Software Requirements Specification

GigConnect: A Hyperlocal Freelance Marketplace

Version: 1.0
Date: November 9, 2025
Prepared by: Development Team
Project Type: MERN Stack Web Application

Table of Contents

1. Introduction

1.1 Purpose

This Software Requirements Specification (SRS) document provides a comprehensive description of the GigConnect platform - a hyperlocal freelance marketplace. It details the functional and non-functional requirements for the system, intended for developers, project managers, testers, and stakeholders involved in the development and deployment of the application.

1.2 Document Conventions

- **Priority Levels**: High (Critical), Medium (Important), Low (Desirable)
- **Requirement Format**: REQ-[Module]-[Number]
- **Text Formatting**:
  - Bold text indicates system components
  - Italics indicate user actions
  - Code blocks indicate technical specifications

1.3 Intended Audience and Reading Suggestions

This document is intended for:

- **Developers**: Focus on Sections 3, 4, and 5 for technical implementation details
- **Project Managers**: Review Sections 2 and 4 for scope and feature understanding
- **Testers**: Concentrate on Section 4 for functional requirements and test case development
- **Stakeholders**: Read Sections 1 and 2 for project overview and business value
- **UI/UX Designers**: Focus on Section 3.1 for interface requirements

1.4 Project Scope

GigConnect is a web-based platform designed to connect local communities with skilled freelancers. The system aims to:

- Streamline the process of finding and hiring local talent
- Provide freelancers with a centralized platform to showcase skills and find opportunities
- Enable secure transactions and transparent communication
- Build trust through community reviews and ratings
- Support hyperlocal service discovery based on geographic proximity

In Scope:

- User registration and authentication for dual roles (Client/Freelancer)
- Profile management and skill showcasing
- Gig posting, browsing, and application system
- Real-time messaging between clients and freelancers
- Secure payment processing with milestone support
- Review and rating system
- Admin dashboard for platform management

Out of Scope:

- Mobile native applications (iOS/Android)
- Video calling functionality
- Advanced AI-based matching algorithms
- Cryptocurrency payment options
- Multi-language support (initial release in English only)

1.5 References

- IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications
- MERN Stack Documentation (MongoDB, Express.js, React.js, Node.js)
- JWT Authentication Standards (RFC 7519)

- Razorpay/Stripe API Documentation
- Socket.IO Documentation for Real-time Communication
- Web Content Accessibility Guidelines (WCAG) 2.1

---

2. Overall Description

2.1 Product Perspective

GigConnect is a new, self-contained web application that operates independently within the freelance marketplace ecosystem. The system consists of:

- **Frontend**: React.js-based responsive web interface
- **Backend**: Node.js with Express.js RESTful API server
- **Database**: MongoDB for data persistence
- **Real-time Layer**: Socket.IO for instant messaging and notifications
- **Payment Gateway**: Integration with Razorpay or Stripe
- **Cloud Storage**: For user profile images and portfolio files

The system interfaces with external services for payment processing but operates as a standalone platform for all core functionalities.

2.2 Product Functions

The major functions of GigConnect include:

1. **User Management**: Registration, authentication, and profile management for both clients and freelancers
2. **Gig Marketplace**: Creation, browsing, and application for freelance opportunities
3. **Discovery System**: Advanced search with hyperlocal filtering capabilities
4. **Communication**: Real-time messaging between platform users
5. **Transaction Processing**: Secure payment handling with milestone support
6. **Reputation System**: Bidirectional reviews and ratings
7. **Administrative Control**: Platform monitoring and management tools

2.3 User Classes and Characteristics

Primary User Classes:

1. **Clients (Service Seekers)**
    - Characteristics: Individuals or businesses seeking local freelance services
    - Technical Expertise: Basic to intermediate web usage
    - Frequency of Use: Occasional to frequent, project-based
    - Key Functions: Post gigs, search freelancers, manage payments, provide reviews

2. **Freelancers (Service Providers)**
    - Characteristics: Skilled professionals offering services locally
    - Technical Expertise: Basic to advanced web usage
    - Frequency of Use: Daily to monitor opportunities and manage work
    - Key Functions: Create profiles, browse gigs, communicate with clients, receive payments
3. **System Administrators**
    - Characteristics: Platform managers responsible for system health
    - Technical Expertise: Advanced technical and administrative skills
    - Frequency of Use: Regular monitoring and intervention as needed
    - Key Functions: User management, gig moderation, dispute resolution, platform analytics

2.4 Operating Environment

Client-Side Requirements:

- Modern web browsers: Chrome 90+, Firefox 88+, Safari 14+, Edge 90+
- Screen resolution: Minimum 320px width (mobile) to 1920px+ (desktop)
- Internet connection: Minimum 2 Mbps for optimal performance
- JavaScript enabled

Server-Side Requirements:

- Operating System: Linux (Ubuntu 20.04 LTS or higher recommended) or Windows Server
- Node.js: Version 16.x or higher
- MongoDB: Version 5.0 or higher
- RAM: Minimum 4GB (8GB recommended for production)
- Storage: Minimum 20GB with scalability for user-generated content

Deployment Environment:

- Cloud hosting platform (AWS, Azure, Google Cloud, or similar)
- SSL/TLS certificate for HTTPS
- CDN for static asset delivery

