Software Requirements Specification (SRS)

SkillBridge - Developer Collaboration Platform

Version: 1.0
Date: February 8, 2026
Prepared by: Hanish Singhal
Project: SkillBridge

Table of Contents

---

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document provides a complete description of the SkillBridge platform. It details the functional and non-functional requirements, system architecture, and design constraints for developers, stakeholders, and project managers.

Intended Audience:

- Development Team
- Project Managers
- Quality Assurance Team
- System Architects
- Stakeholders
- End Users (Developers, Recruiters)

1.2 Scope

Product Name: SkillBridge
Product Type: Web-based Developer Collaboration Platform

Primary Goals:

- Connect developers with project opportunities
- Facilitate team collaboration and project management
- Provide portfolio building capabilities
- Enable skill verification through peer reviews
- Support real-time communication between team members

Key Benefits:

- Developers gain practical project experience
- Project creators find skilled team members
- Recruiters discover verified talent
- Teams collaborate efficiently with integrated tools
- Users build professional portfolios

1.3 Definitions, Acronyms, and Abbreviations

Term	Definition
SRS	Software Requirements Specification
API	Application Programming Interface
JWT	JSON Web Token
MERN	MongoDB, Express.js, React, Node.js
OAuth	Open Authorization
REST	Representational State Transfer
CRUD	Create, Read, Update, Delete
UI	User Interface
UX	User Experience
RBAC	Role-Based Access Control
WebSocket	Full-duplex communication protocol
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
SMTP	Simple Mail Transfer Protocol
NoSQL	Not Only SQL (database)
CDN	Content Delivery Network

1.4 References

- IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications
- MERN Stack Documentation

- React 19 Documentation
- MongoDB Documentation
- Socket.io Documentation
- Razorpay API Documentation
- Cloudinary API Documentation

1.5 Overview

This SRS document is organized into seven main sections:

- **Section 1:** Introduction and scope
- **Section 2:** Overall system description and context
- **Section 3:** Detailed system features and functional requirements
- **Section 4:** External interface specifications
- **Section 5:** Non-functional requirements (performance, security, etc.)
- **Section 6:** Other requirements and constraints
- **Section 7:** Appendices and supporting information

---

2. Overall Description

2.1 Product Perspective

SkillBridge is a standalone web application that operates within the broader ecosystem of developer collaboration tools. The system consists of:

System Components:

1. **Frontend Web Application** (React-based SPA)
2. **Backend API Server** (Node.js/Express)
3. **Database Server** (MongoDB Atlas)
4. **Real-time Communication Server** (Socket.io)
5. **Cloud Storage Service** (Cloudinary)
6. **Payment Gateway Integration** (Razorpay)
7. **Email Service** (Nodemailer with SMTP)

System Context:

- Operates independently without dependencies on other platforms
- Integrates with external services: Google OAuth, Cloudinary, Razorpay
- Accessible via web browsers (Chrome, Firefox, Safari, Edge)
- Mobile-responsive design for tablet and smartphone access

2.2 Product Functions

Core Functionalities:

1. **User Management**
    - User registration and authentication
    - Profile creation and management
    - Google OAuth integration
    - Password recovery and reset
2. **Project Management**
    - Create, read, update, delete projects
    - Browse and search projects
    - Filter projects by skills, difficulty, status
    - AI-powered project recommendations
3. **Team Collaboration**
    - Join request submission
    - Request acceptance/rejection
    - Team member management
    - Role-based permissions
4. **Task Management**
    - Kanban board for task organization
    - Drag-and-drop task manipulation
    - Task assignment and tracking
    - Task comments and attachments
5. **Communication**
    - Real-time chat messaging
    - File sharing in chats
    - Online status indicators
    - Message history
6. **Professional Networking**
    - Talent search and discovery
    - Public profile portfolios
    - Review and rating system
    - Trust badge system
7. **Notifications**

- Real-time activity notifications
- Email notifications
- In-app notification center

8. **Payment Processing**
    - Razorpay payment integration
    - Transaction management
    - Credit system for project creation
9. **Administrative Functions**
    - User and content moderation
    - Report management
    - Platform statistics
    - Admin action logging

2.3 User Classes and Characteristics

2.3.1 Developer (Primary User)

Characteristics:

- Technical background in software development
- Seeking project experience or collaboration
- Age range: 18-45 years
- Tech-savvy, comfortable with web applications
- Values: Learning, networking, portfolio building

Expertise Level: Beginner to Advanced

Frequency of Use: Daily to weekly

2.3.2 Project Creator

Characteristics:

- Has project ideas requiring team collaboration
- May or may not be a developer
- Seeking skilled contributors
- Needs project management tools

Expertise Level: Intermediate to Advanced

Frequency of Use: Weekly to monthly

2.3.3 Recruiter/Hiring Manager

Characteristics:

