**Functional Requirements**

# 目录

Functional Requirements ........................ 2

1. Functional User Requirements ........................ 3

1.1 User Roles ........................ 4

1.2 Customer (Homeowner) Requirements ........................ 5

1.3 Service Provider (Professional) Requirements ........................ 6

1.4 Admin (Platform Manager) Requirements ........................ 7

2. Functional System Requirements ........................ 8

3. Non-Functional Requirements ........................ 9

3.1 Performance Requirements ........................ 10

3.2 Security Requirements ........................ 11

3.3 Reliability Requirements ........................ 12

3.4 Usability Requirements ........................ 13

3.5 Maintainability Requirements ........................ 14

4. Requirements Traceability Matrix ........................ 15

*[在Microsoft Word中，您可以右键点击上方目录区域并选择'更新域'来自动生成准确的页码]*

# **Functional Requirements**

This document outlines the functional requirements for the ArtisanConnect platform, detailing the services the system must provide and how it should behave in various situations. The requirements are categorized into user requirements and system requirements, following the principles of software engineering.

## **1. Functional User Requirements**

Functional user requirements define the services and functions from the perspective of the end-users. These are high-level requirements that describe what users should be able to do with the system.

### **1.1 User Roles**

The system accommodates three primary user roles, each with a distinct set of capabilities:

![](data:None;base64,)

*mermaid chart*

### **1.2 Customer (Homeowner) Requirements**

• **USR-C01: Service Discovery**: Customers shall be able to search for and discover service providers based on various criteria, including service category, location, price, and ratings.

• **USR-C02: Provider Evaluation**: Customers shall be able to view detailed profiles of service providers, including their service descriptions, portfolios of past work, certifications, and customer reviews.

• **USR-C03: Booking Management**: Customers shall be able to book services for a specific date and time, view their upcoming and past bookings, and cancel or reschedule appointments.

• **USR-C04: Secure Payments**: Customers shall be able to pay for services securely through the platform using integrated payment methods.

• **USR-C05: Communication**: Customers shall be able to communicate directly with service providers through an in-app messaging system to discuss job details.

• **USR-C06: Feedback System**: After a service is completed, customers shall be able to rate the provider and leave a detailed review of their experience.

• **USR-C07: Account Management**: Customers shall be able to manage their personal profile, payment information, and notification preferences.

• **USR-C08: Service History**: Customers shall be able to access a complete history of services they have received, including details, costs, and provider information.

### **1.3 Service Provider (Professional) Requirements**

• **USR-P01: Profile Management**: Service providers shall be able to create and manage a detailed business profile, including their areas of expertise, service offerings, pricing, and availability.

• **USR-P02: Portfolio Showcase**: Service providers shall be able to upload and manage a portfolio of their work, including “before and after” photos and project descriptions.

• **USR-P03: Credential Verification**: Service providers shall be able to upload necessary documents for verification, such as licenses, insurance, and certifications.

• **USR-P04: Booking Administration**: Service providers shall be able to view and manage incoming booking requests, accept or decline jobs, and manage their service calendar.

• **USR-P05: Financial Tracking**: Service providers shall be able to track their earnings, view transaction history, and manage their payout settings.

• **USR-P06: Customer Communication**: Service providers shall be able to communicate with customers via the platform’s messaging system to confirm details and provide updates.

• **USR-P07: Service Management**: Service providers shall be able to define, modify, and remove service offerings, including descriptions, pricing, and availability.

• **USR-P08: Marketing Tools**: Service providers shall have access to tools for promoting their services, including featured listings and special offers.

### **1.4 Admin (Platform Manager) Requirements**

• **USR-A01: User Oversight**: Admins shall be able to manage all user accounts (customers and providers), including approvals, suspensions, and deletions.

• **USR-A02: Provider Verification**: Admins shall be able to review and approve or reject service provider applications based on their submitted documentation.

• **USR-A03: Content Moderation**: Admins shall be able to moderate platform content, including reviews and provider profiles, to ensure quality and compliance with platform policies.

• **USR-A04: Dispute Resolution**: Admins shall have the tools to mediate and resolve disputes between customers and service providers, including issues related to payments and service quality.

• **USR-A05: Platform Analytics**: Admins shall be able to access a dashboard with key platform metrics, such as user growth, revenue, and booking trends.

• **USR-A06: System Configuration**: Admins shall be able to configure system settings, including service categories, verification requirements, and platform policies.

• **USR-A07: Role-Based Access Control**: Admins shall be able to manage access permissions for different user roles, ensuring appropriate restrictions on functionality.

