SOFTWARE REQUIREMENTS SPECIFICATION

for

RUN BANGLADESH

Version 1.0

Prepared by : Saif Mohammed (2121913042)
Course: CSE327
Section: 07
Team: 01

Submitted to : Dr. Md. Sazzad Hossain
Professor

November 2, 2025

Contents

Revision History

Table 1: Revision History

Name	Date	Reason for Changes	Version
Saif Mohammed	2025-10-15	Created initial draft of the SRS document, including introduction and scope sections.	1.0
Saif Mohammed	2025-10-25	Added system features, use case diagrams, and functional/non-functional requirements.	1.1
Saif Mohammed	2025-11-02	Finalized document with UI mockups, testing details, glossary, and references for submission.	1.2

1 Executive Summary

Run Bangladesh is a comprehensive full-stack mobile application designed to digitalize and streamline marathon management across Bangladesh. Developed using **React Native** for Android and iOS platforms, the system integrates a **Node.js** backend and an **SQL** database managed through phpMyAdmin. To ensure reliability, security, and performance, the application incorporates the **Stripe API** for payment processing, **Docker** for containerized deployment, and **Jest** for automated testing. The project aims to provide an accessible, scalable, and user-friendly solution that enhances the experience of marathon participants, organizers, and volunteers through an integrated digital platform.

The application extends the existing functionalities of the Run Bangladesh website, offering enhanced accessibility and interactivity in a mobile context. Users can browse events, register, and make secure payments directly from their smartphones. Participants gain access to real-time GPS-based tracking during races, while organizers can monitor live results, manage events, and coordinate volunteers using the mobile dashboard. The interface, designed in **Figma**, ensures a consistent, intuitive, and visually appealing user experience optimized for mobile devices, supporting seamless interaction across different roles and activities.

In essence, **Run Bangladesh Mobile App** represents a modern transformation of traditional marathon management. It integrates technology-driven features such as secure digital payments, real-time analytics, and live leaderboards, fostering engagement and transparency. The solution not only promotes digital inclusivity within Bangladesh's growing running community but also establishes a sustainable model for managing future athletic and community events nationwide. This SRS outlines the complete functional, technical, and design specifications necessary to guide its successful implementation and deployment.

2 Introduction

2.1 Purpose

This Software Requirements Specification (SRS) document defines the requirements for the **Run Bangladesh Mobile Application (Version 1.0)**. It has been prepared as part of the **CSE327: Software Engineering** final course project at North South University.

The purpose of this document is to describe the functional and non-functional requirements for developing a full-stack, cross-platform mobile application that digitizes marathon management in Bangladesh. It serves as a single, authoritative reference for all stakeholders, including developers, project managers, testers, and end users, ensuring a shared understanding of the system's goals, constraints, and features.

This document will guide the project team throughout the software development life cycle (SDLC)—from design and implementation to testing, deployment, and maintenance—while maintaining traceability to user and business needs.

2.2 Scope

Run Bangladesh is a comprehensive mobile application designed to streamline marathon event management and participation across Bangladesh. It extends the existing Run Bangladesh web platform by introducing mobile-first interaction, real-time data synchronization, and improved accessibility for all stakeholders.

The mobile app enables participants to browse events, register, pay securely via the **Stripe API**, and track their performance in real time using GPS. Organizers can manage registrations, volunteers, event logistics, and live analytics through an integrated mobile dashboard. Volunteers can check in, view assignments, and communicate during events.

2.2.1 Primary Objectives

- Modernize marathon management through digital transformation.
- Ensure scalability and security for large-scale public events.
- Enhance participant engagement with real-time updates and media sharing.
- Provide an analytics-driven dashboard for event performance insights.

2.2.2 Key Features

- User registration and authentication (email/phone with OTP).
- Event browsing and category-based registration.
- Secure payment processing via Stripe integration.
- Real-time race tracking with GPS and route visualization.
- Live leaderboard, event results, and multimedia updates.
- Volunteer management and organizer dashboard.
- Data synchronization with the existing Run Bangladesh website.

2.2.3 Users and Stakeholders

- **Participants:** Register for events, make payments, and view live results.
- **Organizers:** Create and manage events, volunteers, and analytics.
- **Volunteers:** Check in, manage tasks, and assist during events.
- **Administrators:** Oversee overall system integrity and data management.

2.3 Overview

The remainder of this SRS document provides a detailed specification of the **Run Bangladesh Mobile Application**. The organization of subsequent chapters is as follows:

- **Chapter 2 – Overall Description:** Describes the product perspective, user classes, operating environment, and design constraints.
- **Chapter 3 – Requirements Modelling:** Defines functional and non-functional requirements, prioritized by importance.
- **Chapter 4 – Use Case Modelling:** Presents UML use case diagrams, templates, and scenarios for key system interactions.
- **Chapter 5 – Requirements Traceability Matrix:** Maps each functional requirement to its associated use case or module.
- **Chapters 6–10 – Supplementary Sections:** Include UI prototypes, glossary, resources, and appendices.

