Software Architecture Document

Insurance Premium Prediction

Vibhu Manikpuri

Version 1.2

09/09/2021

Revision History

Version	Description of Versions / Changes	Responsible Party	Date
1.0	Initial version	Vibhu Manikpuri	8/9/21
1.1	Start of functional view	Vibhu Manikpuri	8/9/21
1.2	-Add the Prediction functional view -Added Entity Data Model diagram -Added interface specifications - Added Web Portal interface definitions. - Changed operation signatures. - Web Portal Interface definition modified Added Use Case View and Deployment View	Vibhu Manikpuri	9/9/21

Approval Block

Version	Comments	Responsible Party	Date

Table of Contents

Software Architecture Document

1. Introduction

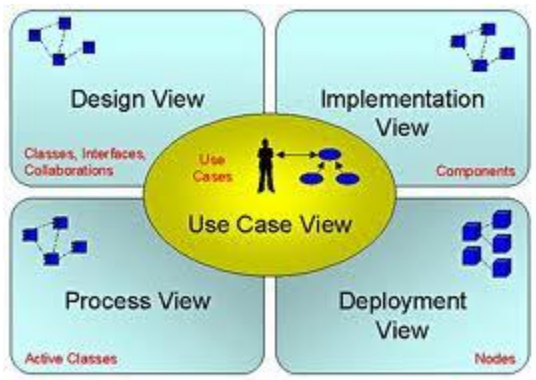
This document provides a high level overview and explains the architecture of the Insurance Premium Predictor.

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.

1.1. Purpose

The Software Architecture Document (SAD) provides a comprehensive architectural overview of the Insurance Premium Predictor (IPP). It presents a number of different architectural views to depict the different aspects of the system.

In order 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].



The "4+1" View Model allows various stakeholders to find what they need in the software architecture.

1.2. Scope

The scope of this SAD is to explain the architecture of the Insurance Premium Predictor.

This document describes the various aspects of the IPP system design that are considered to be architecturally significant. These elements and behaviors are fundamental for guiding the construction of the IPP and for understanding this project as a whole. Stakeholders who require a technical understanding of the IPP are encouraged to start by reading the High Level Document, Low Level Document and Wireframe document developed for this system [HLD, LLD, Wireframe document].

1.3. Definitions, Acronyms, and Abbreviations

- **IPP**- Insurance Premium Predictor
- **Heroku**– Cloud hosting platform
- **SAD -** Software Architecture Document
- **User -** This is any user who is registered on the DMM website

1.4. References

[HLD]:	High Level Document
[LLD]:	Low Level Document
[Wireframe]:	Wireframe document

[MedBiquitous]: Sample SAD, https://projects.cecs.pdx.edu/attachments/download/3146/Software_Architecture_Document.docx

[Kruchten]: The “4+1” view model of software architecture, Philippe Kruchten, November 1995, http://www3.software.ibm.com/ibmdl/pub/software/rational/web/whitepapers/2003/Pbk4p1.pdf

1.5. Overview

In order to fully document all the aspects of the architecture, the Software Architecture Document contains the following subsections.

Section 2: describes the use of each view

Section 3: describes the architectural goals and constraints of the system

Section 4: describes the most important use-case realizations

Section 5: describes logical view of the system including interface and operation definitions.

Section 6: describes how the system will be deployed.

2. Architectural Representation

This document details the architecture using the views defined in the “4+1” model [Kruchten]. The views used to document the IPP 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 Artifacts : 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 Artifacts: Design 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 Artifacts: Deployment model.

3. Architectural Goals and Constraints

There are some key requirements and system constraints that have a significant bearing on the architecture. They are:

1. The system is meant as a proof of concept for a more complete project prediction system to be built in the future. Therefore, one of the primary stakeholders in this document and the system as a whole are future architects and designers, not necessarily users as is normally the case. As a result, one goal of this document is to be useful to future architects and designers.

