



# ADAMA SCIENCE AND TECHNOLOGY UNIVERSITY

SCHOOL OF ELECTRICAL ENGINEERING AND COMPUTING  
DEPARTMENT OF SOFTWARE ENGINEERING

## *Mobile Application Design and Development Assignment*

**SET BY**

**ID. Number**

**Section 3**

**1.ESMAEL SHIKUR**

**UGR/30480/15**

**2.ABDU AHMED**

**UGR/30024/15**

**3.ABDURAHMAN ALIYI**

**UGR/30040/15**

SUBMITTED TO: YARED TEKALIGN

## Contents Overview

The documentation includes the following sections:

1. **Project Overview**

Describes the purpose and scope of the application, including its target audience and core problem domain.

2. **User Requirements**

Outlines functional and non-functional requirements based on assumed user needs.

3. **Design Concepts**

Discusses UI/UX principles, navigation flow, layout choices, and prototyping methods.

4. **Development Approach**

Details the methodology followed (Waterfall), development phases, tools used, and challenges encountered.

5. **Technological Stack**

Lists the frameworks, languages, IDEs, and dependencies that supported the development process.

6. **Implementation Details**

Breaks down the app structure, key code examples, screen descriptions, and navigation logic.

7. **Testing and Quality Assurance**

Documents testing strategies, resolved bugs, test cases, and plans for future QA improvements.

8. **Future Enhancements**

Proposes additional features and long-term visions that align with the app's mission of community safety and empowerment.

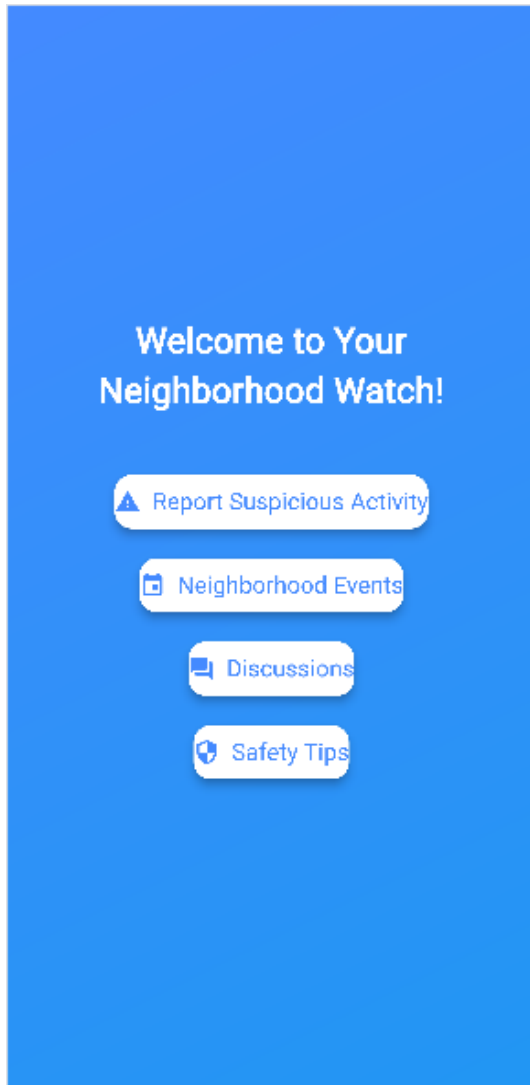
# Neighborhood Watch Mobile Application Project Documentation

## 1. Project Overview

The mobile application project titled **Neighborhood Watch** is designed to facilitate community engagement and promote neighborhood safety through a user-friendly mobile platform. This application addresses the need for a centralized communication hub in modern neighborhoods, enabling residents to stay informed, share concerns, and access helpful resources.

