

Table of Contents**Table of Contents**

<i>Table of Contents</i>	<i>1</i>
<i>Application Design and Testing</i>	<i>3</i>
Class Design	3
UI Design	4
<i>Unit Test Plan</i>	<i>6</i>
Introduction	6
Purpose	6
Overview.....	6
Test Plan	6
Items	6
Features.....	6
Deliverables	6
Tasks.....	7
Needs	7
Pass/Fail Criteria	7
Specifications	7
Procedures	8
Results	8
<i>Hosted Web Application</i>	<i>9</i>
<i>GitLab Repository</i>	<i>9</i>

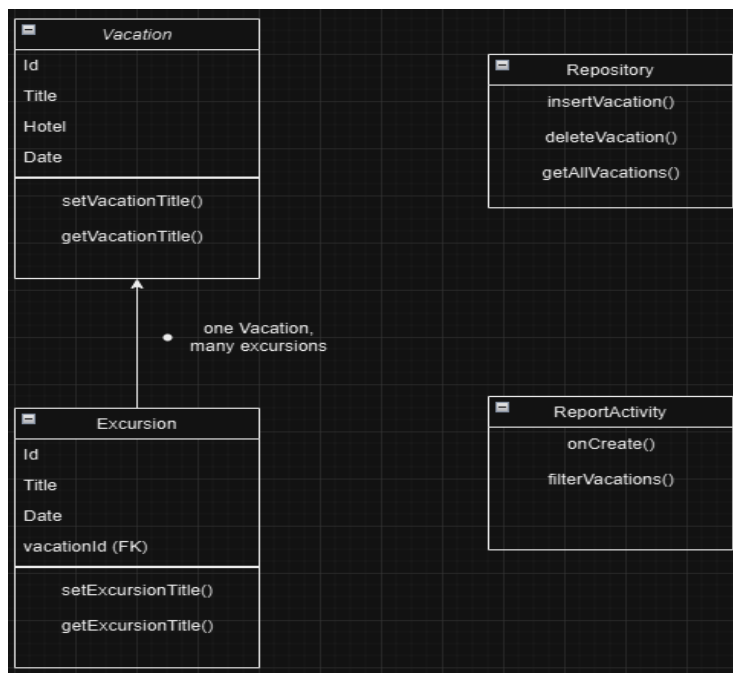
<i>User Guide</i>	9
Introduction	9
Installation and Using the Application	9
Method 1: Directly on Android Device.....	9
Method 2: Using Android Studio	10
<i>Login</i>	10
<i>Using the App</i>	12
Add a Vacation.....	12
Add Excursions	13
Generate a Report	14
Note on Sample Data, Alarms, and Sharing	14

