Software Requirements Specification

for

Quickdeal

Version 1.0

Prepared by:

Mayeesha Musarrat (Team Leader)

Syeda Raisa Rahman and

Maria Sultana

Submitted to:

Md Main Uddin Chisty
Software Engineer - II
Brain Station 23

Submission Date:

April 30, 2025

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Quickdeal v1.0	2025-03-20	Initial SRS Docu	

1. Introduction

1.1 Purpose

This document defines the Software Requirements Specification for **Quickdeal v1.0**, a bid-and-quote mobile application. Quickdeal facilitates communication between customers and vendors for custom product manufacturing. Users can submit product requests with design and specifications, and vendors place bids to fulfill them. This SRS focuses on the mobile application built using **Flutter**, with **Supabase** as the backend, implementing **BLoC** for state management under a **Clean Architecture** structure.

1.2 Document Conventions

- **Bold** text is used for headings and key system components.
- *Italics* represent glossary terms or references to other documents.
- `Monospace` indicates inline code, APIs, or file paths.
- Each system requirement is labeled with a unique identifier for traceability (e.g., `REQ-UI-01`, `REQ-FUNC-02`).
- Priorities for high-level requirements are **not inherited**; each requirement will include its own priority level where applicable.

1.3 Intended Audience and Reading Suggestions

This document is intended for:

- **Developers**: to implement features as described
- **Testers**: to develop test cases from functional requirements
- **Project Managers**: to track development scope and progress
- **Stakeholders**: to ensure project goals align with system behavior

- **UI/UX Designers**: to build screens matching feature descriptions

It is recommended to begin with **Section 1** for context, followed by **Section 2: Overall Description**, and then refer to **Section 4: System Features** for detailed functional requirements.

1.4 Product Scope

Quickdeal aims to bridge the gap between customers and product vendors through a structured and transparent bidding platform. The app enables users to submit detailed Requests for Quotation (RFQs), allows vendors to place and revise bids, and manages post-bid activities like agreements and payments. This improves efficiency in custom product manufacturing, supports small businesses and individuals, and digitizes vendor-client interactions for clarity and accountability.

1.5 References

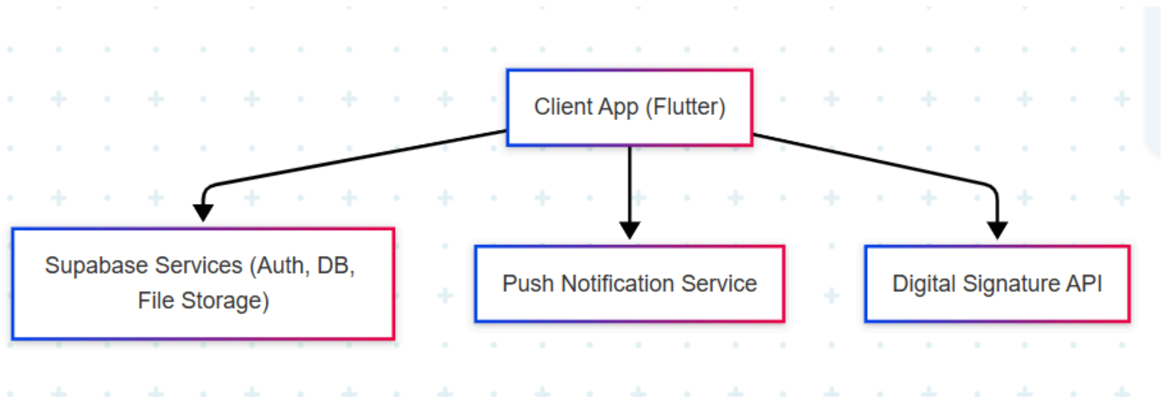
- Flutter Documentation: *https://flutter.dev/docs*
- Supabase Documentation: *https://supabase.com/docs*
- BLoC State Management: *https://bloclibrary.dev*
- IEEE 830-1998: Recommended Practice for Software Requirements Specifications *https://dspmuranchi.ac.in/pdf/Blog/srs_template-ieee.pdf*
- Clean Architecture: https://blog.stackademic.com/building-robust-data-sources-in-clean-architecture-for-flutter-e4c0833f17ed

2. Overall Description

2.1 Product Perspective

Quickdeal is a **new, self-contained mobile application** that provides a platform where individuals and businesses can request customized products or services and vendors can submit bids to fulfill those requests. It is not part of an existing system but is built from the ground up using **Flutter** for frontend development and **Supabase** as the backend solution. The system follows the **Clean Architecture** paradigm and leverages **BLoC (Business Logic Component)** for state management. The product does not depend on any pre-existing internal systems, though it integrates external services such as **digital signature APIs** and may later connect to third-party **payment gateways**.