2.5 Design and Implementation Constraints

1. **Technology Stack Constraints**:
    - Must use MERN stack (MongoDB, Express.js, React.js, Node.js)
    - JavaScript/TypeScript as primary programming languages
2. **Security Constraints**:

   - JWT-based authentication required
   - PCI DSS compliance for payment processing
   - Data encryption for sensitive information
3. **Performance Constraints**:
   - Page load time must not exceed 3 seconds on standard broadband
   - Real-time message delivery within 1 second
   - Support for at least 1000 concurrent users
4. **Regulatory Constraints**:
   - GDPR compliance for user data protection
   - Local data protection regulations
   - Payment processing regulations
5. **Development Timeline**:
   - 4-week development cycle as per project plan
   - Weekly milestones and deliverables

2.6 User Documentation

The following user documentation will be provided:

6. **User Guide**: Comprehensive guide for clients and freelancers covering all platform features
7. **Quick Start Guide**: Brief introduction for new users
8. **FAQ Section**: Common questions and troubleshooting
9. **Video Tutorials**: Screen-recorded walkthroughs for key features
10. **API Documentation**: For potential future integrations (admin access only)
11. **Admin Manual**: Detailed documentation for platform administrators

2.7 Assumptions and Dependencies

Assumptions:

- Users have access to stable internet connectivity
- Users possess valid email addresses for registration
- Users have access to payment methods supported by Razorpay/Stripe
- Freelancers have necessary skills and credentials for offered services
- Geographic location services (GPS) are available for hyperlocal search

Dependencies:

- Third-party payment gateway availability (Razorpay/Stripe)
- Socket.IO library for real-time communication

- MongoDB database availability and performance
- Email service provider for notifications and verification
- Cloud storage service for file uploads
- SMS gateway for optional phone verification

---

3. External Interface Requirements

3.1 User Interfaces

General UI Requirements:

- Responsive design supporting mobile (320px+), tablet (768px+), and desktop (1024px+) screens
- Consistent navigation across all pages
- Accessibility compliance with WCAG 2.1 Level AA standards
- Intuitive user experience with minimal learning curve
- Loading indicators for asynchronous operations
- Clear error messages and validation feedback

Specific Interface Components:

1. **Authentication Pages**:
    - Registration form with role selection (Client/Freelancer)
    - Login page with email and password fields
    - Password recovery interface
    - Email verification screen
2. **Dashboard (Role-specific)**:
    - Client Dashboard: Active gigs, applications received, saved freelancers
    - Freelancer Dashboard: Applied gigs, active projects, earnings overview
    - Quick access to messaging and notifications
3. **Freelancer Profile Page**:
    - Profile photo, bio, and contact information
    - Skills tags and expertise level
    - Portfolio section with work samples
    - Service rates and availability
    - Reviews and ratings display
    - Edit functionality for profile owner

4. **Gig Posting/Browsing Interface**:
    - Gig creation form with title, description, requirements, budget, location
    - Main feed displaying available gigs with thumbnails
    - Detailed gig view with application button
    - Filter sidebar with multiple criteria
5. **Search & Filter Interface**:
    - Search bar with autocomplete
    - Filter options: location radius, skills, price range, ratings, availability
    - Map view showing freelancer/gig locations
    - Sort options: relevance, distance, price, rating
6. **Messaging Interface**:
    - Chat list showing all conversations
    - Individual chat window with message history
    - Real-time message updates
    - Typing indicators and read receipts
    - File sharing capability
7. **Payment Interface**:
    - Payment gateway integration screen
    - Milestone creation and tracking
    - Transaction history view
    - Invoice generation
8. **Review & Rating Interface**:
    - Star rating system (1-5 stars)
    - Text review submission form
    - Review display on profiles and gig pages
    - Report inappropriate reviews functionality
9. **Admin Dashboard**:
    - User management table with search and filter
    - Gig moderation queue
    - Analytics and reporting charts
    - System settings and configuration

3.2 Hardware Interfaces

GigConnect is a web-based application with no direct hardware interfaces. However, it indirectly interfaces with:

- **Client Devices**: Smartphones, tablets, laptops, and desktop computers running web browsers
- **Server Hardware**: Cloud server infrastructure for hosting the application
- **Storage Systems**: Database servers and file storage systems

3.3 Software Interfaces

Frontend Interfaces:

- **React.js** (v18.x): Core frontend framework
- **React Router**: Client-side routing
- **Axios**: HTTP client for API communication
- **Socket.IO Client**: Real-time communication

Backend Interfaces:

- **Node.js** (v16.x+): Runtime environment
- **Express.js** (v4.x): Web application framework
- **MongoDB** (v5.x): Database management system
    - Connection: MongoDB Atlas or self-hosted
    - Driver: Mongoose ODM for object modeling

Third-Party Service Interfaces:

- **Razorpay/Stripe API**: Payment processing
    - API Version: Latest stable
    - Communication: RESTful API over HTTPS
    - Data Format: JSON
- **Email Service** (e.g., SendGrid, AWS SES):
    - Protocol: SMTP or API-based
    - Purpose: Transactional emails and notifications
- **Cloud Storage** (e.g., AWS S3, Cloudinary):
    - Purpose: Store user uploads (images, documents)
    - Access: RESTful API
- **Maps API** (e.g., Google Maps, Mapbox):
    - Purpose: Location services and map visualization
    - Data Format: JSON