Application Design and Testing

Class Design

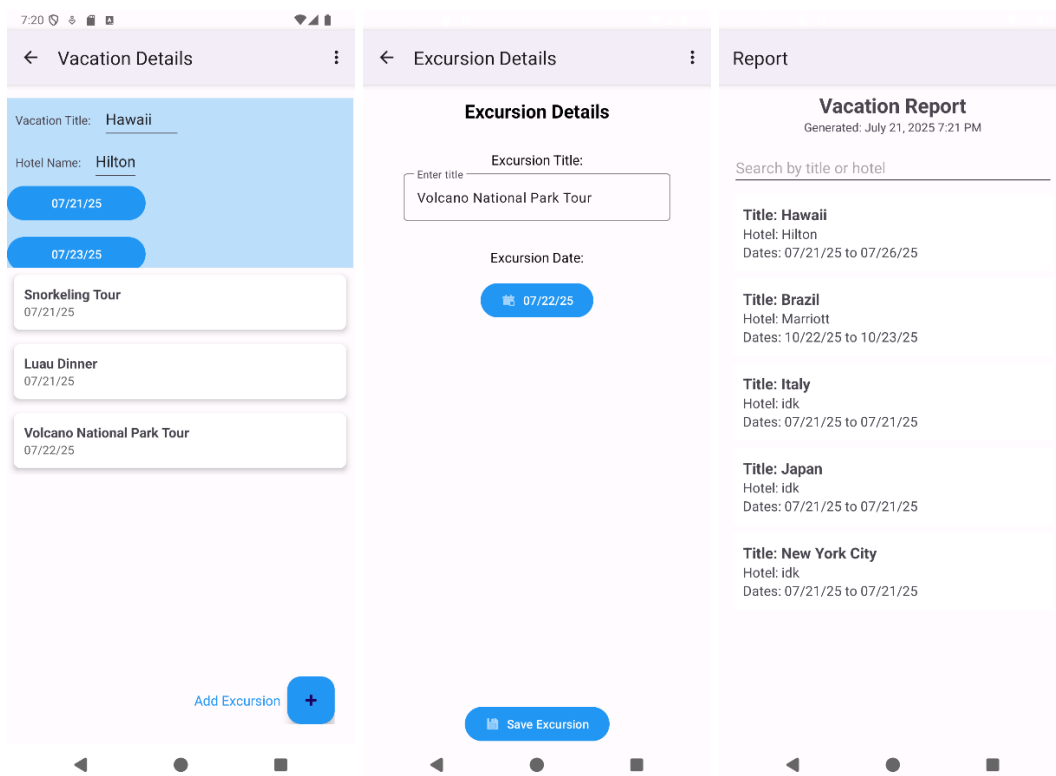
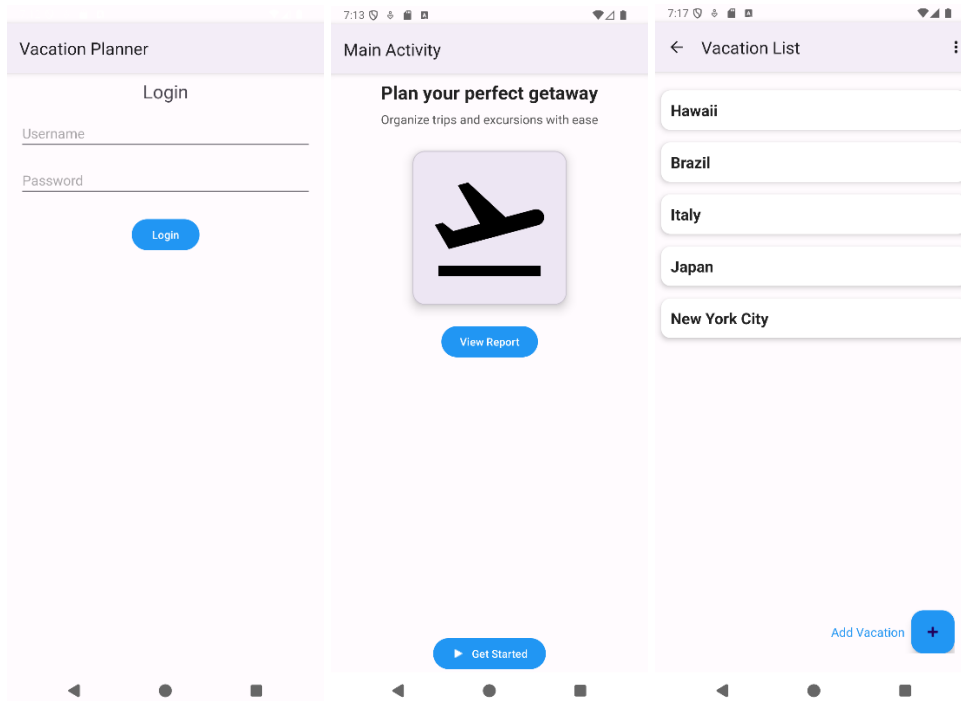
The class diagram illustrates the core structure and relationships of the Vacation Planner Android application. It highlights four primary components: Vacation, Excursion, Repository, and ReportActivity. The Vacation class represents a trip with attributes such as title, hotel, and date, and has a one-to-many relationship with Excursion, which stores individual trip activities. The Repository handles data operations including inserting, deleting, and retrieving vacations, promoting separation of concerns. ReportActivity manages the reporting screen, including filtering logic and timestamp generation.

This diagram provides a simplified yet accurate overview of the app's object-oriented design, helping visualize how classes interact while supporting scalability and maintainability.



UI Design

The user interface design of the Vacation Planner Android application focuses on clarity, simplicity, and ease of use for organizing trips and excursions. The app opens with a login screen that provides secure access. After logging in, users are taken to the home screen where vacations would normally be listed. However, the application does not include any preloaded vacations or excursions by default. Users must either manually create their own entries or use the built-in "Sample Data" feature to populate the app. The interface includes clearly labeled input fields, intuitive navigation, and responsive layouts optimized for mobile use. The screenshots provided demonstrate the application with sample data enabled, showcasing its full functionality.