A simplified component diagram:


Client App (Flutter)
Supabase Services (Auth, DB, File Storage)
Push Notification Service
Digital Signature API

2.2 Product Functions

The core functions of *Quickdeal* include:

- **User Registration and Role Selection**
  - Users sign up as either *Personal* or *Business* accounts.
  - Users can optionally upgrade to become *Vendors*.
- **RFQ Submission**
  - Clients submit custom product or service requests.
  - Upload of supporting files/images.
- **RFQ Viewing and Bidding**
  - Vendors view available RFQs.
  - Vendors place bids with pricing and project details.
- **Bid Management**
  - Clients receive and evaluate multiple bids.
  - Option for follow-up messaging before finalizing.
- **Agreement Generation**
  - Upon bid selection, a *digital agreement* is generated.
  - Users and vendors digitally sign to confirm the project.

- **Project Tracking**
  - Confirmed projects move to “Ongoing Orders”.
  - Project status is updated until completion and payment.

2.3 User Classes and Characteristics

Personal User: Individual clients who seek one-off or occasional custom products. Requires a simple UI and guided form submission.

Business User: Organizations with regular or bulk needs. May require more detailed order tracking and vendor vetting.

Vendor: Suppliers or service providers who respond to RFQs. Needs access to dashboards, bid management tools, and profile editing.

Admin (Future Scope): Oversee user activity, resolve disputes, manage content/categories. Requires access to analytics and user reports.

2.4 Operating Environment

- **Mobile Devices**: Android (v8.0 and above) and iOS (v12 and above).
- **Frontend**: Developed using Flutter.
- **Backend**: Powered by Supabase (Realtime update, Storage).
- **Connectivity**: Requires stable internet for all operations.

2.5 Design and Implementation Constraints

- Must implement **BLoC** for state management.
- Architecture should strictly follow **Clean Architecture** layers: *Presentation*, *Domain*, *Data*.
- All backend interactions must use **Supabase APIs**.
- Digital agreement must be securely stored and cryptographically signed.
- User data must comply with local data protection laws.

2.6 User Documentation

- **In-app onboarding screens** for first-time users.
- **FAQ and Help Center** explaining workflows for both vendors and clients.
- **Vendor Onboarding Guide** (PDF/manual) explaining bidding rules, profile requirements, etc.
- **Future scope**: Video tutorials and community forum access.

2.7 Assumptions and Dependencies

- Users will have access to stable internet connections.
- Supabase services will maintain >99% uptime and provide persistent data storage.
- Future integration may rely on:
  - External **payment gateway APIs** (e.g., Stripe, SSLCommerz).
  - **Digital signature providers** for agreement handling.
- Flutter and the plugin ecosystem remain compatible with target OS versions.

3. External Interface Requirements

3.1 User Interfaces

Quickdeal will follow **Material Design** principles via Flutter's UI toolkit to ensure consistency and usability across Android and iOS platforms. All screens will include standard components like:

- **Navigation Bar** (Home, Bids, Notifications, Profile)
- **Floating Action Button** for key user actions (e.g., posting RFQs)
- **Form Fields** with validation for submitting requests and bids
- **Snackbars/Dialogs** for error messages and confirmations
- **Search and Filter** options for browsing RFQs and bids

- **Vendor Dashboard**: displays bid history, active projects, and revenue
- **Client Dashboard**: shows submitted RFQs, received bids, and project status

Each screen will contain consistent **Help/Info icons**, back navigation, and error handling messages such as *"Please fill all required fields"* or *"Connection lost, try again."*

> *User interface design details will be expanded in a separate User Interface Specification document.*

3.2 Hardware Interfaces

The application does not require any specialized hardware interfaces beyond standard mobile hardware capabilities. Supported interfaces include:

- **Touch Input** (tap, swipe, long press)
- **Camera Access** (for uploading product images)
- **File Storage** (for RFQ document uploads)
- **Biometric Authentication APIs** (optional, for secure login)

All hardware interactions will utilize standard Flutter plugins compatible with Android and iOS platforms.

