# Software Requirements Specification

for

Student Marketplace for Buying & Selling Used Items

Version 1.0 approved

Prepared by :Aaditya Saxena(23BAI1408) Yash Sigchi(23BAI1242) Sanskar Pandey(23BAI1197)

Organization: Vellore Institute of Technology, Chennai. School of Computer Science and Engineering. Course: Software Engineering(BCSE301P)

Date Created: 27th July 2025

## Table of Contents

Revision History

| Name | Date | Reason For Changes | Version |
|---|---|---|---|

# Introduction

## Purpose

This document specifies the software requirements for the ***Student Marketplace for Buying & Selling Used Items*** project. It outlines all major and minor system functionalities needed by students to securely list, buy, or sell used items on a campus-verified platform. It also identifies both functional and non-functional requirements, guiding developers, testers, and stakeholders.

## Document Conventions

<Describe any standards or typographical conventions that were followed when writing this SRS, such as fonts or highlighting that have special significance. For example, state whether priorities for higher-level requirements are assumed to be inherited by detailed requirements, or whether every requirement statement is to have its own priority.>

## Intended Audience and Reading Suggestions

<Describe the different types of reader that the document is intended for, such as developers, project managers, marketing staff, users, testers, and documentation writers. Describe what the rest of this SRS contains and how it is organized. Suggest a sequence for reading the document, beginning with the overview sections and proceeding through the sections that are most pertinent to each reader type.>

## Product Scope

The Student Marketplace is a university-verified, peer-to-peer platform enabling students to buy and sell used books, electronics, furniture, and other goods. Core features include:

- University email authentication
- Item listing with filtering and location support
- Real-time chat
- Secure payments via Stripe (with escrow)
- AI-based fraud detection

- Admin moderation tools
  The platform promotes trust, sustainability, and convenience in local student transactions.

## References

<List any other documents or Web addresses to which this SRS refers. These may include user interface style guides, contracts, standards, system requirements specifications, use case documents, or a vision and scope document. Provide enough information so that the reader could access a copy of each reference, including title, author, version number, date, and source or location.>

# Overall Description

## Product Perspective i/p-task-o/p

The Student Marketplace is a **new, web-based platform** specifically built for verified university students to buy and sell used items locally. It draws inspiration from platforms like OLX but integrates features tailored for student safety, such as university email login, AI fraud detection, and secure in-app payments.

## Product Functions

- User registration and login with university email verification
- Create, view, edit, and delete listings (with images and details)
- Search/filter listings based on category, location, condition, and price
- Real-time chat between buyer and seller
- Payments using Stripe API with escrow model
- Post-transaction ratings and reviews
- Admin dashboard for moderation and flagged content review
- AI-powered fraud detection for spam or anomaly detection
- Notifications for messages, sales, and offers

## User Classes and Characteristics

<Identify the various user classes that you anticipate will use this product. User classes may be differentiated based on frequency of use, subset of product functions used, technical expertise, security or privilege levels, educational level, or experience. Describe the pertinent characteristics of each user class. Certain requirements may pertain only to certain user classes. Distinguish the most important user classes for this product from those who are less important to satisfy.>

## Operating Environment

- **Frontend**: React.js interface running in modern browsers (Chrome, Firefox, Edge)
- **Backend**: Node.js on macOS (development) and Linux (Ubuntu 22.04 LTS for deployment)
- **Database**: MongoDB Atlas (cloud-hosted)
- **Real-time Chat**: Socket.IO
- **Payments**: Stripe API
- **Hosting**: AWS EC2 / GCP with Docker containers
- **Other Services**: Google Maps API, Firebase, SendGrid

## Design and Implementation Constraints

<Describe any items or issues that will limit the options available to the developers. These might include: corporate or regulatory policies; hardware limitations (timing requirements, memory requirements); interfaces to other applications; specific technologies, tools, and databases to be used; parallel operations; language requirements; communications protocols; security considerations; design conventions or programming standards (for example, if the customer's organization will be responsible for maintaining the delivered software).>

## User Documentation

<List the user documentation components (such as user manuals, on-line help, and tutorials) that will be delivered along with the software. Identify any known user documentation delivery formats or standards.>