This structure ensures clear traceability between high-level objectives and system implementation while adhering to the IEEE 830-1998 documentation standard.

3 Requirements Modelling

3.1 Functional Requirements

- **User Registration/Login:** Participants, organisers and volunteers register and login via mobile (email/phone + OTP).
- **Event Browsing:** Users view upcoming, past, and featured events (based on website content). :contentReference[oaicite:3]index=3
- **Event Registration & Payment:** Users select event category, pay via Stripe in-app, receive digital bib/QR code.
- **Run Tracking:** During race day, participants enable GPS tracking via mobile app; organisers monitor live positions, splits.
- **Leaderboard & Results:** Live leaderboard during event and final results posted in app (mirroring website results page). :contentReference[oaicite:4]index=4
- **Notifications & Multimedia Feed:** Push notifications, event updates, photos/videos feed within app (from organisers).
- **Volunteer Management:** Volunteers receive assignments, check-in/out via mobile, view shift details.
- **Organiser Dashboard:** On mobile/tablet, organisers create/edit/publish events, manage registrations/payments, view analytics, route map, manage volunteers.
- **Synchronization:** App syncs with existing website backend so that web content and mobile content remain consistent.

3.2 Non-Functional Requirements (Prioritized on 3 Levels)

The non-functional requirements are categorized into three priority levels — **High**, **Medium**, and **Low** — based on their importance to system reliability, performance, and user experience.

3.2.1 High Priority

These requirements are essential for the correct functioning and reliability of the system.

- **Security:** All user and payment data must be protected using TLS encryption for data in transit, and sensitive information encrypted at rest. The Stripe API ensures PCI-compliant transactions.
- **Performance:** The app should launch within 3 seconds, and real-time tracking updates must occur with a latency below 500 ms. The system must handle a minimum of 1000 concurrent users during peak events.
- **Usability:** The user interface must be intuitive and optimized for mobile devices. The registration and booking flow should be completed within five screens, and accessibility must accommodate users with varied technical experience.

3.2.2 Medium Priority

These requirements enhance system scalability, robustness, and ease of maintenance.

- **Scalability:** The backend should support multiple concurrent events and thousands of participants, with architecture suitable for horizontal cloud scaling.
- **Availability:** The system should achieve at least 99.5% uptime during live events, with limited offline capabilities for viewing event information when network access is unavailable.
- **Maintainability:** The codebase should be modular and well-documented. Unit test coverage using Jest must reach at least 75%. The application should be containerized using Docker for seamless deployment and updates.

3.2.3 Low Priority

These requirements improve overall user satisfaction and compatibility across platforms but are not critical to system operation.

- **Portability:** The mobile app must be deployable on both Android and iOS using a shared React Native codebase, minimizing platform-specific dependencies.
- **Aesthetic Design:** The user interface should follow a clean, modern design consistent with the Run Bangladesh brand identity, though visual polish is secondary to functional stability and performance.