Unit Test Plan

Introduction

Purpose

The unit test plan ensures the core functionality of the Vacation Planner Android app is reliable and behaves as expected. Testing focused on user authentication through the login screen to verify input validation and screen navigation. Any issues encountered during development were addressed immediately, followed by re-running the tests to confirm resolution.

Overview

Two primary login scenarios were tested:

1. Empty login fields - Verified that attempting to log in with an empty username and password triggers the correct error response.
2. Valid login credentials - Confirmed that entering correct credentials allows navigation to the main activity screen.

These tests validate one of the app's most critical functions: secure and accurate user access.

Test Plan

Items

Source code, Android Studio, JUnit4, and Espresso

Features

Login form validation and activity navigation

Deliverables

Test code, screenshots of results.

Tasks

Write and run tests, identify and fix any issues, and document the results.

Needs

Android Studio, Android Emulator, Espresso framework, predefined credentials

Pass/Fail Criteria

A test passes if the UI responds exactly as expected (e.g., "Invalid credentials" message appears or main screen loads). Any failure triggers debugging and re-tests.

Specifications

Screenshot of test code

```
1 package com.example.vacationplanner;
2
3 import androidx.test.ext.junit.rules.ActivityScenarioRule;
4 import androidx.test.ext.junit.runners.AndroidJUnit4;
5
6 import com.example.vacationplanner.UI.LoginActivity;
7
8 import org.junit.Rule;
9 import org.junit.Test;
10 import org.junit.runner.RunWith;
11
12 import static androidx.test.espresso.Espresso.*;
13 import static androidx.test.espresso.action.ViewActions.*;
14 import static androidx.test.espresso.assertion.ViewAssertions.*;
15 import static androidx.test.espresso.matcher.ViewMatchers.*;
16
17 @RunWith(AndroidJUnit4.class)
18 public class LoginActivityTest {
19
20     @Rule
21     public ActivityScenarioRule<LoginActivity> activityRule =
22         new ActivityScenarioRule<>(LoginActivity.class);
23
24     // Test 1: Login with empty fields shows error
25     @Test
26     public void testLoginWithEmptyFields_showsError() {
27         onView(withId(R.id.loginButton)).perform(click());
28         onView(withText("Invalid credentials")).check(matches(isDisplayed()));
29     }
30
31     // Test 2: Login with valid credentials navigates to main screen
32     @Test
33     public void testLoginWithValidCredentials_opensMainActivity() {
34         onView(withId(R.id.usernameField)).perform(typeText("admin"));
35         onView(withId(R.id.passwordField)).perform(typeText("password"));
36         androidx.test.espresso.Espresso.closeSoftKeyboard();
37         onView(withId(R.id.loginButton)).perform(click());
38         onView(withText("View Report")).check(matches(isDisplayed()));
39     }
40 }
```

Procedures

To complete the testing process, Android Studio was used alongside the JUnit4 and Espresso testing frameworks. The tests were written to simulate user interaction with the login screen of the Vacation Planner app. Two unit tests were developed:

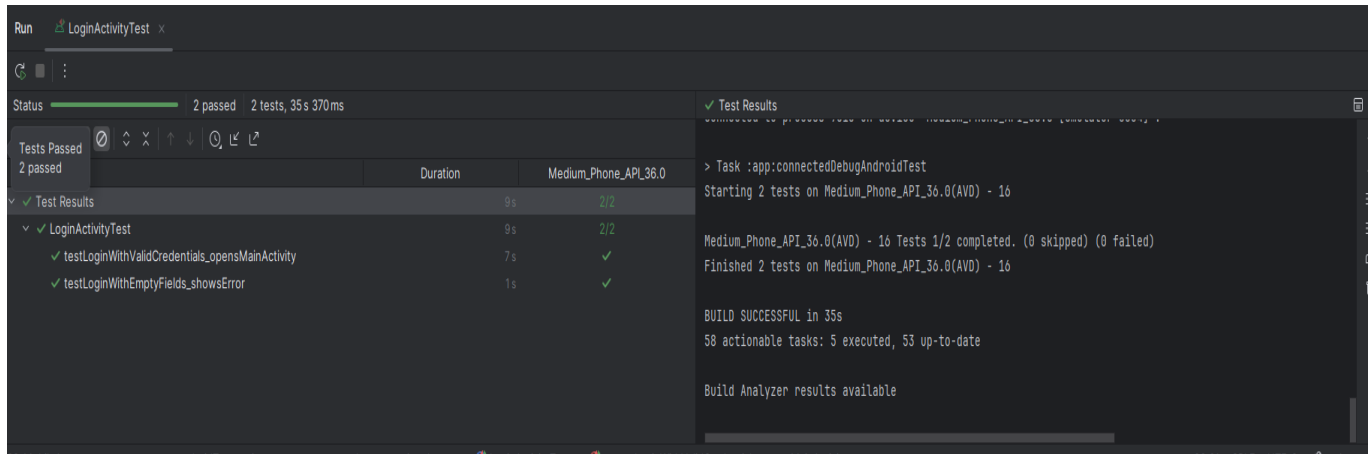
1. A test to ensure that logging in with empty fields displays the proper error message.
2. A test to confirm that entering the correct username and password navigates the user to the main screen.