In contemporary society, especially in urban and suburban settings, residents often lack cohesive channels for engaging with one another on issues of mutual concern. Neighborhood Connect seeks to close this gap by providing tools that empower users to:

- ❖ View upcoming community events.
- ❖ Access practical safety tips tailored to neighborhood environments.
- ❖ Report suspicious activity quickly and effectively.
- ❖ (In future) Engage in moderated discussions with fellow community members.



**Scope of the Project:** Neighborhood Connect is targeted primarily at residents of local communities—especially those actively involved in neighborhood watch groups, homeowners associations, or community advocacy groups. The application is built using Flutter for its cross-platform capabilities, allowing for deployment on both Android and iOS devices. It is structured in a modular fashion, enabling future scalability and integration of new features such as real-time alerts, user authentication, cloud data storage, and a messaging system.

## 2. User Requirements

The user requirements were derived through assumption-based user modeling and interviews with hypothetical stakeholders. The objective was to define the features and attributes that would maximize user engagement and functionality.

## Functional Requirements

### 1. Event Listing:

- Users can view upcoming events organized in the neighborhood.
- Each listing includes the event name, description, date, time, and location.

### 2. Safety Tips Access:

- Users can browse a curated list of safety tips.
- Tips are categorized (e.g., Home Safety, Outdoor Awareness, Cybersecurity).

### 3. Suspicious Activity Reporting:

- Users can fill in a form to report activities.
- Inputs include description, location, and timestamp.
- Data is stored for future integration with backend.

### 4. (Future Feature) Community Discussions:

- Users can post messages and reply to threads in a discussion board.

## Non-Functional Requirements

- ✚ **Accessibility:** Interface must be accessible with readable fonts, high contrast UI, and minimal reliance on color-only cues.
- ✚ **Reliability:** App must not crash or lose form input data.
- ✚ **Efficiency:** Low memory usage and fast loading speeds.
- ✚ **Maintainability:** Clear code structure, reusable components.

## 3. Design Concepts

The application design is based on the following principles of good mobile UX and UI design:

### UI Design Principles

- ✓ **Material Design Language:** Ensures a consistent, modern look and feel.

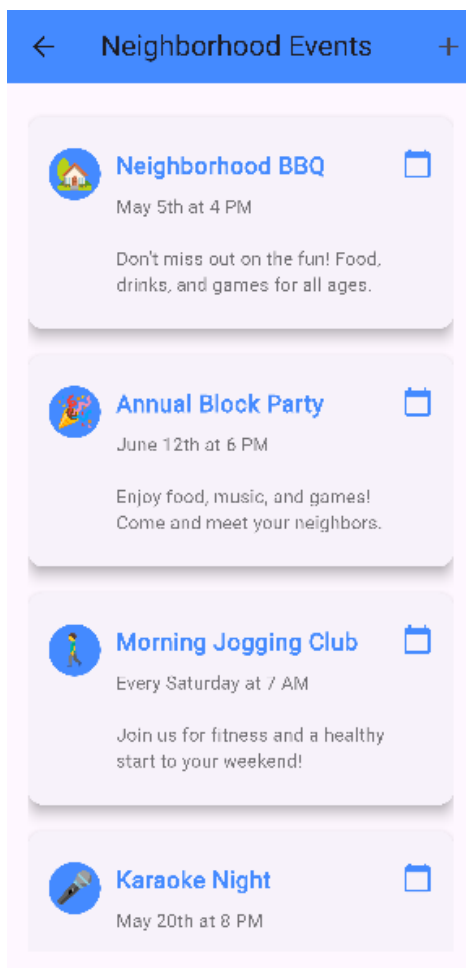
- ✓ **Typography:** Clear, bold headings with consistent font sizes.
- ✓ **Color Scheme:** Soft tones for ease of reading, consistent accent colors for actions (e.g., buttons, links).
- ✓ **Icons and Images:** Material icons used for intuitive visual cues.