## Assumptions and Dependencies

<List any assumed factors (as opposed to known facts) that could affect the requirements stated in the SRS. These could include third-party or commercial components that you plan to use, issues around the development or operating environment, or constraints. The project could be affected if these assumptions are incorrect, are not shared, or change. Also identify any dependencies the project has on external factors, such as software components that you intend to reuse from another project, unless they are already documented elsewhere (for example, in the vision and scope document or the project plan).>

# External Interface Requirements

## User Interfaces

<Describe the logical characteristics of each interface between the software product and the users. This may include sample screen images, any GUI standards or product family style guides that are to be followed, screen layout constraints, standard buttons and functions (e.g., help) that will appear on every screen, keyboard shortcuts, error message display standards, and so on. Define

the software components for which a user interface is needed. Details of the user interface design should be documented in a separate user interface specification.>

### Hardware Interfaces

<Describe the logical and physical characteristics of each interface between the software product and the hardware components of the system. This may include the supported device types, the nature of the data and control interactions between the software and the hardware, and communication protocols to be used.>

### Software Interfaces

<Describe the connections between this product and other specific software components (name and version), including databases, operating systems, tools, libraries, and integrated commercial components. Identify the data items or messages coming into the system and going out and describe the purpose of each. Describe the services needed and the nature of communications. Refer to documents that describe detailed application programming interface protocols. Identify data that will be shared across software components. If the data sharing mechanism must be implemented in a specific way (for example, use of a global data area in a multitasking operating system), specify this as an implementation constraint.>

### Communications Interfaces

<Describe the requirements associated with any communications functions required by this product, including e-mail, web browser, network server communications protocols, electronic forms, and so on. Define any pertinent message formatting. Identify any communication standards that will be used, such as FTP or HTTP. Specify any communication security or encryption issues, data transfer rates, and synchronization mechanisms.>

## System Features

This section describes the main features of the ***Student Marketplace for Buying & Selling Used Items*** system.

Functional requirements are grouped by key system features such as authentication, listings, chat, payments, and more.

Each feature includes its purpose, expected behavior, and related requirements for easy understanding and implementation.

### User Authentication & Profile Management

#### 4.1.1 Description and Priority

This feature enables students to securely create an account, log in, and manage their profile. It includes OAuth login and university email verification to ensure a trusted student-only environment.
**Priority**: High
**Benefit**: 9 **Penalty**: 8 **Cost**: 4 **Risk**: 3

#### 4.1.2 Stimulus/Response Sequences

- User opens the app → Clicks "Sign Up" → Enters university email → System sends verification link → User verifies → Account is created.
- User selects Google Login → System authenticates via OAuth → User is logged in and redirected to dashboard.

#### 4.1.3 Functional Requirements

- **REQ-1**: The system shall allow users to register using a valid university email address.
- **REQ-2**: The system shall verify the email by sending a verification link.
- **REQ-3**: The system shall allow OAuth login via Google and Facebook.
- **REQ-4**: The system shall store user details (name, university, profile photo).
- **REQ-5**: The system shall allow users to update profile information and view their past listings and transactions.

### 4.2 Item Listing and Management

#### 4.2.1 Description and Priority

This feature allows users to post, edit, and manage item listings with images, descriptions, price, and location.
**Priority**: High
**Benefit**: 9 **Penalty**: 7 **Cost**: 4 **Risk**: 2

#### 4.2.2 Stimulus/Response Sequences

- User clicks "Create Listing" → Fills in item details → Uploads photos → Clicks "Submit" → System posts item to marketplace.
- User selects “Edit” on a posted item → Makes changes → Clicks "Save" → Listing is updated.
- User clicks “Delete” → System removes the listing from public view.

#### 4.2.3 Functional Requirements

- **REQ-6**: The system shall allow users to create listings with fields: title, description, category, condition, location, and price.
- **REQ-7**: The system shall support multiple image uploads per listing.
- **REQ-8**: The system shall allow users to edit or delete their listings.
- **REQ-9**: The system shall allow users to mark items as "Sold".
- **REQ-10**: The system shall display listings to all users, ordered by relevance or recency.

## 4.3 Search and Filtering

### 4.3.1 Description and Priority

This feature provides users the ability to search for items using keywords and filter results by category, location, condition, and price range.
**Priority**: High
**Benefit**: 8 **Penalty**: 6 **Cost**: 3 **Risk**: 3

### 4.3.2 Stimulus/Response Sequences

- User enters search term → System displays matching listings.
- User selects filters (category = books, price < ₹1000) → System shows refined results.
- User uses location slider → Listings nearby are displayed on the map.