Each test was run using the Android Emulator on API Level 36. When a test failed, the code was reviewed and corrected before re-running. This iterative process ensured that all functionality behaved as expected before final screenshots were captured.

Results

Both unit tests passed successfully. The first test verified that pressing the login button without filling in credentials displays a snackbar with the message "Invalid credentials." The second test checked that entering valid credentials ("admin" and "password") loads the main activity, where the "View Report" text is visible.

Below is a screenshot showing the final test results in Android Studio, confirming that both tests passed.



Hosted Web Application

Link: [Vacation Planner Android App - Deployment & Documentation](#)

GitLab Repository

Link: <https://gitlab.com/wgu-gitlab-environment/rhosk37/d424-capstone.git>

User Guide

Introduction

This guide provides clear, step-by-step instructions for installing and using the Vacation Planner Android application. It explains how to install the app via APK, log in, view and manage vacations and excursions, and generate reports. Each step has been tested to ensure a smooth experience for users with basic IT knowledge.

Installation and Using the Application

Method 1: Directly on Android Device

1. Download the APK file from the provided link on the project portfolio page.

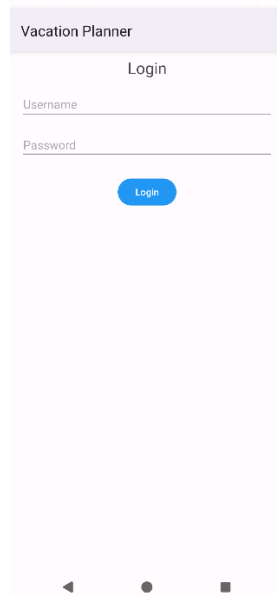
2. On your Android phone:
3. Go to Settings > Security (or Privacy).
4. Enable Install from Unknown Sources (this may vary slightly depending on device).
5. Locate the APK in your Downloads folder (or wherever saved).
6. Tap the APK file and select Install.
7. Once installed, open the app from your app drawer.

Method 2: Using Android Studio

1. Download the APK onto your PC or Mac.
2. Open Android Studio and ensure:
3. Your emulator is running, or
4. Your physical device is connected via USB and visible in the Device Manager.
5. Drag and drop the APK file onto the emulator window or device.
6. Android Studio will automatically install the app.
7. Once installed, launch the app from the device's app drawer.

