Assignment 1: Testing the MittArv Mobile

Application

Objective:

The purpose of this assignment is to test the MittArv application on a mobile device, identify critical UI/UX issues, and suggest improvements. At MittArv,it's essential that the user experience is smooth, intuitive, and free from major issues.

Scope:

1.Download and install the MittArv application on your mobile device (iOS/Android).
2.Explore the application's features.
3.Identify critical UI/UX issues that hinder user experience.
4.Provide detailed steps for reproducing the issue.

Instructions:

Installation & Initial Setup

1.Download and install the MittArv app from the App Store (iOS) or Google Play Store (Android).

2.Open the app and complete the initial setup process, including account creation and any Necessary permissions

User Interface (UI) & User Experience (UX) Testing

1.Navigate through the app to explore all its features.

2.Pay attention to the layout, design consistency, ease of navigation, and overall user experience.

Identify Any Functional Issues

1.Test all functionalities of the app, including asset vault, emotional will, and multiple email login.

2.Document any bugs, crashes, or performance issues encountered during testing.

Deliverables:

1.Detailed Report:

1.Document Your Testing Process, Including Steps taken, observations, and Issues encountered.

2.Describe any critical UI/UX issues you found.

3.suggest improvements for each issue.

2.Annotated Screenshots:

1.Highlight critical UI/UX issues with annotated screenshots.

2.Use arrows, circles, and text to clearly indicate the problem areas.

3.Summary Report:

1.Provide an overall assessment of the app.

2.List the top 3 recommendations for improving the user experience.

Example Report Structure:

Detailed Report

Installation&Initial Setup

1.Open the MittArv app.

2.Familiarize yourself wit its main features, navigation, and functionality.

3.pay attention to the user interface(UI) elements, such as buttons, menus, forms, and icons.

UI/UX Testing:

Issue 1: Navigation Problem

*Description: Difficulty in finding the “Emotional Will” feature.

*Steps to Reproduce:

1.Open the MittArv app.

2.Try to find the “Emotional Will” feature.

3.Observe the ease of navigation.

*Expected Result: The “Emotional Will” feature should be easily accessible from the main menu.

*Actual Result: The feature is buried under multiple sub-menus, making it hard to find.

*Suggested Improvement: Move the “Emotional Will” feature to the main menu for easier access.

Functional Issues:

Issue 2: Slow Loading Time

*Description: The assets list takes more than 10 seconds to load.

*Steps to Reproduce:

1.Open the MittArv app.

2.Navigate to the “Asset Vault” section.

3.Observe the loading time for the assets list.

*Expected Result: The assets list should load within 2-3 seconds.

*Actual Result: The assets list takes more than 10 seconds to load.

*Suggested Improvement: Optimize the loading process to reduce the time.

Summary Report

Overall Assessment: Provide a brief overview of the app’s strengths and weaknesses.

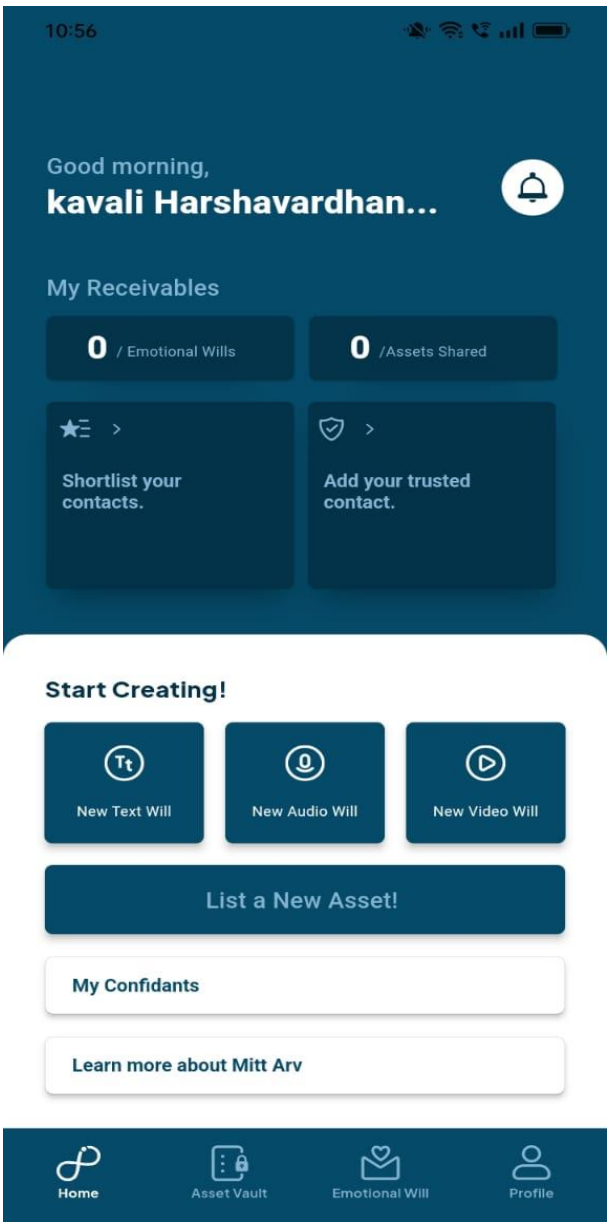
Top 3 Recommendations:

