Verification and Validation for 4ZP6: DieSpy

Team #9

Jackson Cassidy
Wyatt Habinski
Christian Majid
Paul Puscas

February 7, 2025

Revision History

Date	Version	Notes
February 7, 2025	0	
April 4,, 2025	1	added test results to components

Project Description

DieSpy is a machine learning-based mobile application that detects and analyzes dice rolls in real time using a mobile device’s camera. Its primary goal is to automate dice recognition for tabletop gaming, eliminating manual counting while enhancing user experience with group collaboration features. The system includes a camera module for capturing frames, an ML-based dice detection engine, and a statistical manager for tracking roll outcomes. Additionally, it leverages network connectivity agents and party managers to support multi-user interaction in a collaborative environment

SRS Document Link   Design Document Link

Component Test Plan

As most screen components have very similar ui performance metrics and tests, to save space, we will refer to the following as **Default UI Tests**:

- **Screen load time**: time to render screen
- **UI Responsiveness**: frame rate, smooth animations
- **Button responsiveness**: button click delay

Component	**Login Screen**
Unit Tests	- **Input response:** All screen inputs (Buttons, TextFields, etc) respond correctly
Performance Test/Metrics	- **Default UI tests** - **Input responsiveness**: keyboard delay
Results	- All input fields responded correctly during testing. No crashes or freezes observed. - Button presses are registered without delay. - Keyboard showed/hid as expected on focus/blur. - Measured keyboard response time < 100ms — within acceptable thresholds for mobile UX.

Component	**Authentication Manager**
Unit Tests	- **Authentication handling**: Valid and invalid login attempts - **Account creation**: Valid registration, username uniqueness - **Input validation**: special character, case sensitivity, empty fields, - **UI navigation**: redirects to correct screen - **Error Handling**: displays correct error message - **Persistence Handling**: Credentials successfully stored / saved
Performance Test/Metrics	- **Login Authentication**: Overarching test to determine if valid logins will be accepted, and invalid ones rejected with no side effects.

	- **Input sanitization**: Check that every possible utf-character will be coded properly and supported - **UI navigation**: Any invalid authentication will be kept at the login screen, any valid authentication will advance to the Party Action screen
Results	- Valid login attempts redirected correctly to the Home screen. - Invalid credentials displayed appropriate error messages with no crashes. - Input validation handled edge cases including special characters, empty fields, and case sensitivity. - Credentials persisted successfully and were retrieved on relaunch. - Login performance remained stable across all tested conditions, with no noticeable delays or side effects. - Input sanitization handled all tested UTF characters without breaking UI or backend logic.

Component	**Firebase Manager**
Unit Tests	**- Data Retrieval:** Confirmed successful fetch of documents using valid IDs. **- Data Creation:** Verified creation of new documents with correct data mapping. **- Data Update:** Ensured fields were updated without overwriting unrelated fields. **- Data Deletion:** Confirmed that deleted documents were removed permanently. **- Query Handling:** Validated that documents were correctly returned based on field value filters **- Error Handling:** Tested non-existent document access and verified safe exception handling.
Performance Test/Metrics	**- Latency Check:** Measured average round-trip time for read/write operations (≤ 150ms). **- Concurrency Handling:** Simulated simultaneous reads/writes with no data corruption.
Results	- Queries returned expected results with minimal latency. - Data integrity was maintained under all tested conditions. - Error handling prevented app crashes during network failures or invalid queries. - Components supported real-time updates

Component	**Cache Manager**
Unit Tests	**- Data Storage:** Verified that userIds, usernames, and turnIndex values are stored correctly. **- Data Retrieval:** Confirmed that cached data can be accessed accurately across app components. **- Reset Behavior:** Tested proper clearing of cached data when user leaves or switches parties. **- Edge Cases:** Ensured behavior is consistent when no data is cached or cache is

	accessed before initialization.
Performance Test/Metrics	**- Access Time:** Measured instant access to cached values **- Consistency:** Compare cache against Firestore data to confirm synchronization accuracy.
Results	- Successfully reduced redundant Firestore calls - Maintained consistent data for the current session - Cleared and reset cleanly with no stale data carried over between parties or sessions

Component	**Dice Simulation Manager**
Unit Tests	**- Roll Accuracy:** Confirmed that dice values are randomized correctly and fall between 1–6 for each die. **- Dice Count Handling:** Tested edge cases like 0 dice, single dice, and maximum supported dice count. **- User Interaction:** Verified correct updates when the simulate button is pressed and results are displayed. **- UI State Sync:** Ensured simulated rolls are shown in the log interface and reflect accurate data.
Performance Test/Metrics	**- Roll Generation Speed:** All simulated rolls generated and displayed in reasonable time **- Concurrency Handling:** Simultaneous simulate actions prevented through UI lockout. No lag or duplication observed. **- Result Display Latency:** Minimal delay between user input and result rendering
Results	- Simulated rolls are randomized and reflect proper dice logic. - Works seamlessly with the log system and UI without any noticeable delay or bugs