Login

1. Launch the app.
2. You will see the Login Screen.

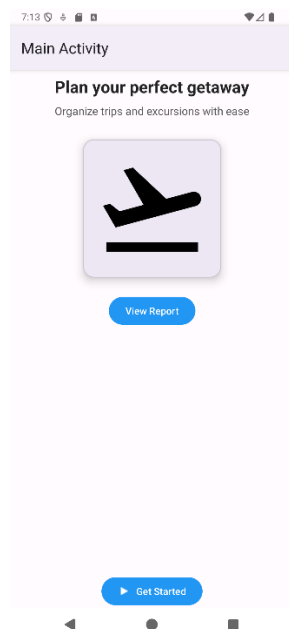


3. Enter the following credentials:

Username: admin

Password: password

4. Tap Login.
5. Upon success, you will be taken to the home screen.

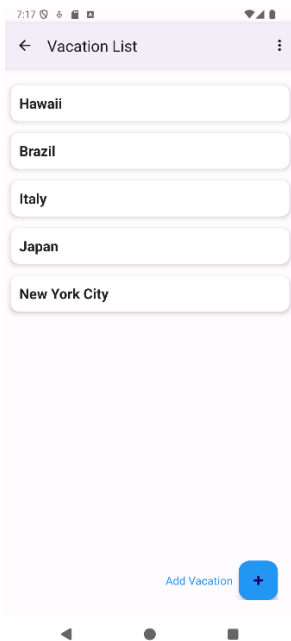


If credentials are incorrect, an “Invalid credentials” message will appear at the bottom of the screen.

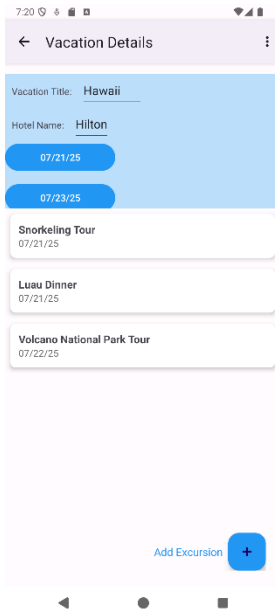
Using the App

Add a Vacation

1. From the Vacation List screen, tap the + icon to add a new vacation. Or use sample data.

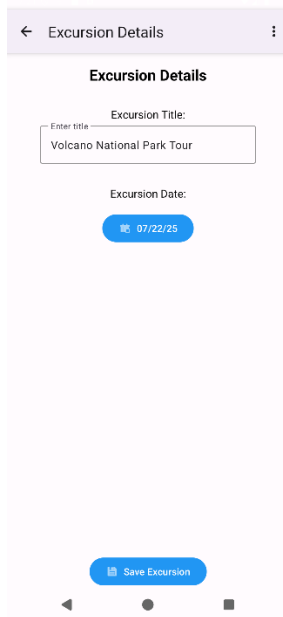


2. Fill out the fields:
 - Title
 - Hotel
 - Start and End Dates
3. Tap the three dots at the top right of the screen to save



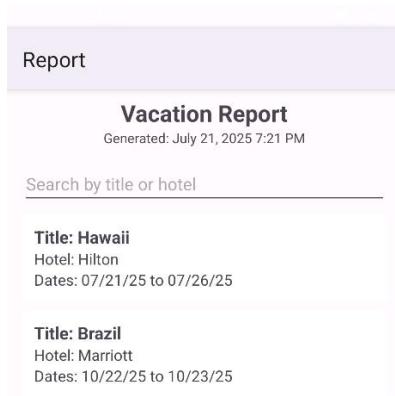
Add Excursions

1. Tap on an existing vacation from the list.
2. Tap Add Excursion.
3. Enter:
 - Title
 - Date
4. Tap the three dots at the top right of the screen to save



Generate a Report

1. On the Main Screen, click “View Report”.
2. View all vacations in a searchable list.
3. Use the search bar to filter by vacation or hotel name.
4. The current timestamp is displayed at the top for report generation time.

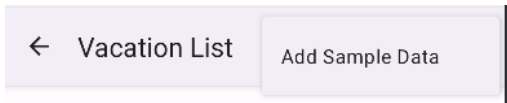


Note on Sample Data, Alarms, and Sharing

- On the first launch, the app does not contain any vacations or excursions.

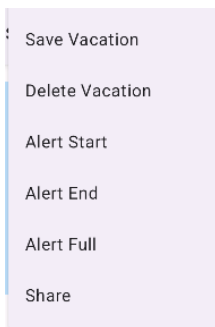
You can either manually add them or use the built-in “Sample Data” option to pre-populate test entries for demonstration.

Tap the three-dot menu in the top right of the Vacation List screen and select “Sample Data.”



- You can also set alarms to remind yourself when a vacation starts or ends.

To do this, tap the three-dot menu on the Vacation List screen and select either “Alert Start”, “Alert End”, or “Alert Full”.



- To share a vacation, open a specific vacation from the list and tap the three-dot menu on the top right.

Select “Share” to send the vacation summary via your device’s built-in share sheet (e.g., to copy, text, email, etc.).

The shared content includes the vacation title, hotel name, dates, and a list of excursions with dates.

