Software Architecture Document

Online Doctor Visit Appointment (Web Application)

Shubham Gantayat, Suraj Kumar

Version 1.0

13/08/2021

Version Control

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Description | Responsible Party | Date |
| 1.0 | Initial version | Shubham Gantayat,  Suraj Kumar | 13-08-2021 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Introduction

This document provides a high-level overview and explains the architecture of the Online Doctor Visit Appointment.

The document defines goals of the architecture, the use cases supported by the system, architectural styles and components that have been selected. The document provides a rationale for the architecture and design decisions made from the conceptual idea to its implementation.

## Purpose

The Software Architecture Document (SAD) provides a comprehensive architectural overview of the Online Doctor Visit Appointment Application (ODVA). It presents several different architectural views to depict the different aspects of the system.

To depict the software as accurately as possible, the structure of this document is based on Philippe Kruchten’s “4+1” model view of architecture [Kruchten].



## Scope

The scope of this SAD is to explain the architecture of the Online Doctor Visit Appointment Application (ODVA).

This document describes the various aspects of the ODVA system design that are architecturally significant. These elements and behaviours are fundamental for guiding the construction of the ODVA system and for understanding this project.

## Definitions, Acronyms and Abbreviations

* + - WWW – World Wide WEB
    - HTTP – Hyper Text Transfer Protocol
    - SAD – Software Architecture Document
    - ODVA – Online Doctor Visit Appointment Application
    - User – Any user who is registered on our database

# Architectural Representation

This document details the architecture using the views defined in the “4+1” model [Kruchten]. The views used to document the ODVA system are:

**Use Case view**

**Audience**: all the stakeholders of the system, including the end-users.

**Area**: describes the set of scenarios and/or use cases that represent some significant, central functionality of the system. Describes the actors and use cases for the system, this view presents the needs of the user and is elaborated further at the design level to describe discrete flows and constraints in more detail. This domain vocabulary is independent of any processing model or representational syntax (i.e. XML).

**Related Artefacts**: Use-Case Model, Use-Case documents

**Logical view**

**Audience**: Designers.

**Area**: Functional Requirements: describes the design's object model. Also describes the most important use-case realizations and business requirements of the system.

**Related Artefacts**: Design model

**Process view**

**Audience**: Data specialists, Database administrators

**Area**: Persistence: describes the architecturally significant persistent elements in the data model as well as how data flows through the system.

**Related Artefacts**: Data model.

**Deployment view**

**Audience**: Deployment managers.

**Area**: Topology: describes the mapping of the software onto the hardware and shows the system's distributed aspects. Describes potential deployment structures, by including known and anticipated deployment scenarios in the architecture we allow the implementers to make certain assumptions on network performance, system interaction and so forth.

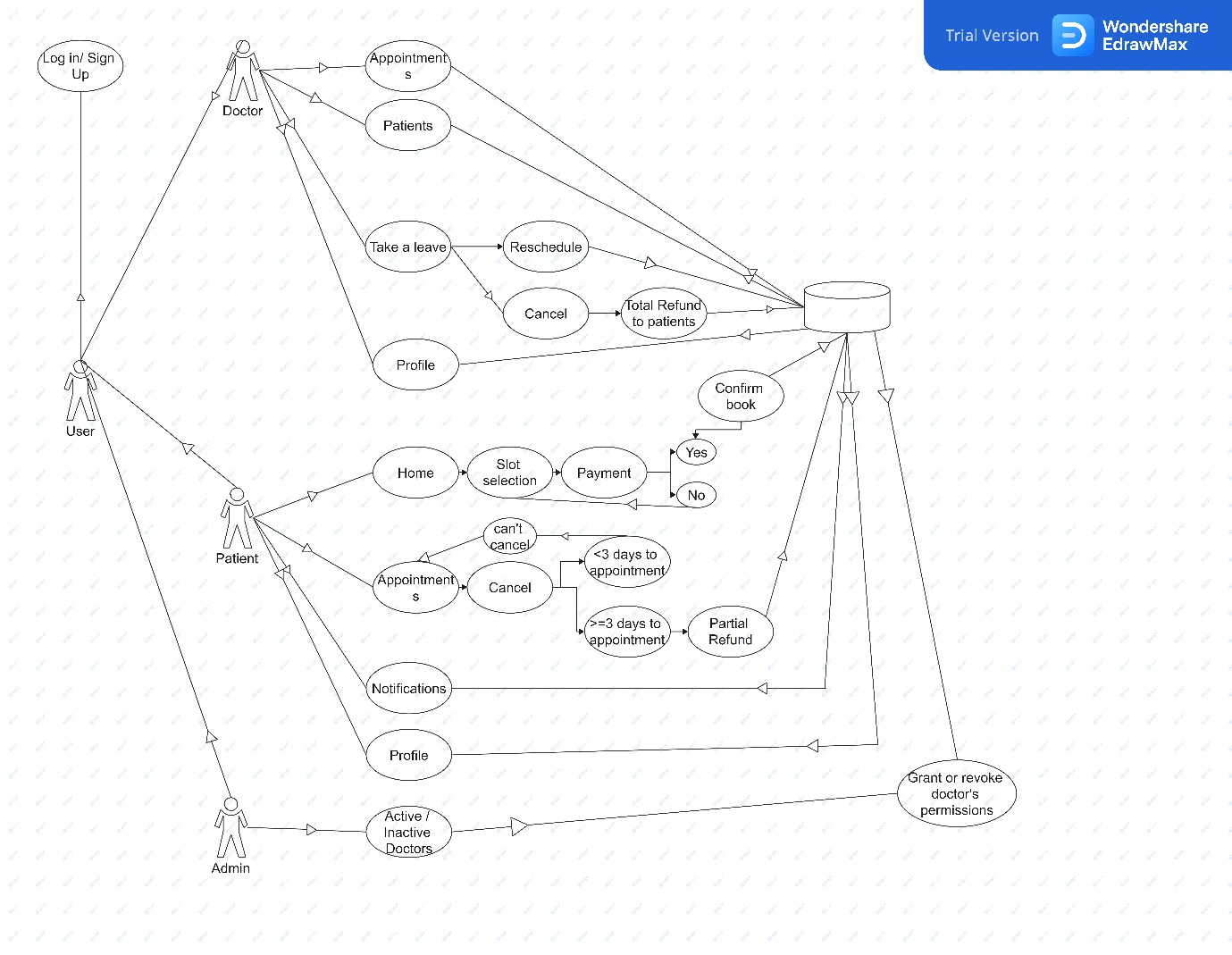
**Related Artefacts**: Deployment model.