3.4 Communications Interfaces

HTTP/HTTPS Communication:

- Protocol: HTTPS (TLS 1.2 or higher)
- Port: 443 (standard HTTPS)
- Data Format: JSON for RESTful API
- Authentication: JWT tokens in Authorization headers

WebSocket Communication:

- Protocol: WSS (WebSocket Secure)
- Library: Socket.IO over HTTPS
- Purpose: Real-time messaging and notifications
- Connection: Persistent, bidirectional

Email Communication:

- Protocol: SMTP with TLS encryption
- Purpose: User notifications, verification emails
- Frequency: Event-driven (registration, gig updates, messages)

Database Communication:

- Protocol: MongoDB Wire Protocol
- Connection: Secured with authentication
- Connection Pooling: Enabled for performance

---

4. System Features

4.1 Dual-Role User Authentication

Priority: High

Description: A secure authentication system that allows users to register and login as either a Client or a Freelancer, with JWT-based session management.

Functional Requirements:

REQ-AUTH-001: The system shall allow users to register with email, password, full name, phone number, and role selection (Client or Freelancer).

- Input: Registration form data
- Processing: Validate data, hash password, create user record, send verification email
- Output: User account creation confirmation

REQ-AUTH-002: The system shall validate email addresses to ensure uniqueness and proper format.

- Validation: Email format regex, database uniqueness check

REQ-AUTH-003: The system shall enforce password complexity requirements (minimum 8 characters, including uppercase, lowercase, and numbers).

REQ-AUTH-004: The system shall send a verification email upon registration with a unique verification link.

- Expiration: Link valid for 24 hours

REQ-AUTH-005: The system shall authenticate users with email and password credentials.

- Output: JWT access token and refresh token upon successful authentication

REQ-AUTH-006: The system shall generate JWT tokens with user ID, role, and expiration time.

- Access Token Expiration: 1 hour
- Refresh Token Expiration: 7 days

REQ-AUTH-007: The system shall allow users to reset forgotten passwords via email verification.

- Process: Send reset link, validate link, allow password update

REQ-AUTH-008: The system shall implement role-based access control based on user type (Client/Freelancer/Admin).

REQ-AUTH-009: The system shall automatically logout users when tokens expire and prompt re-authentication.

REQ-AUTH-010: The system shall provide "Remember Me" functionality for extended sessions.

4.2 Freelancer Profile Management

Priority: High

Description: Comprehensive profile creation and management system for freelancers to showcase their skills, portfolio, rates, and receive reviews.

Functional Requirements:

REQ-PROFILE-001: The system shall allow freelancers to create detailed profiles including bio, skills, expertise level, and service categories.

- Maximum bio length: 1000 characters

REQ-PROFILE-002: The system shall allow freelancers to upload a profile picture.

- Supported formats: JPEG, PNG
- Maximum file size: 5MB
- Image dimensions: Minimum 200x200px

REQ-PROFILE-003: The system shall allow freelancers to add multiple skills with proficiency levels (Beginner, Intermediate, Expert).

- Maximum skills: 20 per profile

REQ-PROFILE-004: The system shall allow freelancers to set hourly rates or project-based pricing.

- Currency: Local currency with conversion display

REQ-PROFILE-005: The system shall allow freelancers to upload portfolio items with titles, descriptions, and images.

- Maximum portfolio items: 15
- Maximum 5 images per item

REQ-PROFILE-006: The system shall display aggregate ratings and individual reviews on freelancer profiles.

- Rating scale: 1-5 stars
- Display: Average rating, total reviews count

REQ-PROFILE-007: The system shall allow freelancers to set their location (city, address) for hyperlocal discovery.

- Location accuracy: City level mandatory, specific address optional

REQ-PROFILE-008: The system shall allow freelancers to indicate their availability status (Available, Busy, Unavailable).

REQ-PROFILE-009: The system shall display profile completion percentage to encourage complete profiles.

REQ-PROFILE-010: The system shall allow freelancers to edit and update their profile information at any time.

REQ-PROFILE-011: The system shall display freelancer response time and completion rate statistics.

4.3 Gig Posting & Management

Priority: High

Description: A comprehensive system for clients to create, manage, and track job postings (gigs) with clear requirements and budget specifications.

Functional Requirements:

REQ-GIG-001: The system shall allow clients to create new gigs with title, description, category, required skills, budget, and location.

- Title: Maximum 100 characters
- Description: Maximum 2000 characters

REQ-GIG-002: The system shall allow clients to specify gig type (one-time project, hourly, recurring).

REQ-GIG-003: The system shall allow clients to set budget ranges or fixed prices for gigs.

- Budget types: Fixed price, hourly rate, price range

REQ-GIG-004: The system shall allow clients to specify project duration and deadline.

REQ-GIG-005: The system shall allow clients to add attachments or reference files to gig postings.

- Maximum attachments: 5 files
- Maximum file size: 10MB per file
- Supported formats: PDF, DOC, DOCX, PNG, JPEG

REQ-GIG-006: The system shall allow clients to edit gig details before receiving applications.