1.Improve navigation by making key features more accessible.

2.Optimize loading times for better performance.

3.Enhance design consistency to improve the overall user experience.

Screen Shorts of User Interface (UI) & User Experience (UX) Testing and Suggested Improvements.



Text Truncation: The text truncation issue could be improved by adjusting the font size or providing a tooltip on hover.

Unlabeled Navigation Icons: The bottom navigation bar contains five icons, but only two are labeled.

Adding labels to all icons would improve clarity and reduce ambiguity.

Suggested Improvements:

1.Resize the user's name to prevent truncation.

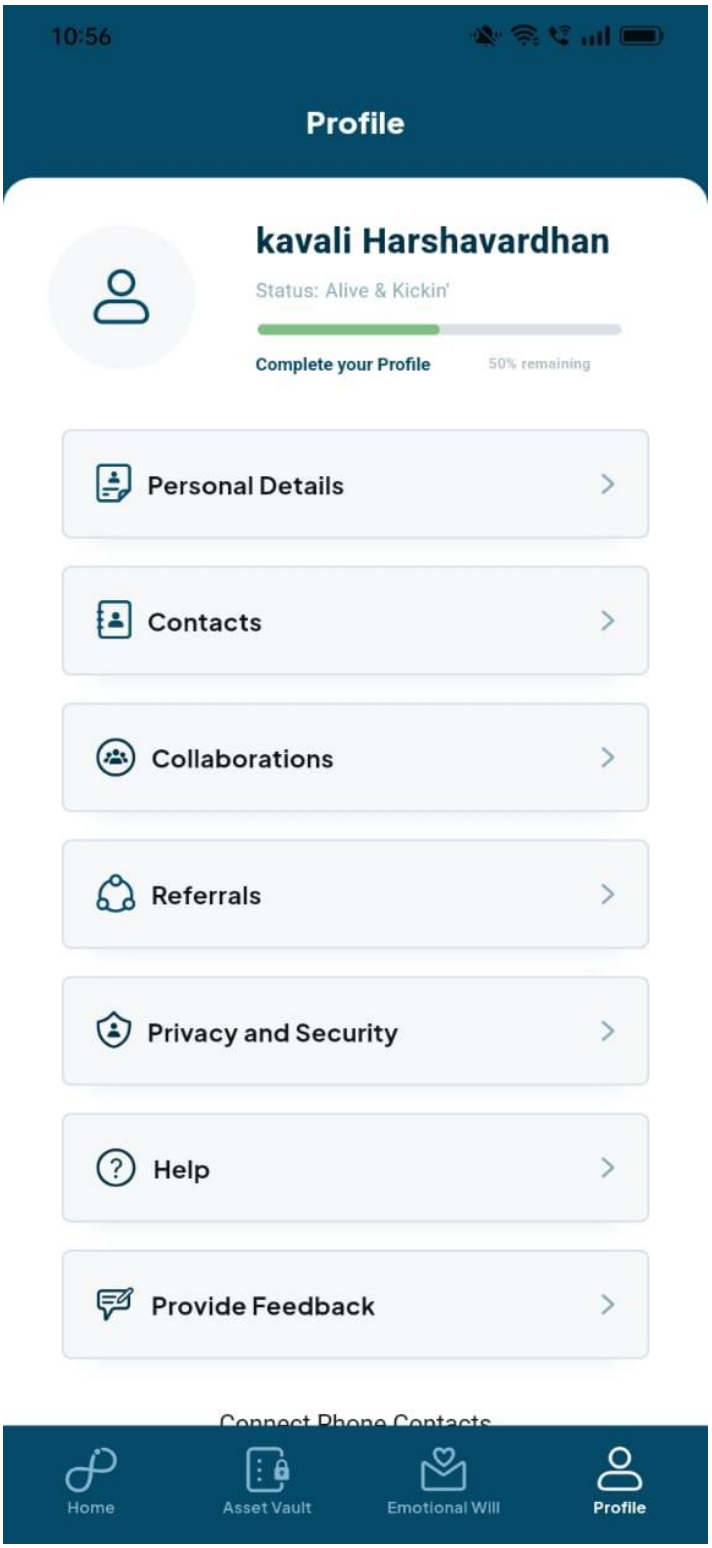
2.Provide clear labels for all navigation icons.