4 Use Case Modelling

4.1 Use Case Diagram

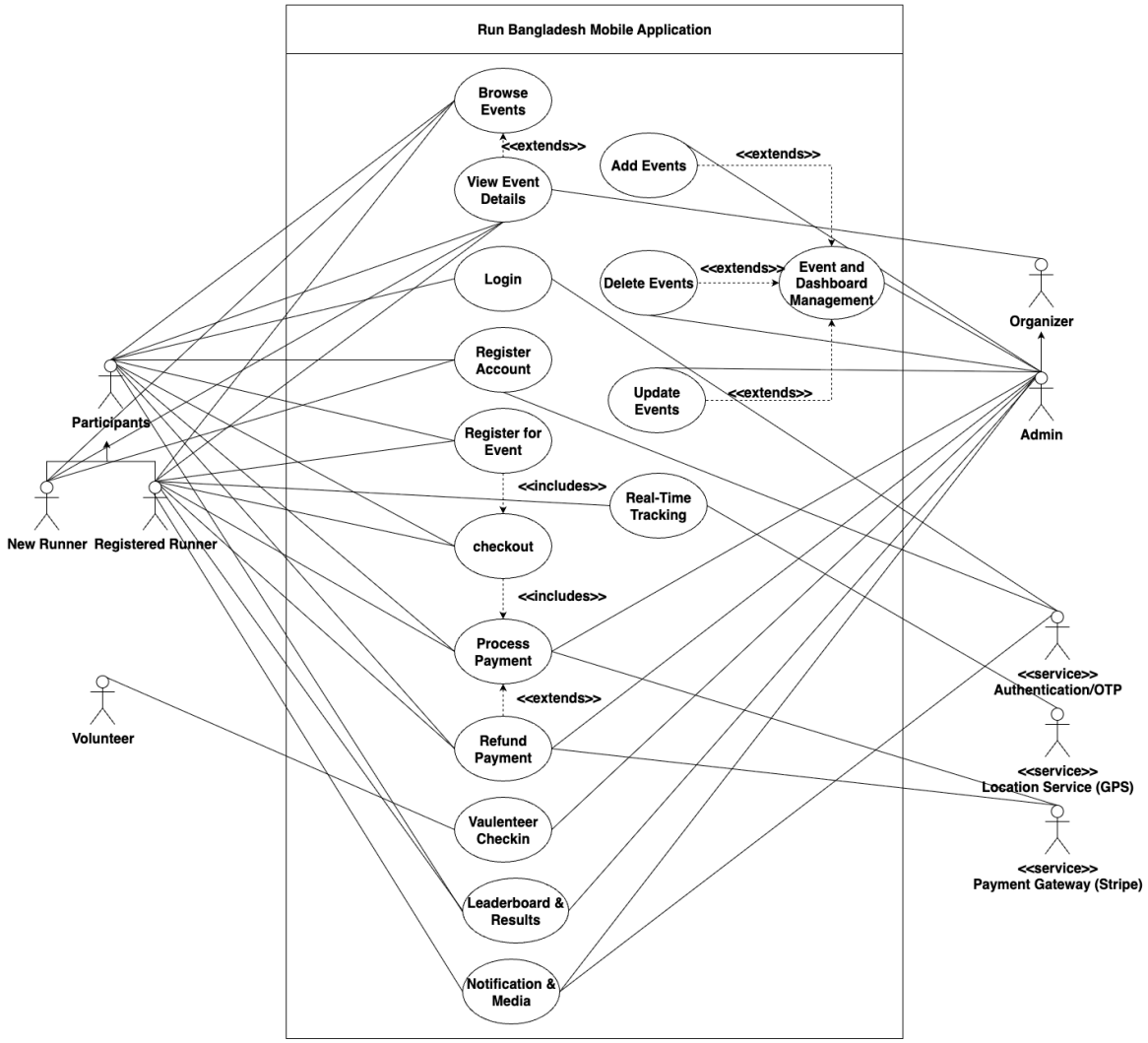

Figure 4.1: Run Bangladesh Mobile Application — UML Use Case Diagram

4.2 Use Cases Using Templates and Text Descriptions

This section describes the key use cases of the **Run Bangladesh Mobile Application** in tabular IEEE format. Each use case includes actors, preconditions, normal flow, alternative flows, and related non-functional requirements.

UC1: User Registration and Login

Use Case Name	User Registration and Login
Actors	New Runner, Registered Runner, Authentication/OTP Service
Preconditions	User has installed the mobile app and has an active internet connection.
Normal Flow	**Description:** • User opens the app and selects "Register" or "Login". • System prompts for email/phone and password. • System sends an OTP for authentication. • User enters OTP; system verifies and grants access. **Postconditions:** • User is authenticated and lands on the dashboard.
Alternative Flows and Exceptions	• Invalid/expired OTP → prompt retry or resend OTP. • Forgot password → launch password reset flow. • Existing user taps Register → redirect to Login.
Non-Functional Requirements	Secure auth (OTP), TLS in transit, encryption at rest, high availability, usability.

UC2: Event Browsing and View Details

Use Case Name	Event Browsing and View Details
Actors	Participant, Registered Runner
Preconditions	User is logged in (or guest for browse-only if permitted); events exist in the system.
Normal Flow	**Description:** • User opens "Browse Events". • System lists upcoming/past events with filters (date, city, distance). • User selects an event to view details (route map, fees, categories, rules). • User may proceed to registration. **Postconditions:** • Event details displayed; user can continue to register or exit.
Alternative Flows and Exceptions	• No events found → show empty state with "Notify me". • Network error → retry or offline cached list.
Non-Functional Requirements	Fast list rendering, responsive filtering, accessibility, graceful offline fallback.