REQ-GIG-007: The system shall allow clients to mark gigs as "Open", "In Progress", "Completed", or "Cancelled".

REQ-GIG-008: The system shall display a list of applicants for each gig with their profiles and proposals.

REQ-GIG-009: The system shall allow clients to shortlist, accept, or reject applications.

REQ-GIG-010: The system shall notify freelancers when their application status changes.

REQ-GIG-011: The system shall allow clients to close gig postings when no longer accepting applications.

REQ-GIG-012: The system shall maintain a history of all gigs posted by each client.

REQ-GIG-013: The system shall automatically close gigs after 90 days of inactivity unless renewed.

4.4 Advanced Search & Filtering

Priority: High

Description: A powerful search and discovery system with hyperlocal capabilities, enabling users to find freelancers or gigs based on multiple criteria including location, skills, price, and ratings.

Functional Requirements:

REQ-SEARCH-001: The system shall provide a search bar for keyword-based search of freelancers and gigs.

- Search scope: Titles, descriptions, skills, categories

REQ-SEARCH-002: The system shall implement hyperlocal search based on user's current location or specified location.

- Location detection: GPS or manual entry
- Distance filter: 5km, 10km, 25km, 50km, 100km radius options

REQ-SEARCH-003: The system shall allow filtering by specific skills or skill categories.

- Multiple skill selection: Supported with OR logic

REQ-SEARCH-004: The system shall allow filtering by price range (minimum and maximum).

REQ-SEARCH-005: The system shall allow filtering by user ratings (minimum rating threshold).

- Filter options: 3+, 4+, 4.5+ stars

REQ-SEARCH-006: The system shall allow filtering by availability status for freelancers.

REQ-SEARCH-007: The system shall allow filtering by gig type (one-time, hourly, recurring).

REQ-SEARCH-008: The system shall provide sorting options (relevance, distance, price, rating, newest).

REQ-SEARCH-009: The system shall display search results with pagination (20 results per page).

REQ-SEARCH-010: The system shall show distance from the user for each freelancer/gig in search results.

REQ-SEARCH-011: The system shall allow users to save search filters as presets for quick access.

REQ-SEARCH-012: The system shall display an interactive map view showing freelancer locations.

REQ-SEARCH-013: The system shall implement autocomplete suggestions for search queries based on popular searches and existing data.

REQ-SEARCH-014: The system shall display "no results found" message with suggestions when search yields no results.

4.5 Real-time Messaging System

Priority: High

Description: An integrated, real-time chat system enabling seamless communication between clients and freelancers throughout the engagement process.

Functional Requirements:

REQ-MSG-001: The system shall provide a messaging interface accessible from user dashboards.

REQ-MSG-002: The system shall allow users to initiate conversations with other users (clients to freelancers and vice versa).

REQ-MSG-003: The system shall deliver messages in real-time using WebSocket technology.

- Maximum delivery latency: 1 second under normal conditions

REQ-MSG-004: The system shall display message read receipts when a message has been viewed.

REQ-MSG-005: The system shall display typing indicators when the other party is composing a message.

REQ-MSG-006: The system shall maintain conversation history for all users.

- History retention: Indefinite or until account deletion

REQ-MSG-007: The system shall display timestamps for each message.

- Format: Relative time for recent messages, absolute time for older messages

REQ-MSG-008: The system shall allow users to share files through the messaging system.

- Maximum file size: 25MB
- Supported formats: Images, PDFs, documents

REQ-MSG-009: The system shall show a list of all active conversations with the most recent message preview.

REQ-MSG-010: The system shall display unread message indicators and counts.

REQ-MSG-011: The system shall send push notifications for new messages when the user is not actively viewing the chat.

REQ-MSG-012: The system shall allow users to search within conversation history.

REQ-MSG-013: The system shall allow users to block or report other users for inappropriate behavior.

REQ-MSG-014: The system shall maintain separate conversation threads for each client-freelancer pair per gig.

REQ-MSG-015: The system shall support emoji reactions to messages.

4.6 Review & Rating System

Priority: High

Description: A bidirectional feedback system allowing both clients and freelancers to rate and review each other after project completion, building trust and reputation on the platform.

Functional Requirements:

REQ-REVIEW-001: The system shall allow clients to rate freelancers on a 5-star scale after gig completion.

- Rating categories: Quality, Communication, Timeliness, Professionalism

REQ-REVIEW-002: The system shall allow freelancers to rate clients on a 5-star scale after gig completion.

- Rating categories: Communication, Payment promptness, Clarity of requirements

REQ-REVIEW-003: The system shall allow users to write text reviews accompanying their ratings.

- Maximum review length: 500 characters
- Minimum review length: Optional, but encouraged

REQ-REVIEW-004: The system shall only allow reviews after a gig has been marked as completed.

REQ-REVIEW-005: The system shall allow one review per user per completed gig.

REQ-REVIEW-006: The system shall display average ratings prominently on user profiles.

- Display format: Stars with numerical value (e.g., 4.7/5.0)

REQ-REVIEW-007: The system shall display all individual reviews on user profiles in chronological order (newest first).

REQ-REVIEW-008: The system shall calculate and display rating breakdowns (5-star, 4-star, 3-star, etc.).