3.3 Software Interfaces

The following software interfaces are essential to the system:

- **Supabase Backend Services**:
  - Auth: for user login, registration, and role-based access control.
  - PostgreSQL Database: stores user data, RFQs, bids, messages, agreements.
  - Storage: for uploading files and digital agreement PDFs.
  - Realtime: to deliver notifications and track project status updates.
- **Flutter SDK**: Frontend framework for building the cross-platform mobile app.
- **Digital Signature API** (e.g., Docusign or similar): generates and verifies user-vendor agreements.

- **Optional Future Interfaces**:
    - Payment Gateway APIs (e.g., SSLCommerz, Stripe) for managing secure payments.
    - Email Notification Services (SMTP or SendGrid) for transactional updates.

Data shared between frontend and backend uses secure HTTPS calls with JSON payloads. Authentication tokens (JWT) are used for secure session control.

3.4 Communications Interfaces

- **Internet Connectivity** is required for all operations.
- **HTTPS** protocol will be used for all client-server communication to ensure encryption.
- **Push Notifications** will be handled through Firebase Cloud Messaging (FCM) or Supabase Realtime.
- **WebSocket** connections may be used for real-time updates (e.g., new bids on an RFQ).
- **Digital signature agreements** will be exchanged securely using encrypted endpoints from third-party APIs.
- **No local offline mode** is currently planned; all data interactions require network availability.

4. System Features

4.1 Request for Quotation (RFQ) Submission

4.1.1 Description and Priority

This feature enables users (clients) to submit detailed *Requests for Quotation* to request customized products or services. Users can define product specifications, attach reference files, and set deadlines.

Priority: High

Benefit: 9 | Penalty: 8 | Cost: 3 | Risk: 4

4.1.2 Stimulus/Response Sequences

- User logs in → selects *Submit RFQ* → fills out form with product details → uploads optional files → submits request

- System validates input → stores RFQ in database → notifies vendors via real-time update and push notification

4.1.3 Functional Requirements

- REQ-FUNC-01: The system shall allow users to fill in RFQ fields including product name, category, quantity, budget, and deadline.
- REQ-FUNC-02: The system shall validate all required fields and alert the user on missing or invalid input.
- REQ-FUNC-03: The system shall allow users to upload one or more image or document files with an RFQ.
- REQ-FUNC-04: The system shall store RFQ data securely in the backend database.
- REQ-FUNC-05: The system shall notify vendors in relevant categories when a new RFQ is submitted.

4.2 Vendor Bidding System

4.2.1 Description and Priority

This feature allows vendors to view submitted RFQs and respond with detailed bids including price quotes, estimated delivery time, and comments.

Priority: High

Benefit: 8 | Penalty: 7 | Cost: 4 | Risk: 5

4.2.2 Stimulus/Response Sequences

- Vendor logs in → navigates to *Available RFQs* → selects an RFQ → fills in bid form → submits bid
- System validates the bid → stores bid data → notifies the client

4.2.3 Functional Requirements

- REQ-FUNC-06: The system shall allow vendors to view a list of open RFQs based on their registered product categories.

- REQ-FUNC-07: The system shall allow vendors to submit bids including unit price, total cost, expected delivery time, and additional notes.
- REQ-FUNC-08: The system shall validate bid entries and ensure duplicate bids for the same RFQ by the same vendor are not allowed.
- REQ-FUNC-09: The system shall notify the client when a new bid is received.

4.3 Bid Finalization and Digital Agreement

4.3.1 Description and Priority

Once a client selects a suitable bid, both parties finalize the project with a *digital signature agreement*. This legally binds the vendor to the agreed terms.

Priority: High

Benefit: 9 | Penalty: 9 | Cost: 5 | Risk: 6

4.3.2 Stimulus/Response Sequences

- Client selects a bid → initiates agreement → system generates contract → both parties sign digitally
- System confirms agreement → moves project to *Ongoing Orders*

4.3.3 Functional Requirements

- REQ-FUNC-10: The system shall allow the client to choose a preferred bid for any RFQ.
- REQ-FUNC-11: The system shall generate a digital agreement with terms pulled from the bid and RFQ.
- REQ-FUNC-12: The system shall use an external API to enable digital signatures from both the client and vendor.
- REQ-FUNC-13: The system shall store the digitally signed agreement in a secure, read-only format (PDF).
- REQ-FUNC-14: The system shall move the project to the *Ongoing Confirmed Orders* section after both parties have signed.

