**D424 – Software Engineering**

**Task 3**

|  |  |
| --- | --- |
| **Capstone Proposal Project Name:** | http://www.idevnews.com/views/images/uploads/general/wgu_logo.png  Glorious Travel Vacation Mobile App |
| **Student Name:** | Eddie Gehrmann |

**Table of Contents**

Contents

[Table of Contents 2](#_Toc189051417)

[Application Design and Testing 4](#_Toc189051418)

[Class Design 4](#_Toc189051419)

[UI Design 5](#_Toc189051421)

[Unit Test Plan 10](#_Toc189051422)

[Introduction 10](#_Toc189051423)

[Purpose 10](#_Toc189051424)

[Overview 10](#_Toc189051425)

[Test Plan 11](#_Toc189051426)

[Items 11](#_Toc189051427)

[Features 11](#_Toc189051428)

[Deliverables 12](#_Toc189051429)

[Tasks 12](#_Toc189051430)

[Needs 13](#_Toc189051431)

[Pass/Fail Criteria 13](#_Toc189051432)

[Specifications 14](#_Toc189051433)

[Procedures 15](#_Toc189051434)

[Results 15](#_Toc189051435)

[Introduction 17](#_Toc189051436)

[Installation and Using the Application 17](#_Toc189051437)

[*Login and Register* 18](#_Toc189051438)

[Logging In 18](#_Toc189051439)

[*Creating an Account* 18](#_Toc189051443)

[Pass/Fail Criteria for Key Functions 21](#_Toc189051450)

[Login Functionality 21](#_Toc189051451)

# Application Design and Testing

## Class Design

The **Vacation Planner Mobile App** is structured around the core classes that define its primary functionalities. The application uses **Java with Android Studio** and **Room Database (SQLite)** for local storage.

The key entities in this system include:

* **Vacation**: Represents a vacation with attributes such as title, hotel, start date, and end date.
* **Excursions: Represent planned activities during vacations**. Each excursion is associated with a vacation.
* **User**: Handles authentication, registration, and authorization for accessing the application.
* **Repository**: Acts as an intermediary between the UI and the Room Database, ensuring data operations are performed asynchronously.

## Output image

## UI Design

The **Vacation Planner App** features a user-friendly interface with a simple and intuitive design. It allows users to **plan vacations, add excursions, generate reports, and receive reminders**.

**Screens & Features**

1. **Login & Registration Screen**: Users can create an account and log in securely.
2. **Main Dashboard**: Displays a search bar and a list of vacations.
3. **Vacation Details Screen**: Allows users to edit vacation details, add excursions, and set alerts.
4. **Excursion Management**: Users can add, edit, or delete excursions within a vacation.
5. **Upcoming Vacations Report**: Generates a report of vacations that are coming up.
6. **Notifications & Alerts**: Users receive notifications for upcoming vacations.
7. **Vacation Sharing**: Allows users to share vacation details with others.

A screenshot of a login screen

Description automatically generatedA screenshot of a login screen

Description automatically generatedA screenshot of a phone

Description automatically generatedA screenshot of a phone

Description automatically generatedA screenshot of a phone

Description automatically generatedA screenshot of a phone

Description automatically generatedA screenshot of a phone

Description automatically generatedA screenshot of a phone

Description automatically generatedA screenshot of a phone

Description automatically generated

# Unit Test Plan

## Introduction

### Purpose

The purpose of this **Unit Test Plan** is to ensure the reliability, correctness, and security of the **Vacation Planner App**. By isolating and testing individual components of the system, we aim to identify defects early in the development cycle, ensuring a high-quality user experience.

The **Vacation Planner App** includes essential functionalities such as **user authentication, vacation management, excursion management, and reporting**. Each of these components must be tested to ensure correct behavior across different scenarios.

This **test plan** defines the scope, approach, resources, and activities required for effective testing. Additionally, it establishes the **pass/fail criteria**, outlines the **tools** and **frameworks** used, and provides detailed test scripts and expected results.

### Overview

This unit test plan focuses on testing **critical functionalities** of the **Vacation Planner App**, specifically:

* **User Authentication**: Ensuring users can securely log in and register.
* **Vacation Management**: Validating that users can **add, edit, and delete vacations** with correct date validation.
* **Excursion Management**: Confirming that excursions are correctly associated with vacations.
* **Report Generation**: Ensuring reports display accurate data based on stored vacations and excursions.