REQ-REVIEW-009: The system shall send notifications to users when they receive new reviews.

REQ-REVIEW-010: The system shall allow users to respond to reviews they have received.

- Response character limit: 300 characters

REQ-REVIEW-011: The system shall allow users to report inappropriate or fake reviews for admin moderation.

REQ-REVIEW-012: The system shall prevent users from editing reviews after submission (but allow deletion within 24 hours).

REQ-REVIEW-013: The system shall display review verification badges for confirmed completed projects.

REQ-REVIEW-014: The system shall factor ratings into search result rankings.

4.7 Secure Payment Integration

Priority: High

Description: Integration with third-party payment gateways (Razorpay/Stripe) to facilitate secure, transparent transactions between clients and freelancers with support for milestone-based payments.

Functional Requirements:

REQ-PAY-001: The system shall integrate with Razorpay or Stripe for payment processing.

- Supported payment methods: Credit/Debit cards, UPI, Net Banking, Wallets

REQ-PAY-002: The system shall allow clients to fund gigs by depositing payments into an escrow system.

REQ-PAY-003: The system shall support milestone-based payment structures for projects.

- Maximum milestones per gig: 10
- Milestone definition: Description, amount, deadline

REQ-PAY-004: The system shall release milestone payments to freelancers upon client approval.

REQ-PAY-005: The system shall hold funds in escrow until work is delivered and approved.

REQ-PAY-006: The system shall automatically release funds after a specified period if no disputes are raised.

- Auto-release period: 7 days after delivery

REQ-PAY-007: The system shall charge platform fees on transactions.

- Fee structure: Configurable percentage (e.g., 5-10% of transaction value)
- Fee calculation: Transparent and displayed before transaction

REQ-PAY-008: The system shall provide transaction history for all users.

- History includes: Amount, date, gig details, status

REQ-PAY-009: The system shall generate invoices for completed transactions.

- Format: PDF download
- Contents: Transaction details, tax information, platform details

REQ-PAY-010: The system shall support refund processing for cancelled gigs or disputes.

- Refund processing time: As per payment gateway standards

REQ-PAY-011: The system shall send payment confirmation emails to both parties.

REQ-PAY-012: The system shall allow freelancers to add bank account details for payouts.

- Required information: Account number, IFSC/routing number, account holder name

REQ-PAY-013: The system shall facilitate automated payouts to freelancer accounts.

- Payout frequency: On-demand or scheduled (weekly/monthly)

REQ-PAY-014: The system shall maintain PCI DSS compliance for handling payment information.

REQ-PAY-015: The system shall display pending, completed, and failed payment statuses clearly.

REQ-PAY-016: The system shall implement fraud detection mechanisms in collaboration with payment gateways.

4.8 Admin Dashboard

Priority: Medium

Description: A comprehensive administrative interface for platform management, including user moderation, content management, analytics, and system configuration.

Functional Requirements:

REQ-ADMIN-001: The system shall provide a secure admin login separate from regular user authentication.

REQ-ADMIN-002: The system shall display a dashboard with key metrics (total users, active gigs, transactions, revenue).

REQ-ADMIN-003: The system shall allow admins to view, search, and filter all registered users.

- Search criteria: Name, email, role, registration date, status

REQ-ADMIN-004: The system shall allow admins to suspend or ban user accounts with reason documentation.

REQ-ADMIN-005: The system shall allow admins to view and moderate all gig postings.

- Actions: Approve, flag, remove, edit

REQ-ADMIN-006: The system shall provide a reported content queue for admin review.

- Content types: Users, gigs, reviews, messages

REQ-ADMIN-007: The system shall allow admins to view detailed transaction logs.

REQ-ADMIN-008: The system shall provide analytics on platform usage, popular categories, and user growth.

- Visualization: Charts and graphs with export capability

REQ-ADMIN-009: The system shall allow admins to send platform-wide announcements or notifications.

REQ-ADMIN-010: The system shall maintain an audit log of all admin actions with timestamps and admin IDs.

REQ-ADMIN-011: The system shall allow admins to configure platform settings (fees, policies, feature flags).

REQ-ADMIN-012: The system shall allow admins to manage skill categories and tags.

REQ-ADMIN-013: The system shall provide dispute resolution tools for payment and service disputes.

REQ-ADMIN-014: The system shall generate reports on demand (user reports, financial reports, activity reports).

- Export formats: PDF, CSV, Excel

REQ-ADMIN-015: The system shall implement role-based admin access (Super Admin, Moderator, Support).

5. Other Nonfunctional Requirements

5.1 Performance Requirements

REQ-PERF-001: The system shall support at least 1,000 concurrent users without performance degradation.

REQ-PERF-002: The system shall load pages within 3 seconds on a standard broadband connection (5 Mbps).

REQ-PERF-003: The system shall deliver real-time messages with a maximum latency of 1 second under normal network conditions.

REQ-PERF-004: The system shall perform search operations and return results within 2 seconds for datasets up to 100,000 records.

REQ-PERF-005: The system shall handle file uploads of up to 25MB with progress indication and completion within 30 seconds on standard connections.

REQ-PERF-006: The system shall maintain 99.5% uptime excluding scheduled maintenance windows.

