

American International University-Bangladesh (AIUB)  
**Department of Computer Science  
Faculty of Science & Technology (FST)**

***SafeHer* – *A Women’s Safety Application***

A Software Engineering Project Submitted

By

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Semester: Summer\_21\_22** | | **Section:** | **Group Number:** | |
| SN | Student Name | Student ID | Contribution (CO3+CO4) | Individual Marks |
| 1 | **TABASSUM AKTAR SADIA** | 23-52427-2 |  |  |
| 2 | **SHAMA ZERIN** | 22-49210-3 |  |  |
| 3 | **SABIHA BINTE MAHABUB LEYA** | 23-50978-1 |  |  |
| 4 | **ANTARA BHOWMIC** | 23-51952-2 |  |  |
|  |  |  |  |  |

The project will be evaluated for the following Course Outcomes



|  |  |  |
| --- | --- | --- |
| ***CO3 (PO-g-1)***  ***Select appropriate software engineering models, project management roles and their associated skills for the complex software engineering project and evaluate the sustainability of developed software, taking into consideration the societal and environmental aspects*** | Total Marks | |
|  | |
| Selection of Software Engineering Models: Process model selection and presents sufficient evidence to support argument for the model selection | [5 Marks] |  |
| Role identification and Responsibility Allocation: Well-planned project with proper role identification and responsibility allocation in the project management activities | [5Marks] |  |
| Formatting and Submission: Submission, Defense, Completeness, Spelling, grammar, and Organization of the Project report | [5Marks] |  |
| ***CO4 (PO-k-1)***  ***Apply engineering management principles and economic decision making to develop software engineering project management plan.*** | Total Marks | |
|  | |
| Project WBS and Testcases: Relevant WBS (project task list) and testcases for the proposed project are stated properly. | [5Marks] |  |
| Effort Estimation and Scheduling: Project estimation was described using proper effort estimation or schedules based on available project resources | [5Marks] |  |
| Risk Management: Sufficient and appropriate risks are identified, analyzed, and properly categorized or prioritized. | [5Marks] |  |



# PROJECT PROPOSAL

## Background to the Problem

In recent years, women’s safety has become an increasingly critical concern worldwide, especially in developing nations like Bangladesh. Reports of harassment, assault, and abduction are distressingly frequent, particularly in urban areas. Many women feel unsafe when traveling alone or in unfamiliar places due to the lack of immediate access to help and safety resources. One of the core issues is the absence of an accessible, real-time safety support system that can empower women to act quickly in emergencies. Traditional emergency hotlines are often slow, confusing, or inaccessible during real-time danger. Moreover, a lack of awareness and proper reporting channels contributes to underreporting of incidents, thereby weakening the response system.

The root cause of this problem lies in insufficient infrastructure, underreporting, poor accessibility of safety services, and a lack of trust in existing systems. This issue is important not just for individual well-being, but also for societal development, gender equality, and national safety

## Solution to the Problem and Process Model Selection

#### ****Project Scope and Features****

The scope of the SafeHer project is to develop a mobile-based safety application that addresses the urgent need for accessible, real-time safety solutions for women in Bangladesh. The application aims to empower women by providing emergency tools, anonymous reporting mechanisms, and location-based safety information.

**Key Features include:**

* **Emergency Help Button:** One-tap SOS with voice activation and live GPS location sharing with trusted contacts and authorities.
* **Live Location Tracking:** Real-time monitoring with optional sharing features for enhanced safety.
* **Incident Reporting:** Anonymous or identified reporting to encourage barrier-free communication.
* **Safe Zone Directory:** Nearby verified safe locations such as police stations and hospitals, complete with contact details and distance.
* **Feedback System:** Users can rate and comment on the safety of specific areas, contributing to a community-driven safety map.
* **Admin Dashboard:** For law enforcement and NGOs to monitor, manage reports, and provide rapid response.

The project involves two primary user roles—**End Users (Women)** and **Admins (Authorities/NGOs)**—with distinct functionalities.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  | | --- | --- | --- | --- | | User Story ID | As a | I want to | So that I can | | US-01 | User | Register and log in securely | Access personalized emergency features | | US-02 | User | Tap a single SOS button | Instantly alert my contacts and share my live location | | US-03 | User | Enable live location tracking | Share my location with trusted contacts during unsafe situations | | US-04 | User | Report incidents anonymously | Ensure privacy and encourage reporting without fear | | US-05 | User | View nearby safe zones | Identify and navigate to police stations or hospitals | | US-06 | User | Provide feedback on locations | Help others stay informed about safety | | US-07 | Admin | Log in securely | Manage incident reports and users | | US-08 | Admin | View real-time user alerts | Respond quickly to emergencies | | US-09 | Admin | Manage feedback and safe zones | Ensure reliable and verified safety information | |

1. **User Story Board (Trello)**

The team used **Trello** to create a Scrum-based user story board with the following lists:

* **Backlog** – Collecting new features (e.g., voice-activated SOS, feedback mechanism).
* **To Do** – Selected stories for upcoming sprint.
* **In Progress** – Tasks under active development (e.g., login system, emergency alert module).
* **Testing** – Completed features awaiting validation.
* **Done** – Completed and reviewed features.

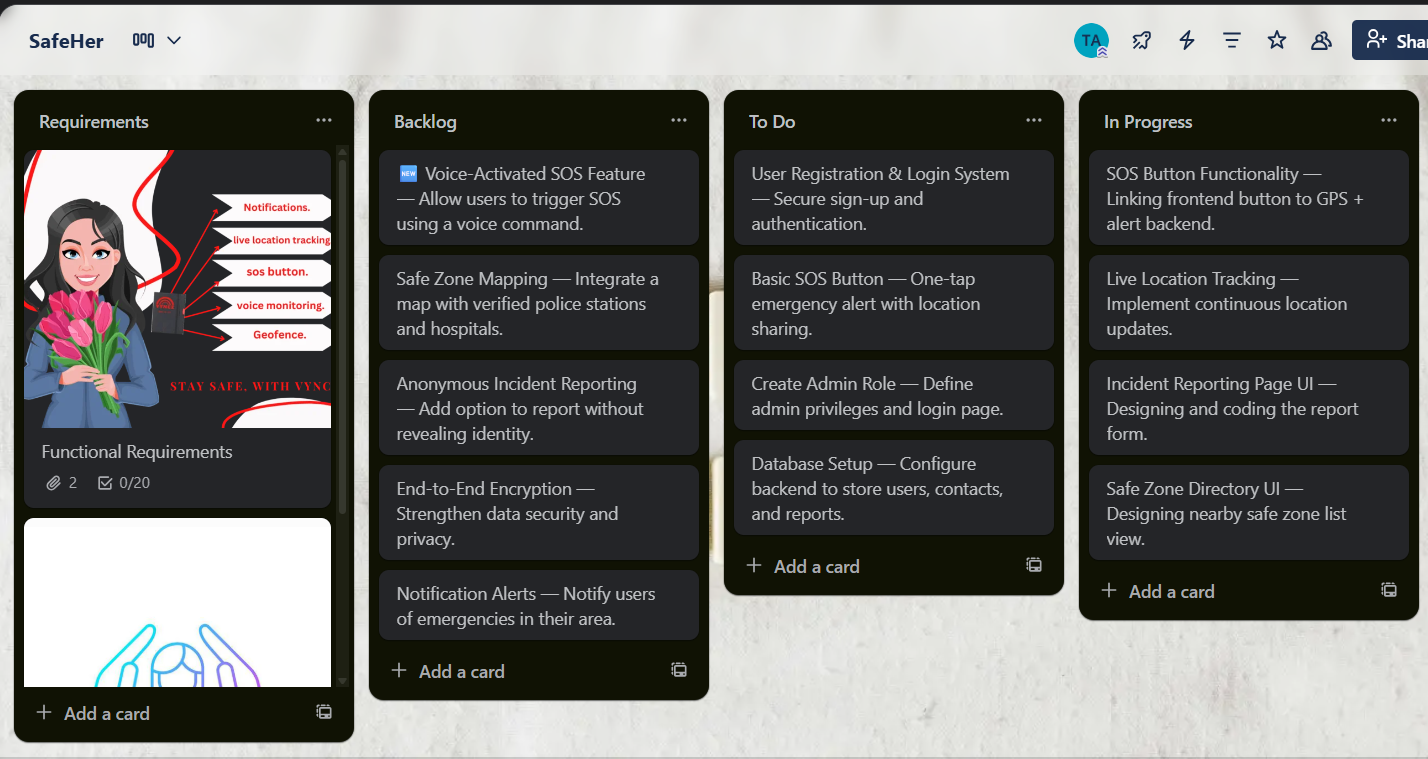


Figure: User Story Board (Trello)