UC3: Event Registration and Payment

Use Case Name	Event Registration and Payment
Actors	Participant, Payment Gateway (Stripe)
Preconditions	User is authenticated; selected event is open for registration; payment method available.
Normal Flow	**Description:** • User selects event category (e.g., 5K, 10K, 25K). • System collects participant info (age, emergency contact, shirt size). • User proceeds to Checkout. • Stripe processes payment securely. • System confirms registration and issues QR bib/ID. **Postconditions:** • Registration recorded; receipt and digital bib available in "My Events".
Alternative Flows and Exceptions	• Payment failure → show reason; allow retry/change method. • Category full → waitlist option or choose another category. • Event cancelled → initiate Refund Payment (see UC8).
Non-Functional Requirements	PCI-compliant Stripe integration, response < 3s, reliability at peak load, idempotent payment calls.

UC4: Real-Time Tracking

Use Case Name	Real-Time Tracking
Actors	Participant, Organizer, Location Service (GPS)
Preconditions	Participant is registered; device GPS permission granted; event tracking window active.
Normal Flow	**Description:** • Participant taps "Start Tracking". • App streams GPS updates at defined intervals. • Organizer views positions/pace on dashboard map. • System logs splits, distance, and time. **Postconditions:** • Tracking data stored; leaderboard and results can be computed.
Alternative Flows and Exceptions	• GPS lost → cache locally and backfill on reconnection. • Battery saver active → prompt to disable throttling.
Non-Functional Requirements	Latency < 500 ms, power efficiency, location accuracy, data integrity, secure transport.

UC5: Event and Dashboard Management

Use Case Name	Event and Dashboard Management
Actors	Organizer, Admin
Preconditions	Organizer/Admin authenticated with proper role; connectivity available.
Normal Flow	**Description:** • Organizer opens Dashboard. • Creates/updates event (title, date, route, fees, capacity). • Assigns volunteers; reviews registrations and payments. • Publishes updates; monitors analytics. **Postconditions:** • Event configuration saved and reflected to users.
Alternative Flows and Exceptions	• Invalid input → field-level validation errors. • Admin audit override → changes logged with rationale.
Non-Functional Requirements	RBAC, audit logging, strong consistency for writes, responsive UI.

UC6: Volunteer Check-In

Use Case Name	Volunteer Check-In
Actors	Volunteer, Organizer
Preconditions	Volunteer assigned to an event/shift; authenticated.
Normal Flow	**Description:** • Volunteer opens "My Assignments". • Reviews duty location/time and instructions. • Taps "Check In"; status visible to Organizer. **Postconditions:** • Attendance recorded; check-in timestamp stored for reporting.
Alternative Flows and Exceptions	• No assignment found → contact Organizer. • Late arrival → flagged for dashboard alert.
Non-Functional Requirements	Low-friction UI, reliable sync, role-based permissions, auditability.

UC7: Notifications and Media Updates

Use Case Name	Notifications and Media Updates

Actors	Organizer, Participant
Preconditions	Participants subscribed to event; push service operational.
Normal Flow	**Description:** • Organizer composes announcement or uploads media. • System pushes notification to subscribers. • Participants open feed to view content. **Postconditions:** • Announcement/media visible to intended audience; stored for later access.
Alternative Flows and Exceptions	• Delivery delay → queued; retry with backoff. • Large media → background upload with progress.
Non-Functional Requirements	Delivery $<$ 3s where possible, reliability, content moderation hooks, efficient media loading.

UC8: Refund Payment

Use Case Name	Refund Payment
Actors	Participant, Payment Gateway (Stripe), Organizer/Admin
Preconditions	A prior successful transaction exists; refund window and policy permit refund.
Normal Flow	**Description:** • Participant requests refund from "My Events" or Organizer triggers refund (cancellation). • System validates eligibility and calls Stripe refund API. • User receives confirmation; status updated to "Refunded". **Postconditions:** • Refund recorded; financial logs and user history updated.
Alternative Flows and Exceptions	• Refund denied (policy violation) → show reason; escalate to Organizer. • Network/API error → retry with idempotency key; notify user.
Non-Functional Requirements	Idempotent APIs, audit trail, policy compliance, user notification, data consistency.

4.3 Sample Scenarios

This section presents key real–world interaction sequences illustrating how the **Run Bangladesh Mobile Application** operates in practice. Each scenario demonstrates the logical flow of user actions, system responses, and outcomes based on the functional use cases described in Section 4.2.

Primary Scenario: Participant Registers and Completes a Marathon

Actors Involved: Participant (Registered Runner), Organizer, Authentication/OTP Service, Payment Gateway (Stripe), GPS Tracking Service
Goal: To enable a participant to register for a marathon, complete secure payment, run using the tracking feature, and appear on the leaderboard.

