automated test to measure the response time for various game actions.
    - **Actions:**
        - Programmatically initiate actions such as placing bets and spinning the wheel.
        - Measure the time taken for each action to complete.
    - **Expected Result:** Response times for game actions meet specific criteria ensuring smooth gameplay experience.

Reasons why the above are manual tests:

Automation ensures that test cases are executed consistently and repeatedly, reducing the likelihood of human error. Automated tests can be run more efficiently, saving time and effort over manual testing. Automating critical functionalities like bet placement and outcome validation ensures thorough coverage and accuracy. Automation makes it easier to test under various conditions (load, response time), allowing for a more comprehensive performance assessment.

Assumptions and Decisions:

Assumption 1: The test environment accurately simulates the production environment of csgoempire.com.

Assumption 2: The automated test scripts cover a sufficient number of test scenarios to validate the roulette game's functionality.

Decision 1: The choice of Selenium WebDriver with Python aligns with the test objectives for many reasons. Selenium WebDriver offers efficient automation for web testing tasks, like interacting with game elements and checking if the game functions correctly. Python's simplicity and clear writing style make it easy for our testing team to understand and maintain the test scripts. Selenium and Python are both cross-platform, so our tests can run smoothly on different operating systems without needing changes. There is a supportive community for both Python and Selenium, providing helpful documentation, tutorials, and resources for creating tests and solving any issues. With Selenium and Python working together, tests will run reliably, giving accurate and valuable results.

Decision 2: Manual testing will focus on exploratory testing in addition to executing predefined test cases to uncover any unforeseen issues.

If I had more time, I would focus on a few key areas to enhance the testing process:

1. I would adopt exploratory testing. I would spend some time playing around with the game to uncover any unexpected issues or problems that might not have been considered in the initial test plan.
2. I would implement an iterative testing approach. This means testing in cycles, refining and improving the test plan based on feedback and changing requirements throughout the development process. This allows for continuous enhancement of the testing process to ensure thorough evaluation and quality assurance of the game.