### 4.3.3 Functional Requirements

- **REQ-11**: The system shall provide a search bar for keyword-based listing search.
- **REQ-12**: The system shall support filters: category, price range, item condition, and location.
- **REQ-13**: The system shall display search results with pagination or infinite scroll.
- **REQ-14**: The system shall integrate Google Maps API to show listings near the user's location.
- **REQ-15**: The system shall auto-detect the user’s location with permission or allow manual location entry.

## 4.4 Real-Time Chat and Messaging

### 4.4.1 Description and Priority

Enables buyers and sellers to communicate securely through real-time messaging.
**Priority**: High
**Benefit**: 8 **Penalty**: 6 **Cost**: 4 **Risk**: 5

### 4.4.2 Stimulus/Response Sequences

- Buyer clicks “Message Seller” on a listing → Chat window opens → User types and sends message → Seller receives instant notification and reply.
- User opens Messages → Views previous conversations → System displays chat history.

### 4.4.3 Functional Requirements

- **REQ-16**: The system shall allow real-time chat between buyer and seller using Socket.IO.
- **REQ-17**: The system shall store chat messages persistently in the database.
- **REQ-18**: The system shall notify users of new messages via push or in-app

alerts.
- **REQ-19**: The system shall allow users to block/report conversations.
- **REQ-20**: The system shall support emojis, basic formatting, and file/image sharing (optional).

### 4.5 Secure Payment Integration

#### 4.5.1 Description and Priority

Allows users to make secure transactions via Stripe, with escrow held until delivery is confirmed.
**Priority**: High
**Benefit**: 9 **Penalty**: 7 **Cost**: 5 **Risk**: 6

#### 4.5.2 Stimulus/Response Sequences

- Buyer clicks “Buy Now” → Proceeds to payment → Stripe processes the amount → Seller is notified → Amount is held in escrow.
- Buyer marks item as received → Funds are released to the seller.

#### 4.5.3 Functional Requirements

- **REQ-21**: The system shall use Stripe API for payment processing.
- **REQ-22**: The system shall support escrow functionality until delivery is confirmed.
- **REQ-23**: The system shall show payment history in the user dashboard.
- **REQ-24**: The system shall send email receipts after each transaction.
- **REQ-25**: The system shall validate payments securely with Stripe webhooks.

### 4.6 Ratings and Reviews

#### 4.6.1 Description and Priority

After each transaction, buyers and sellers can rate and review each other. This builds trust in the platform.
**Priority**: Medium
**Benefit**: 7 **Penalty**: 5 **Cost**: 3 **Risk**: 2

#### 4.6.2 Stimulus/Response Sequences

- Transaction completed → System prompts both users to rate and leave feedback → Rating is submitted and shown on profile.

#### 4.6.3 Functional Requirements

- **REQ-26**: The system shall allow users to rate other users (1–5 stars) after a completed transaction.
- **REQ-27**: The system shall allow text reviews with optional anonymity.
- **REQ-28**: The system shall display average rating and review count on the user profile.
- **REQ-29**: The system shall allow users to view reviews before contacting a

seller or buyer.

## 4.7 AI-Powered Fraud Detection

### 4.7.1 Description and Priority

Detects suspicious listings or activity using ML models and flags them for admin review.
**Priority**: Medium
**Benefit**: 8 **Penalty**: 9 **Cost**: 6 **Risk**: 7

### 4.7.2 Stimulus/Response Sequences

- New listing created → AI module scans for anomalies → Suspicious listings are auto-flagged → Admin receives notification for review.

### 4.7.3 Functional Requirements

- **REQ-30**: The system shall evaluate each listing using an AI fraud detection model.
- **REQ-31**: The system shall flag listings with suspicious pricing or duplicate patterns.
- **REQ-32**: The system shall allow admins to view flagged listings in a dashboard.
- **REQ-33**: The system shall allow users to manually report listings or users.
- **REQ-34**: The system shall periodically retrain the AI model with new data (future scope).