- Looking for verified talent
- Reviews portfolios and project history
- Assesses collaboration skills
- Makes hiring decisions

Expertise Level: Non-technical to technical

Frequency of Use: As needed (irregular)

2.3.4 Administrator

Characteristics:

- Platform management responsibility
- Handles moderation and reports
- Technical expertise in system administration
- Ensures platform quality and safety

Expertise Level: Advanced

Frequency of Use: Daily

2.4 Operating Environment

Client-Side:

- **Web Browsers:** Chrome 90+, Firefox 88+, Safari 14+, Edge 90+
- **Operating Systems:** Windows 10/11, macOS 11+, Linux (Ubuntu 20.04+), iOS 14+, Android 10+
- **Screen Resolutions:** 320px (mobile) to 2560px+ (desktop)
- **JavaScript:** Enabled and ECMAScript 2015+ support required

Server-Side:

- **Operating System:** Linux (Ubuntu 20.04 LTS or higher)
- **Runtime:** Node.js 18.x or higher
- **Database:** MongoDB 5.0 or higher
- **Web Server:** Nginx or Apache (reverse proxy)
- **SSL/TLS:** Required for HTTPS

Third-Party Services:

- **Cloud Storage:** Cloudinary
- **Payment Gateway:** Razorpay
- **Email Service:** SMTP-compatible service
- **OAuth Provider:** Google

2.5 Design and Implementation Constraints

2.5.1 Technology Constraints

- **Frontend:** Must use React 19.x
- **Backend:** Must use Node.js with Express.js
- **Database:** Must use MongoDB (NoSQL)
- **Real-time:** Must use WebSocket (Socket.io)

2.5.2 Regulatory Constraints

- GDPR compliance for EU users
- Data protection and privacy laws
- Payment processing compliance (PCI DSS)
- Accessibility standards (WCAG 2.1 Level AA)

2.5.3 Security Constraints

- HTTPS encryption required for all communications
- Password hashing using bcryptjs
- JWT token-based authentication
- HttpOnly cookies for session management
- Input validation and sanitization

2.5.4 Business Rules

- Users must be authenticated to access most features
- Project creation requires payment (₹1.00)
- Users can only edit/delete their own projects
- Administrators have override permissions
- Reviews can only be given after project completion

2.6 Assumptions and Dependencies

Assumptions:

- Users have stable internet connection (minimum 2 Mbps)
- Users have modern web browsers with JavaScript enabled
- Users have valid email addresses for registration
- Third-party services (Cloudinary, Razorpay) remain operational
- MongoDB Atlas provides reliable database hosting

Dependencies:

- **External APIs:**
  - Google OAuth API
  - Cloudinary API
  - Razorpay Payment API
  - SMTP service for emails
- **Libraries and Frameworks:**
  - React and React Router
  - Express.js
  - Mongoose ORM
  - Socket.io
  - Tailwind CSS
  - Framer Motion
- **Infrastructure:**
  - MongoDB Atlas cloud hosting
  - Cloudinary cloud storage
  - Deployment platform (Vercel/Heroku)

---

3. System Features

3.1 User Authentication and Authorization

3.1.1 Description

Secure user registration, login, and session management with role-based access control.

3.1.2 Functional Requirements

FR-AUTH-001: User Registration

- **Priority:** High
- **Description:** The system shall allow users to register with email and password
- **Input:** Name, email, password, confirm password
- **Process:** Validate input, hash password, create user record, send verification email

- **Output:** User account created, confirmation message displayed

FR-AUTH-002: User Login

- **Priority:** High
- **Description:** The system shall authenticate users via email/password or Google OAuth
- **Input:** Email and password OR Google account credentials
- **Process:** Verify credentials, generate JWT token, set HttpOnly cookie
- **Output:** Authenticated session, redirect to dashboard

FR-AUTH-003: Google OAuth Integration

- **Priority:** Medium
- **Description:** The system shall support Google sign-in via OAuth 2.0
- **Input:** Google account authorization
- **Process:** OAuth flow, retrieve user info, create/update account, authenticate
- **Output:** Authenticated session

FR-AUTH-004: Password Reset

- **Priority:** Medium
- **Description:** The system shall allow users to reset forgotten passwords
- **Input:** Email address
- **Process:** Generate reset token, send email with reset link, verify token, update password
- **Output:** Password updated, confirmation message

FR-AUTH-005: Session Management

- **Priority:** High
- **Description:** The system shall maintain secure user sessions using JWT
- **Input:** JWT token from cookie
- **Process:** Verify token validity, extract user info, check permissions
- **Output:** Authorized access to protected resources

FR-AUTH-006: Role-Based Access Control

- **Priority:** High
- **Description:** The system shall enforce permissions based on user roles
- **Input:** User role (User, Admin, Moderator)
- **Process:** Check role against required permissions for action
- **Output:** Access granted or denied

3.2 User Profile Management

3.2.1 Description

Comprehensive user profile creation and management with skills, social links, and portfolio information.