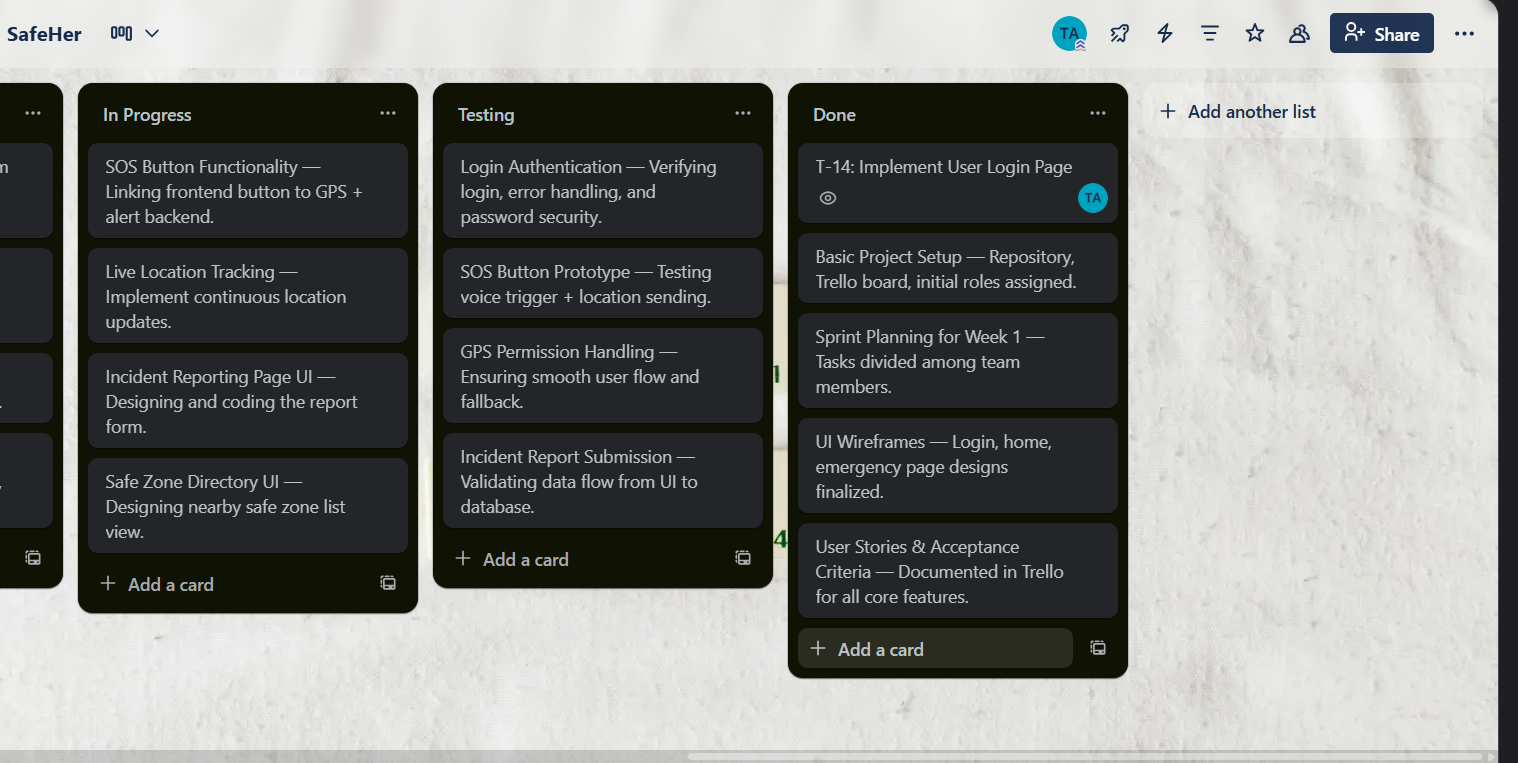


Figure: User Story Board (Trello)

#### ****Existing Software Solutions****

Existing applications such as **bSafe** and **Circle of 6** offer emergency alerts and live tracking features. However, they lack several aspects critical for the Bangladeshi context:

* No integration with **local safe zones** (police stations, hospitals).
* **No anonymous reporting** support.
* Lack of **multi-role admin panels** for real-time monitoring.
* Not culturally localized for Bangladesh.

SafeHer addresses these gaps with **localized design**, **voice activation**, **safe zone mapping**, and **admin dashboards** for response coordination.

1. **Selected Process Model: Scrum**

For SafeHer, the **Scrum process model** is chosen as the most suitable software development methodology.

**Rationale for Selection:**

* **Iterative and Incremental Delivery:** Enables rapid development of a Minimum Viable Product (MVP), followed by frequent updates based on user feedback.
* **Flexibility to Evolving Requirements:** The women’s safety domain is dynamic; Scrum’s sprint-based structure accommodates changes easily.
* **User-Centric Development:** Allows continuous integration of user feedback, especially from pilot groups.
* **Team Collaboration:** Daily standups, sprint planning, and retrospectives ensure clear communication within the 4-member team.
* **Rapid Release Cycles:** Ensures timely delivery of core safety features such as emergency alerts and live tracking.

#### ****Project Environment Analysis****

The **requirements** for SafeHer are **dynamic** and expected to evolve based on real-world testing, user behavior, and emerging safety needs.

The **team size** (4 members) fits well with Scrum, as the framework supports **small, cross-functional teams** with clearly defined roles such as Product Owner, Scrum Master, and Scrum Team.

**Communication** is managed through regular sprint meetings, Trello boards, and shared documentation.

1. **Feasibility and Model Fit**

The Scrum model supports:

* **Business objectives** — enabling rapid release of a community-centric safety platform.
* **Scope changes** — through product backlog refinement and sprint reprioritization.
* **Technology adaptation** — easily integrating GPS APIs, cloud services, and security updates.
* **User requirement changes** — incorporating new features without halting the project flow.

#### ****Flexibility and Creativity****

SafeHer’s solution integrates **voice activation**, **localized safe zones**, and **real-time feedback** — demonstrating creative adaptation of existing safety concepts to the Bangladeshi context. The app is not just a panic button but a **community-driven safety platform**.

1. **Contribution to Scientific Knowledge**

The project contributes to the field of **mobile safety technologies** for developing nations, offering insights into:

* Integrating **real-time GPS** with **localized safety data**.
* Encouraging **anonymous reporting** in culturally sensitive contexts.
* Applying **Agile (Scrum)** in small-scale, socially impactful projects.

The app can serve as a **prototype for other vulnerable groups**, including children and the elderly.

**6. Evidence for Model Selection**

* According to **Sommerville (2016)**, Agile methods such as Scrum are particularly well-suited for applications with rapidly changing or uncertain requirements, which is true for safety-focused applications like SafeHer.
* A study by **Boehm and Turner (2004)** also emphasizes Agile’s success in projects that demand user involvement and rapid adaptability, which aligns with SafeHer’s iterative feedback and admin-managed content features.
* Research by **Petersen & Wohlin (2010)** indicates that Agile approaches improve software quality and delivery speed in mobile application development — both critical for SafeHer’s usability and trustworthiness.
* In **Koch (2005)**, it is noted that Agile (Scrum in particular) shortens time-to-market while maintaining adaptability, a key factor for the rapid prototyping and release strategy of SafeHer.

#### ****Risk Management Through Scrum****

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Risk Name** | **Likelihood** | **Impact** | **Risk Response** | **Mitigation Plan** |
| Changing Requirements | High | High | The team will handle this risk by adapting sprint plans and updating the product backlog so that new requirements can be managed without breaking sprint goals. | Use backlog refinement, adjust sprint goals, involve Product Owner for reprioritization. |
| Data Privacy Leak | Low | High | The team will prevent this risk by applying proactive measures such as strong security practices, daily Scrum meetings, and early testing to avoid problems before they occur. | Apply encryption, secure login, security testing as part of Definition of Done. |
| System Downtime | Medium | High | The team will reduce the risk by preventing issues through redundancy and will also prepare a backup plan (contingency) to keep the system working if downtime occurs. | If the system goes down, the app will automatically send the last available tracking location to emergency contacts and continue with SMS-based SOS alerts. Cloud hosting with redundancy will also be used to reduce downtime. |
| GPS Inaccuracy | Medium | Medium | The team will manage this risk by reducing its effect, such as using alternative technical solutions, adjusting sprint priorities, or improving system design based on feedback. | Combine GPS with network triangulation, allow manual location confirmation. |
| Delayed Authority Response | Medium | High | The team will manage this risk by reducing its effect, such as using alternative technical solutions, adjusting sprint priorities, or improving system design based on feedback. | Partnerships with police/NGOs, escalation alerts if first responder fails. |
| Integration Issues | Medium | Medium | The team will manage this risk by reducing its effect, such as using alternative technical solutions, adjusting sprint priorities, or improving system design based on feedback. | Continuous integration, test after each sprint, peer code review. |

Scrum’s **iterative nature** allows:

* Early identification of changing requirements.
* Continuous integration and security testing.
* Regular sprint reviews to catch integration issues.
* Clear mitigation strategies for GPS inaccuracies, data privacy, and downtime.

#### ****Relation to Project Schedule****

The **10-week project schedule** aligns with multiple **short Scrum sprints** (e.g., 2-week sprints):

* Sprint 1: Core authentication and SOS
* Sprint 2: Live tracking and reporting
* Sprint 3: Admin dashboard
* Sprint 4: Safe zones, feedback, testing, and deployment

This structure ensures **timely delivery** while allowing adaptability throughout development.

1. **Justification for Process Model Selection**

The **Scrum model** is more suitable for the *SafeHer* project compared to other software development models due to its **flexibility, adaptability, and user‐centric approach**, which align perfectly with the dynamic and sensitive nature of women’s safety requirements.

The Scrum model is chosen over traditional models like **Waterfall**, **V-Model**, or even **Incremental** for the following reasons:

* **Iterative and Incremental Development**: Scrum supports frequent releases of working software, which is ideal for quickly deploying an MVP (Minimum Viable Product) version of SafeHer and progressively adding features like feedback mechanisms or voice activation.
* **User-Centric Design**: Frequent user feedback (from testers or pilot groups) can be incorporated into successive sprints, aligning perfectly with SafeHer’s goal of evolving based on community input.
* **Flexibility to Change**: Unlike the Waterfall model, which is rigid and linear, Scrum accommodates changing requirements—critical for an app meant to respond to new safety risks and user needs.
* **Short Sprint Cycles**: Allows rapid development and testing of core features like emergency alerting and live tracking.
* **Cross-functional Collaboration**: Encourages close collaboration between developers, testers, domain experts, and potential users — ensuring SafeHer meets both technical and social expectations.

**Waterfall Model**

* Follows a rigid, linear process, making it difficult to adapt to changing requirements. Once a phase is completed, user feedback is hard to incorporate—unsuitable for the evolving nature of *SafeHer*.

