**Activity-Style Flowcharts**

# 目录

Activity-Style Flowcharts ........................ 2

1. Customer Booking Process ........................ 3

2. Service Provider Registration Workflow ........................ 4

3. Admin Approval Workflow ........................ 5

4. Role-Based Access Control Flow ........................ 6

*[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]*

# **Activity-Style Flowcharts**

This document presents a series of flowcharts for the ArtisanConnect platform, styled to represent system activities. These diagrams illustrate the dynamic workflows for key processes, providing a step-by-step view of activities and decisions, in line with software engineering best practices. The diagrams use the correct and validated flowchart syntax with swimlanes to clearly separate responsibilities.

## **1. Customer Booking Process**

This diagram shows the end-to-end flow for a customer booking a service, from initial search to final confirmation.

![](data:None;base64,)

*mermaid chart*

**Description of Customer Booking Workflow:**

This flowchart details the customer’s journey through the booking process. It begins with the customer’s initial search and selection of a business. The “System” swimlane highlights the backend and frontend logic that occurs within the BookingSection.jsx component, such as fetching available slots to prevent double-booking. The flow then proceeds to the decision point of checking if the user is authenticated before finalizing the booking via the saveBooking function and providing feedback to the user. The system also sends email notifications to both the customer and service provider upon successful booking creation.

## **2. Service Provider Registration Workflow**

This diagram illustrates the process a new service provider follows to register their business on the platform.

![](data:None;base64,)

*mermaid chart*

**Description of Provider Registration Workflow:**

This flowchart shows the structured, multi-step registration process for service providers, which is primarily handled by the ProviderRegistrationForm.jsx component. It highlights the critical dependency on document uploads, with clear separation between provider actions and system processing. The system logic prevents the provider from submitting their application until the minimum required documents are provided, ensuring data completeness before it reaches the admin for review. The workflow concludes with the business being saved with a PENDING status, awaiting approval, and an email notification being sent to administrators.

## **3. Admin Approval Workflow**

This diagram details the process an administrator follows to review and either approve or reject a new business registration.

![](data:None;base64,)

*mermaid chart*

**Description of Admin Approval Workflow:**

This flowchart illustrates the admin’s critical quality control function, centered around the BusinessApprovalSystem.jsx component. The process begins with the system presenting a queue of pending applications fetched from the /api/admin/pending-businesses API route. The admin’s decision to “Approve” or “Reject” triggers a call to a specific API endpoint (/api/admin/approve-business), which handles the state change of the business document in the MongoDB database and notifies the provider. The rejection flow includes capturing a reason for rejection, which is included in the notification to the provider. This ensures a clear and auditable approval process with proper feedback mechanisms.

## **4. Role-Based Access Control Flow**

This diagram illustrates how the system enforces role-based access restrictions across the platform.

![](data:None;base64,)

*mermaid chart*

**Description of Role-Based Access Control Flow:**

This flowchart details how the system enforces role-based access restrictions using the middleware and component-level checks. The process begins when a user attempts to access a route or feature. The middleware extracts the JWT token from the request and validates it. If valid, it extracts the user’s role and checks if the requested route is protected and requires a specific role. Based on these checks, the system either allows access, redirects to the login page, or shows an error page. For allowed requests, the system renders appropriate UI components based on the user’s role, hiding restricted elements and processing the request with role-specific logic. This implementation is primarily handled in the middleware.js file and the RoleAccessMessage.jsx component.