4.4 Project Tracking and Notifications

4.4.1 Description and Priority

This feature provides real-time tracking of active projects, including timeline milestones, communication, and status updates.

Priority: Medium

Benefit: 7 | Penalty: 6 | Cost: 3 | Risk: 4

4.4.2 Stimulus/Response Sequences

- Vendor updates delivery status → system logs update → client gets notified
- Client views progress from dashboard or messages vendor for clarification

4.4.3 Functional Requirements

- REQ-FUNC-15: The system shall allow vendors to update project progress (e.g., design stage, production, shipment).
- REQ-FUNC-16: The system shall notify the client when status updates are made.
- REQ-FUNC-17: The system shall allow both parties to exchange secure messages related to the order.
- REQ-FUNC-18: The system shall display all active projects in a dashboard sorted by most recent update.

5. Other Nonfunctional Requirements

5.1 Performance Requirements

- *QuickDeal* shall support up to **500 concurrent users** without significant degradation in performance.
- The system shall return search results within **2 seconds** for 95% of queries.
- All notification delivery (e.g., RFQ updates, bid submissions) shall occur within **1 second** of the triggering event.
- Page load time for authenticated dashboards shall not exceed **3 seconds** under average server load.

5.2 Safety Requirements

- All user actions that involve **critical operations** (e.g., accepting bids, submitting payments, digital signatures) must be **confirmed by the user** via dialog pop-ups.
- The system shall include an **auto-save and recovery** mechanism for RFQ or bid drafts to avoid data loss.
- Error messages must not disclose system internals or database details that could be exploited.
- *QuickDeal* shall follow **OWASP** secure coding guidelines to minimize risk of data compromise.

5.3 Security Requirements

- All user data must be encrypted using **AES-256** at rest and **TLS 1.3** in transit.
- System access shall require **multi-factor authentication (MFA)** for all vendor accounts and admin roles.
- Role-based access control (RBAC) shall enforce **privilege separation** between clients, vendors, and administrators.
- Sensitive actions (e.g., agreement signing, payment initiation) must be logged and **auditable**.
- The system must comply with **GDPR** and local data protection laws in the regions of deployment.

5.4 Software Quality Attributes

- **Reliability**: 99.5% uptime required, excluding planned maintenance.
- **Usability**: Interfaces must follow intuitive navigation patterns with **WCAG 2.1 AA** accessibility compliance.
- **Maintainability**: Code must follow modular structure with documentation to enable third-party maintenance.
- **Scalability**: Architecture should allow horizontal scaling to accommodate increasing vendor-client interactions.

5.5 Business Rules

- Only **registered vendors** may place bids on RFQs.
- Users must confirm receipt of project completion before payment is finalized.
- **Inactive users** (no activity for 6 months) will be flagged and notified before account deactivation.
- **Digital signatures** are required to finalize any vendor-client agreement.

5.6 Other Requirements

- **Database**: Must use **PostgreSQL** with full-text search capabilities.
- **Deployment**: The application shall support deployment using **Docker** containers.
- **Localization**: The platform shall support both **English** and **Bangla** languages.
- **Logging**: System shall implement structured logging for user actions, errors, and warnings.
- **Legal**: All terms of use shall be agreed to via a **checkbox** on registration, stored with timestamp.

Appendix A: *Glossary*

- **RFQ**: *Request for Quotation* – A form used by clients to describe a custom product or service they want vendors to bid on.
- **Bid**: A response from a vendor that includes pricing, delivery time, and conditions.
- **Digital Signature**: A legally binding e-signature used to confirm vendor-client agreements.
- **MFA**: *Multi-Factor Authentication* – A security protocol requiring more than one method of identity verification.
- **RBAC**: *Role-Based Access Control* – A security approach where users have access based on their role in the system.

Appendix B: *Analysis Models*

- **User Flow Diagram**: QuickDeal user flow
- **Schema Diagram**: QuickDeal schema diagram
- **UI Diagram:** QuickDeal UI diagram

Appendix C: *To Be Determined (TBD) List*

1. TBD-01: Final integration API for digital signature service
2. TBD-02: Decision on which payment gateway to integrate (e.g., Stripe, SSLCommerz)
3. TBD-03: Language packs for future localization beyond English and Bangla
4. TBD-04: SLA definition for vendor response times
5. TBD-05: Final list of categories for RFQs