REQ-PERF-007: The system shall scale horizontally to accommodate growing user base and data volume.

REQ-PERF-008: The system shall optimize database queries to complete within 500ms for 95% of operations.

REQ-PERF-009: The system shall implement caching mechanisms for frequently accessed data (user profiles, gig listings).

- Cache invalidation: Automatic on data updates

REQ-PERF-010: The system shall compress and optimize images automatically upon upload to reduce load times.

5.2 Safety Requirements

REQ-SAFE-001: The system shall implement automated data backup procedures.

- Backup frequency: Daily incremental, weekly full backup
- Backup retention: Minimum 30 days

REQ-SAFE-002: The system shall maintain transaction logs to enable data recovery in case of system failure.

REQ-SAFE-003: The system shall implement graceful error handling to prevent data loss during failures.

REQ-SAFE-004: The system shall validate all user inputs to prevent injection attacks and data corruption.

REQ-SAFE-005: The system shall implement circuit breakers for external service dependencies to prevent cascade failures.

5.3 Security Requirements

REQ-SEC-001: The system shall encrypt all passwords using bcrypt hashing with a minimum of 10 salt rounds.

REQ-SEC-002: The system shall transmit all data over HTTPS with TLS 1.2 or higher encryption.

REQ-SEC-003: The system shall implement JWT-based authentication with token expiration and refresh mechanisms.

REQ-SEC-004: The system shall protect against common web vulnerabilities (SQL injection, XSS, CSRF).

REQ-SEC-005: The system shall implement rate limiting on API endpoints to prevent DDoS attacks.

- Login attempts: Maximum 5 failed attempts per 15 minutes per IP
- API requests: Maximum 100 requests per minute per user

REQ-SEC-006: The system shall sanitize and validate all user-generated content before storage and display.

REQ-SEC-007: The system shall implement role-based access control (RBAC) to restrict access to sensitive operations.

REQ-SEC-008: The system shall not store sensitive payment information (credit card numbers, CVV) and shall use payment gateway tokenization.

REQ-SEC-009: The system shall implement Content Security Policy (CSP) headers to prevent XSS attacks.

REQ-SEC-010: The system shall log all security-relevant events (failed login attempts, unauthorized access attempts, data modifications).

REQ-SEC-011: The system shall implement two-factor authentication (2FA) as an optional security enhancement for user accounts.

REQ-SEC-012: The system shall expire user sessions after 30 minutes of inactivity.

REQ-SEC-013: The system shall mask sensitive information in logs and error messages.

REQ-SEC-014: The system shall implement API authentication for all backend endpoints except public routes.

REQ-SEC-015: The system shall conduct regular security audits and vulnerability assessments.

REQ-SEC-016: The system shall comply with OWASP Top 10 security best practices.

5.4 Software Quality Attributes

Usability:

REQ-QUAL-001: The system shall provide intuitive navigation with maximum 3 clicks to reach any major feature.

REQ-QUAL-002: The system shall provide consistent UI/UX across all pages and features.

REQ-QUAL-003: The system shall provide helpful error messages with clear guidance on resolution.

REQ-QUAL-004: The system shall include tooltips and contextual help for complex features.

REQ-QUAL-005: The system shall support keyboard navigation for accessibility.

REQ-QUAL-006: The system shall be usable without prior training for basic functions.

Reliability:

REQ-QUAL-007: The system shall have a Mean Time Between Failures (MTBF) of at least 720 hours (30 days).

REQ-QUAL-008: The system shall recover from failures within 5 minutes (Mean Time To Recovery).

REQ-QUAL-009: The system shall maintain data consistency across all operations.

REQ-QUAL-010: The system shall handle unexpected inputs gracefully without crashing.

Maintainability:

REQ-QUAL-011: The system shall follow modular architecture principles for easy maintenance and updates.

REQ-QUAL-012: The system code shall follow consistent coding standards and conventions.

REQ-QUAL-013: The system shall include comprehensive inline documentation and comments.

REQ-QUAL-014: The system shall use version control (Git) for all source code.

REQ-QUAL-015: The system shall maintain separate development, staging, and production environments.

Portability:

REQ-QUAL-016: The system shall be browser-agnostic and function correctly on all major browsers (Chrome, Firefox, Safari, Edge).

REQ-QUAL-017: The system shall be responsive and functional on various screen sizes (mobile, tablet, desktop).

REQ-QUAL-018: The system backend shall be deployable on different cloud platforms without major modifications.

Scalability:

REQ-QUAL-019: The system architecture shall support horizontal scaling to handle increased load.

REQ-QUAL-020: The system database shall be designed to handle growth to millions of records without performance degradation.

REQ-QUAL-021: The system shall implement microservices architecture principles where appropriate for independent scaling.

Accessibility:

REQ-QUAL-022: The system shall comply with WCAG 2.1 Level AA accessibility standards.

REQ-QUAL-023: The system shall provide adequate color contrast ratios (minimum 4.5:1 for normal text).

REQ-QUAL-024: The system shall support screen reader compatibility.

REQ-QUAL-025: The system shall provide alternative text for all images and icons.

REQ-QUAL-026: The system shall be fully navigable using keyboard only.