3.Adjust button icon alignment for consistency.

4.Ensure full visibility of all text elements.

5.Clarify the meaning of zero values (no data or actual zero).

6.Enhance color contrast for better visibility.



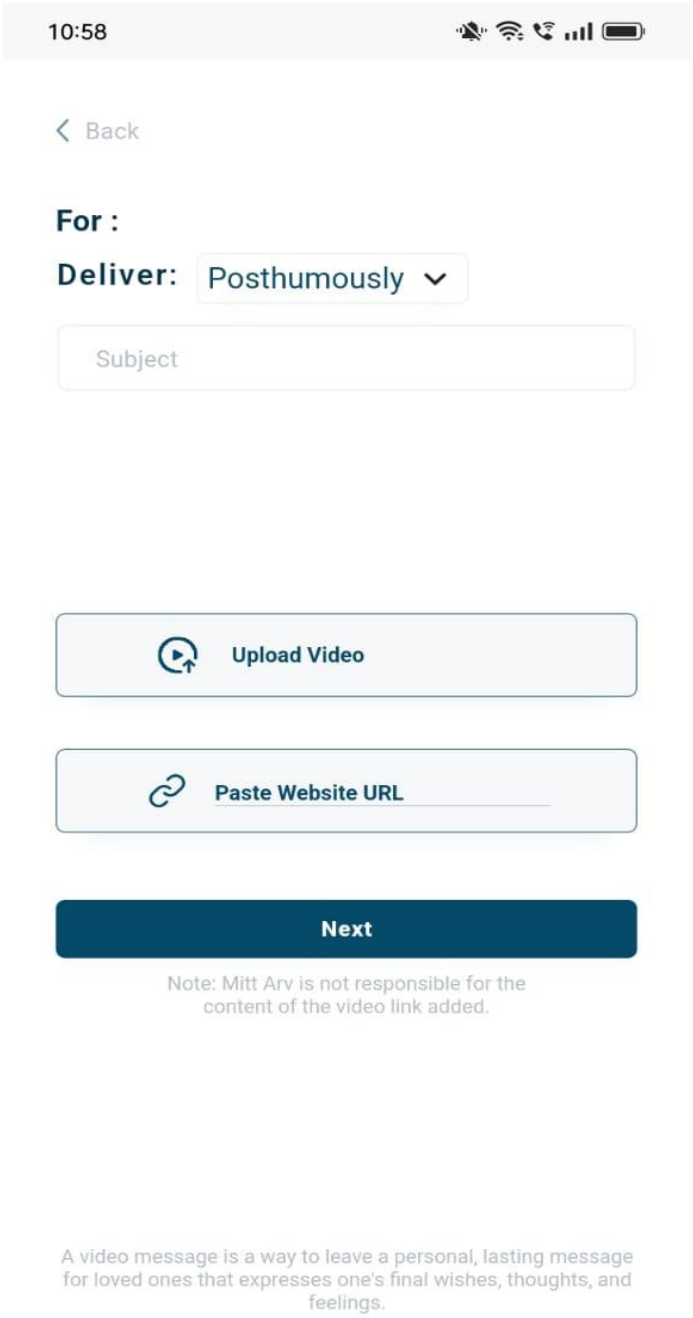
1.Inconsistent Icon Styles:Some icons (e.g., “Personal Details”) have a filled style, while others (e.g., “Contacts”) have an outline style.