**V-Model**

* A structured, test-heavy approach ideal for hardware or regulatory systems. It lacks the flexibility and quick iteration needed for a dynamic mobile app like *SafeHer*.

**Incremental Model**

* Delivers the product in parts but lacks defined roles (e.g., Scrum Master, Product Owner) and structured feedback loops, making it less collaborative than Scrum.

**Iterative Model**

* Focuses on repeated development cycles but lacks formal team practices like daily standups or sprint reviews, which are essential for team coordination and fast delivery.

**Extreme Programming (XP)**

* Emphasizes technical practices like pair programming. While agile, it’s developer-centric and lacks formal management roles. It doesn’t support broader collaboration with non-technical stakeholders, which is key for *SafeHer*.

**Scrum ensures frequent stakeholder engagement**, rapid response to changing requirements, and continuous delivery of usable software. Its sprint-based structure allows the team to deploy a **Minimum Viable Product (MVP)** early and then progressively enhance the app with features such as anonymous reporting, live tracking, and safe zone directories. This flexibility, combined with its lightweight management structure and strong emphasis on communication, makes Scrum the **most appropriate and effective model** for achieving SafeHer’s objectives within the project’s scope, timeline, and resource constraints.

# 1.3 Project Role Identification and Responsibilities

The successful development of the *SafeHer* application relies on a well-defined distribution of roles and responsibilities across both **development** and **management** domains. This ensures that the team functions efficiently, meets deadlines, and maintains high-quality standards throughout the software development life cycle.

1. **4.1 Main Roles**
2. **Product Owner**
3. **Scrum Master / Project Manager**
4. **Development Team**
   * Frontend Developer
   * Backend Developer
   * UI/UX Designer
5. **Quality Assurance (QA) & Testing Lead**
6. **Deployment & Maintenance Coordinator**

**Role Descriptions and Responsibilities**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | Role | Responsibilities | | Product Owner | Acts as the voice of stakeholders and end users (women, NGOs, law enforcement). Responsible for collecting and prioritizing requirements, maintaining the product backlog, making key functional decisions, and ensuring that the developed features align with project objectives. Oversees requirement gathering and approves sprint goals. | | Scrum Master / Project Manager | Facilitates team communication and Scrum ceremonies (sprint planning, daily standups, retrospectives). Removes obstacles, monitors progress, manages the project timeline, and ensures smooth coordination between team members. Also responsible for resource management and risk mitigation. | | Frontend Developer | Responsible for implementing the user interface using mobile-friendly design principles. Works on login pages, SOS button, incident reporting forms, and safe zone directory interfaces. Collaborates closely with the backend developer to integrate APIs. | | Backend Developer | Designs and develops the server-side logic, database structure, and integration with GPS and authentication services. Handles data storage for incident reports, user management, and admin dashboard functionalities. | | UI/UX Designer | Creates wireframes, mock-ups, and prototypes for the SafeHer application. Focuses on usability, accessibility, and culturally appropriate design for female users in Bangladesh. | | QA & Testing Lead | Ensures software quality through unit testing, integration testing, security testing, and user acceptance testing. Validates that all features meet acceptance criteria and function correctly under real-world scenarios. | | Deployment & Maintenance Coordinator | Manages the app deployment process, server configuration, database hosting, and ongoing maintenance. Monitors system performance, applies updates, and handles post-release issue tracking. | |

**Responsibilities in Key Project Stages**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | Project Stage | Responsible Roles | Key Activities | | Requirements Gathering | Product Owner, Scrum Master | Collect user needs, analyze existing solutions, define functional & non-functional requirements, create user stories. | | Design | UI/UX Designer, Frontend Developer, Backend Developer | Develop wireframes, UI mockups, class diagrams, database schemas, and architecture planning. | | Implementation | Frontend Developer, Backend Developer | Develop core features such as registration/login, SOS button, GPS tracking, incident reporting, and admin dashboard. | | Testing | QA & Testing Lead, Developers | Conduct unit tests, integration tests, performance tests, and user acceptance tests. Verify GPS accuracy, SOS speed, and data security. | | Deployment | Deployment & Maintenance Coordinator, Scrum Master | Deploy the MVP, manage app hosting, ensure uptime, fix post-deployment bugs, and monitor feedback for future sprints. | |

1. **Decision Making, Quality Assurance, and Resource Management**

* **Decision Making:**
  + *Product-related decisions* are made by the **Product Owner** based on user needs and priorities.
  + *Technical decisions* are made collaboratively between the **Development Team** and **Scrum Master**.
  + *Strategic project decisions* (e.g., timeline, resource allocation) are taken by the **Scrum Master/Project Manager**.
* **Quality Assurance:**
  + Primarily handled by the **QA & Testing Lead**, with support from developers through continuous testing during sprints.
  + The Product Owner also validates deliverables against user expectations.
* **Resource Management:**
  + Overseen by the **Scrum Master**, ensuring optimal utilization of team members, scheduling tasks efficiently, and maintaining required infrastructure.

#### ****Distribution of Responsibilities Among Team Members****

The project team consists of **four members**, and responsibilities are distributed based on individual strengths and contributions:

* **Tabassum Aktar Sadia** – Worked on project proposal, activity diagram, SOS page design, effort estimation, budgeting, and scheduling. Given her organizational and analytical skills, she fits the **Scrum Master.**
* **Shama Zerin** – Focused on use case diagram, login system design, and functional requirements. She is assigned as **Frontend Developer** and assists the Product Owner.
* **Sabiha Binte Mahabub Leya** – Contributed to class diagrams, admin dashboard design, risk analysis, and non-functional requirements. She plays the role of **Backend Developer** and **QA Assistant**.
* **Antara Bhowmic** – Developed sequence diagrams and supported backend logic. She is involved in **UI/UX design** and **database integration**.

## 2. SOFTWARE REQUIREMENTS SPECIFICATIONS (SRS) / PRODUCT REQUIREMENTS DOCUMENT (PRD)

## 2.1 Functional Requirements

The **SafeHer** application focuses on providing **real-time emergency response**, **secure user interaction**, and **community-driven safety features**. The following functional requirements outline the **core services, operations, and workflows** that support the overall objective of enhancing women’s safety in Bangladesh through a mobile application.

**A. Major Functionalities**

1. **User Registration and Authentication**
   * Users can create an account using personal information such as name, email, phone number, and password.
   * The system securely authenticates users before granting access to the app.
   * Accounts will be locked temporarily after multiple failed login attempts to enhance security.

**Objective Supported:** Ensures that only verified users can use emergency services and submit reports, protecting system integrity.

1. **Emergency SOS Feature**
   * A single tap on the SOS button triggers a **loud voice alert** and shares the user’s **real-time GPS location** with pre-selected emergency contacts and admins.
   * The SOS can also be activated through **voice commands** for hands-free operation during dangerous situations.

**Objective Supported:** Enables rapid response during emergencies, ensuring users can seek help instantly even under pressure.

1. **Live Location Tracking**
   * Users can enable live GPS tracking at any time.
   * The app allows users to share their location with trusted contacts and administrators selectively.
   * Admins can track users during active emergency alerts for quicker intervention.

**Objective Supported:** Enhances safety by allowing continuous monitoring during unsafe situations.

1. **Incident Reporting**
   * Users can report incidents **anonymously** or using their identity.
   * Admins can view, filter, and manage submitted reports through the dashboard.
   * Reports include details such as incident type, location, and time.

**Objective Supported:** Encourages reporting of safety issues without fear, improving data collection and community awareness.

1. **Safe Zone Directory**
   * Displays nearby **verified safe locations** such as police stations and hospitals.
   * Users can view contact details, map directions, and distance from their location.

**Objective Supported:** Provides quick access to help points, supporting safer navigation for women in real time.

1. **Feedback Mechanism**
   * Users can rate and provide feedback on the safety of specific areas or safe zones.
   * Admins can review and moderate feedback to ensure reliability.

**Objective Supported:** Builds a community-informed safety network, helping users make informed decisions.

1. **Admin Dashboard & Monitoring**
   * Admins can log in securely to access a **central dashboard**.
   * The dashboard includes summaries of reports, SOS alerts, active users, and safe zone management.
   * Admins can manage user accounts, deactivate suspicious users, and track ongoing emergencies.

**Objective Supported:** Enables authorities and NGOs to coordinate quick, organized responses to incidents.

**B. User Story / Use Case Style Descriptions**

|  |  |  |
| --- | --- | --- |
| ID | User Story | Acceptance Criteria |
| FR-01 | As a user, I want to register securely so that I can access emergency services. | Registration form validates input, securely stores data, and redirects to login after success. |
| FR-02 | As a user, I want to trigger an SOS alert with one tap so that my contacts and admins are immediately notified. | SOS alert activates voice, sends location, and shows confirmation on screen. |
| FR-03 | As a user, I want to share my live location with trusted contacts so that I can be tracked during unsafe situations. | Location updates are transmitted accurately and can be turned on/off. |
| FR-04 | As a user, I want to report incidents anonymously so that I can share information safely. | Report is stored with “anonymous” tag and appears in admin dashboard without personal data. |
| FR-05 | As a user, I want to find nearby safe zones so that I can quickly navigate to a secure location. | Safe zone list loads within 3 seconds and provides map + contact info. |
| FR-06 | As an admin, I want to view and manage all reports and SOS alerts so that I can respond quickly. | Dashboard displays live data, with filtering and action options. |

**C. Typical Workflow Example**

**Task:** *Triggering an SOS Alert and Receiving Help*

1. **User** taps the SOS button on the Home page.
2. The system immediately **activates a voice alarm**.
3. The app fetches the **GPS location** and sends it to emergency contacts and the admin dashboard.
4. **Admins** receive the alert, track the user’s location, and coordinate emergency response.
5. The user can view status updates (e.g., “Admin Received,” “Help On the Way”).

This workflow illustrates how SafeHer provides **real-time assistance** through minimal user interaction.