3.2.2 Functional Requirements

FR-PROFILE-001: Profile Creation

- **Priority:** High
- **Description:** The system shall allow users to create detailed profiles
- **Input:** Bio, location, timezone, skills, social links, avatar
- **Process:** Validate and store profile information
- **Output:** Profile created and displayed

FR-PROFILE-002: Avatar Upload

- **Priority:** Medium
- **Description:** The system shall support profile picture upload via Cloudinary
- **Input:** Image file (PNG, JPG, max 5MB)
- **Process:** Upload to Cloudinary, store URL in database
- **Output:** Avatar displayed on profile

FR-PROFILE-003: Skills Management

- **Priority:** High
- **Description:** The system shall allow users to add, update, and remove skills
- **Input:** Skill name, proficiency level
- **Process:** Validate skill, add to user profile
- **Output:** Skills displayed on profile

FR-PROFILE-004: Social Links

- **Priority:** Medium
- **Description:** The system shall support GitHub, LinkedIn, and portfolio links
- **Input:** Valid URLs for social platforms
- **Process:** Validate URL format, store in profile
- **Output:** Social links displayed with icons

FR-PROFILE-005: Public Portfolio

- **Priority:** Medium
- **Description:** The system shall generate shareable public profile pages
- **Input:** User ID or username
- **Process:** Retrieve user data, render public view

- **Output:** Public profile page accessible via URL

FR-PROFILE-006: Profile Completion Tracking

- **Priority:** Low
- **Description:** The system shall calculate and display profile completion percentage
- **Input:** User profile data
- **Process:** Check for bio, skills (≥3), social links, projects
- **Output:** Completion percentage and checklist

3.3 Project Management

3.3.1 Description

Complete project lifecycle management including creation, discovery, team management, and collaboration.

3.3.2 Functional Requirements

FR-PROJECT-001: Project Creation

- **Priority:** High
- **Description:** The system shall allow users to create new projects
- **Input:** Title, description, required skills, difficulty, team size, budget
- **Process:** Validate input, deduct credit, create project record
- **Output:** Project created, owner assigned

FR-PROJECT-002: Project Listing

- **Priority:** High
- **Description:** The system shall display all active projects
- **Input:** Optional filters (skills, difficulty, status)
- **Process:** Query database with filters, paginate results
- **Output:** List of projects matching criteria

FR-PROJECT-003: Project Search

- **Priority:** Medium
- **Description:** The system shall support text-based project search
- **Input:** Search keywords
- **Process:** Full-text search in title and description
- **Output:** Relevant project results

FR-PROJECT-004: Advanced Filtering

- **Priority:** Medium
- **Description:** The system shall filter projects by multiple criteria
- **Input:** Difficulty level, required skills, status, duration
- **Process:** Apply filters to project query
- **Output:** Filtered project list

FR-PROJECT-005: Project Details View

- **Priority:** High
- **Description:** The system shall display comprehensive project information
- **Input:** Project ID
- **Process:** Retrieve project, team, tasks, and activity data
- **Output:** Detailed project page with all information

FR-PROJECT-006: Project Update

- **Priority:** High
- **Description:** The system shall allow project owners to edit project details
- **Input:** Updated project fields
- **Process:** Verify ownership, validate input, update record
- **Output:** Project updated, changes reflected

FR-PROJECT-007: Project Deletion

- **Priority:** Medium
- **Description:** The system shall allow project owners to delete projects
- **Input:** Project ID
- **Process:** Verify ownership, remove project and related data
- **Output:** Project deleted, confirmation message

FR-PROJECT-008: AI Project Recommendations

- **Priority:** Medium
- **Description:** The system shall recommend projects based on user skills
- **Input:** User skills profile
- **Process:** Match user skills with project requirements, calculate match %
- **Output:** Ranked list of recommended projects

FR-PROJECT-009: My Projects View

- **Priority:** High
- **Description:** The system shall display projects owned or joined by user
- **Input:** User ID
- **Process:** Query projects where user is owner or member

- **Output:** Grid of user's projects

FR-PROJECT-010: Project Statistics

- **Priority:** Low
- **Description:** The system shall calculate user's project statistics
- **Input:** User ID
- **Process:** Count active, pending, and completed projects
- **Output:** Statistics displayed on dashboard

3.4 Team Collaboration

3.4.1 Description

Join request management, team member coordination, and role-based permissions.