2. The system will be written using Python programing language and will take User Input and predict output without storing any data entered by the User and will be deployed to a cloud hosting platform Heroku.

3. One of the primary goals of the system architecture is to define how the system will react if the Input entered by the User is not how the Machine Learning model expected it. The architecture seeks to do this through the use of error handling and information hiding to isolate components.

4. Use-Case View

The purpose of the use-case view is to give additional context surrounding the usage of the project and the interactions between its components. For the purposes of this document, each component is considered a use-case actor. Section 4.1 lists the current actors and gives a brief description of each in the overall use context of the system. In section 4.2, the most common use-cases are outlined and illustrated using UML use-case diagrams and sequence diagrams to clarify the interactions between components.

4.1. Actors

User

The user will drive all operation of the software. No distinction is made in regards to type of user. The user interacts with all available interfaces to initiate and monitor all application operations.

Web Portal

The web portal is the main user interface for the system.

Prediction Service

The Prediction Service is the link between our application and the User Input which provides the desired predicted value.

4.2. Use-Case Realizations

4.2.1. Request Analysis (Get Report)

User requests a report for a user-specified inputs and report is displayed.

Figure 4.1 Request Analysis Use Case Diagram

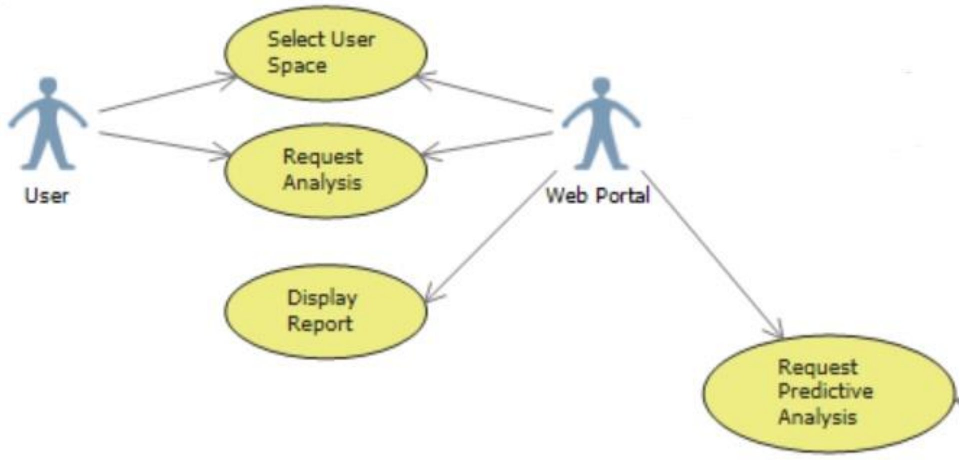
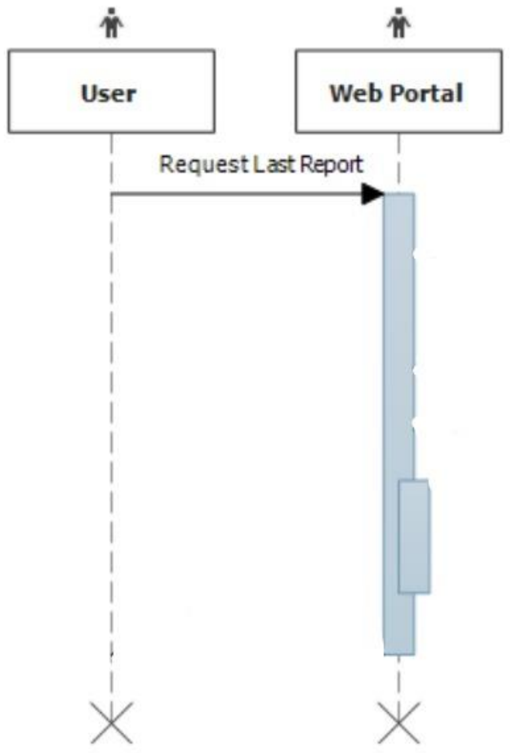


Figure 4.2 Request Analysis Use Sequence Diagram



4.2.2. Retrieve Report

User requests to view the generated report.