Component	**Profile Screen**
Unit Tests	- **Displays Correct User Information:** Verify that user information retrieved from the profilemanager is accurate - **Input response:** All screen inputs (Buttons, TextFields, etc) respond correctly
Performance Test/Metrics	- **Default UI tests**
Results	All UI elements functioned as intended 100% of the time

Component	Settings Screen
Unit Tests	- **Real Time Changes:** Settings changed are reflected immediately - **Input response:** All screen inputs (Buttons, TextFields, etc) respond correctly
Performance Test/Metrics	- **Default UI tests** - **Input responsiveness**: keyboard delay
Results	All UI elements functioned as intended 100% of the time

Component	Profile Manager
Unit Tests	- **Profile Data Retrieval:** Ensure user game data is fetched - **Profile Update:** Validate new data appears on profile immediately - **Error Handling:** Simulate failures like network errors or database unavailability.
Performance Test/Metrics	- **Profile Load Time:** Time taken to fetch user data: < 2 seconds. - **Database Query Efficiency:** Queries are optimized for fetching/updating data **Profile Update Time:** User profile updated before they can open it
Results	- **Profile Data Retrieval:** User data correctly fetched 100% of the time - **Profile Update:** User data updated on the database within 1 second - **Error Handling:** All errors are correctly logged with proper fallbacks.

Component	Home Screen
Unit Tests	- **Input Response:** All screen inputs (Buttons, TextFields, etc) respond correctly
Performance Test/Metrics	- **Default UI tests**
Results	All UI elements functioned as intended 100% of the time

Component	Create Party Screen
Unit Tests	- **Input Response:** All screen inputs (Buttons, TextFields, etc) respond correctly
Performance Test/Metrics	- **Default UI tests** - **Input responsiveness**: keyboard delay
Results	All UI elements functioned as intended 100% of the time

Component	Join Party Screen
Unit Tests	- **Input Response:** All screen inputs (Buttons, TextFields, etc) respond correctly
Performance	- **Default UI tests**

Test/Metrics	- **Input responsiveness**: keyboard delay
Results	All UI elements functioned as intended 100% of the time

Component	**Party Screen**
Unit Tests	- **Parties Updates:** Refreshing screen as new parties update - **Invalid Party Code:** Error for users trying to join with an incorrect code - **Correctly Displays Data:** Display correct number of members in parties in real time - **Input response:** All screen inputs (Buttons, TextFields, etc) respond correctly
Performance Test/Metrics	- **Data Update Time:** Must be less than 100ms for accurate info - **Default UI tests**
Results	All UI elements functioned as intended 100% of the time

Component	**Member Screen**
Unit Tests	- **Correctly Displays Data:** Display correct the provided information - **Input response:** All screen inputs (Buttons, TextFields, etc) respond correctly
Performance Test/Metrics	- **Default UI tests** - **Input responsiveness**: keyboard delay
Results	All UI elements functioned as intended 100% of the time

Component	**Party Manager**
Unit Tests	**- Turn Management:** Verified that turn cycling works correctly and updates persist across sessions. **- Data Synchronization:** Ensured that party-related data remain in sync between users **- Real-Time Updates:** Validated that all changes to the party state are immediately reflected in the UI. **- Error Handling:** Confirmed proper behavior when party data is missing or incorrect, with no app crashes.
Performance Test/Metrics	**- Update Latency:** Measured responsiveness of real-time updates, ensuring changes are reflected within milliseconds. **- Multi-User Consistency:** Simulated concurrent user interactions to confirm consistent behavior across devices. **- Stability Under Load:** Tested under rapid changes to party state to evaluate robustness and performance.
Results	- Real-time syncing of party state was consistent and reliable

	- Turn updates and member changes reflected accurately across all devices - No crashes or major performance issues were observed during stress testing

Component	**Network Manager**
Unit Tests	- **Recieve Party Name:** Can read BLE messages, parse by correct UUID, and discard duplicate messages - **Broadcast Party Name:** Can advertise the party name on the correct UUID -**Stop Broadcasting:** Can stop broadcasting the BLE message
Performance Test/Metrics	- **Receive Party Name:** Correctly distinguish duplicate and unique parties (2 players in one party, another player in another party) - **Broadcast Party Name**: Advertise a signal that is detectable by an OTS bluetooth detector such as LightBlue - **Uptime:** Can have 99% uptime - **Parsing:** Can correctly parse a JSON message by header, and catch incorrectly formatted messages.
Results	- **Recieve Party Name:** Could properly receive and distinguish 2 duplicate and one unique parties - **Broadcast Party Name:** Correctly advertised party name with observed 100% uptime -**Stop Broadcasting:** Correctly stopped broadcasting when needed 100% of the time.