3.4.2 Functional Requirements

FR-TEAM-001: Submit Join Request

- **Priority:** High
- **Description:** The system shall allow users to request joining projects
- **Input:** Project ID, cover letter message
- **Process:** Create request record, notify project owner
- **Output:** Request submitted, confirmation message

FR-TEAM-002: View Join Requests

- **Priority:** High
- **Description:** The system shall display incoming requests to project owners
- **Input:** Project ID
- **Process:** Query requests for project, display with user info
- **Output:** List of pending requests

FR-TEAM-003: Accept Join Request

- **Priority:** High
- **Description:** The system shall allow owners to accept join requests
- **Input:** Request ID
- **Process:** Add user to project members, update request status, notify user
- **Output:** User added to team, request marked accepted

FR-TEAM-004: Reject Join Request

- **Priority:** High
- **Description:** The system shall allow owners to reject join requests
- **Input:** Request ID
- **Process:** Update request status to rejected, notify user
- **Output:** Request marked rejected

FR-TEAM-005: View Team Members

- **Priority:** Medium
- **Description:** The system shall display all project team members
- **Input:** Project ID
- **Process:** Retrieve owner and assigned members
- **Output:** List of team members with roles

FR-TEAM-006: Remove Team Member

- **Priority:** Medium
- **Description:** The system shall allow owners to remove members
- **Input:** Project ID, user ID
- **Process:** Verify ownership, remove user from assignedTo array
- **Output:** Member removed, notification sent

3.5 Task Management (Kanban Board)

3.5.1 Description

Visual task organization with drag-and-drop functionality, task assignment, and progress tracking.

3.5.2 Functional Requirements

FR-TASK-001: Create Task

- **Priority:** High
- **Description:** The system shall allow team members to create tasks
- **Input:** Task title, description, column (status)
- **Process:** Validate input, add task to project
- **Output:** Task created, displayed on Kanban board

FR-TASK-002: Update Task Status

- **Priority:** High
- **Description:** The system shall support drag-and-drop task movement
- **Input:** Task ID, new column/status

- **Process:** Update task status, persist change, notify team
- **Output:** Task moved, board updated

FR-TASK-003: Edit Task

- **Priority:** Medium
- **Description:** The system shall allow task detail editing
- **Input:** Task ID, updated fields (title, description, assigned members)
- **Process:** Validate input, update task record
- **Output:** Task updated

FR-TASK-004: Delete Task

- **Priority:** Medium
- **Description:** The system shall allow task deletion
- **Input:** Task ID
- **Process:** Verify permissions, remove task
- **Output:** Task deleted from board

FR-TASK-005: Assign Task

- **Priority:** Medium
- **Description:** The system shall support task assignment to team members
- **Input:** Task ID, user IDs
- **Process:** Add users to task assignedTo field
- **Output:** Members assigned, notifications sent

FR-TASK-006: Task Comments

- **Priority:** Medium
- **Description:** The system shall support task-specific comments
- **Input:** Task ID, comment text
- **Process:** Add comment with timestamp and author
- **Output:** Comment displayed in task modal

FR-TASK-007: Task Attachments

- **Priority:** Low
- **Description:** The system shall support file attachments on tasks
- **Input:** Task ID, file upload
- **Process:** Upload to Cloudinary, link to task
- **Output:** Attachment displayed in task modal

FR-TASK-008: Real-time Sync

- **Priority:** High
- **Description:** The system shall synchronize board updates in real-time
- **Input:** Task modification event
- **Process:** Broadcast change via WebSocket to all team members
- **Output:** Board updated for all connected users

3.6 Real-Time Chat System

3.6.1 Description

Instant messaging with file sharing, online status, and conversation management.

3.6.2 Functional Requirements

FR-CHAT-001: Send Message

- **Priority:** High
- **Description:** The system shall allow users to send text messages
- **Input:** Recipient user ID, message text
- **Process:** Create message record, emit via Socket.io, store in DB
- **Output:** Message delivered in real-time

FR-CHAT-002: File Sharing

- **Priority:** Medium
- **Description:** The system shall support file attachments in chat
- **Input:** File upload (images, documents, max 10MB)
- **Process:** Upload to Cloudinary, attach URL to message
- **Output:** File shared, preview/download available

FR-CHAT-003: Conversation List

- **Priority:** High
- **Description:** The system shall display user's conversations
- **Input:** User ID
- **Process:** Query conversations where user is participant
- **Output:** List of conversations with last message preview

FR-CHAT-004: Message History

- **Priority:** High
- **Description:** The system shall load and display message history
- **Input:** Conversation ID

- **Process:** Query messages for conversation, paginate if needed
- **Output:** Chronological list of messages

FR-CHAT-005: Online Status

- **Priority:** Medium
- **Description:** The system shall show online/offline status
- **Input:** User connection state
- **Process:** Track Socket.io connections, broadcast status
- **Output:** Green/gray indicator on user avatar

FR-CHAT-006: Typing Indicator

- **Priority:** Low
- **Description:** The system shall show when user is typing
- **Input:** Typing event from client
- **Process:** Broadcast typing status to conversation
- **Output:** "User is typing..." indicator

FR-CHAT-007: Unread Count

- **Priority:** Medium
- **Description:** The system shall track unread message counts
- **Input:** Message read status
- **Process:** Count unread messages per conversation
- **Output:** Badge count displayed

3.7 Notifications System

3.7.1 Description

Real-time and email notifications for user activities and system events.