2. **Misalignment of Text and Icons**:The text labels and icons are not vertically centered with each other.

Improving Image Quality:

1.Providing a higher-resolution image.

2.Ensuring proper lighting and focus during image capture.



1.**Misalignment of Text and Icons**:

1.The text labels and icons are not vertically centered with each other.

2.Proper alignment ensures a balanced and aesthetically pleasing layout.

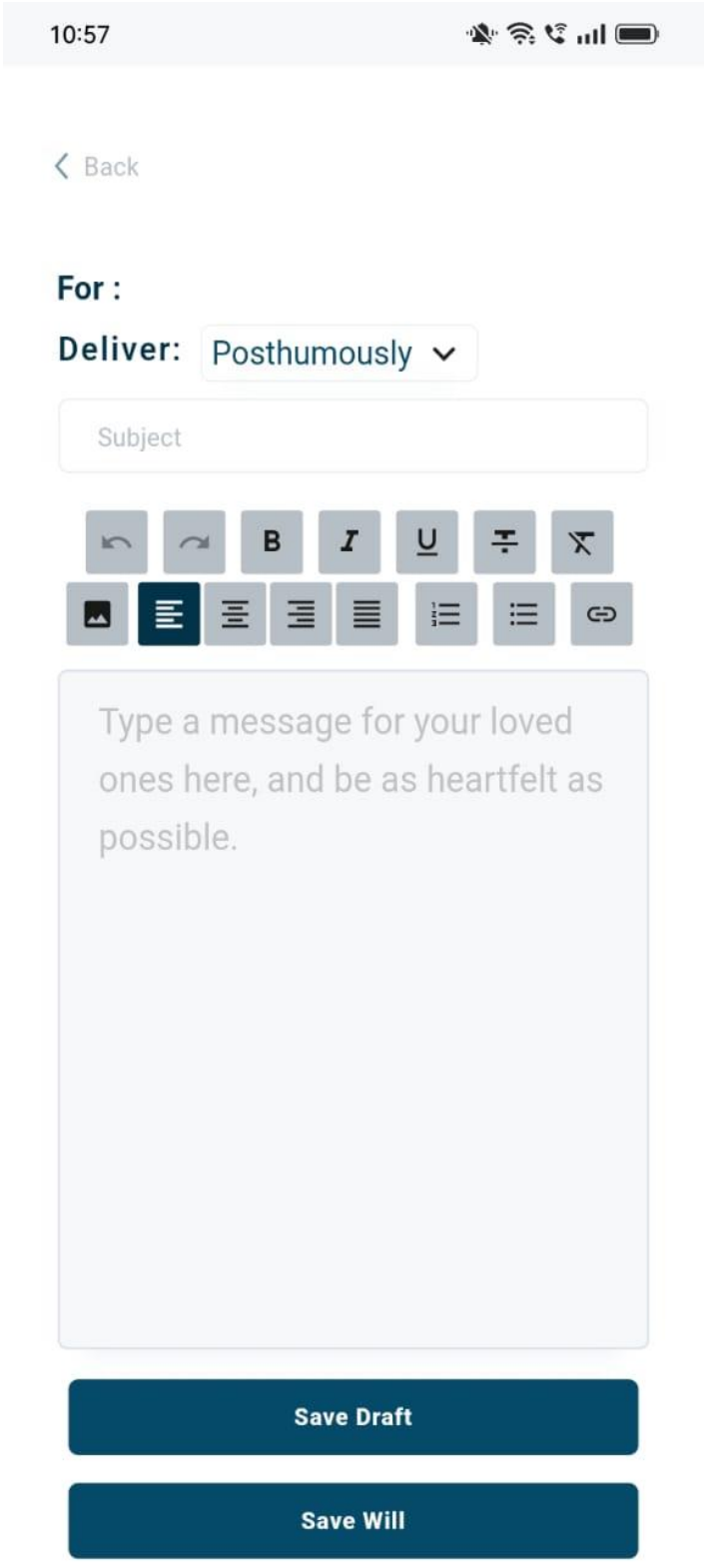
2.Ambiguous Status Message:The status message "Alive & Kickin'" may not clearly convey relevant information within the app context.