Each test case will validate functionality **individually** and **in combination** with related modules to ensure system-wide consistency. Errors and inconsistencies will be addressed by refining validation logic, database queries, and UI behaviors.

## Test Plan

### Items

To complete the unit tests, the following are required:

**Software & Dependencies**

* **Android Studio (Latest Version)** – For running and debugging the app.
* **Gradle (Latest Version)** – For dependency management and building the project.
* **JUnit 4** – Unit testing framework for Java-based Android applications.
* **Mockito** – Mocking framework for simulating database interactions.
* **AndroidX Testing Libraries** – Required for UI and unit testing.

**Database & Data Requirements**

* **Pre-filled SQLite Database** containing sample vacations and excursions.
* **Test user accounts** to simulate different authentication scenarios.
* **Dummy reports** generated based on stored vacations and excursions.

### Features

**User Authentication**

* Valid login and logout functionality
* Handling incorrect login attempts
* Ensuring password hashing and security

**Vacation Management**

* Adding new vacations with correct validation
* Editing vacation details
* Preventing invalid date selections (e.g., end date before start date)
* Deleting vacations (only if no associated excursions exist)

**Excursion Management**

* Associating excursions with vacations
* Ensuring valid excursion dates
* Editing and deleting excursions

### Deliverables

* **Test Scripts**: Automated unit test scripts in **JUnit** for backend logic validation.
* **Test Results Report**: Detailed report of test execution, including pass/fail status.
* **Bug Reports**: Any issues identified will be logged with steps for reproduction and resolution.
* **Test Data**: Sample data for user authentication, vacations, and excursions.

### Tasks

1. **Set up a controlled test environment** with a fresh SQLite database.
2. **Implement mock objects** for data access using Mockito.
3. **Write unit test scripts** for critical features and expected user interactions.
4. **Execute unit tests** and log results.
5. **Analyze failures** and update code accordingly.
6. **Re-run tests** to confirm successful fixes.

### Needs

* **JUnit & Mockito** integrated into the Android Studio project.
* **A SQLite database instance** for testing database transactions.
* **Simulated user input** to replicate real-world interactions.

### Pass/Fail Criteria

**Vacation Retrieval Test (testGetAllVacations)**

* **Pass Criteria:**
  + The repository successfully retrieves a **list of vacations** from the database.
  + The returned list contains the expected **two vacation records**.
  + The retrieved vacations match the expected **title, hotel, and dates**.
* **Fail Criteria:**
  + The repository returns an **empty list** instead of expected data.
  + The retrieved vacation **titles, hotels, or dates** do not match the expected values.

**User Authentication Test (testAuthenticateUser)**

* **Pass Criteria:**
  + The repository successfully authenticates a **valid user**.
  + The retrieved user object contains the correct **username**.
* **Fail Criteria:**
  + Authentication fails for a **valid user**.
  + The repository returns a **null user** for valid credentials.
  + Authentication fails for **invalid credentials** (security risk).

## Specifications

The following **JUnit & Mockito** scripts test core functionalities:

Test: User Login with Valid Credentials

A screenshot of a computer program

Description automatically generated

Test: Prevent Vacation Deletion if Excursions Exist

A screenshot of a computer code

Description automatically generated

## Procedures

1. **Prepare test environment** – Install dependencies and set up database.
2. **Run individual test scripts** – Execute unit tests in Android Studio.
3. **Log results** – Document pass/fail status.
4. **Debug and refine code** – Fix validation errors, refine queries, improve UI messaging.
5. **Re-run tests** – Ensure resolved issues do not reoccur.

## Results

**Vacation Retrieval Test (testGetAllVacations)**

* ✅ **Passed**:
  + Successfully retrieved **two vacation records** from the mock database.
  + Data returned matched the expected values:
    - **Vacation 1**: Title - "Orlando", Hotel - "Mariott", Start Date - "2025-05-01"
    - **Vacation 2**: Title - "Paris", Hotel - "Hilton", Start Date - "2025-06-10"
* ❌ **Failed**: (Initial Attempt)
  + **Issue:** getmAllVacations() originally returned an **empty list** instead of the expected two vacations.
  + **Fix Implemented:** Ensured that mockVacationDAO.getAllVacations() was correctly mocked to return the expected data.