3.7.2 Functional Requirements

FR-NOTIF-001: Create Notification

- **Priority:** High
- **Description:** The system shall generate notifications for user actions
- **Input:** Event type, target user, related entity
- **Process:** Create notification record, emit via WebSocket
- **Output:** Notification created and delivered

FR-NOTIF-002: Notification Types

- **Priority:** High
- **Description:** The system shall support multiple notification types:
    - Join request received
    - Request accepted/rejected
    - New comment on project
    - Task assigned
    - New chat message
    - Review received

FR-NOTIF-003: Display Notifications

- **Priority:** High
- **Description:** The system shall show notifications in dropdown
- **Input:** User ID
- **Process:** Query unread notifications, display with icon
- **Output:** Notification list in navbar dropdown

FR-NOTIF-004: Mark as Read

- **Priority:** Medium
- **Description:** The system shall allow marking notifications as read
- **Input:** Notification ID
- **Process:** Update isRead flag to true
- **Output:** Notification marked read, badge count updated

FR-NOTIF-005: Email Notifications

- **Priority:** Medium
- **Description:** The system shall send email notifications for important events
- **Input:** Notification event, user email
- **Process:** Generate email template, send via Nodemailer
- **Output:** Email delivered to user

3.8 Review and Rating System

3.8.1 Description

Peer review submission and display for building trust and credibility.

3.8.2 Functional Requirements

FR-REVIEW-001: Submit Review

- **Priority:** Medium
- **Description:** The system shall allow users to review team members
- **Input:** Reviewee ID, project ID, rating (1-5), comment
- **Process:** Verify project completion, create review record
- **Output:** Review saved, reviewee notified

FR-REVIEW-002: View Reviews

- **Priority:** Medium
- **Description:** The system shall display user reviews on profiles
- **Input:** User ID
- **Process:** Query reviews for user, calculate average rating
- **Output:** Reviews displayed with stars and comments

FR-REVIEW-003: Review Restrictions

- **Priority:** Medium
- **Description:** The system shall enforce review rules:
    - Only after project completion
    - One review per project per user
    - Cannot review self

FR-REVIEW-004: Average Rating Calculation

- **Priority:** Low
- **Description:** The system shall calculate and display average ratings
- **Input:** User's received reviews
- **Process:** Calculate mean rating, count total reviews
- **Output:** Average rating displayed on profile

3.9 Payment Integration

3.9.1 Description

Razorpay integration for project creation fees and transaction management.

3.9.2 Functional Requirements

FR-PAY-001: Create Payment Order

- **Priority:** High
- **Description:** The system shall create Razorpay payment orders
- **Input:** Amount (₹1.00), currency (INR)
- **Process:** Call Razorpay API, generate order ID
- **Output:** Order created, ID returned to client

FR-PAY-002: Payment Checkout

- **Priority:** High
- **Description:** The system shall display Razorpay checkout modal
- **Input:** Order ID, user details
- **Process:** Load Razorpay SDK, open checkout UI
- **Output:** Payment interface displayed

FR-PAY-003: Payment Verification

- **Priority:** High
- **Description:** The system shall verify payment signatures
- **Input:** Payment ID, order ID, signature from Razorpay
- **Process:** Verify HMAC signature, validate payment
- **Output:** Payment confirmed or rejected

FR-PAY-004: Credit Allocation

- **Priority:** High
- **Description:** The system shall add credits upon successful payment
- **Input:** Verified payment
- **Process:** Increment user's projectCredits, create transaction record
- **Output:** Credit added, user can create project

FR-PAY-005: Transaction History

- **Priority:** Low
- **Description:** The system shall maintain payment transaction logs
- **Input:** Payment events
- **Process:** Store transaction with details
- **Output:** Transaction history available

3.10 Administrative Functions

3.10.1 Description

Platform moderation, user management, and analytics for administrators.

