**Software Architecture Document (SAD) for Service App**

Version: 1.0

Date: [16-08-2023]

Contents

[1. Introduction 2](#_Toc143120767)

[1.1 Purpose: 2](#_Toc143120768)

[1.2 Scope: 2](#_Toc143120769)

[1.3 Document Overview: 2](#_Toc143120770)

[2. Architectural Overview 2](#_Toc143120771)

[2.1 System Architecture: 3](#_Toc143120772)

[2.2 Key Components: 3](#_Toc143120773)

[2.3 Technology Stack: 3](#_Toc143120774)

[3. Stakeholders and User Roles 3](#_Toc143120775)

[3.1 Customer: 3](#_Toc143120776)

[3.2 Business Owner: 3](#_Toc143120777)

[3.3 Admin: 3](#_Toc143120778)

[4. Functional Requirements 3](#_Toc143120779)

[4.1 Splash Page: 4](#_Toc143120780)

[4.2 Login and Signup: 4](#_Toc143120781)

[4.3 Customer's Functional Requirements: 4](#_Toc143120782)

[4.4 Business Owner's Functional Requirements: 4](#_Toc143120783)

[4.5 Admin's Functional Requirements: 5](#_Toc143120784)

[4.5.1 Customer Support: 5](#_Toc143120785)

[5. Non-Functional Requirements 5](#_Toc143120786)

[5.1 Performance: 5](#_Toc143120787)

[5.2 Security: 5](#_Toc143120788)

[5.3 Scalability: 5](#_Toc143120789)

[5.4 Usability: 5](#_Toc143120790)

[6. System Design 6](#_Toc143120791)

[6.1 High-Level Design: 6](#_Toc143120792)

[6.3 Database Design: 6](#_Toc143120793)

[7. Deployment 6](#_Toc143120794)

[7.1 Environments: 6](#_Toc143120795)

[7.2 Deployment Strategy: 6](#_Toc143120796)

[8. Monitoring and Maintenance 7](#_Toc143120797)

[8.1 Logging and Monitoring: 7](#_Toc143120798)

[8.2 Maintenance Procedures: 7](#_Toc143120799)

[9. Risks and Mitigation Strategies 7](#_Toc143120800)

[9.1 Data Privacy and Security Risks: 7](#_Toc143120801)

[9.2 System Downtime Risks: 7](#_Toc143120802)

[9.3 User Adoption Risks: 7](#_Toc143120803)

[10. Conclusion 7](#_Toc143120804)

# 1. Introduction

## 1.1 Purpose:

This Software Architecture Document outlines the architecture of the Collaborative Request Handling Services App, developed using the Python programming language (or any suggested development approach). It defines the system's components, user roles, functional and non-functional requirements, design, deployment, and maintenance strategies.

## 1.2 Scope:

The Service App is designed to connect **customers** and **business owners** in a seamless manner to provide and receive services. It offers a platform for users to browse, book, and manage services. The app includes functionalities for both customers, business owners, and admin control.

## 1.3 Document Overview:

This document provides an in-depth understanding of the Service App's architecture, its user roles, interactions, and the technical decisions made during its design and development.

# 2. Architectural Overview

## 2.1 System Architecture:

The Service App follows a client-server architecture, where clients (mobile app) interact with the backend server through APIs. The backend manages data storage, business logic, and communication with external services.

## 2.2 Key Components:

- Mobile App (iOS and Android) for Customer, Business Owner, Admin.

- Application Server (APIs and Business Logic)

- User Authentication.

- SQL database

- External Services Integration (Payments..etc)

## 2.3 Technology Stack:

- TBD

# 3. Stakeholders and User Roles

## 3.1 Customer:

Customers use the app to search for services, view service providers' profiles, book appointments, make payments, and provide reviews.

## 3.2 Business Owner:

Business owners use the app to list their services, manage their availability, receive booking requests, confirm appointments, and interact with customers.

## 3.3 Admin:

Admins manage the overall system, including user accounts approvals, service categories.

# 4. Functional Requirements

## 4.1 Splash Page:

* Display a visually appealing splash page when the app is launched.
* Provide a brief introduction to the app's services and features.
* Offer an option to proceed to the login or signup screens or skip to view categories.