1. The user installs the *Run Bangladesh App* and selects **"Register / Login"**.
2. The system prompts for phone or email and password, then sends a one-time OTP.
3. The user enters the OTP, and the system authenticates and redirects to the home dashboard.
4. The participant browses available events and selects the **"Dhaka City Half Marathon 2025"**.
5. The system displays event details (date, category, fees, map, and rules).
6. The user chooses the 10 K category and taps **"Register Now"**.
7. The app collects personal information and proceeds to the **Checkout** screen.
8. The participant confirms payment via **Stripe Gateway**; transaction is processed securely.
9. The system issues a digital bib ID and QR code, confirming registration.
10. On race day, the participant launches the app and starts the **Real-Time Tracking** mode.
11. GPS data are transmitted to the server; the organizer monitors positions on the dashboard map.
12. Upon finishing, the participant stops tracking; results are automatically calculated.
13. The leaderboard updates in real time, showing ranks and timings.
14. The participant receives a completion notification and can download the digital certificate.

Outcome: The user successfully registers, pays, tracks the run, and appears on the leaderboard—demonstrating the complete primary workflow of the mobile application.

Secondary Scenario 1: Volunteer Check-In and Event Assistance

Actors Involved: Volunteer, Organizer
Goal: To allow volunteers to check in via the mobile app, confirm attendance, and assist during the event.

1. The organizer assigns volunteers to specific checkpoints and shift times through the dashboard.
2. The volunteer logs into the mobile app and opens **"My Assignments."**
3. The system displays shift details, location, and instructions.
4. On arrival, the volunteer taps **"Check In."**
5. The system records the timestamp and updates the organizer's dashboard.
6. The volunteer assists participants during the race and reports incidents through the in-app messaging feature.
7. After the event, the volunteer taps **"Check Out."**

8. Attendance and activity logs are stored for the organizer's analytics report.

Outcome: The volunteer's presence and performance are accurately logged, improving event coordination and transparency.

Secondary Scenario 2: Organizer Publishes Event and Pushes Updates

Actors Involved: Organizer, Admin, Participants, Notification Service
Goal: To demonstrate how an organizer manages an event and communicates with participants via the app.

1. The organizer logs in and navigates to the **Event Dashboard**.
2. Creates a new event, uploads route map, sets fees, date, and participant limit.
3. Reviews and publishes the event, making it visible in the participant feed.
4. Sends a **push notification** announcing registration opening.
5. During the event, uploads photos and videos to the media feed.
6. Participants receive instant notifications and can view updates in the app.
7. After the event, the organizer posts final results and thanks participants.
8. Admin reviews logs and approves the post-event summary.

Outcome: The organizer effectively manages event creation, communication, and post-race engagement through the mobile application.
Summary: These scenarios collectively represent the system's core interactions:

- The **Primary Scenario** showcases a full participant lifecycle—from registration to race completion.
- The **Secondary Scenarios** illustrate supporting workflows: volunteer management and organizer communication.

Together, they validate the system's ability to support multiple user roles in an integrated, real-time environment.

5 Requirements Traceability Matrix

The Requirements Traceability Matrix (RTM) links each functional and non-functional requirement to its corresponding use case(s), related quality attributes, and verification method. This ensures that every requirement can be verified through inspection, testing, analysis, or demonstration.

Req. ID	**Requirement Description**	**Associated Use Case(s)**	**Related NFR(s)**	**Priority**	**Verification Method**
FR1	Support user login and registration via email/-phone and OTP.	UC1	Security, Usability	High	Test
FR2	Display upcoming and past marathon events.	UC2	Performance, Usability	High	Test
FR3	Show event details (map, fee, category, rules).	UC2	Usability	Medium	Inspection
FR4	Enable event registration and secure payment (Stripe).	UC3	Security, Reliability	High	Test
FR5	Generate digital bib/QR after payment confirmation.	UC3	Availability, Integrity	High	Test
FR6	Track participant runs via GPS in real time.	UC4	Accuracy, Performance	High	Demonstration
FR7	Update leaderboard and race results automatically.	UC4	Reliability, Speed	High	Test
FR8	Provide dashboard for event management and analytics.	UC5	Maintainability, Security	High	Demonstration
FR9	Allow volunteer check-in/out with timestamps.	UC6	Reliability, Auditability	Medium	Test

Req. ID	Requirement Description	Associated Use Case(s)	Related NFR(s)	Priority	Verification Method
FR10	Send notifications and media updates to users.	UC7	Timeliness, Performance	Medium	Test
FR11	Process event refunds securely through Stripe.	UC8	Security, Integrity	High	Test
FR12	Enforce role-based access for Admin and Organizer.	UC5	Security, Maintainability	High	Inspection
FR13	Sync data between mobile app and web backend.	UC1–UC8	Consistency, Availability	High	Test
FR14	Cache data offline for browsing and tracking continuity.	UC2, UC4	Availability, Usability	Medium	Test
FR15	Generate reports and event analytics for organizers.	UC5	Scalability, Maintainability	Medium	Analysis

Table 5.1: Requirements Traceability Matrix for Run Bangladesh Mobile Application

Legend:

- **Inspection** — Requirement verified by document or code review.
- **Test** — Verified through functional, integration, or system testing.
- **Analysis** — Verified by examining performance metrics or log data.
- **Demonstration** — Verified by executing the feature in a controlled scenario.