3.10.2 Functional Requirements

FR-ADMIN-001: User Management

- **Priority:** Medium
- **Description:** The system shall allow admins to manage users
- **Input:** User ID, action (suspend, delete, change role)
- **Process:** Verify admin permissions, execute action
- **Output:** User account modified

FR-ADMIN-002: Report Management

- **Priority:** Medium
- **Description:** The system shall allow admins to review reports
- **Input:** N/A (automatic query)
- **Process:** Retrieve all reports, display with status
- **Output:** Report list with details

FR-ADMIN-003: Resolve Report

- **Priority:** Medium
- **Description:** The system shall allow admins to resolve reports
- **Input:** Report ID, resolution action
- **Process:** Update report status, take action if needed
- **Output:** Report marked resolved

FR-ADMIN-004: Platform Statistics

- **Priority:** Low
- **Description:** The system shall display platform analytics
- **Input:** Date range (optional)
- **Process:** Aggregate user count, projects, messages, etc.
- **Output:** Dashboard with statistics and charts

FR-ADMIN-005: Admin Logs

- **Priority:** Medium
- **Description:** The system shall log all admin actions
- **Input:** Admin action event
- **Process:** Create log entry with timestamp, admin ID, action
- **Output:** Audit trail maintained

---

4. External Interface Requirements

4.1 User Interfaces

4.1.1 General UI Requirements

UI-001: Responsive Design

- The system shall adapt to screen sizes from 320px (mobile) to 2560px+ (desktop)
- Breakpoints: Mobile (320-640px), Tablet (641-1024px), Desktop (1025px+)

UI-002: Theme Support

- The system shall support Light and Dark themes
- Theme preference shall persist across sessions
- Theme toggle shall be accessible in navbar

UI-003: Accessibility

- The system shall comply with WCAG 2.1 Level AA
- All interactive elements shall be keyboard accessible
- Proper ARIA labels and semantic HTML required

UI-004: Loading States

- The system shall display loading indicators for async operations
- Skeleton screens for data-heavy components
- Progress bars for file uploads

UI-005: Error Handling

- The system shall display user-friendly error messages
- Form validation feedback in real-time
- Toast notifications for success/error/info messages

4.1.2 Specific UI Components

Navigation Bar:

- Logo and brand name
- Main navigation links (Dashboard, Projects, Talent, Chat)
- Search bar (optional)
- Notification bell with badge
- User avatar with dropdown menu
- Theme toggle button

Dashboard:

- Welcome message with user name
- Statistics cards (Active Projects, Pending Tasks, Completed)

- Profile completion widget
- My Projects grid
- Suggested Projects (scrollable, max-height 400px)

Project Pages:

- Project listing with filters sidebar
- Project cards with image, title, skills, budget
- Project details with Kanban board, discussion, overview tabs
- Join/Apply button with modal form

Profile Pages:

- Avatar, name, bio, location
- Skills badges
- Social media links
- Project history
- Reviews section

Chat Interface:

- Conversation list sidebar
- Message area with scrollable history
- Message input with file upload button
- Online status indicators

4.2 Hardware Interfaces

No direct hardware interfaces required. The system operates entirely through web browsers and does not interact with client hardware beyond standard browser capabilities (camera for avatar upload, if browser permits).

4.3 Software Interfaces

4.3.1 Database Interface

Database: MongoDB (via MongoDB Atlas)

- **Connection:** Mongoose ODM
- **Protocol:** MongoDB Wire Protocol
- **Authentication:** Username/Password
- **Data Format:** BSON (Binary JSON)

Collections:

- users
- projects
- requests
- messages
- conversations
- notifications
- reviews
- reports
- skills
- transactions
- adminlogs

4.3.2 External API Interfaces

Google OAuth API

- **Purpose:** User authentication via Google accounts
- **Protocol:** OAuth 2.0
- **Endpoints:**
    - Authorization: `https://accounts.google.com/o/oauth2/v2/auth`
    - Token: `https://oauth2.googleapis.com/token`
- **Data Format:** JSON
- **Authentication:** Client ID and Secret

Cloudinary API

- **Purpose:** Image and file storage
- **Protocol:** REST API over HTTPS
- **Endpoints:**
    - Upload: `https://api.cloudinary.com/v1_1/{cloud_name}/upload`
    - Transformation: Dynamic URLs
- **Data Format:** JSON, multipart/form-data
- **Authentication:** API Key and Secret

Razorpay API

- **Purpose:** Payment processing
- **Protocol:** REST API over HTTPS
- **Endpoints:**
    - Create Order: `POST /v1/orders`
    - Verify Payment: Server-side signature verification

- **Data Format:** JSON
- **Authentication:** Key ID and Secret

SMTP Email Service

- **Purpose:** Transactional emails
- **Protocol:** SMTP over TLS
- **Port:** 587 or 465
- **Authentication:** Username and Password
- **Libraries:** Nodemailer