Component	**Chat Screen**
Unit Tests	- **Typing Messages** : Can interact with the screen to type a message, and send it (to the manager). - **Displaying Messages** : When passed a message from the manager, properly renders and displays it on screen
Performance Test/Metrics	- **Default UI tests** - **Input responsiveness**: keyboard delay
Results	All UI elements functioned as intended 100% of the time

Component	**Chat Manager**
Unit Tests	- **Properly Syncing Messages:** Ensure that in a conversation with > 3 users, all messages are consistent. - **Loading chats from previous sessions:** Ensure that upon restarting a lobby, previous chats are loaded - **Saving Chats:** Ensure that chats are regularly saved and backed up every 30 seconds, so that they save upon closing

Performance Test/Metrics	- **Chat Write Time:** <10ms per entry - **Chat Retrieval Speed:** Fetching chat logs should be quick, rendering as scrolling if need <50ms - **Storage Efficiency:** Ensure chat histories do not consume excessive storage - **Time Syncing**: Ensure that all messages are sent within <10ms of each other, and have the same timestamp.
Results	

Component	**Logs Screen**
Unit Tests	- **Logs Updates:** Refreshing screen as new logs are added - **Correctly Displays Data:** Display correct logs in correct order in real time
Performance Test/Metrics	- **Default UI tests**
Results	All UI elements functioned as intended 100% of the time

Component	**Logs Manager**
Unit Tests	**- Retaining Logs:** Retains last 100 logs - **Log Deletion:** App removes all traces of logs from phone to reclaim space - **Error Handling:** Error handling of failed logs writes and corrects them - **Storage Limit:** Will prompt user when logs are close to full, and stop when space is less than 50mb on device
Performance Test/Metrics	- **Log Write Time:** <10ms per entry - **Log Retrieval Speed:** Fetching logs should be quick, rendering as scrolling if need <50ms - **Storage Efficiency:** Ensure logs do not consume excessive storage
Results	

Component	**Dice Stats Manager**
Unit Tests	- **Calculates Sums correctly:** Given a list of dice, properly aggregates the sum of each dice
Performance Test/Metrics	- **Accuracy:** All calculations should be mathematically correct
Results	- **Calculates Sums correctly:** Aggregated the dice properly 100% of the time

Component	**Dice Detection Screen**

Unit Tests	- **Frequent Screen Updates:** Screen updates within 3 frames of the camera focusing on the dice
Performance Test/Metrics	- **Bounding Boxes:** Drawn accurately around the dice (dependant on model accuracy) - **Default UI tests**
Results	All UI elements functioned as intended 100% of the time

Component	**Camera X**
Unit Tests	- **Camera Initialization:** Ensure the camera starts correctly. - **Analysis Use Case:** Validate image frames are processed correctly for dice detection. - **Lifecycle Handling:** Test if the camera stops/restarts correctly when the app lifecycle changes. - **High Resolution:** Camera output should be of high resolution
Performance Test/Metrics	- **Frame Processing Time: Goal:** <16ms for 60 FPS (or as fast as phone can handle) - **FPS (Frames Per Second):** Camera can processes at least 30 FPS smoothly, and syncs with overlay - **CPU & Memory Usage:** The app should have the same cpu and memory usage as the native camera - **Latency:** Camera should operate as fast as native camera - **No Motion Blurring:** Camera should not blur when moving
Results	

Component	**Dice Detection Manager**
Unit Tests	- **Frequent Updates:** Facilitates communication between Dice Detection Screen and Dice Detection Agent every 3 frames.
Performance Test/Metrics	- **Quick Communication:** The output from the Dice Detection Agent is fed to the Dice Detection Screen quickly. Minimal latency between camera view and bounding box placement.

Component	**Dice Detection Agent**
Unit Tests	- **Model Loading:** Ensure the tensorflow model loads properly - **Image Recognition:** Ensure model runs inference on input images - **Class Labels**: Ensure agent detects all classes (faces) correctly
Performance Test/Metrics	- **Bounding Box Accuracy:** - **mAP50:** Mean average precision with an intersection over union threshold of 0.5

	– Target > 0.9 - **mAP50-95:** Mean average precision with intersection over union thresholds ranging from 0.5-0.95 – Target > 0.75 - **Classification Accuracy:** - **Overall Average Accuracy:** Target > 0.95 - **Peak F1-Score (from F1-Confidence Curve):** Balance of Precision and Recall across confidence thresholds – Target > 0.9
Results	- Results were obtained using the ultralytics YOLO library using the val() function. More info about this, and more detailed testing results can be found in ml/README.md in our github repo. - **mAP50:** achieved 0.983 - **mAP50-95:** achieved 0.811 - **Overall Average Accuracy:** achieved 0.946 (target was 0.95) - Note: accuracy was calculated using the confusion matrix that can be found in the readme mentioned above - **Peak F1-Score:** achieved 0.96 We chose Accuracy and F1-Score as metrics because F1-Score balances precision and recall, precision to measure how often the model is correct when it predicts a specific class, and recall to measure how often the model detects and classifies a die correctly. We chose accuracy because we have balanced classes, false positives and false negatives are of equal importance in this task, and overall "correctness" is a good representation of the performance goal of our model. We chose mAP50 and mAP50-95 because very precise object location is not a priority, and these balance class detection with bounding box accuracy, which suits our goals better.