## 4.8 Notifications

### 4.8.1 Description and Priority

Keeps users informed of new messages, listings, price drops, and system alerts via in-app and email notifications.
**Priority**: Medium
**Benefit**: 6 **Penalty**: 4 **Cost**: 2 **Risk**: 1

### 4.8.2 Stimulus/Response Sequences

- New message received → Notification bell icon updates → Email and/or push alert sent to user.
- Price drop or new listing alert → User receives notification if subscribed to similar item.

### 4.8.3 Functional Requirements

- **REQ-35**: The system shall send push notifications using Firebase Cloud Messaging.
- **REQ-36**: The system shall send email notifications using SendGrid or Mailgun.
- **REQ-37**: The system shall display unread notification count in the UI.
- **REQ-38**: The system shall allow users to mute or customize notification

preferences.

### 4.9 Admin Dashboard

#### 4.9.1 Description and Priority

Admins manage reported listings, users, flagged content, and view platform analytics.
**Priority**: High
**Benefit**: 8 **Penalty**: 6 **Cost**: 4 **Risk**: 5

#### 4.9.2 Stimulus/Response Sequences

- Admin logs in → Views flagged content → Takes action (delete, warn, ban user) → Action logged.
- Admin views analytics dashboard → Sees active users, number of listings, and payment volume.

#### 4.9.3 Functional Requirements

- **REQ-39**: The system shall allow admins to log in via a secure admin interface.
- **REQ-40**: The system shall allow admins to view and moderate reported listings or users.
- **REQ-41**: The system shall provide an analytics dashboard showing platform usage statistics.
- **REQ-42**: The system shall allow banning or warning users.
- **REQ-43**: The system shall log all admin actions for auditing.

# Other Nonfunctional Requirements

## Performance Requirements

- The system shall support up to **500 concurrent users** without performance degradation.
- Average page load time shall not exceed **2 seconds** on standard internet connections.
- Chat messages shall be delivered in **real-time (<1 second latency)** via WebSocket communication.
- The backend must respond to standard API requests within **500 milliseconds** under normal load.

## Safety Requirements

- The platform does not involve physical components, but it must prevent **data loss**, **unauthorized payments**, or **fraudulent listings**.
- Payment operations will be handled by **Stripe**, ensuring PCI compliance.
- Admins shall review suspicious listings flagged by the AI model before removal to avoid false positives.

- All transactions will be **reversible or refundable** under defined conditions to protect users.

## Security Requirements

- All communication between client and server shall occur over **HTTPS** using SSL/TLS encryption.
- Users must be authenticated via **JWT tokens** after login.
- OAuth integration must follow **Google/Facebook API security protocols**.
- Passwords shall be stored using **secure hashing algorithms (e.g., bcrypt)**.
- User data shall comply with **data privacy regulations (e.g., GDPR or university IT policy)**.
- Admin functions must be protected by **role-based access control (RBAC)**.

## Software Quality Attributes

- **Usability**: The platform should be intuitive and simple to navigate for non-technical users (students).
- **Reliability**: System uptime shall be **99.5%** or higher.
- **Maintainability**: Codebase shall follow modular design for easy updates.
- **Scalability**: The backend must support **horizontal scaling** through Docker containers.
- **Portability**: The system should run consistently across macOS (dev) and Linux (prod).
- **Testability**: Automated tests shall be written for critical features like login, payments, and listings.

## Business Rules

- Only users with a **verified university email address** may create accounts or post listings.
- Buyers and sellers must complete the transaction on the platform for ratings to be enabled.
- Admins have the authority to **ban users**, **remove listings**, or **suspend accounts** for policy violations.
- Payment release will occur **only after buyer confirmation** or **auto-release after 7 days**, whichever comes first.
- One user account per verified email — **duplicate or fake profiles are prohibited**.

# Other Requirements

<Define any other requirements not covered elsewhere in the SRS. This might include database requirements, internationalization requirements, legal requirements, reuse objectives for the project, and so on. Add any new sections that are pertinent to the project.>

## Appendix A: Glossary

<Define all the terms necessary to properly interpret the SRS, including acronyms and abbreviations. You may wish to build a separate glossary that spans multiple projects or the entire organization, and just include terms specific to a single project in each SRS.>

## Appendix B: Analysis Models

<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams.>

## Appendix C: To Be Determined List

<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>