4.3.3 WebSocket Interface

Socket.io

- **Purpose:** Real-time bidirectional communication
- **Protocol:** WebSocket with fallback to HTTP long-polling
- **Events:**
  - `connection` - User connects
  - `disconnect` - User disconnects
  - `sendMessage` - Send chat message
  - `receiveMessage` - Receive chat message
  - `taskUpdate` - Kanban board update
  - `newNotification` - New notification
  - `typing` - User typing indicator
  - `online` - User online status

4.4 Communication Interfaces

4.4.1 HTTP/HTTPS

- **Protocol:** HTTP/1.1, HTTP/2
- **Security:** TLS 1.2 or higher required
- **Ports:** 80 (HTTP redirect), 443 (HTTPS)
- **Methods:** GET, POST, PUT, DELETE, PATCH
- **Content Types:** application/json, multipart/form-data

4.4.2 WebSocket

- **Protocol:** ws:// (development), wss:// (production)
- **Port:** Same as HTTP server (typically 443 for production)
- **Upgrade:** HTTP Upgrade header required
- **Heartbeat:** Ping/pong every 30 seconds

---

5. Non-Functional Requirements

5.1 Performance Requirements

NFR-PERF-001: Response Time

- API responses shall return within 2 seconds for 95% of requests
- Page load time shall not exceed 3 seconds on standard broadband
- Real-time messages shall be delivered within 100ms under normal load

NFR-PERF-002: Throughput

- The system shall support at least 1000 concurrent users
- The system shall handle 100 requests per second minimum
- WebSocket server shall maintain 500+ concurrent connections

NFR-PERF-003: Database Performance

- Database queries shall execute within 500ms for 95% of operations
- Index optimization for frequently queried fields
- Caching strategy for repeated queries

NFR-PERF-004: File Upload

- Image uploads shall complete within 5 seconds for files up to 5MB
- File uploads shall support resumable uploads for files > 10MB
- Progress indicators shall update at least every 500ms

NFR-PERF-005: Scalability

- System architecture shall support horizontal scaling
- Database shall support sharding for future growth
- Load balancing capability for API servers

5.2 Safety Requirements

NFR-SAFE-001: Data Backup

- Database backups shall occur daily
- Backup retention period of 30 days minimum
- Automated backup verification

NFR-SAFE-002: Disaster Recovery

- Recovery Time Objective (RTO): 4 hours
- Recovery Point Objective (RPO): 24 hours
- Documented disaster recovery plan

NFR-SAFE-003: Graceful Degradation

- System shall function with reduced features if third-party services fail
- File uploads shall queue if Cloudinary is unavailable
- Payments shall notify users if Razorpay is down

5.3 Security Requirements

NFR-SEC-001: Authentication

- Passwords shall be hashed using bcryptjs with minimum 10 salt rounds
- JWT tokens shall expire after 7 days
- Refresh token rotation for enhanced security

NFR-SEC-002: Authorization

- Role-Based Access Control (RBAC) enforced on all protected routes
- Principle of least privilege applied to all user roles
- API endpoint authorization checks before data access

NFR-SEC-003: Data Encryption

- All data transmission shall use HTTPS (TLS 1.2+)
- Sensitive data in database shall be encrypted at rest
- Payment information shall never be stored locally

NFR-SEC-004: Input Validation

- All user inputs shall be validated and sanitized
- XSS prevention through input escaping
- SQL/NoSQL injection prevention through parameterized queries

NFR-SEC-005: Session Management

- HttpOnly cookies for session tokens
- CSRF protection on state-changing operations
- Automatic session timeout after 30 minutes of inactivity

NFR-SEC-006: API Security

- Rate limiting: 100 requests per minute per IP
- API key authentication for admin endpoints
- Logging of all failed authentication attempts

NFR-SEC-007: File Upload Security

- File type validation (whitelist approach)
- Virus scanning for uploaded files (if implemented)
- File size limits enforced (5MB for images, 10MB for documents)

5.4 Software Quality Attributes

5.4.1 Reliability

- System uptime shall be 99.5% (excluding planned maintenance)
- Mean Time Between Failures (MTBF) > 720 hours
- Error rate shall be < 0.1% of requests

5.4.2 Availability

- Planned maintenance window: Maximum 2 hours per month
- Notification of maintenance at least 24 hours in advance
- Critical bugs shall be patched within 24 hours

5.4.3 Maintainability

- Code shall follow ESLint and Prettier standards
- Minimum 60% code coverage for unit tests
- Comprehensive API documentation using Swagger/OpenAPI
- Code comments for complex logic
- Modular architecture for easy updates

5.4.4 Portability

- Frontend code shall be browser-agnostic
- Backend shall run on Linux, Windows, and macOS
- Database migrations for schema changes
- Environment-based configuration (dev, staging, production)

5.4.5 Usability

- New users shall complete registration within 3 minutes
- Core features shall be accessible within 3 clicks
- Consistent UI/UX patterns across all pages
- Helpful onboarding tooltips and empty states