**2.2 Non-Functional Requirements**

In addition to its functional behavior, **SafeHer** must satisfy several **quality attributes** to ensure reliability, security, and long-term sustainability.

**1. Performance**

* The **SOS alert** must be triggered and sent **within 2 seconds** of user action.
* All pages should load within **3 seconds** under standard network conditions.
* The app should support **up to 500 concurrent users** during initial deployment, with minimal lag.

**2. Reliability**

* The system must maintain **99% uptime**, especially for emergency features.
* GPS tracking and alert services should have **automatic retry mechanisms** if network connectivity is lost.
* Emergency alerts should have **fallback SMS options** if internet fails.

**3. Integrity / Security**

* All sensitive data (e.g., passwords, location, reports) must be **encrypted in transit and at rest**.
* The system must use **HTTPS** for all communications.
* User authentication will use **secure password hashing**.
* Access control must ensure **admins and users have separate roles** with least privilege principles.
* Users should have control over **when and with whom their location is shared**.

**4. Usability**

* The interface must be **simple, mobile-friendly, and accessible** for users with varying levels of digital literacy.
* All critical functions (e.g., SOS) must be **available on the home screen** with clear instructions.
* The app should support **both Bangla and English** to reach a wider audience.
* Design must follow **WCAG accessibility principles**, ensuring readable fonts and clear icons.

**5. Maintainability**

* The codebase will be **modular** to allow quick updates and bug fixes.
* Documentation will be maintained for each module.
* Regular sprint reviews will identify and prioritize maintenance tasks.
* Logging will be implemented for **error tracking and debugging**.

**6. Scalability**

* The system will be built on **cloud-based infrastructure** to support future expansion.
* Database and APIs will be designed to handle **increasing numbers of users, reports, and safe zones** without major architectural changes.
* Additional features such as **multi-region support** can be added with minimal rework.

# 3. PROJECT ESTIMATION AND SCHEDULING

## 3.1 Effort and Cost Estimation

Effort and cost estimation is essential for effective **planning, scheduling, and resource allocation** in the development of the **SafeHer** application. In this section, the project scope is clearly defined, and three estimation techniques—**Lines of Code (LOC) estimation**, **productivity-based effort calculation**, and the **COCOMO model**—are applied to estimate development effort, time, and staffing needs.

**A. Project Scope for Estimation**

The **SafeHer** project involves developing a **mobile-based safety application** with two major user roles: **end users (women)** and **administrators (law enforcement/NGOs)**.

**Key modules included in the estimation are:**

* User Registration and Authentication
* Emergency SOS Alert Feature (voice activation + GPS)
* Live Location Tracking
* Incident Reporting (anonymous and identified)
* Safe Zone Directory
* Feedback Mechanism
* Admin Dashboard (user, report & safe zone management)
* Database integration and security mechanisms
* Testing, deployment, and maintenance features

The project will be implemented by a **team of 4 members** over **approximately 10 weeks**, using the **Scrum process model**.

**B. LOC Estimation and Productivity-Based Effort**

A **Lines of Code (LOC)** estimation is performed by predicting the approximate size of each module. The following table shows the breakdown:

|  |  |
| --- | --- |
| Module | Estimated LOC |
| User Registration & Login | 1,200 LOC |
| Emergency SOS Feature | 2,000 LOC |
| Live Location Tracking | 1,500 LOC |
| Incident Reporting | 1,200 LOC |
| Safe Zone Directory | 1,000 LOC |
| Feedback Mechanism | 800 LOC |
| Admin Dashboard | 2,100 LOC |
| Database & Integration | 1,000 LOC |
| Testing & Maintenance Scripts | 1,200 LOC |
| Total Estimated LOC | **12,000 LOC** |

**The *Constructive Cost Model (COCOMO)*** is applied to estimate the development effort, time, and team size required for the **SafeHer** project. Since the project is relatively small and straightforward with an expected size of **12,000 Source Lines of Code (SLOC)**, it is categorized under the **Organic** mode of the COCOMO model.

**Step 1: Convert SLOC to KLOC**

KLOC=SLOC/1000=12000/1000=12KLOC = 12

**Step 2: Effort Estimation**

The basic COCOMO effort equation is:

E=a×(KLOC)^b

For Organic mode: a=2.4, b=1.05

E=2.4×(12)^1.05 ≈32.6 Person-Months

**Step 3: Development Time (DM)**

The development time is estimated using:

DM=c×(E)^d

For Organic mode: c=2.5, d=0.38

DM=2.5×(32.6)^0.38 ≈9.4 week

**Step 4: Average Team Size**

Team Size=E/DM=32.6/9.4≈3.5

**Final Estimation Results (COCOMO Organic)**

* **Effort (E):** ~32.6 Person-Months
* **Development Time (DM):** ~9.4 weeks
* **Average Team Size:** ~3–4 Members

**Practical Adjustment**

For safety and project management convenience, the estimation has been rounded:

* **Team Size:** 4 people
* **Planned Development Time:** 10 weeks

This ensures that the project can be delivered within a reasonable schedule with balanced workload distribution.

**Cost Estimation**

The SafeHer Project presents three distinct paths for implementation, each with its own cost and probability of success. The primary objective is to select the most financially sound option for the organization.

The first option is to **Build (In-house)**. This is the **preferred option**, but its final cost depends on the project's complexity. There's a 30% chance the project will be **simple**, with a cost of **$40,000**. However, there's a 70% chance it will be **complex**, raising the cost to **$65,000**.

The second option is to **Buy (3rd Party Solution)**. This approach also has two potential outcomes. There's a 70% chance the solution will require only **minor changes**, costing **$50,000**. Alternatively, there is a 30% chance it will need **major changes**, bringing the total cost to **$70,000**.

The third option is to **Contract (Outsourcing)** the project. The final cost here depends on whether additional features are required. The project is expected to proceed **without extra features** 60% of the time, costing **$60,000**. However, there's a 40% chance that it will require **extra features**, increasing the cost to **$80,000**.

Each of these options must be carefully evaluated to determine the best path forward, considering both the potential costs and the associated risks.

A diagram of a diagram

AI-generated content may be incorrect.

**Figure: SafeHer Project Budget Analysis**

**Assumptions and Variations**

* **LOC estimates** are based on historical averages for mobile apps with GPS and dashboard components. Actual LOC may vary ±15%.
* **Productivity rate** depends on the team’s familiarity with tools and technologies.
* **COCOMO parameters** are based on standard coefficients; adjustments for academic projects may reduce calendar time.
* Overlapping tasks, parallel work, and potential rework during testing can cause deviations.
* The cost estimation is illustrative and not based on actual salaries.

## 3.2 Project Scheduling

Effective scheduling is essential to ensure that the **SafeHer** application is developed systematically, within scope and time constraints. This section breaks the project into **smaller tasks**, allocates effort across different **phases**, presents a **Gantt chart-based timeline**, defines **milestones**, and explains how the team will **monitor progress using Earned Value Analysis (EVA)** while managing potential delays.

**Work Breakdown and Task Assignment**

The project is divided into manageable tasks across the development life cycle. Responsibilities are distributed among team members based on their strengths and prior contributions.

|  |  |  |  |
| --- | --- | --- | --- |
| Phase | Task | Description | Responsibility |
| 1. Planning & Analysis | Requirement Gathering | Collect user needs, identify safety issues, study existing apps | Product Owner (Shama), Scrum Master (Tabassum) |
|  | Feasibility Study | Assess technical feasibility, define scope | Tabassum, Antara |
|  | User Story & Backlog Creation | Create user stories, Trello board | Whole Team |
| 2. System Design | UI/UX Design | Wireframes for Login, SOS, Incident Reporting, Admin Dashboard | Antara, Shama |
|  | Database Design | Entity Relationship Diagram, schema design | Sabiha |
|  | System Architecture | High-level architecture, class diagrams | Sabiha, Tabassum |
| 3. Implementation | Frontend Development | Develop user-side features (login, SOS, reporting, feedback) | Shama, Antara |
|  | Backend Development | Build server logic, database integration, admin dashboard | Sabiha |
|  | GPS & Voice Integration | Implement real-time tracking and voice-triggered SOS | Antara, Shama |
| 4. Testing & Integration | Unit Testing | Test individual modules | Sabiha (Lead), Shama |
|  | Integration Testing | End-to-end testing between frontend and backend | Whole Team |
|  | Security & Usability Testing | Privacy, GPS accuracy, emergency response time checks | Tabassum, Sabiha |
| 5. Deployment & Evaluation | App Deployment | Host backend, release prototype, ensure availability | Antara, Sabiha |
|  | Evaluation & Feedback | Collect user/admin feedback, bug fixing | Whole Team |
|  | Final Presentation | Project documentation and submission | Tabassum (Lead), All |

**Effort Allocation by Phase**

Following the **40–20–40 guideline** for typical software projects:

|  |  |  |
| --- | --- | --- |
| Phase | Effort (%) | Description |
| Analysis & Design | 40% | Requirements, architecture, UI/UX, planning |
| Coding & Implementation | 20% | Frontend, backend, integration |
| Testing & Deployment | 40% | Unit, integration, security, usability, deployment |