## **2. Functional System Requirements**

Functional system requirements provide a more detailed specification of the system’s functions, services, and operational constraints. They describe the system’s behavior from a technical standpoint, grounded in the existing codebase.

• **SYS-01: User Authentication**:

• **SYS-01.1**: The system provides secure user registration and login using a CredentialsProvider. The implementation in app/api/auth/[...nextauth]/route.js handles email and password authentication.

• **SYS-01.2**: Before authorizing, the system performs a health check on the database connection using Database.healthCheck() as defined in lib/database.js.

• **SYS-01.3**: The system validates user credentials against the User model (models/User.js) using the static method User.validatePassword().

• **SYS-01.4**: The system enforces email verification. A user cannot log in if the emailVerified flag in their User document is false.

• **SYS-01.5**: Role-based access control is managed via JSON Web Tokens (JWT). The user’s role is encoded into the JWT upon login and attached to the session object, as configured in the callbacks of authOptions.

• **SYS-01.6**: Custom pages for authentication are defined in authOptions, with /auth/signin as the entry point for login.

• **SYS-01.7**: The system implements middleware protection for role-specific routes, restricting access based on user roles as defined in middleware.js.

• **SYS-02: Data Retrieval and Search**:

• **SYS-02.1**: The system uses Mongoose models to interact with the MongoDB database for all data operations (e.g., models/BusinessList.js, models/Category.js).

• **SYS-02.2**: API routes located in app/api/ handle data retrieval. For example, /api/categories/route.js fetches all service categories.

• **SYS-02.3**: Searching for businesses within a specific category is handled by server-side logic that queries the businessLists collection in MongoDB with a filter on the category field.

• **SYS-02.4**: The business details pages are powered by API routes like /api/businesses/[businessId]/route.js which fetch a single document by its ID.

• **SYS-02.5**: The system implements advanced search filters including location, price range, and rating as defined in app/(routes)/search/\_components/SearchFilters.jsx.

• **SYS-03: Booking Engine**:

• **SYS-03.1**: New bookings are created by sending a POST request to the /api/bookings endpoint.

• **SYS-03.2**: The API handler uses the Booking.create() Mongoose method to save the new booking document, which includes provider and customer details, date, and time, into the bookings collection.

• **SYS-03.3**: To prevent double-booking, the frontend calls an endpoint that queries the bookings collection for a given business ID and date before rendering available time slots.

• **SYS-03.4**: A user’s booking history is retrieved by a dedicated API that finds all bookings associated with their user ID.

• **SYS-03.5**: The system enforces role-based restrictions on booking functionality, allowing only customers to create bookings as implemented in the booking API and UI components.

• **SYS-04: Review and Rating System**:

• **SYS-04.1**: Users can submit reviews via a POST request to the /api/reviews endpoint.

• **SYS-04.2**: The system uses the Review.create() Mongoose method to create a new review document, linking it to the specific business being reviewed.

• **SYS-04.3**: All new reviews can have a default status (e.g., pending) that requires an Admin to approve them before they are publicly visible, ensuring content quality.

• **SYS-04.4**: The system verifies that a user has completed a booking with the business before allowing them to submit a review, as implemented in the review submission validation.

• **SYS-05: Service Provider Onboarding**:

• **SYS-05.1**: Service providers register new businesses via a POST request to the /api/businesses endpoint.

• **SYS-05.2**: The system creates a new document in the businessLists collection with an approvalStatus field set to PENDING by default.

• **SYS-05.3**: Admins fetch all businesses with a PENDING status from the /api/admin/pending-businesses endpoint to populate their review queue.

• **SYS-05.4**: Admins use the /api/admin/approve-business endpoint to update the approvalStatus of a business to APPROVED or REJECTED.

• **SYS-05.5**: The system sends email notifications to providers when their business status changes, using the EmailService component.

• **SYS-06: Payment Processing**:

• **SYS-06.1**: The system shall integrate with Stripe to process payments securely.

• **SYS-06.2**: The system shall create a payment intent before confirming a booking to authorize the transaction.

• **SYS-06.3**: The system shall store transaction records, linking them to the corresponding booking and user accounts.

• **SYS-06.4**: The system shall implement payment status tracking (PENDING, PAID, REFUNDED) as part of the booking model.

• **SYS-07: Real-time Chat**:

• **SYS-07.1**: The system shall use Socket.IO to enable real-time, bi-directional communication between customers and service providers.

• **SYS-07.2**: Chat messages shall be stored in MongoDB and associated with a specific booking ID.

