**Traceability and Testing**

Timothy Beers

College of Science, Engineering, & Technology, Grand Canyon University

CST-452: Senior Project II

Professor Michael Landreth

9 August 2025

**Traceability and Testing**

**Implementation Plan**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| User Story ID | Use Case or User Story | List of Detailed Development Tasks | Estimate (hrs.) | Actual (hrs.) | Percent Complete |
| US-1 | As a user, I want to launch the app and view a home screen with navigation options so that I can select a game mode. | 1. Define a SwiftUI HomeView 2. Implement NavigationStack with destinations | 1 | 2 | 100% |
| US-2 | As a user, I want to select the Alien Invasion mode from the game select screen so that I can start the typing challenge. | 1. Define a GameSelectView  2. Add a button with a destination for the GameView | 1 | 1 | 100% |
| US-3 | As a user, I want to see a QWERTY-styled emoji keyboard that I can resize for better usability. | 1. Define a KeyboardView  2. Implement an ObservableObject to track key strokes and scale, publishing any changes so that Views are redrawn accordingly | 2 | 2.5 | 100% |
| US-4 | As a user, I want my key taps to be captured accurately by the game scene and reflected in animations to know if I’ve typed correctly or not. | 1. Hook up the Keyboard’s ObservableObject to the GameView  2. Use a callback from the game scene to interpret any key presses | 2 | 3 | 100% |
| US-5 | As a user, I want the game to adjust its speed and timing based on my input accuracy and response times so that the challenge remains engaging. | 1. Define difficulty thresholds in the game scene  2. Increase or decrease difficulty based on time intervals between correct guesses | 3 | 3 | 100% |
| US-6 | As a user, I want the system to record my typing performance (ex., time-to-type per emoji) and display this data on a stats page so I can monitor my progress. | 1. Create and populate a RoundStats model with user statistics as the game progresses  2. Define a callback for scene to pass RoundStats back to the View  3. Define a StatsRepository in the environment and append RoundStats when rounds are completed | 2 | 1.5 | 100% |
| US-7 | As a user, I want to access a settings page to modify game parameters such as keyboard size and difficulty levels so that the experience can be tailored to my needs | 1. Define a SettingsView with buttons for keyboard sizing and difficulty adjustment  2. Hook up difficulty setting to a UserStats object in the environment, to be read by the GameView and passed to the scene  3. Hook up keyboard sizing to the KeyboardObserver object so that scale is reflected across the app | 3 | 2 | 100% |
| US-8 | As a user, I want the application to process my key inputs with minimal latency (e.g., within 200ms) so that gameplay feels responsive and fluid. | 1. Use SwiftUI’s built in State management to facilitate fast updates and response times | 0 | 0 | 100% |
| US-9 | As a user, I want the interface to adhere to accessibility standards, including standard levels of contrast and scalable emoji sizes, so that it accommodates users with visual or motor impairments. | 1. Use color contrast tools in design phase to determine the Color catalog to be used by the app  2. Use built in SwiftUI components for Text to automatically support resizing | 0 | 0 | 100% |
| US-10 | As a user, I want the system to be highly reliable and stable, with minimal crashes or disruptions, ensuring continuous and uninterrupted gameplay. | 1. Test the app in debug mode so that any inefficiencies are automatically reported through the console or breakpoints.  2. Monitor device statistics in debug mode (CPU, RAM, etc.), ensuring none of them near the maximum allowed | 1 | 0.5 | 100% |
| US-11 | As a user, I want the application to securely handle any performance data it records, ensuring that my progress and personal data are protected from unauthorized access. | 1. Keep all persistent data on device  2. Use iCloud’s built in security for cross-device persistent storage if needed | 1.5 | 2 | 100% |
| US-12 | As a developer, I want the application to be modular and scalable so that future game modes and features can be added with minimal rework. | 1. Use POP principles for the game scene and any reusable models | 2 | 2 | 100% |

**Functional Requirements Mapping**

|  |  |  |  |
| --- | --- | --- | --- |
| **User Story ID** | **Architecture Plan**  Page or section where designed | **Code Module** Class and Method where implemented | **Test Case ID** |
| US-1 | Page 11 - UI Diagram | HomeView.body | TC-1 |
| US-2 | Page 11 - UI Diagram | GameSelectView.body | TC-1 |
| US-3 | Page 11 - UI Diagram and Page 5 - System Modules | KeyboardView.body KeyboardSizingView.body | TC-3 |
| US-4 | Page 5 - System Modules | GameView.body | TC-1 |
| US-5 | Page 10 - Flowchart | UFOTypingScene.update(\_:) | TC-4 |
| US-6 | Page 5 - System Modules | UFOTypingScene.endGame() StatsView.body | TC-2 |
| US-7 | Page 11 - UI Diagram and Page 5 - System Modules | SettingsView.body | TC-3 |
| US-8 | Page 5 - System Modules | KeyboardObserver View.onKeyTap(\_:) | TC-1 |
| US-9 | Page 6 – Accessibility and Usability | All structs within the Views folder | TC-3 |
| US-10 | Page 6 – Reliability and Fault Tolerance | All code | N/A |
| US-11 | Page 6 - Security | FileStatsRepository.save(\_:) UserSettings.init() | TC-5 |
| US-12 | Page 6 – Maintainability and Extensibility | TypingScene All reusable subviews | N/A |

**Test Cases**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case#** | **Test Title** | **Test Steps** | **Expected Result** | **Actual Result** | **Pass/Fail Status** | **Notes** |
| TC-1 | UFO Destruction | 1. Launch Alien Invasion mode  2. Type correct emoji for the closest UFO  3. Observe if UFO is destroyed | UFO should explode or disappear with corresponding animation | Projectile fired from bottom of screen when key is tapped. Corresponding UFO disappeared with sparks animating outwards. | Pass |  |
| TC-2 | Metrics Display | 1. Launch the Alien Invasion mode  2. Complete a round  3. Open Stats Page | Stats Page displays updated time-to-type and trouble keys. | Stats page showed completed round with a “1m ago” label. Average time to type was displayed with trouble keys. | Pass |  |
| TC-3 | Keyboard Resize Settings | 1. Go to Settings  2. Adjust keyboard size  3. Return to game and verify new layout | Keyboard layout resizes accordingly, maintaining functionality | Keyboard layout resizes and maintains functionality. | Pass | While the keyboard resized accordingly, the background behind it remained the same size, resulting in a lot of blank space within the game mode. |
| TC-4 | Difficulty Adjustment | 1. Intentionally make rapid correct inputs  2. Observe UFO speed increase  3. Make multiple errors  4. Observe UFO speed decrease | Speed dynamically adjusts up or down based on current accuracy/time-to-type metrics | After rapid correct inputs, the speed ramped up slowly. When multiple errors were made, the UFOs still approached the bottom but did slow down to a stop at a certain point. | Pass | Game was not long enough to observe how consistent the speed ramp-up remains on repeated correct attempts. |
| TC-5 | No External Connections | 1. Launch game in offline mode  2. Check gameplay, stats, and settings | All features continue to function correctly without any external dependencies | All functionality appeared the same as when testing with an internet connection enabled. | Pass |  |