Harmony

Quality Assurance Document

Ali Doğan Çamur 231101017

Savaş Solak 231101025

Ege Azca 231101057

Yankı Yağız Ozan 231101059

1. **Quality Assurance Strategy:**
    - **Overview:** Our approach to quality assurance makes sure that the software meets the functional requirements by having a mix of manual and automated testing. Tests are integrated into the software development lifecycle to make sure problems can be identified and fixed as soon as possible.
    - **Testing Methodologies:** We will use the following testing techniques:
        - **Unit Testing:** The developers will test individual functions.
        - **Integration Testing:** The developers will test to make sure different modules work together.
        - **Usability Testing:** Software will be used by real users.
        - **Functional Testing:** Test the functional requirements of the software to ensure they are met.
        - **Non-functional Testing:** Test the non-functional requirements such as performance, security and usability to ensure they are met.
    - **Manual vs. Automated Testing:**
        - **Automated Testing:** Unit Testing, Integration Testing
        - **Manual Testing:** Usability Testing, Functional/Non-functional Testing

2. **Quality Factors & Metrics:**
    - **Accessibility:**
        - Description: Linking accounts
    - **Ease of Use:**
        - Description: Interface and features shouldn't be too complex and should be easy to use.
    - **Performance:**
        - Description: Project, app and/or servers need to be responsive enough.
    - **Maintainability:**
        - Description: Codes need to be written clearly and with necessary documentation in order for it to keep it maintainable.
3. **Test Plan:**

    a. Google Sign-In Functionality

    Test Scenario: Verify that users can sign in using Google authentication.
    Test Steps:
    1. Launch the app and navigate to the login screen.
    2. Click the "Sign in with Google" button.
    3. Select a Google account for authentication.
    4. Verify that the user is redirected to the main activity (map screen)

    Expected Result: The user is successfully authenticated and navigated to the main screen.

b. Location Permission Handling

Test Scenario: Verify that location permissions are requested and handled properly.
Test Steps:

1. Install and launch the app for the first time.
2. Observe the permission request prompt.
3. Select "Allow" and check if the map centers on the user's location.
4. Relaunch the app and verify that it does not request permissions again.

Expected Result: The app requests location permissions once and correctly enables location-based features upon approval.

c. Map Display and Marker Placement

Test Scenario: Verify that the map loads correctly and markers are placed at specific locations.
Test Steps:

1. Open the app and navigate to the map screen.
2. Check if the map displays properly.
3. Verify that a marker is placed at the user’s current location.

Expected Result: The map loads successfully, showing the user's location and markers.

d. Handling Location Fetch Failure

Test Scenario: Verify the app’s behavior when the user’s location cannot be retrieved.
Test Steps:

1. Disable location services on the device.
2. Launch the app and observe the behavior.
3. Check for an appropriate error message (e.g., "Unable to fetch location").
4. Enable location services and restart the app.

Expected Result: The app displays a clear error message when the location is unavailable and functions properly once location services are enabled.

e. Application Behavior on Denied Permissions
Test Scenario: Verify how the app behaves when the user denies location permissions.
Test Steps:

1. Install and launch the app for the first time.
2. When prompted for location permission, select "Deny."
3. Observe the app’s response (e.g., error message or restricted functionality).
4. Try enabling location manually from the device settings and relaunch the app.

Expected Result: The app should display a message indicating that location access is required and provide instructions to enable it from settings.

Bug Tracking: Bugs can be reported and tracked using GitHub Issues, where each issue will include a title, description, steps to reproduce, expected vs. actual results, importance level, relevant screenshots/logs, and other details. Bugs and common issues can also be reported locally.