This RTM ensures that every requirement from Chapters 3 and 4 is traceable to its implementation and has a clear verification approach. It supports project validation, quality assurance, and audit readiness throughout the development lifecycle.

Each functional requirement (FR) corresponds directly to one or more detailed Use Cases defined in Chapter 4, while non-functional requirements (NFR) span across multiple use cases to ensure system-wide qualities such as performance, security, and maintainability. This traceability matrix supports verification and validation activities during development and testing phases.

6 Initial Snapshots of the User Interface

6.1 Design Overview

The user interface (UI) of the **Run Bangladesh** mobile application was meticulously designed in **Figma** to ensure a seamless, engaging, and visually consistent user experience. **UC1–UC4:** Participant-facing features: Registration, Login, Event Browsing, and **UC5–UC8:** Organizer and Volunteer features: Dashboard Management, Tracking, Leaderboard, and Notifications.

6.2 Figma Mockups for Use Cases

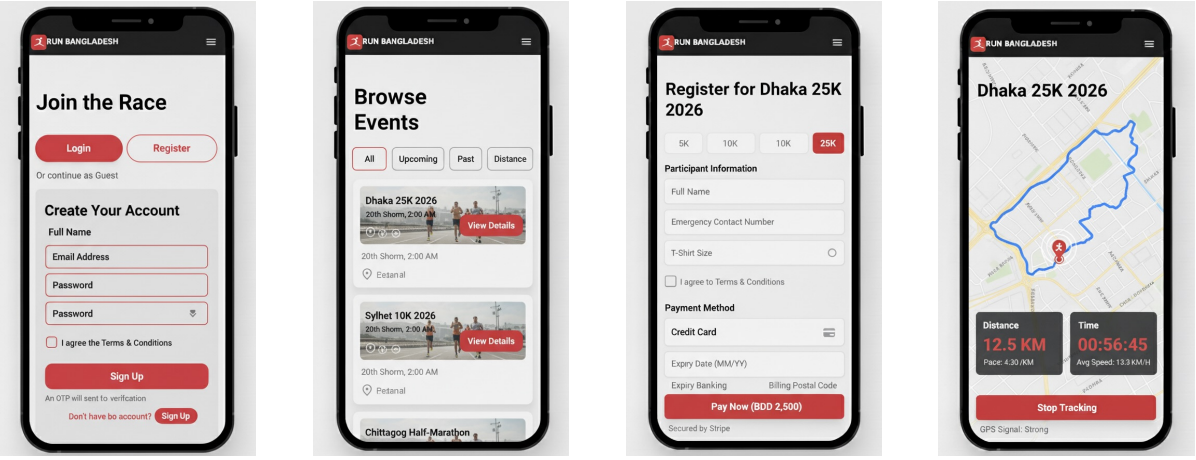

Figure 6.1: Initial Figma UI Mockups for Run Bangladesh Mobile App (UC1–UC4)

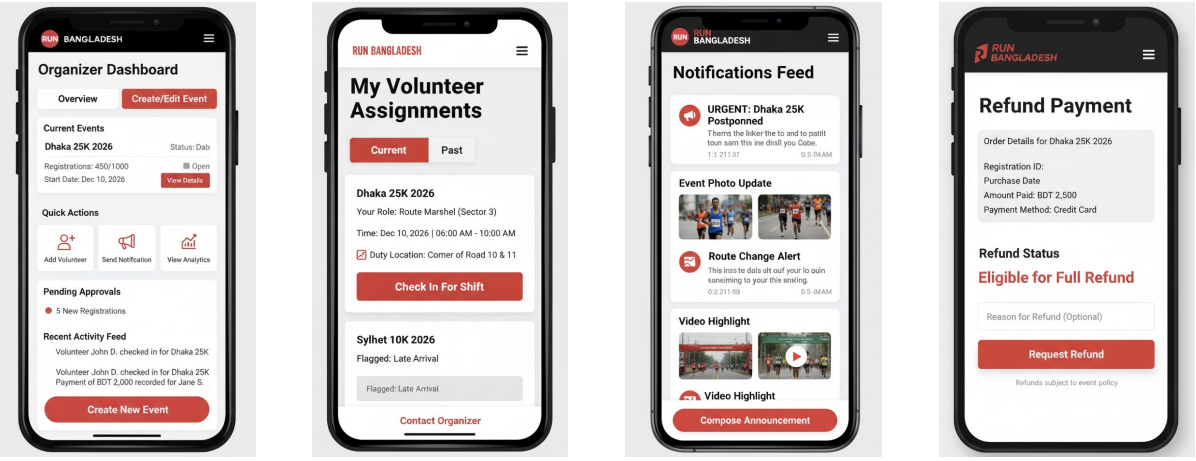

Figure 6.2: Initial Figma UI Mockups for Run Bangladesh Mobile App (UC1–UC8)