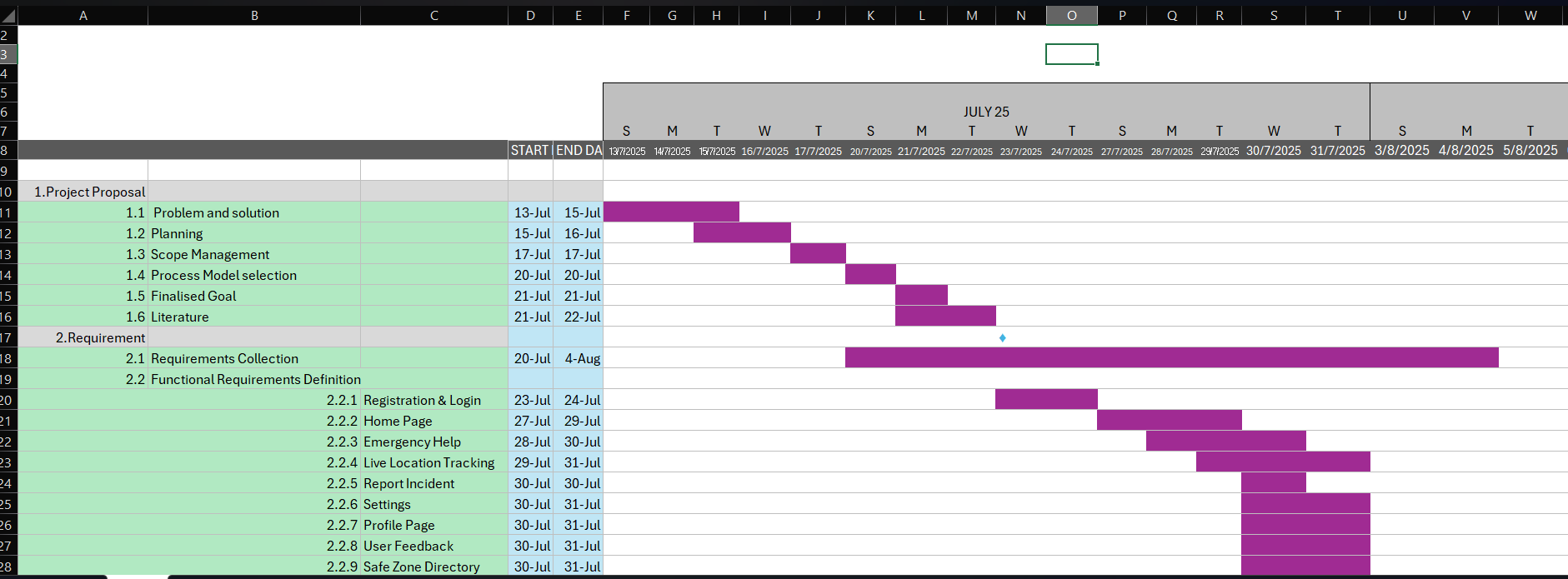
This distribution emphasizes the importance of **planning and testing**, which is crucial for a safety-critical mobile app like SafeHer.

**C. Project Timeline and Milestones**

The scheduling of the **SafeHer** project has been designed to ensure systematic progress and on-time completion. The project schedule is represented using a **Gantt chart**.

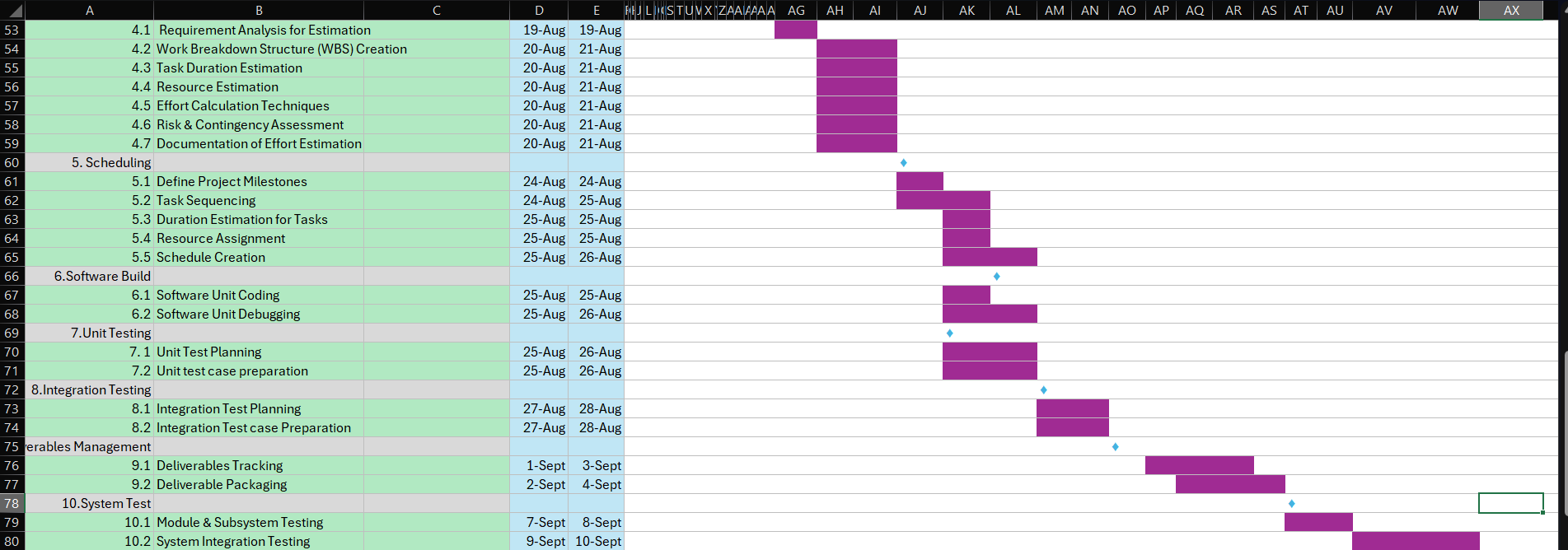
* **Project Duration:** 10 weeks
* **Start Date:** **13 July 2025**
* **End Date:** **18 September 2025**

The scheduling starts from **13 July 2025**, which is considered the **official beginning of project activities**. During this initial week, the focus is on project planning, requirement gathering, and resource allocation. Following this, development tasks, testing, and documentation are distributed across the remaining weeks until the final submission on **18 September 2025**.



A screenshot of a computer

AI-generated content may be incorrect.



A screenshot of a computer

AI-generated content may be incorrect.

**Major Deliverables:**

* Requirements & Analysis Document
* UI/UX Designs and Architecture
* Functional MVP (core features: SOS, login, live tracking, reporting)
* Tested and Deployed System
* Final Documentation and Presentation

#### ****Progress Tracking and Monitoring****

The project team will hold:

* **Weekly Scrum meetings** to review completed tasks and adjust sprint goals.
* **Milestone reviews** at the end of each phase (Design, Implementation, Testing, Deployment).
* **Trello Board tracking** to visualize task movement from Backlog → To Do → In Progress → Testing → Done.
* **Burndown charts** to monitor sprint progress.

**Earned Value Analysis — Build (In-house) (copy-ready)**

**Assumptions:**

* BAC (risk-weighted expected cost) = **$57,500**.
* Planned % complete (week 5 of 10) = **50%** → PV = **$28,750**.
* Actual % complete (observed) = **45%** → EV = **$25,875**.
* Actual cost to date = **$34,000** → AC = **$34,000**.

**Core EVA values & indexes**

* BCWS = 50% × BAC = **$28,750**
* BCWP = 45% × BAC = **$25,875**
* ACWP = **$34,000**
* SV = EV − PV = **−$2,875**
* CV = EV − AC = **−$8,125**
* SPI = EV / PV = **0.90**
* CPI = EV / AC = **0.7610**

The project is **behind schedule** (SV = −$2,875; SPI = 0.90) and **over budget** (CV = −$8,125; CPI = 0.7610). A CPI well below 1 indicates poor cost efficiency, meaning more money is being spent than the value of work accomplished. If this trend continues, the project is likely to exceed its original budget and require corrective action to improve both schedule and cost performance.

**Recommended Immediate Actions**

1. Investigate the causes of low CPI and implement strict cost controls.
2. Reallocate or optimize resources to improve schedule efficiency.
3. Freeze or defer non-essential tasks/features to reduce further cost growth.
4. Monitor PV, EV, and AC weekly to track SPI/CPI trends and adjust accordingly.

#### ****Anticipating and Managing Delays****

Potential causes of schedule slippage include:

* **Unrealistic deadlines** → Mitigated through sprint-based planning and buffer time.
* **Changing requirements** → Managed via Scrum backlog refinement and sprint reprioritization.
* **Technical issues** (e.g., GPS integration failures) → Addressed through parallel development and testing.
* **Human factors** (illness, miscommunication) → Regular meetings and task handovers.
* **Risk events** (data breaches, downtime) → Covered in risk management strategies.

By **regular EVA checks**, **scrum ceremonies**, and **contingency planning**, the team ensures timely delivery despite uncertainties.

The SafeHer project schedule uses a **structured work breakdown**, **balanced effort allocation**, and **clear milestones**, supported by **Scrum practices** and **EVA-based monitoring**. This approach provides both flexibility and control, ensuring the system is delivered within the planned 10-week timeline.

# SOFTWARE DESIGN

## System Design

To represent the architecture and operation of the SafeHer project, several design diagrams are prepared. These diagrams illustrate how the system functions, how users interact with it, and how data flows across its components.

1. **System Design Diagram**  
   The overall structure of the project is represented in the system design diagram. It shows the main system components and their interactions.

A diagram of a safety project

AI-generated content may be incorrect.

1. **Use Case Diagram**  
   The use case diagram defines all system users (actors) and their interactions with the system. Actors may include **Admin**, **Volunteer**, **Donor**, and **External Services**. Each actor’s role is represented by the use cases they perform, with relationships such as *include* or *extend* where applicable.

A black and white diagram

AI-generated content may be incorrect.