5.5 Business Rules

REQ-BUS-001: Users must be at least 18 years old to register on the platform (verified through date of birth).

REQ-BUS-002: Freelancers must complete at least 80% of their profile before appearing in search results.

REQ-BUS-003: Users can only have one active account per email address.

REQ-BUS-004: Gig payments must be deposited into escrow before freelancers can begin work.

REQ-BUS-005: Platform commission fees are non-refundable once a transaction is completed.

REQ-BUS-006: Reviews can only be submitted within 30 days of gig completion.

REQ-BUS-007: Users with an average rating below 2.0 stars may be subject to account review or suspension.

REQ-BUS-008: Freelancers must complete identity verification before receiving payments above a threshold amount (e.g., $500).

REQ-BUS-009: Disputed payments are held in escrow pending admin resolution, with a maximum resolution time of 14 days.

REQ-BUS-010: Users cannot delete their accounts if they have pending transactions or disputes.

REQ-BUS-011: Inactive gigs (no activity for 90 days) are automatically archived and removed from public listings.

REQ-BUS-012: Freelancers can apply to a maximum of 20 open gigs simultaneously.

REQ-BUS-013: Platform fees range from 5% to 15% based on transaction volume and user tier.

REQ-BUS-014: Refunds for cancelled gigs follow a tiered structure based on cancellation timing:

- Before work begins: 100% refund minus platform fee
- After work begins: Pro-rated based on milestones completed
- After completion: No refund unless disputed and approved

REQ-BUS-015: Users must accept the Terms of Service and Privacy Policy during registration.

---

6. Other Requirements

6.1 Legal and Compliance Requirements

REQ-LEGAL-001: The system shall comply with GDPR regulations for users in applicable regions.

- Right to access personal data
- Right to data portability
- Right to be forgotten (account deletion)

REQ-LEGAL-002: The system shall comply with local data protection and privacy laws.

REQ-LEGAL-003: The system shall maintain Terms of Service and Privacy Policy documents accessible to all users.

REQ-LEGAL-004: The system shall obtain explicit user consent for data collection and processing during registration.

REQ-LEGAL-005: The system shall comply with payment card industry standards (PCI DSS) for payment processing.

REQ-LEGAL-006: The system shall implement age verification mechanisms to comply with child protection laws.

REQ-LEGAL-007: The system shall maintain audit trails for all financial transactions for regulatory compliance.

REQ-LEGAL-008: The system shall provide mechanisms for legal authorities to request user data through proper channels.

6.2 Internationalization Requirements

REQ-I18N-001: The system shall support multiple currencies with real-time conversion rates (future enhancement).

REQ-I18N-002: The system shall use UTC for all timestamp storage and convert to user's local timezone for display.

REQ-I18N-003: The system architecture shall be designed to support future multi-language capabilities.

REQ-I18N-004: The system shall handle various date and time formats based on user locale (future enhancement).

6.3 Environmental Requirements

REQ-ENV-001: The system shall be hosted on cloud infrastructure with renewable energy commitments.

REQ-ENV-002: The system shall implement efficient resource utilization to minimize carbon footprint.

REQ-ENV-003: The system shall optimize data transfer to reduce bandwidth consumption and energy usage.

6.4 Installation and Deployment Requirements

REQ-DEPLOY-001: The system shall support containerized deployment using Docker.

REQ-DEPLOY-002: The system shall implement CI/CD pipelines for automated testing and deployment.

REQ-DEPLOY-003: The system shall support blue-green deployment strategy for zero-downtime updates.

REQ-DEPLOY-004: The system shall provide deployment documentation and scripts for easy setup.

REQ-DEPLOY-005: The system shall separate configuration from code using environment variables.

6.5 Support and Maintenance Requirements

REQ-SUPPORT-001: The system shall provide in-app help documentation and FAQ section.

REQ-SUPPORT-002: The system shall include a contact/support form for user inquiries.

REQ-SUPPORT-003: The system shall log errors and exceptions for troubleshooting and debugging.

REQ-SUPPORT-004: The system shall provide admin tools for user support (view user details, transaction history, message threads).

REQ-SUPPORT-005: The system shall implement a ticketing system for tracking user support requests.

REQ-SUPPORT-006: The system shall schedule regular maintenance windows during off-peak hours with advance user notification.

6.6 Training Requirements

REQ-TRAIN-001: Video tutorials shall be created for key user workflows (registration, posting gigs, applying to gigs, payments).

REQ-TRAIN-002: Admin training documentation shall be provided covering all admin dashboard features.

REQ-TRAIN-003: API documentation shall be created for potential future integrations.

REQ-TRAIN-004: The system shall include onboarding walkthroughs for new users highlighting key features.