7 Glossary of Terms

This chapter defines the key technical and domain-specific terms used throughout the **Run Bangladesh Mobile Application** Software Requirements Specification (SRS). The glossary ensures clarity and consistency for all stakeholders involved in the project — including developers, testers, organizers, and users.

Term / Acronym	Definition and Description
SRS (Software Requirements Specification)	A comprehensive document that outlines all functional and non-functional requirements of the system, serving as a reference for development, testing, and maintenance.
Run Bangladesh	The mobile application being developed to simplify marathon event management in Bangladesh by enabling participants to register, pay, track, and view results, while providing organizers with event management tools.
React Native	An open-source JavaScript framework used to build cross-platform mobile applications that run on both Android and iOS using a single codebase.
Figma	A collaborative UI/UX design tool used to create wireframes and high-fidelity mockups for the Run Bangladesh mobile application.
Node.js	A back-end runtime environment for building scalable server-side applications, used in this project to manage data, API endpoints, and server logic.
phpMyAdmin (SQL)	A web-based interface used for managing the SQL database that stores user profiles, event data, payments, and run statistics.
Stripe API	A secure payment gateway integrated within the mobile application to process user registration fees and manage transactions.
Docker	A containerization platform used to ensure consistent deployment and scalability of the application across different environments.
Jest	A JavaScript testing framework used for unit and integration testing of the application to ensure system reliability and performance.
GPS (Global Positioning System)	A satellite-based system used to provide real-time location tracking of participants during marathon events.
OTP (One-Time Password)	A temporary authentication code sent to users during registration or login to enhance account security.
UI (User Interface)	The visual and interactive elements of the application through which users interact with the system.
UX (User Experience)	The overall experience and satisfaction of users while interacting with the application, influenced by ease of use, accessibility, and visual appeal.
Participant	A user who registers for and participates in marathon events through the mobile app, accessing registration, payment, tracking, and results features.

Organizer	A user role responsible for creating, managing, and monitoring marathon events, including handling registrations, volunteers, and results.
Volunteer	A user assisting in event operations such as runner support, check-ins, and event logistics, managed through the volunteer module of the app.
Leaderboard	A dynamic ranking system that displays real-time positions, times, and performance statistics of participants during an event.
Push Notification	Alerts sent by the system to inform users about event updates, payment confirmations, or new announcements.
API (Application Programming Interface)	A set of defined rules and protocols that allow different software systems (mobile app and backend) to communicate with each other.
Authentication Service	The component responsible for verifying user identity through login credentials and OTP verification.
Database	The structured data storage system used to maintain all persistent information such as users, events, payments, and logs.

Resources and References

[1] IEEE, *IEEE Recommended Practice for Software Requirements Specifications (IEEE Std 830-1998)*, Institute of Electrical and Electronics Engineers, 1998. Available: https://ieeexplore.ieee.org/document/720574

[2] Department of Electrical and Computer Engineering, North South University, *CSE327: Software Engineering – Course Specification*, Fall 2025.

[3] Run Bangladesh, *Official Website*, 2025. [Online]. Available: https://runbangladesh.com

[4] Figma Inc., *Figma Design Tool Documentation*, 2025. [Online]. Available: https://help.figma.com/

[5] Meta Platforms Inc., *React Native Documentation*, Version 0.75, 2025. [Online]. Available: https://reactnative.dev/docs/getting-started

[6] OpenJS Foundation, *Node.js Documentation*, Version 20.x, 2025. [Online]. Available: https://nodejs.org/en/docs/

[7] The phpMyAdmin Project, *phpMyAdmin Documentation*, Version 5.2, 2025. [Online]. Available: https://www.phpmyadmin.net/docs/

[8] Stripe Inc., *Stripe API Reference*, Version 2025-10, 2025. [Online]. Available: https://stripe.com/docs/api

[9] Docker Inc., *Docker Documentation*, Version 26.0, 2025. [Online]. Available: https://docs.docker.com/

[10] Meta Platforms Inc., *Jest Testing Framework Documentation*, Version 29.x, 2025. [Online]. Available: https://jestjs.io/docs/getting-started

[11] ISO/IEC/IEEE 29148:2018, *Systems and Software Engineering — Life Cycle Processes — Requirements Engineering*, International Organization for Standardization, 2018.