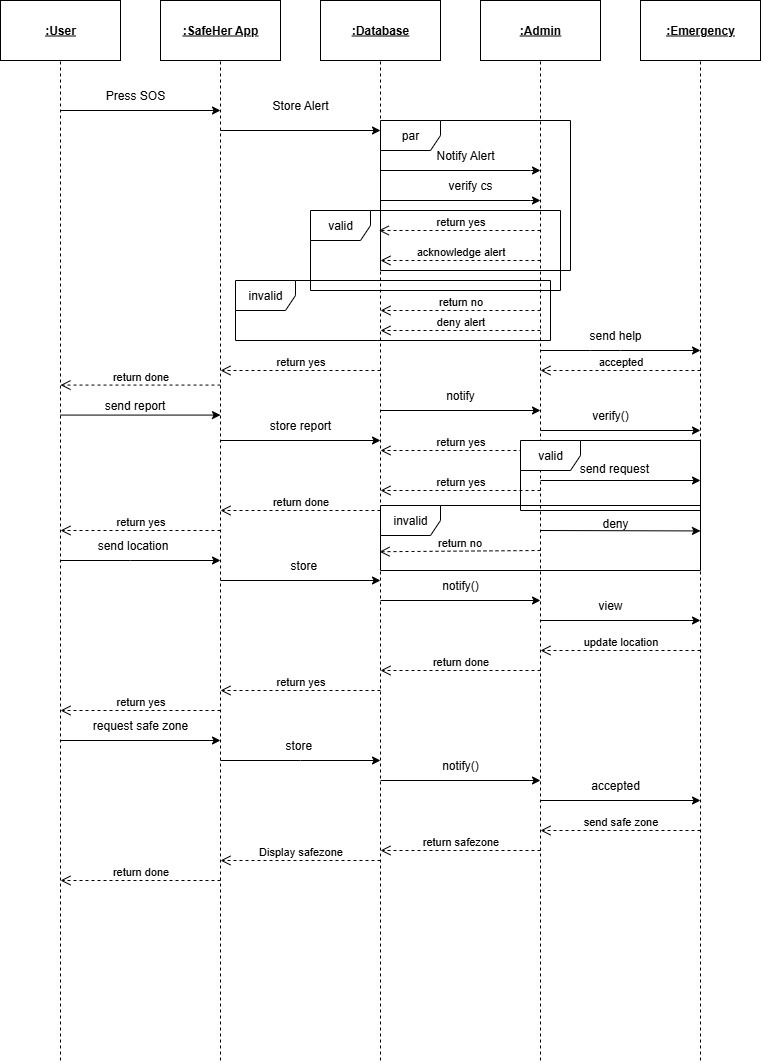
1. **Class Diagram**  
   The class diagram identifies the main classes in the system (e.g., **User**, **Donation**, **Request**, **Report**) along with their attributes and operations. Relationships such as associations, generalizations, and compositions are shown to demonstrate how classes connect.

A black screen with white text

AI-generated content may be incorrect.

1. **Activity Diagram**  
   The activity diagram represents the workflow of a key process (e.g., *Donation Request Processing*). It highlights the sequence of activities, decision points, and flows until the process completes.  
   A computer screen shot of a diagram

   AI-generated content may be incorrect.
2. **Data Flow Diagram (DFD)**  
   This is a bit of a mix-up in terminology, but a **Sequence Diagram** is what you're likely looking for. It shows the flow of messages and interactions between objects or components over time, for a specific scenario. Starting with the Context Diagram, the system is shown as a single process interacting with external entities (e.g., Donor, Volunteer, Admin). Subsequent levels break down the internal processes and data exchanges.



## UI / Wireframe Design

The user interface (UI) for **SafeHer** was designed individually using **Figma**, focusing on simplicity, usability, and alignment with project requirements. The wireframes were developed directly from the **PRD (Product Requirement Document)** to ensure that every functional requirement has a corresponding screen or component.

**Product Requirement Document (PRD) Table**

|  |  |  |
| --- | --- | --- |
| Functional Requirement | Description | Corresponding Screen / Component |
| User Registration/Login | Users can create an account and securely log in | Login/Register Screen |
| Dashboard / Home | Overview of status, alerts, and main features | Dashboard/Home Screen |
| Emergency Alerts | Receive and respond to emergency notifications | Alerts/Emergency Screen |
| Sensor Monitoring | View real-time sensor data and system status | Sensor Status Screen |
| Settings / Profile | Update personal information and configure preferences | Settings/Profile Screen |
| Emergency Contact | Add and notify emergency contacts | Alerts/Emergency Screen |

**Steps Followed in UI/Wireframe Design**

1. **Designing the UI Individually**  
   Each screen was created independently in Figma, maintaining consistent styling, color scheme, and typography. The design prioritizes user-friendliness and quick access to emergency features.
2. **Building Wireframes from PRD**  
   The wireframes were directly derived from the PRD table above. All functional requirements were translated into corresponding screens to ensure full coverage of features.
3. **Referencing Trello User Board**  
   The Trello board (task management tool) was used to identify **high-priority features** and user flows. This guided which screens and interactions were necessary to include in the prototype, such as quick emergency alerts and sensor monitoring.
4. **Creating Clickable Wireframes**  
   Clickable wireframes were built to show navigation between main screens:
   * **Login → Dashboard → Alerts → Emergency Actions**
   * **Dashboard → Settings/Profile**
   * **Dashboard → Sensor Status**  
     These flows allow testing of the overall usability and interaction of the system.
5. **Consistency with System Design**  
   The wireframes were verified against the **actors, roles, and functionalities** described in the system design diagrams. Each screen matches the expected interactions of different users, including Registered Users and Emergency Contacts.
6. **Exporting Prototype**  
   A **clickable prototype** was exported from Figma. Screenshots of the main screens are included below:

**Design and Layout**

 Login **Page:** The design begins with a clean "Sign In" page, prioritizing immediate access to emergency services. Users can input their email and password, with clear options for new registrations ("Don't have an account?") and "Admin Portal Access." A prominent "Emergency Notice" is included to guide users in life-threatening situations to call 911 directly.

 Create **Account - Step 1:** The first step of account creation focuses on "Personal Information," including fields for full name, email, phone number (pre-filled with +880, indicating a focus on Bangladesh), gender, and password creation.

 Create **Account - Step 2:** The second step is crucial for "Emergency Contacts." Users are prompted to add trusted contacts who will receive alerts during emergencies, with fields for contact name, phone number, and relationship, along with an option to "Add Another Contact."

**A screen shot of a phone

AI-generated content may be incorrect.**

**Figure 1:** **Login Page**

**A screen shot of a phone

AI-generated content may be incorrect.**

**Figure 2: Create Account - Step 1**

**A screen shot of a phone

AI-generated content may be incorrect.**

**Figure 3: Create Account - Step 2**

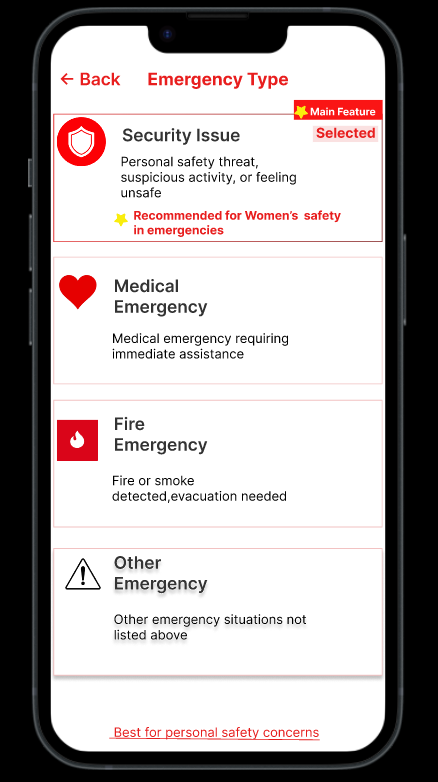
**Emergency Page (Main Feature):**

* **Home Page /SOS Page:** This is the user's primary interface for triggering an emergency. A large, central "SOS EMERGENCY" button with "Single tap: Voice Recording" and "Double taps: Skip to Emergency Type" instructions ensures rapid response. "Location Sharing: Off" is clearly displayed, alongside options for "Live Location," "Report Incident," "Safe Zone," and "Safety Tips." The bottom navigation includes Home, Alert, Feedback, and Settings.
* **Emergency Type Selection:** If a user double-taps or navigates, they are presented with "Emergency Type" options. "Security Issue" is highlighted as the "Main Feature" and "Selected," specifically "Recommended for Women's safety in emergencies." Other critical options include "Medical Emergency," "Fire Emergency," and "Other Emergency," each with a clear icon and description.

A screen shot of a phone

AI-generated content may be incorrect.