• **SYS-07.3**: The system shall deliver push notifications for new messages.

• **SYS-07.4**: The system shall maintain message history and support message status tracking (sent, delivered, read).

• **SYS-08: Provider Document Management**:

• **SYS-08.1**: The system shall provide a secure interface for providers to upload verification documents.

• **SYS-08.2**: Documents shall be stored securely, and access will be restricted to Admins for verification purposes.

• **SYS-08.3**: The system shall track the verification status of each provider (Pending, Approved, Rejected).

• **SYS-08.4**: The system shall integrate with Cloudinary for document storage and retrieval.

• **SYS-09: Admin Dashboard**:

• **SYS-09.1**: The system shall present aggregated data on the admin dashboard, including total revenue, new users, and pending provider applications.

• **SYS-09.2**: The system shall provide interfaces for admins to perform management tasks, such as approving businesses and managing user accounts.

• **SYS-09.3**: The system shall implement role-based access control to restrict admin functionality to users with the ADMIN role.

• **SYS-09.4**: The system shall provide analytics visualizations for key performance metrics.

• **SYS-10: Role-Based Access Control**:

• **SYS-10.1**: The system shall enforce role-based access restrictions across all components.

• **SYS-10.2**: The system shall restrict booking functionality to users with the CUSTOMER role.

• **SYS-10.3**: The system shall restrict admin dashboard access to users with the ADMIN role.

• **SYS-10.4**: The system shall restrict provider dashboard access to users with the PROVIDER role.

• **SYS-10.5**: The system shall display appropriate UI elements based on user role.

## **3. Non-Functional Requirements**

Non-functional requirements specify criteria that judge the operation of a system, rather than specific behaviors. They are critical for ensuring the quality, reliability, and usability of the platform.

### **3.1 Performance Requirements**

• **NFR-P01: Response Time**: The system shall respond to user interactions within 2 seconds under normal load conditions.

• **NFR-P02: Load Capacity**: The system shall support at least 1000 concurrent users without degradation in performance.

• **NFR-P03: Scalability**: The system architecture shall allow horizontal scaling to accommodate growing user base.

• **NFR-P04: Database Performance**: Database queries shall execute within 500ms for standard operations.

• **NFR-P05: Image Loading**: Service provider images and portfolio items shall load within 3 seconds on standard connections.

### **3.2 Security Requirements**

• **NFR-S01: Authentication**: The system shall implement secure authentication using industry-standard protocols.

• **NFR-S02: Data Encryption**: All sensitive data shall be encrypted both in transit and at rest.

• **NFR-S03: Authorization**: The system shall enforce role-based access control for all protected resources.

• **NFR-S04: Input Validation**: All user inputs shall be validated and sanitized to prevent injection attacks.

• **NFR-S05: Session Management**: User sessions shall be securely managed with appropriate timeout mechanisms.

• **NFR-S06: Secure API**: All API endpoints shall be protected against unauthorized access and abuse.

### **3.3 Reliability Requirements**

• **NFR-R01: Availability**: The system shall maintain 99.9% uptime during business hours.

• **NFR-R02: Fault Tolerance**: The system shall handle component failures gracefully without data loss.

• **NFR-R03: Backup**: The system shall perform automated backups of all data at least once daily.

• **NFR-R04: Recovery**: The system shall be able to recover from failures within 1 hour.

• **NFR-R05: Error Handling**: The system shall provide meaningful error messages and graceful degradation.

### **3.4 Usability Requirements**

• **NFR-U01: Responsive Design**: The user interface shall adapt to different screen sizes (mobile, tablet, desktop).

• **NFR-U02: Accessibility**: The system shall comply with WCAG 2.1 Level AA accessibility standards.

• **NFR-U03: Consistency**: The user interface shall maintain consistent design patterns throughout.

• **NFR-U04: Learnability**: New users shall be able to complete basic tasks without assistance.

• **NFR-U05: Feedback**: The system shall provide clear feedback for all user actions.

### **3.5 Maintainability Requirements**

• **NFR-M01: Code Structure**: The codebase shall follow a clear component structure with separation of concerns.

• **NFR-M02: Documentation**: All code shall be documented with appropriate comments and external documentation.

• **NFR-M03: Testing**: The system shall have automated tests covering critical functionality.

• **NFR-M04: Modularity**: The system shall be built with modular components that can be updated independently.

• **NFR-M05: Configuration**: System parameters shall be configurable without code changes.

## **4. Requirements Traceability Matrix**

The following matrix maps key user requirements to system requirements and implementation components:

![](data:None;base64,)

*mermaid chart*