[12] OMG (Object Management Group), *Unified Modeling Language (UML) Specification*, Version 2.5.1, 2017. [Online]. Available: https://www.omg.org/spec/UML/

[13] Stack Overflow, *Developer Q&A Platform*, 2025. [Online]. Available: https://stackoverflow.com/

[14] FreeCodeCamp, *REST API Tutorials and Documentation*, 2025. [Online]. Available: https://www.freecodecamp.org/news/tag/rest-api/

[15] World Wide Web Consortium (W3C), *Web and Mobile Accessibility Guidelines*, 2025. [Online]. Available: https://www.w3.org/WAI/standards-guidelines/

8 Contributions of Team Members

This chapter outlines the specific roles and contributions of each team member involved in the development of the **Run Bangladesh Mobile Application**. All members collaborated throughout the project lifecycle, contributing to analysis, design, implementation, and documentation phases.

Name	Student ID	Role and Responsibilities
Saif Mohammed	2121913042	**Project Manager & Team Lead** — Coordinated project planning, managed tasks and timelines, integrated team deliverables, and ensured overall SRS documentation consistency.
Humayra Rahman Nipa	2121128042	**Designer & Tester** — Designed Figma UI mockups, contributed to front-end layout planning, and performed functional and usability testing to validate user interface elements.
Sinthia Ahmed Rachona	2211916042	**Software Developer** — Developed system modules using React Native and Node.js, managed database integration, and supported backend logic and API implementation.

Table 8.1: Roles and Contributions of Team Members

All team members participated in requirement analysis, system design discussions, and documentation reviews to ensure alignment with IEEE SRS standards and project goals.

9 Appendices

This chapter contains supplementary materials and supporting documents that complement the main sections of the **Run Bangladesh Mobile Application** Software Requirements Specification (SRS). These appendices include additional interface wireframes, test planning details, and proposed enhancements for future iterations of the system.

9.1 Appendix A: Additional Interface Wireframes

This section includes additional user interface mockups created in **Figma** to illustrate detailed design flows not covered in Chapter 6. These wireframes demonstrate extended functionalities such as:

- Event detail view with location map and participant statistics.
- Payment confirmation and digital bib generation screens.
- Volunteer schedule management and check-in confirmation layouts.
- Organizer analytics dashboard with event KPIs and participant insights.

Each screen adheres to the same design principles defined in the Design Overview, ensuring a consistent and accessible experience across all modules.

9.2 Appendix B: Testing Summary

The testing phase will follow a structured and iterative approach, including both functional and non-functional tests. The primary testing methods are outlined below:

- **Unit Testing:** Conducted using **Jest** to validate individual modules such as login, payment, and GPS tracking.
- **Integration Testing:** Ensures smooth interaction between frontend (React Native) and backend (Node.js + SQL).
- **System Testing:** Verifies complete workflows, including event registration, tracking, and leaderboard updates.
- **User Acceptance Testing (UAT):** Confirms usability and accuracy through feedback from pilot participants and organizers.

Test results, performance logs, and error reports will be documented in subsequent project iterations as the prototype evolves toward deployment.

9.3 Appendix C: Future Enhancements

Future development of the **Run Bangladesh** mobile application may include:

- Integration of wearable device data (smartwatches, fitness trackers) for advanced health metrics.
- Offline mode support for event-day check-ins and run tracking.

- Enhanced analytics dashboards using AI-driven insights for organizers.
- Social engagement features (leaderboard sharing, participant chat groups).
- Multi-language interface support (Bangla and English).

9.4 Appendix D: Project Repository and Version Control

All source code, design assets, and documentation for this project are maintained in a private **GitHub Repository**. Version control ensures collaborative development, allowing the team to track progress, manage issues, and maintain proper documentation.

- **Repository Name:** Run Bangladesh Mobile App
- **Platform:** GitHub
- **Branching Model:** Main (Stable) and Dev (Feature Development)
- **Primary Technologies:** React Native, Node.js, SQL, Docker, Jest

9.5 Appendix E: Acknowledgment of Tools and Frameworks

The successful preparation of this SRS and system design was made possible through the use of the following tools and technologies:

- **Figma** — Frontend UI/UX design and mockup creation.
- **Overleaf** — LaTeX-based documentation and report preparation.
- **GitHub** — Version control and team collaboration.
- **React Native & Node.js** — Development of frontend and backend systems.
- **Docker** — Application containerization for deployment.
- **Jest** — Testing framework for frontend and backend validation.

Note: These appendices serve as supplementary documentation and will be updated in future development phases as additional prototypes, test results, and deployment details become available.