**User Authentication Test (testAuthenticateUser)**

* ✅ **Passed**:
  + User successfully authenticated with **valid credentials** ("testUser", "hashedpassword").
  + The system returned the correct **User object** with the expected username.
* ❌ **Failed**: (Initial Attempt)
  + **Issue:** Authentication initially failed because mockUserDAO.authenticate() was not properly mocked.
  + **Fix Implemented:** Correctly configured when(mockUserDAO.authenticate(...)).thenReturn(fakeUser) in the test setup.

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

*C2. Provide a link to where the web app is hosted with HTML code (if applicable).*

*https://play.google.com/apps/internaltest/4701609292064584645*

*C3. Provide a link to the GitLab repository of the code indicating the version included in this submission.*

*https://gitlab.com/wgu-gitlab-environment/student-repos/egehrma/d424-software-engineering-capstone/-/tree/working\_branch?ref\_type=heads*

. **User Guide**

## Introduction

This guide provides step-by-step instructions on installing, logging into, and using the Vacation Planner application effectively. It details the key features, such as vacation and excursion management, and ensures all procedures are clearly defined, tested, and user-friendly.

## Installation and Using the Application

**Installation Steps**

1. **System Requirements**:
   * Android device with minimum **SDK version 24**.
   * Internet access for initial download and updates.
2. **Downloading the App**:
   * Download the APK from the designated repository or application store.
   * Ensure the downloaded APK is saved on your Android device.
3. **Install the APK**:
   * Navigate to your device’s “Downloads” folder.
   * Tap on the APK file and select **Install**.
   * If prompted, enable the “Install from Unknown Sources” option in your device settings.
4. **Launch the Application**:
   * After installation, tap the **Vacation Planner** icon on your home screen to open the app.

## *Login and Register*

## Logging In

## *Open the app to view the Login Screen.*

## *Enter your registered username and password.*

## *Tap Login to access the main application dashboard.*

## *Creating an Account*

## *On the login screen, tap the “Don’t have an account? Register” link.*

* Click “Register”

## *Enter the following details:*

## *Username: Must be unique.*

## *Password: Requires at least 6 characters.*

## *Tap Register to create your account.*

## *Return to the Login Screen to log in with your new credentials.*

**Vacation Management**

**Add a New Vacation**

1. Navigate to the **Vacation List** screen from the main dashboard.
2. Tap the **Add Vacation** button (floating action button at the bottom right).
3. Enter the following details:
   * **Vacation Title** (e.g., "Beach Getaway").
   * **Hotel Name** (e.g., "Seaside Resort").
   * **Start Date**: Tap the field to select a date.
   * **End Date**: Tap the field to select a date.
4. Tap **Save** to store the vacation in the database.

**Edit an Existing Vacation**

1. On the **Vacation List** screen, tap on a vacation entry to view its details.
2. Tap the **Edit** button at the top right of the **Vacation Details** screen.
3. Modify the vacation title, hotel name, or dates.
4. Tap **Save Changes** to update the information.

**Delete a Vacation**

1. Open the **Vacation Details** screen for the vacation you want to remove.
2. Tap the **Delete** button in the menu.
3. Confirm the deletion to permanently remove the vacation.

**Excursion Management**

**Add a New Excursion**

1. Navigate to the **Vacation Details** screen of a specific vacation.
2. Tap the **Add Excursion** button (floating action button at the bottom).
3. Enter the excursion details:
   * **Title** (e.g., "Snorkeling Adventure").
   * **Date**: Tap the field to select a date.
4. Tap **Save** to link the excursion to the vacation.

**View Excursions**

1. Open the **Vacation Details** screen.
2. Scroll down to view a list of excursions linked to the vacation.

**Reports**

**View Upcoming Vacations**

1. On the main dashboard, tap the **View Upcoming Vacations** button.
2. The app displays a report with:
   * Vacation Title
   * Hotel Name
   * Start Date
   * End Date
3. Use the search bar at the top to filter vacations by keyword.

## Pass/Fail Criteria for Key Functions

## Login Functionality

## *Pass: User logs in successfully with correct credentials and is directed to the main dashboard.*

## *Fail: Incorrect credentials display an appropriate error message.*

## Add New Vacation

## *Pass: The vacation is added successfully and appears in the Vacation List.*

## *Fail: Validation errors (e.g., missing fields or end date before start date) display appropriate error messages.*

**Conclusion**

This user guide ensures that all functionalities of the Vacation Planner app are utilized effectively. Following these instructions guarantees a seamless experience while managing vacations and excursions.