# Use-Case View

A description of the use-case view of the software architecture. The Use Case View is important input to the selection of the set of scenarios and/or use cases that are the focus of an iteration. It describes the set of scenarios and/or use cases that represent some significant, central functionality. It also describes the set of scenarios and/or use cases that have a substantial architectural coverage (that exercise many architectural elements) or that stress or illustrate a specific, delicate point of the architecture.

The ODVA use cases are –

* Login
* Signup
* Doctor - Appointments
* Doctor - Patients
* Doctor - Profile
* Doctor - Take a leave section
* Patient - Home
* Patient - Appointments
* Patient - Notification
* Patient - Profile
* Admin - Active Doctors
* Admin - Inactive Doctors



## Login

First thing anyone will see is the login window, which will ask the user for username and password to login. Sign Up toggler will redirect to signup window.

## Signup

The signup window opens when you click the signup button at the login page. Sign up page is different for both doctor and patients.

## Doctor – Appointments

This page will show the appointments for upcoming 15 days along with patient details.

## Doctor – Patients

This section allows the doctor to see the past patient details whom they have diagnosed.

## Doctor – Take a leave

Doctor can take a leave for a specific period. During that time all the appointments will be rescheduled to the nearest possible opening and the patients will be notified.

## Doctor – Profile

Doctor can visit their profile for setting up payment details and edit their initial information.

## Patient – Home

This section allows patients to search for a doctor by name or speciality and allows them to book an appointment.

This window shows a list of doctors. On clicking doctor image the booking window opens

## Patient – Appointments

This allows patients to see their past as well as upcoming appointments.

## Patient – Notification

This section displays reminder, rescheduling and cancellation of appointments.

## Patient – Profile

Patients can visit their profile for setting up payment details and edit their initial information.

## Admin – Active

Admin can view all the active doctors and can cancel their active license on the site in case of a dispute.

## Admin – Inactive

Admin can activate the account of a doctor after verifying the account details and doing a background check.

# Logical View

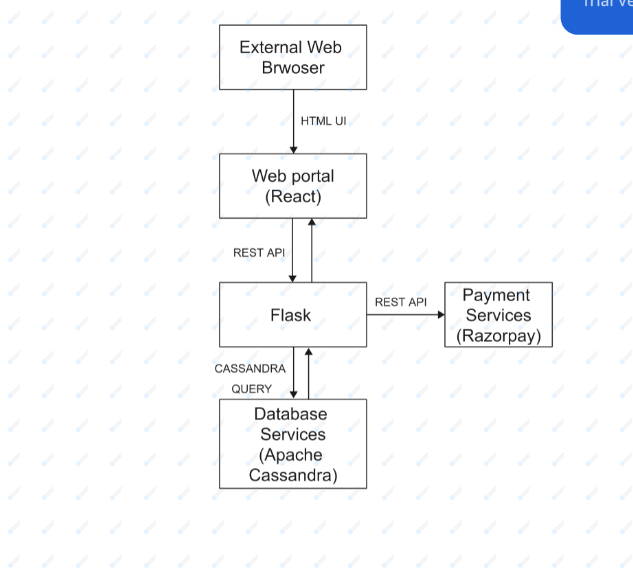
A description of the logical view of the architecture. Describes the most important classes, their organization in service packages and subsystems, and the organization of these subsystems into layers. Also describes the most important use-case realizations, for example, the dynamic aspects of the architecture. Class diagrams may be included to illustrate the relationships between architecturally significant classes, subsystems, packages and layers.

The logical view of the ODVA is comprised of the 3 main packages: User Interface, Database Services and Payment Services.

The user interface package contains classes for each of the forms that the actors use to communicate with the system. Boundary classes exist to support log in, sign up, maintaining appointment schedules, cancellation and rescheduling of appointments, and notifying patient is something comes up.

The database package maintains the user data and is responsible for retrieval, inserting, updating and deletion of data.

The payment package is responsible for accepting payments, transfers and refunds.



# Process View

A description of the process-view of the architecture. Describes the tasks (processes and threads) involved in the system's execution, their interactions and configurations. Also describes the allocation of objects and classes to tasks.

For process view diagram please visit –

<https://github.com/shubhamgantayat/MedHub/blob/main/Documents/ProcessView.jpg>

# Deployment View

The web application is hosted on AWS EC2 instance.

