Integration Testing

Last edited by **Priya Ashok Sardhara** 4 hours ago

In integration testing, our plan implements Top Down Integration. We started by testing higher-level GUI modules such as MainMenuScreen, PetSelectionScreen, and GamePlayScreen, and integrated them with underlying logic modules. Stubs have been created for modules not yet completed, especially in earlier sprints.

Test Case 1

Test Case Name	**Main Menu → Pet Selection → Gameplay Screen**
Test Case Description	Ensure user flow transitions from the Main Menu to Pet Selection and then into the main gameplay with pet data.
Test Steps	1. Launch application (Main.java). 2. On MainMenuScreen, click "Start Game". 3. On PetSelectionScreen, choose a pet and click "Start". 4. Check if GamePlayScreen launches with correct pet data.
Pre-Requisites	• JSONs for pets loaded • PetType and PetState working
Expected Results	Gameplay screen is shown with chosen pet sprite and attributes
Test Category	Integration
Requirements	3.1.2 (Main Menu), 3.1.3 (Pet Selection), 3.1.5 (Game State Management), 3.1.6 (Vital Statistics & Rules), 3.1.10 (Pet Sprite)
Automation	Manual
Date Run	March 30, 2025
Pass/Fail	pass
Test Result	Flow works; pet attributes are passed correctly
Remarks	SoundManager stub used in early versions

Test Case 2

Test Case Name	**Gameplay Actions → Save Data**
Test Case Description	Ensure that when the user performs actions like "Feed" or "Exercise", the pet's state updates correctly and is reflected in JSON or UI.
Test Steps	1. Open GamePlayScreen. 2. Click the Feed or Exercise command. 3. Check if the PetState and StatType are updated. 4. Confirm visual feedback and internal model state.
Pre-Requisites	Pet model, JSON UI, and command handlers (FeedCommand, ExerciseCommand, PlayCommand, TakeToVetCommand, etc.)
Expected Results	Pets fullness, sleepiness, gift, vet etc stat changes and reflect on UI
Test Category	Integration
Requirement	3.1.6 (Vital Statistics & Rules), 3.1.7 (Commands), 3.1.9 (Keeping Score), 3.1.10 (Pet Sprite), 3.1.12 (Housekeeping & Error Handling)
Automation	Partial (command invoked manually)
Date Run	March 30th, 2025
Pass/Fail	Pass
Test Results	State updates worked across logic and screen.
Remarks	SoundManager integration needs refinement

Test Case 3

Test case Name	**GamePlayScreen → SaveManager → SaveData JSON**
Test Case Description	Verify that exiting the game triggers save logic and writes correct data to JSON using SaveManager
Test Steps	1. Modify the pet state during gameplay. 2. Exit via MainMenuScreen or GameOverScreen 3. Check savedGameList.JSON and SavedPet.JSON contents.
Pre-Requisites	• SaveManager and SaveData classes implemented. • Save triggers wired on exit.
Expected Results	JSONs are written with the most recent game state.
Test Category	Integration
Requirement	3.1.5 (Game State Management), 3.1.9 (Keeping Score), 3.1.12 (Housekeeping & Error Handling), 3.1.13 (Extra Functional Requirement - Sound)
Automation	Manual
Date Run	March 31st 2025
Test Results	Save files were generated with the correct values
Remarks	Needs further testing for corrupted exits

Test Case 4

Test Case name	**Parental Controls → Timer Limit → Gameplay Lock**
Test Case Description	If a time limit is set in ParentalControlScreen, user should be blocked from playing once the time exceeds.
Test Steps	1. Set time limit via ParentalLimitationsScreen. 2. Launch GamePlayScreen. 3. Simulate time exceeding limit. 4. Observe system behavior.
Pre-Requisites	SettingsManager, time-tracking logic
Expected Results	Gameplay is stopped or redirected to parental warning screen.
Test Category	Integration
Requirement	3.1.11 (Parental Controls), 3.1.12 (Housekeeping & Error Handling)
Automation	Manual
Date Run	March 31st 2025
Pass/Fail	Pass
Test Results	Game transitioned as expected after time expired.
Remarks	Time simulation was done manually for this test.

Test Case 5

Test Case Name	**Tutorial Flow: Main → TutorialScreen**
Test Case Description	Ensure that tutorial java files load correctly and buttons allow navigation between screens.
Test Steps	1. Launch MainMenuScreen. 2. Click “Tutorial”. 3. Ensure TutorialScreen loads and displays the information screenshots correctly. 4. Click “Next” and confirm the flow from the first tutorial screen to the last.

Pre-Requisites	• JSON tutorial files in place. • Logic for navigation implemented.
Expected Results	Smooth transition through all tutorial screens.
Test Category	Integration
Requirement	3.1.2 (Main Menu), 3.1.4 (Instructions/ Tutorial), 3.1.12 (Housekeeping & Error Handling)
Automation	Manual
Date Run	March 31, 2025
Pass/Fail	Pass
Test Results	No missing files; progression worked as expected.
Remarks	Consider adding a skip option for experienced users.

Test Case 6

Test Case Name	**SettingsScreen → SettingsManager → JSON Persistence**
Test Case Description	Verify that changes made in the SettingsScreen (e.g., sound toggle, difficulty level, parental control toggle) are correctly passed to SettingsManager.
Test Steps	1. Launch the application. 2. Navigate to SettingsScreen from MainMenuScreen 3. Modify one or more settings (e.g., mute sound, enable parental control). 4. Click “Save” or exit the settings screen (depending on implementation).
Pre-Requisites	• SettingsScreen GUI fully functional. • SettingsManager correctly linked to JSON read/write operations. • Default setting.JSON exists or can be created.
Expected Results	• Changes in the GUI are reflected immediately in the application's behaviour. • The corresponding fields in setting.JSON are updated accordingly (e.g., "soundEnabled":false).
Test Category	Integration Test
Requirement	3.1.2 (Main Menu), 3.1.8 (Settings), 3.1.12 (Housekeeping & Error Handling)
Automation	Manual
Date Run	March 31, 2025
Pass/Fail	Pass
Test Results	Modified settings correctly updated in settings.JSON and were effective in-game.
Remarks	Will add validation for corrupted settings files in future tests.