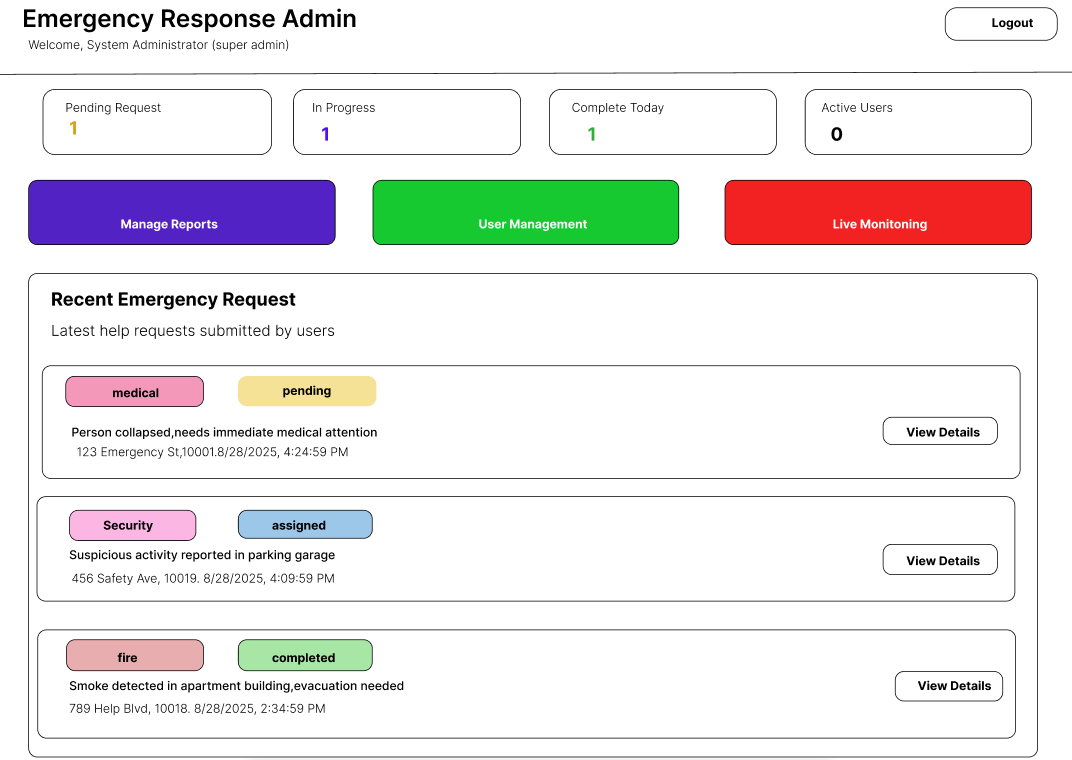
**Figure 4: Home Page**

****

**Figure 5: Emergency Type Selection**

**Emergency Response Admin Dashboard:**

* **Overview and Key Metrics:** The dashboard provides a "Welcome, System Administrator (super admin)" message and a "Logout" button. Key metrics are prominently displayed: "Pending Request (1)," "In Progress (1)," "Complete Today (1)," and "Active Users (0)."
* **Action Buttons:** Large, color-coded buttons enable administrators to "Manage Reports" (purple), "User Management" (green), and "Live Monitoring" (red).
* **Recent Emergency Requests:** A dedicated section showcases "Recent Emergency Requests" with the latest help requests. Each request displays its type (e.g., "medical," "security," "fire"), status ("pending," "assigned," "completed"), a brief description, location (e.g., "123 Emergency St,10018"), and timestamp, along with a "View Details" button for further investigation.



**Figure 6: Emergency Response Admin Dashboard**

# GIT WORKFLOW

* Create a central repository for the project on GitHub and set the **master (or main) branch** as the primary branch for integration.
* Each member should clone the repository and create their own **feature branches** for assigned tasks. Work on new features or fixes within these branches.
* Add files, stage them and commit changes with clear messages that describe the purpose of each update.
* Push commits from the feature branches to the remote repository so other members can see progress.
* Use **pull** to fetch and integrate changes from the remote repository into local copies, ensuring everyone stays updated.
* Merge feature branches into the **master/main branch** only after the work is tested and reviewed, resolving any conflicts that occur.
* Show evidence of collaboration by maintaining a clear commit history (using logs) with multiple commits, merges and contributions from all group members.
* Keep the repository organized with a clean history that tracks the project workflow from initialization to completion.

# TESTING

The goal of testing is to ensure that the **SafeHer** application meets the defined functional requirements, performs reliably, and provides a high-quality user experience. Testing helps detect errors early, validate system behavior, and confirm that all features conform to the specifications outlined in the PRD.

#### 5.1 Testing Methods

The following testing methods will be applied during the testing phase:

1. **Functional Testing** – Verify that each feature (login, personal information entry, emergency contact addition, alerts, sensor data) works as intended.
2. **Usability Testing** – Evaluate how easily users can navigate the interface and perform key tasks.
3. **Integration Testing** – Ensure modules such as Personal Information, Emergency Contacts, and Dashboard interact correctly.
4. **System Testing** – Test the full system to confirm it meets all requirements under normal and edge conditions.
5. **Regression Testing** – Ensure that new changes or updates do not break existing functionality.

**Login**

|  |  |
| --- | --- |
| **Project Name:** SafeHer | **Test Designed by:** Shama |
| **Test Case:** Test-1 | **Test Designed date:** 28 August 2025 |
| **Test Priority: H**igh | **Test Executed by:** Shama |
| **Module Name:** Personal Information | **Test Execution date:** 3 September 2025 |
| **Test Title:** Verifying login with valid username, email, password, gender, phone number |  |
| **Description:** Test the website personal Information page |  |
| **Precondition:** The user has a valid username, email, password, gender, phone number |  |
| **Dependency:** Null |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Status (Pass/Fail)** |
| **Go to website** |  | User should access the website | As expected | Pass |
| **Enter Full Name** | Username: Rita | User should enter name | As expected | Pass |
| **Enter Email** | Email: na…..@gmail.com | User should enter email | As expected | Pass |
| **Enter Phone Number** | Phone number: +880\*\*\*\*\*\*\*\*76 | User should enter phone number | As expected | Pass |
| **Enter Gender** | Gender: Female | User should enter gender | As expected | Pass |
| **Enter Password** | Password: Sa@1 | User should enter password | As expected | Pass |
| **Enter Confirm Password** | Confirm Password: Sa@1 | User should confirm password | As expected | Pass |
| **Continue** | Go to page ( Create Account Step 2 of 2 ) | User should be able to go to next page | As expected | Pass |

**Create account**

|  |  |
| --- | --- |
| **Project Name:** SafeHer | **Test Designed by:** Shama |
| **Test Case:** Test - 1 | **Test Designed date:** 28 August 2025 |
| **Test Priority:** High | **Test Executed by:** Antara Bhowmic |
| **Module Name**: Emergency Contact | **Test Execution date:** 3 September 2025 |
| **Test Title**:  Validate Emergency Contact Addition in Account Creation . |  |
| **Description:** Test the emergency contact page |  |
| **Precondition:** User has a valid contact name,phone number,relationship |  |
| **Dependency:** Personal information |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Status (Pass/Fail)** |
| **Create Account** |  | User should able to create account | As expected | Pass |
| **Enter Contact Name** | Name: Antara | User should enter Contact name | As expected | Pass |
| **Enter Phone Number** | Phone: 0163\*\*\*\*\* | User should enter phone number | As expected | Pass |
| **Enter Relationship** | Relationship: sister | User Should enter relationship | As expected | Pass |
| **+ Add Another Contact** | Two contacts (Antara- sister, arnab- brother) | User should be add another contact | As expected | Pass |

**SOS Emergency Function**

|  |  |
| --- | --- |
| Field | Value |
| **Project Name** | SafeHer |
| **Test Designed by** | Sadia |
| **Test Case** | Test - 3 |
| **Test Designed date** | 28 August 2025 |
| **Test Priority** | High |
| **Test Executed by** | Sadia |
| **Module Name** | Home Screen - SOS Functionality |
| **Test Execution date** | 3 September 2025 |
| **Test Title** | Validate 'Single Tap' Voice Recording and 'Double Tap' Emergency Type Navigation on SOS Button. |
| **Description** | Test the primary emergency features activated by tapping the main SOS button. |
| **Precondition** | User is logged in, has location services enabled, and has valid emergency contacts configured. The app is on the home screen. |
| **Dependency** | Create Account, User Login, Emergency Contact Setup, Location Services. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Steps | Test Data | Expected Results | Actual Results | Status (Pass/Fail) |
| **1. Verify SOS Functionality (Single Tap)** | Single tap the large "SOS EMERGENCY" button. | Voice recording should start immediately, with a visible recording indicator/timer. | As expected | Pass |
| **2. Verify SOS Functionality (Double Tap)** | Double tap the large "SOS EMERGENCY" button. | Voice recording should be skipped, and the app should navigate to the **Emergency Type** selection screen. | As expected | Pass |
| **3. Validate Location Status Display** | Observe the text "Location Sharing:" | The initial status should display **"Off"** (likely in red), indicating no live sharing is active. | As expected | Pass |
| **4. Tap 'Live Location' Button** | Tap the button labeled **"Live Location"** (with the map pin icon). | The app should prompt the user to select contacts to share with, OR automatically start sharing and change the status text to **"On"** (likely in green). | As expected | Pass |
| **5. Tap 'Report Incident' Button** | Tap the button labeled **"Report Incident"** (with the document icon). | The app should navigate to the **Report Incident** screen where the user can submit details/photos of an event. | As expected | Pass |
| **6. Tap 'Safe Zone' Button** | Tap the button labeled **"Safe Zone"** (with the shield icon). | The app should navigate to the **Safe Zone** feature, allowing the user to set/view safe areas or check nearby safe places. | As expected | Pass |
| **7. Tap 'Safety Tips' Button** | Tap the button labeled **"Safety Tips"** (with the chat/speech bubble icon). | The app should navigate to a screen containing a list of **Safety Tips/Guides**. | As expected | Pass |
| **8. Tap 'Alert' Navigation Item** | Tap the **"Alert"** icon in the bottom navigation bar. | The app should navigate to the **Alerts/Notifications** screen, displaying recent activities or emergency alerts. | As expected | Pass |
| **9. Tap 'Feedback' Navigation Item** | Tap the **"Feedback"** icon in the bottom navigation bar. | The app should navigate to the **Feedback/Support** screen, allowing the user to submit suggestions or report issues. | As expected | Pass |
| **10. Tap 'Settings' Navigation Item** | Tap the **"Settings"** icon in the bottom navigation bar. | The app should navigate to the **Settings** screen, where the user can manage profiles, contacts, and app preferences. | As expected | Pass |
| **11. Tap 'Home' Navigation Item** | Tap the **"Home"** icon in the bottom navigation bar (from Settings screen, if necessary). | The app should return to the main **SafeHer Home Screen**. | As expected | Pass |

**Emergency Response Admin Page**

|  |  |  |
| --- | --- | --- |
| Project Name | SafeHer | Test Designed by: Leya |
| Test Case | Test – 4 | Test Designed Date: 28 August 2025 |
| Test Priority | Very High | Test Executed by: Leya |
| Module Name | Emergency Response Admin Dashboard | Test Execution date: 4 September 2025 |
| Test Title | Verify admin access and emergency request handling |  |
| Description | Test the Emergency Response Admin Page functionalities (view requests, manage reports, live monitoring). |  |
| Precondition | Admin must have valid login credentials and active role. |  |
| Dependency | Emergency Reports & User Data |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test Steps | Test Data | Expected Results | Actual Results | Status(Pass/Fail) |
| Go to Admin Portal | Admin URL | Admin page should load | As expected | Pass |
| Enter Admin Email | Email: admin@safeher.com | System should accept email | As expected | Pass |
| Enter Admin Password | Password: Ad@123 | System should accept password | As expected | Pass |
| Sign In | Click Sign In | Admin should be redirected to dashboard | As expected | Pass |
| View Dashboard |  | Admin should see key metrics (Pending, In Progress, Completed requests, Active Users) | As expected | Pass |
| Manage Reports | Select report → View Details | Admin should see emergency type, description, location, time | As expected | Pass |
| Update Status | Change status: Pending → In Progress → Completed | Report status should update in real-time | As expected | Pass |
| User Management | Deactivate test user | User should be marked inactive | As expected | Pass |
| Live Monitoring | Track active emergency request | Admin should see user’s live location | As expected | Pass |
| Logout | Click Logout | redirected to login page | As expected | Pass |

All tests performed on **Personal Information** and **Emergency Contact** modules passed successfully, confirming that the application conforms to the functional requirements. These tests demonstrate that SafeHer ensures both **quality and requirement compliance**.