5.4.6 Testability

- Unit tests for business logic functions
- Integration tests for API endpoints
- End-to-end tests for critical user flows
- Automated testing in CI/CD pipeline

5.5 Accessibility Requirements

NFR-ACCESS-001: WCAG Compliance

- Comply with WCAG 2.1 Level AA standards
- Minimum color contrast ratio of 4.5:1
- Alt text for all images

NFR-ACCESS-002: Keyboard Navigation

- All functionality accessible via keyboard
- Visible focus indicators
- Logical tab order

NFR-ACCESS-003: Screen Reader Support

- Semantic HTML5 elements
- ARIA labels where necessary
- Descriptive link text

5.6 Internationalization Requirements

NFR-I18N-001: Language Support (Future)

- UI structure shall support multiple languages
- Date and time formats shall respect user locale
- Prepared for i18n library integration (e.g., react-i18next)

NFR-I18N-002: Character Encoding

- UTF-8 encoding throughout the application
- Support for international characters in names and content

---

6. Other Requirements

6.1 Legal and Compliance Requirements

Legal-001: Terms of Service

- Users must accept Terms of Service before registration
- ToS shall be clearly accessible from footer links
- ToS updates shall notify users

Legal-002: Privacy Policy

- Privacy policy shall comply with GDPR for EU users
- Clear data collection and usage disclosure
- User consent for data processing

Legal-003: Cookie Policy

- Notification of cookie usage on first visit
- Option to manage cookie preferences
- Essential cookies vs. non-essential distinction

Legal-004: Copyright and Intellectual Property

- User-generated content ownership remains with users
- Platform license to display user content
- DMCA takedown process for copyright violations

6.2 Data Retention Requirements

Data-001: User Data

- Active user data retained indefinitely
- Deleted account data purged after 30 days
- Right to data export (GDPR compliance)

Data-002: Message History

- Chat messages retained for 1 year
- Option to delete individual messages
- Conversation history export feature

Data-003: Logs

- Application logs retained for 90 days
- Admin action logs retained for 5 years
- Error logs retained for 1 year

6.3 Business Rules

Business-001: Project Creation Fee

- ₹1.00 fee required to create a project

- Fee non-refundable
- Purpose: Quality control and spam prevention

Business-002: Review Eligibility

- Reviews only allowed after project completion
- One review per user per project
- Cannot review own profile

Business-003: Project Ownership

- Only project owner can delete project
- Owner cannot be removed from project
- Ownership transfer not supported (v1.0)

---

7. Appendix

7.1 Glossary

Term	Definition
Kanban Board	Visual project management tool with columns representing workflow states
Markdown	Lightweight markup language for formatting text
OAuth	Open standard for access delegation
Portfolio	Collection of work samples and project history
Real-time	Data transmission and update with minimal latency
Socket.io	JavaScript library for real-time web applications
Trust Badge	Visual indicator of user verification or achievement
WebSocket	Protocol for full-duplex communication channels

7.2 Analysis Models

7.2.1 Use Case Diagram

Primary Actors:

- Developer
- Project Creator
- Recruiter
- Administrator

Key Use Cases:

- Register/Login
- Create/Browse Projects
- Join Project
- Manage Tasks (Kanban)
- Chat with Team
- Submit/View Reviews
- Search Talent
- Report Content (Admin)

7.2.2 Entity-Relationship Diagram

Key Entities:

- User
- Project
- Request
- Message
- Conversation
- Review
- Notification
- Report
- Transaction

Relationships:

- User creates Project (1:N)
- User submits Request (1:N)
- Project has Requests (1:N)
- User belongs to Projects (N:M)
- Conversation has Messages (1:N)
- User sends Messages (1:N)
- User receives Reviews (1:N)
- Project generates Notifications (1:N)

7.3 To Be Determined (TBD) Items

1. **AI Matching Algorithm:** Exact algorithm for project-skill matching score calculation
2. **Video Chat Integration:** Specific video conferencing provider (future feature)
3. **Mobile App:** Timeline and platform for native mobile app (React Native vs. Flutter)
4. **Advanced Analytics:** Metrics and KPIs for dashboard charts
5. **Gamification:** Badge system and achievement types
6. **Code Repository Integration:** Direct linking to GitHub/GitLab repositories
7. **Payment Expansion:** Additional payment methods beyond Razorpay
8. **Multi-language Support:** Specific languages to support first

Document Version History:

Version	Date	Author	Changes
1.0	Feb 8, 2026	Hanish Singhal	Initial SRS document created

Approval Signatures:

Role	Name	Signature	Date
Project Manager			
Lead Developer			
QA Lead			
Product Owner			

This Software Requirements Specification document is subject to updates and revisions as the project evolves. All stakeholders will be notified of major changes.