Appendix A: Glossary

- **Gig**: A job posting or project listing created by a client seeking freelance services
- **Freelancer**: A service provider who offers skills and completes gigs for clients
- **Client**: A user who posts gigs and hires freelancers
- **Milestone**: A defined checkpoint in a project with associated payment and deliverables
- **Escrow**: A financial arrangement where payment is held by a third party until conditions are met
- **Hyperlocal**: Focused on a very small geographic area, typically within a few kilometers
- **JWT (JSON Web Token)**: A compact, URL-safe means of representing claims to be transferred between two parties
- **MERN Stack**: MongoDB, Express.js, React.js, Node.js - a JavaScript-based technology stack
- **Socket.IO**: A JavaScript library for real-time, bidirectional communication
- **PCI DSS**: Payment Card Industry Data Security Standard
- **WCAG**: Web Content Accessibility Guidelines
- **RBAC**: Role-Based Access Control
- **API**: Application Programming Interface
- **HTTPS**: Hypertext Transfer Protocol Secure
- **CRUD**: Create, Read, Update, Delete operations
- **UI/UX**: User Interface / User Experience

Appendix B: Analysis Models

B.1 Use Case Diagram Components

Primary Actors:

- Client
- Freelancer
- Administrator
- Payment Gateway (External System)

Key Use Cases:

- User Registration and Login
- Create/Manage Profile
- Post/Browse Gigs
- Search and Filter
- Apply to Gigs
- Real-time Messaging
- Process Payments
- Submit Reviews
- Moderate Platform (Admin)

B.2 Data Flow Overview

1. **User Authentication Flow**: User → Frontend → Backend API → Database → JWT Token → Frontend
2. **Gig Creation Flow**: Client → Form Input → API → Validation → Database → Confirmation
3. **Search Flow**: User Query → Search API → Database Query → Results Processing → Filtered Results → UI
4. **Payment Flow**: Client → Payment Intent → Payment Gateway → Confirmation → Escrow → Milestone Completion → Freelancer Payout
5. **Messaging Flow**: User → Message → Socket.IO Server → Recipient Client → Database Storage

B.3 Technology Stack Summary

Frontend:

- React.js 18.x
- React Router for navigation
- Axios for HTTP requests
- Socket.IO Client for real-time features
- Tailwind CSS / Material-UI for styling

Backend:

- Node.js 16.x+
- Express.js 4.x
- Socket.IO for WebSocket management
- JWT for authentication
- Bcrypt for password hashing
- Multer for file uploads

Database:

- MongoDB 5.x
- Mongoose ODM

Third-Party Services:

- Razorpay/Stripe for payments
- SendGrid/AWS SES for emails
- AWS S3/Cloudinary for file storage
- Google Maps API for location services

Appendix C: Development Timeline

Week 1: Foundation & Authentication

Backend:

- Project setup and architecture
- JWT authentication implementation
- User registration and login APIs
- Profile management APIs
- Gig CRUD APIs

Frontend:

- React project initialization
- Authentication UI (login, registration)
- Dashboard layout
- Freelancer profile pages
- Gig posting forms

Week 2: Search & Real-time Features

Backend:

- Advanced search and filtering logic
- Location-based search implementation
- Socket.IO setup for real-time chat
- Chat message APIs

Frontend:

- Main gig feed with search interface
- Filter components
- Real-time messaging UI
- Search results display

Week 3: Payments & Reviews

Backend:

- Payment gateway integration (Razorpay/Stripe)
- Escrow and milestone APIs
- Review and rating system APIs
- Transaction history APIs

Frontend:

- Payment interface integration
- Review submission and display UI
- Transaction history view
- Rating components

Week 4: Admin & Finalization

Backend:

- Admin dashboard APIs
- User management endpoints
- Analytics and reporting
- Final testing and optimization

Frontend:

- Admin panel UI
- Analytics dashboards
- Final UI polish and responsiveness
- Cross-browser testing

Appendix D: Risks and Mitigation Strategies

Technical Risks

Risk 1: Real-time messaging scalability

- Impact: High
- Probability: Medium
- Mitigation: Implement connection pooling, use Redis for Socket.IO scaling, conduct load testing

Risk 2: Payment gateway integration complexity

- Impact: High
- Probability: Medium
- Mitigation: Thorough API documentation review, sandbox testing, fallback payment methods

Risk 3: Database performance with large datasets

- Impact: Medium
- Probability: Medium
- Mitigation: Proper indexing, query optimization, implement caching, database sharding if needed

Security Risks

Risk 4: Data breaches and unauthorized access

- Impact: Critical
- Probability: Medium
- Mitigation: Encryption, regular security audits, penetration testing, secure coding practices

Risk 5: Payment fraud

- Impact: High
- Probability: Medium
- Mitigation: Fraud detection mechanisms, transaction monitoring, user verification

Business Risks

Risk 6: Low user adoption

- Impact: High
- Probability: Medium
- Mitigation: User-friendly design, effective onboarding, marketing strategy, competitive pricing

Risk 7: Legal and compliance issues

- Impact: High

- Probability: Low
- Mitigation: Legal consultation, compliance audits, clear Terms of Service, Privacy Policy

Operational Risks

Risk 8: Third-party service downtime

- Impact: Medium
- Probability: Medium
- Mitigation: Multiple service providers, fallback mechanisms, monitoring and alerts

Document Approval

This Software Requirements Specification has been reviewed and approved by:

Role	Name	Signature	Date
Project Manager	[Name]	__________	__________
Lead Developer	[Name]	__________	__________
QA Lead	[Name]	__________	__________
Product Owner	[Name]	__________	__________
Stakeholder	[Name]	__________	__________

Revision History

Version	Date	Author	Description
1.0	November 9, 2025	Development Team	Initial SRS document creation