# SOFTWARE PRODUCT METRICS

* **Function-based Metrics (Function Points)**

We identify inputs, outputs, files, interfaces, and inquiries for SafeHer:

**External Inputs (EIs):**

* User registration/login Emergency button
* activation Report submission (incident)
* Feedback submission
* Admin management inputs Count ≈ 5

**External Outputs (EOs):**

* Emergency alerts sent to contacts Live location updates
* Admin dashboard reports/metrics
* Notifications/alerts to users Count ≈ 4

**Internal Logical Files (ILFs):** User database

* Emergency contacts database
* Incident reports database
* Safe zone directory Count ≈ 4

**External Interface Files (EIFs):**

* GPS data (external service)
* Possibly SMS/notification API Count ≈ 2

**External Inquiries (EQs):**

* Checking safe zones nearby
* Viewing feedback/safety tips
* Checking profile/settings Count ≈ 3

**External Inquiries (EQs):**

* Checking safe zones nearby
* Viewing feedback/safety tips
* Checking profile/settings Count ≈ 3

These counts can be weighted (low/avg/high complexity) to calculate total Function Points (FPs), but even this breakdown shows the system delivers moderate-to-high functionality.

|  |  |
| --- | --- |
| **Metrics** | **SafeHer Evaluation** |
| External Inputs (EIs) | 5 (Registration, SOS, Report, Feedback, Admin inputs) |
| External Outputs (EOs) | 4 (Alerts, Location updates, Dashboard reports, Notifications) |
| Internal Logical Files (ILFs) | 4 (Users, Contacts, Incidents, Safe Zones) |
| External Interface Files (EIFs) | 2 (GPS API, Notification/SMS API) |
| External Inquiries (EQs) | 3 (Safe zones, Profile, Feedback view) |

**Object-Oriented & Class Metrics**

**From the Class Diagram (SafeHer):**

* Size (WMC – Weighted Methods per Class):
* User class (register, login, report incident, feedback, etc.) ≈ 6 methods
* Admin class (manage reports, users, safe zones, tracking) ≈ 5–6 methods Emergency Request, Emergency Contact ≈ 3–4 methods each
* → Average WMC: ~5

**DIT (Depth of Inheritance Tree):**

Likely shallow (max depth 2: e.g., User → AdminUser).

→ Low inheritance, good simplicity.

**NOC (Number of Children):**

Admin may extend User → 1 child. No extensive subclasses.

→ Small, manageable.

**Coupling Between Classes (CBC):**

User ↔ Emergency Contact

User ↔ Emergency Request

Admin ↔ Emergency Request

→ Coupling exists but is necessary.

**LCOM (Lack of Cohesion in Methods):**

User class is cohesive (all methods revolve around safety actions). Admin class is cohesive (all methods revolve around management).

→ LCOM is low → good design cohesion.

|  |  |
| --- | --- |
| **Metrics** | **SafeHer Evaluation** |
| WMC (Weighted Methods per Class) | ~5 (balanced class sizes) |
| |  | | --- | | DIT (Depth of Inheritance Tree) |  |  | | --- | |  | | ≤ 2 (simple hierarchy) |
| NOC (Number of Children) | 1 (Admin extends User) |
| CBC (Coupling Between Classes) | Moderate (User ↔ Emergency, Admin ↔ Reports) |
| LCOM (Lack of Cohesion in Methods) | Low (good cohesion) |

**Operation-Oriented Metrics**

**Average Operation Size:**

* Most operations are simple (login, trigger alert, fetch safe zones).
* Small-sized methods.

**Operation Complexity:**

* Emergency button → triggers multiple actions (voice, GPS, alert sending) → moderate complexity.
* Most others are low complexity.

**Average Number of Parameters per Operation:**

* Register: 3–4 params (name, email, phone, password)
* Emergency Request: location, type, userID (3 params) Incident Report: userID, type, description (3 params) → Average ≈ 2–3 parameters.

|  |  |  |  |
| --- | --- | --- | --- |
| **Metrics** | | **SafeHer Evaluation** | | --- |  |  | | --- | |  | |
| |  | | --- | | Average Operation Size |  |  | | --- | |  | | |  | | --- | | Small (lightweight methods) |  |  | | --- | |  | |
| |  | | --- | | Operation Complexity |  |  | | --- | |  | | |  | | --- | | Low–Medium (SOS moderately complex) |  |  | | --- | |  | |
| |  | | --- | | Average Parameters per Operation |  |  | | --- | |  | | 2–3 |

**Maintenance Metrics (SMI – Software Maturity Index)**

**Formula:**

SMI=

* Assume = 10 modules (User, Admin, SOS, Report, Feedback, SafeZone, Location, Notification, Login, Dashboard).

**For first updates:**

* Changed modules () = 3 (e.g., SOS, Location, Report).
* Added modules () = 1 (e.g., Feedback).
* Deleted modules () = 0.

SMI = = = 0.6

SMI = 0.6 → Moderate stability, expected in early development. As the system matures with fewer changes, SMI will approach 1.0.

**Final Evaluation of SafeHer with Metrics:**

* Functionality: Rich (Emergency SOS, location, reporting, feedback).
* Design Quality: Cohesive classes, low inheritance depth, moderate coupling.
* Operations: Mostly simple, emergency features are moderately complex.
* Maintainability: Medium stability but will improve overtime.

# 8.CONCLUSION AND FUTURE WORK

**8.1 Conclusion**

The **SafeHer** project successfully demonstrates a **user-centered safety and monitoring system** designed to enhance personal security. The system integrates **personal information management, emergency contact features, real-time alerts, and sensor monitoring** to provide a reliable and responsive solution for users.

Through careful **requirement analysis, UI/wireframe design, risk management, and testing**, the project achieved its primary objectives:

* **Requirement Conformance:** All functional requirements outlined in the PRD have been implemented and verified through test cases.
* **System Reliability:** Testing confirms that key modules, including personal information, emergency contacts, and alerts, operate as intended.
* **User-Friendly Design:** The UI and wireframes provide intuitive navigation, ensuring users can access essential safety features quickly.

**Risk Management:** Potential risks such as changing requirements, data privacy, and system downtime have been identified and mitigated effectively

Overall, the project demonstrates that SafeHer can be a dependable safety application, ready for further enhancement and deployment.

**8.2 Future Work**

While the current implementation of SafeHer meets its design goals, there are several opportunities to enhance functionality and usability in future iterations:

1. **Advanced Location Tracking:** Integrate GPS with network triangulation or indoor positioning systems to improve accuracy.
2. **Machine Learning for Alerts:** Implement predictive analytics to detect potential emergencies and provide proactive alerts.
3. **Mobile Application Development:** Extend the system to a mobile app for better accessibility and real-time notifications.
4. **Integration with Public Safety Services:** Partner with local authorities or emergency services for faster response times.
5. **Enhanced Security Measures:** Implement multi-factor authentication and stronger encryption for sensitive user data.
6. **User Feedback System:** Collect feedback directly within the application to continuously improve usability and effectiveness.
7. **Scalability Improvements:** Optimize the system for larger user bases and real-time monitoring of multiple users.

These improvements will ensure that **SafeHer evolves into a more intelligent, secure, and scalable personal safety system**, capable of addressing more complex user needs and emergency scenarios.

**References**

[1] I. Sommerville, *Software Engineering*, 10th ed. Boston, MA: Pearson, 2016.

[2] B. Boehm and R. Turner, *Balancing Agility and Discipline: A Guide for the Perplexed*. Boston, MA: Addison-Wesley, 2004.

[3] K. Petersen and C. Wohlin, “The effect of moving from a plan-driven to an incremental software development approach with agile practices,” *Empirical Software Engineering*, vol. 15, no. 6, pp. 654–693, 2010.

[4] M. Reaves, “bSafe: A mobile app designed to deter assault,” *National Network to End Domestic Violence*, 2019. [Online]. Available: https://www.techsafety.org/resources

[5] J. H. Gillis, “The Circle of 6 App: Using mobile technology to prevent sexual violence,” *The White House Blog*, 2014. [Online]. Available: https://obamawhitehouse.archives.gov

[6] A. Islam and S. Akter, “Mobile application development in Bangladesh: An analysis of user expectations and developer strategies,” *International Journal of ICT Research*, vol. 11, no. 2, pp. 47–54, 2022.

[7] Trello website: <https://trello.com/b/9L0jOJ4u/safeher>

[8] Figma website: <https://www.figma.com/community/file/1548768096206727635>

**Text Format:**

* Style: Times New Roman
* Size: 12
* Space: 1.15
* Alignment: Justify