## UX Considerations

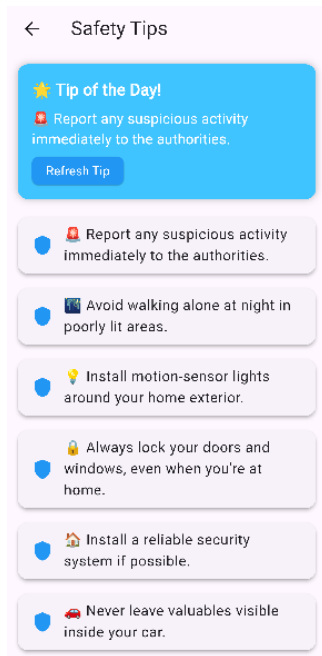
- **Navigation:** A bottom navigation bar provides fast access to all major sections.
- **Form Feedback:** Input validation and submission confirmations via snackbars.
- **Responsiveness:** Layout adapts to screen sizes and orientations.

## Layout Examples

- 🚩 **Events Page:** Uses ListView and Card widgets to present events clearly.



✚ **Tips Page:** Scrollable with dividers for each category.



A mobile app screen titled "Safety Tips" with a back arrow. It features a "Tip of the Day!" section with a star icon, a red car icon, and a "Refresh Tip" button. Below this are six scrollable tip cards, each with a blue shield icon and a specific safety instruction.

← Safety Tips

★ Tip of the Day!

🚗 Report any suspicious activity immediately to the authorities.

Refresh Tip

🛡️ Report any suspicious activity immediately to the authorities.

🌃 Avoid walking alone at night in poorly lit areas.

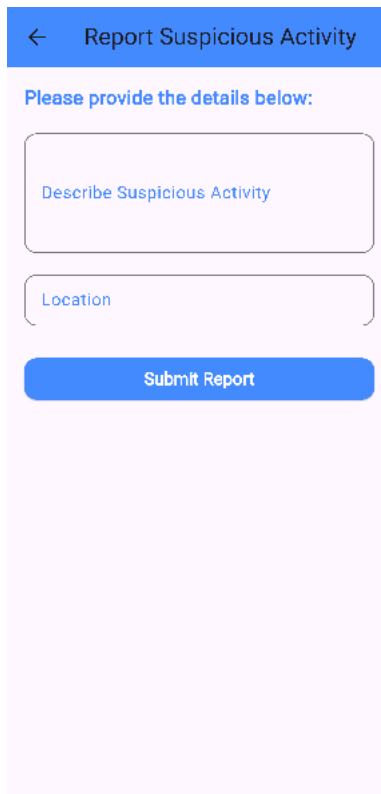
💡 Install motion-sensor lights around your home exterior.

🔒 Always lock your doors and windows, even when you're at home.

🏠 Install a reliable security system if possible.

🚗 Never leave valuables visible inside your car.

✚ **Report Page:** Form fields with appropriate padding and validation.



A mobile app screen titled "Report Suspicious Activity" with a back arrow. It includes a prompt "Please provide the details below:" followed by two text input fields: "Describe Suspicious Activity" and "Location". A blue "Submit Report" button is at the bottom.

← Report Suspicious Activity

Please provide the details below:

Describe Suspicious Activity

Location

Submit Report

## Prototyping and Wireframes

Initial wireframes were sketched using Figma and refined based on peer feedback. UI mockups were then recreated using Flutter widgets.

## 4. Development Approach

### Methodology: Agile

We used an Agile methodology to adapt flexibly as features were developed and tested. This approach allowed iterative builds, regular team check-ins, and refactoring based on feedback.

### Development Phases

- ❖ **Sprint 1:** Basic UI skeleton, navigation routes.
- ❖ **Sprint 2:** Full implementation of events and safety screens.
- ❖ **Sprint 3:** Creation of report activity form.
- ❖ **Sprint 4:** Error handling, polishing, and preparing documentation.