## 4.2 Login and Signup:

* Allow new users to create accounts by signing up.
* Collect essential user information during signup (name, phone number, password).
* Validate phone number by OTP.
* Provide an option for users to log in using their registered email and password, or phone number.
* Implement password reset functionality for forgotten passwords.
* Offer social media login integration Gmail quick signup..etc (if applicable).
* Ensure proper encryption and security measures for storing user credentials.

## 4.3 Customer's Functional Requirements:

* Browse available services for request handling.
* Submit specific requests for various services (Events and Parties, Songs, Flowers, etc.).
* Allow user to select numbers such as attendants, how many tables required..etc
* Allow user to write note to the business owner.
* Provide detailed descriptions and requirements for each request.
* Allow to select delivery date for the service.
* Receive notifications and updates on request status.
* Review and approve proposals from service providers.
* Make payments for approved proposals.
* Provide feedback and ratings after the service is completed.
* Contact customer support for issues, inquiries, and assistance.

## 4.4 Business Owner's Functional Requirements:

* List services offered for request handling.
* Receive notifications about new customer requests.
* Review customer requests and requirements.
* Create and submit proposals with detailed plans and pricing.
* Allow to create dynamic subcategory for user to select its info. For ex(create menu for tables numbers, chairs..etc) by allowing business owner to make his own label.
* Allow each business owner to select his delivery time/date.
* Communicate with customers to clarify details before approving.
* Track the status of submitted proposals.
* Once approved, execute the requested service.
* Receive payments upon completion of the service.
* Contact customer support for issues, inquiries, and assistance.

## 4.5 Admin's Functional Requirements:

* Manage the platform for request handling services.
* Review and approve business owners providing request handling services.
* Monitor customer requests and proposal submissions.
* Assist in dispute resolution between customers and business owners.
* Provide customer support for inquiries and issues.
* Ensure compliance with platform policies and guidelines.
* Analyze data to identify trends in customer requests and service performance.

### 4.5.1 Customer Support:

* Handle customer inquiries, complaints, and feedback.
* Provide timely responses to customer queries.
* Assist in resolving disputes and conflicts.
* Maintain a knowledge base for frequently asked questions.
* Collaborate with business owners to address customer concerns.
* Continuously improve customer support processes.

## 5. Non-Functional Requirements

## 5.1 Performance:

- App responsiveness: Load times and interactions should be swift.

- Scalability: The system should handle a growing number of users and services.

## 5.2 Security:

- User data protection: Sensitive information should be securely stored and transmitted.

- Authentication and authorization: Secure user authentication and role-based access control.

## 5.3 Scalability:

- The system should scale horizontally to accommodate increased user load.

## 5.4 Usability:

- Intuitive user interface for all user roles.

- Consistent user experience across platforms.

## 6. System Design

## 6.1 High-Level Design:

- Mobile apps with a user-friendly UI for customers and business owners.

- Web app for admin functionalities.

- API layer for communication between clients and server.

- Database schema to store user data, services, bookings, and reviews.

## 6.2 Database Design:

- User tables for customers, business owners, and admins.

- Service and category tables.

- Booking and appointment tables.

- Review and rating tables.

## 7. Deployment

## 7.1 Environments:

- Development: Local development environments for testing.

- Staging: Testing environment resembling production for final testing.

- Production: Live environment for end-users.

## 7.2 Deployment Strategy:

- TBD if needed.

## 8. Monitoring and Maintenance

## 8.1 Logging and Monitoring:

- Implement logging to track system behavior and errors.

- Use monitoring tools to track system performance and uptime (if applicable).

## 8.2 Maintenance Procedures:

- Regularly update software components and libraries.

- Scheduled maintenance windows for database backups and updates.

---

## 9. Risks and Mitigation Strategies

## 9.1 Data Privacy and Security Risks:

- Encryption of sensitive data during storage and transmission (if applicable).

- Regular security audits and vulnerability assessments.

## 9.2 System Downtime Risks:

- Load balancing and redundancy to minimize downtime.

- Effective disaster recovery and backup procedures.

## 9.3 User Adoption Risks:

- User training and onboarding materials.

- Continuous user feedback and feature improvement.

---

## 10. Conclusion

This Software Architecture Document provides an overview of the architecture, user roles, and functionality of the Service App. It serves as a guide for development, deployment, and maintenance, ensuring the creation of a reliable, secure, and user-friendly service platform for customers, business owners, and administrators. Feel free to add some features or suggest any technology to the application.