3.Inconsistent Icon Styles: Consistency in icon design enhances visual appeal and user understanding.

4. **Inconsistent Capitalization**:The menu items use different capitalization styles ("Connect Bank Account" vs. "Privacy and Security").

Improving Model Performance:

1.Data Collection and Labeling
2. Fine-Tuning and Transfer Learning
3. Evaluate on Real-World Data
4. **Iterative Improvement**



Improving Model Performance:
1.Data Collection and Labeling:Collect more diverse data to train the model on various user profiles and scenarios.

*Ensure accurate labeling of UI elements (e.g., icons, buttons, text) during data annotation.

2.Fine-Tuning and Transfer Learning:Fine-tune the model using transfer learning on a large dataset of UI/UX images.

3. **Evaluate on Real-World Data**:Test the model on real-world user interfaces to assess its performance in practical scenarios.

4. **Iterative Improvement**:Continuously update the model based on user feedback and new data.

Regularly retrain the model to adapt to evolving UI/UX patterns.

Assignment 2: Automation Testing of the MittArv Website

*Minimum of 2 end-to-end user functionalities to be tested
*Automated preparation of the reports post the execution of the script.

Instructions:

1.Set Up Your Automation Testing Environment:

1.Automation Test Im Done in Python Lang
2.Install necessary tools and libraries (e.g., Selenium).

2.Identify End-to-End User Functionalities to Test:

1.Example

Functionality 1:

UserLogin:

Test the login process with valid and invalid credentials.

2.Example

Functionality 2:

Asset Management:

Test adding, editing, and deleting assets in the user area.

3.Develop Automation Scripts:

*Write scripts to automate the identified functionalities.
*Ensure scripts include assertions to validate expected outcomes.

4.Generate Automated Reports:

*Use reporting tools (e.g., Allure, ExtentReports) to generate reports post-execution.
*Ensure reports include details of test cases, execution status, and any errors encountered.

Deliverables:

1. Output of the Report Generated by the Automation Scripts:
*Provide the generated report in a readable format (e.g., PDF).

2. Link to the GitHub Repository:
*Check-in your automation scripts to a GitHub repository.
*Share the link to the repository.

Example Report Structure:

1.Automation Scripts

Script 1: User Login Test (Python with Selenium)

Example Code in Python

```
from selenium import webdriver
import unittest

class MittArvLoginTest(unittest.TestCase):
    def setUp(self):
        self.driver = webdriver.Chrome()

    def tearDown(self):
        self.driver.quit()

    def test_login(self):
        self.driver.get("https://app.mittarv.com")
        # Login steps
        self.driver.find_element_by_id("username").send_keys("valid_user")
        self.driver.find_element_by_id("password").send_keys("valid_password")
        self.driver.find_element_by_id("loginButton").click()
        # Verify that the dashboard page is loaded
        self.assertIn("Dashboard", self.driver.title)

if __name__ == "__main__":
   unittest.main()
```

Script 2: Asset Management Test (Python with Selenium)

Example Code in Python

```
from selenium import webdriver
import unittest

class MittArvAssetTest(unittest.TestCase):
    def setUp(self):
        self.driver = webdriver.Chrome()

    def tearDown(self):
```

```
        self.driver.quit()

    def test_add_asset(self):
        self.driver.get("https://app.mittarv.com")
        # Login steps
        self.driver.find_element_by_id("username").send_keys("valid_user")
        self.driver.find_element_by_id("password").send_keys("valid_password")
        self.driver.find_element_by_id("loginButton").click()
        # Add asset steps
        self.driver.find_element_by_id("addAssetButton").click()
        self.driver.find_element_by_id("assetName").send_keys("New Asset")
        self.driver.find_element_by_id("saveAssetButton").click()
        # Verify that the asset was added
        self.assertTrue("New Asset" in self.driver.page_source)

if __name__ == "__main__":
    unittest.main()
```

Thank You