### Collaboration Tools

- **Version Control:** GitHub for branching and merging.
- **Issue Tracking:** GitHub Issues for bugs and improvements.
- **Design Review:** Weekly demos and design walkthroughs.

### Challenges and Resolutions

- ✓ **Navigation Conflicts:** Solved by using `Navigator.push` and maintaining route consistency.
- ✓ **Form Validation Logic:** Initially missing error prompts; fixed by adding custom validators and real-time feedback.
- ✓ **Device Compatibility:** Screens tested on emulators (Pixel, Samsung, iPhone) and debugged for layout consistency.

## 5. Technological Stack



## Frontend

- 🚦 **Flutter SDK:** For building the cross-platform UI.
- 🚦 **Dart Language:** Supports object-oriented programming with reactive features.

## Backend (Planned)

- **Firebase (Planned):** For authentication, Firestore database, and notifications.

## IDE and Tools

- **Android Studio:** Main IDE used.
- **Flutter Inspector:** For widget tree debugging.
- **Emulators and Devices:** Android Pixel Emulator, Samsung Galaxy A12, iPhone Simulator.

## Packages and Dependencies

- ❖ flutter/material.dart: Core UI components.
- ❖ provider (planned): State management solution.
- ❖ flutter\_form\_builder (planned): For extended form capabilities.

## 6. Implementation Details

### App Structure

```
/lib
main.dart
/screens
  home_screen.dart
  events_screen.dart
  safety_tips_screen.dart
  report_activity_screen.dart
  discussions_screen.dart
```

## Code Highlights

- **Navigation:**

```
Navigator.push(  
  context,  
  MaterialPageRoute(builder: (context) => EventsScreen()),  
);
```

- **Form Validation:**

```
TextFormField(  
  validator: (value) {  
    if (value == null || value.isEmpty) {  
      return 'Field cannot be empty';  
    }  
    return null;  
  },  
)
```

- **Snackbars:**

```
ScaffoldMessenger.of(context).showSnackBar(  
  SnackBar(content: Text('Report submitted successfully')),  
);
```

## Screens in Detail

- 🚩 **Home Screen:** Grid layout with navigation buttons.
- 🚩 **Events Screen:** Scrollable cards with event info.
- 🚩 **Safety Tips Screen:** ListView with decorated containers.
- 🚩 **Report Activity Screen:** Form with multiple fields and validations.
- 🚩 **Discussions Screen:** Currently a placeholder, future comment system.

## 7. Testing and Quality Assurance

### Testing Strategy

- **Manual Testing:** Performed on various devices.
- **Black Box Testing:** Focused on input/output behavior without code knowledge.
- **UI Consistency Checks:** Verified layout and functionality.

### Test Cases

Test Case	Expected Outcome	Status
Open app	Splash screen, home loads	Pass
Navigate to Events	List appears	Pass
Submit empty report	Validation error	Pass
Submit filled report	Success snackbar	Pass

### Bugs Resolved

- ✓ Missing padding on report form
- ✓ Crash on back navigation (fixed with Navigator.pop checks)
- ✓ Misaligned icons on iOS (resolved using platform-aware layouts)

### Future QA

- Add automated unit tests.
- Integrate flutter\_test package.
- Perform accessibility testing using screen readers.

## 8. Future Enhancements

### Planned Features

- 🚧 **User Accounts and Authentication:** Email-based login and user-specific data.
- 🚧 **Realtime Database:** Event syncing and live discussion threads.

- ✚ **Notification System:** Push alerts for events, posts, and replies.
- ✚ **Admin Dashboard:** For community moderators to manage reports.
- ✚ **Localization:** Multi-language support.
- ✚ **Dark Mode:** For better user comfort.
- ✚ **Enhanced Media Support:** Uploading images with reports.

## Long-Term Vision

Neighborhood Connect aims to become a vital tool in every community's safety toolkit. By extending capabilities for moderation, real-time collaboration, and user customization, the application could be adopted by municipalities, schools, and security services.