Figure 4.3 Retrieve Report Use Case Diagram

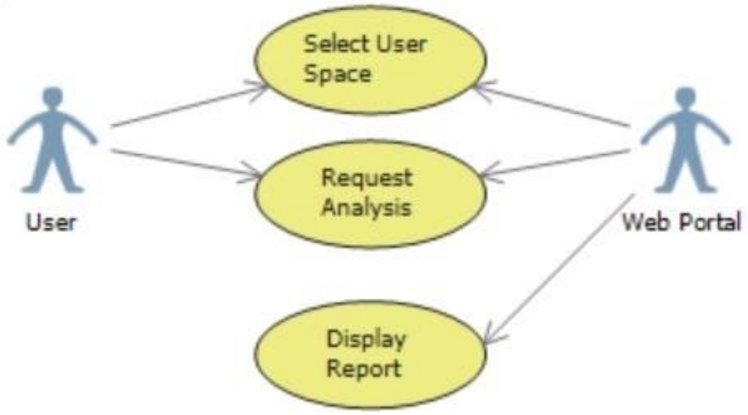
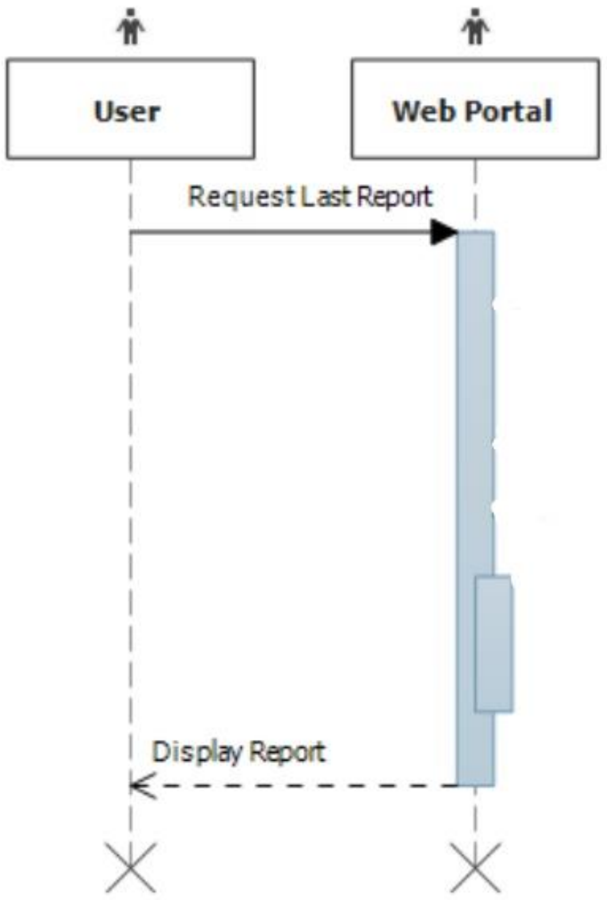


Figure 4.4 Retrieve Last Report Sequence Diagram



5. Logical View

5.1. Overview

The main goal of the logical view is to define the components that will make up the system and to define the interfaces through which they will communicate and interact with one another. The primary decision-making factor behind defining the system components is the need to isolate the components that are likely to change. A summary of these changes and how the logical decomposition of the architecture addresses them is as follows:

1. Changes to the metrics used to construct the predictive model
    a. All business logic dealing with predictive model, including what metrics are sent to it to construct it's model are isolated in a Prediction Service component (see figure 5.1). Changes to the metrics used to construct the model need only be made in this component without affecting the rest of the system.

2. Changes to the content of the prediction report generated for the user
    a. This report is generated in the Web Portal module (see figure 5.1). As such changes to its content need only be made in this module.

Figure 5.1 Logical Component Diagram.

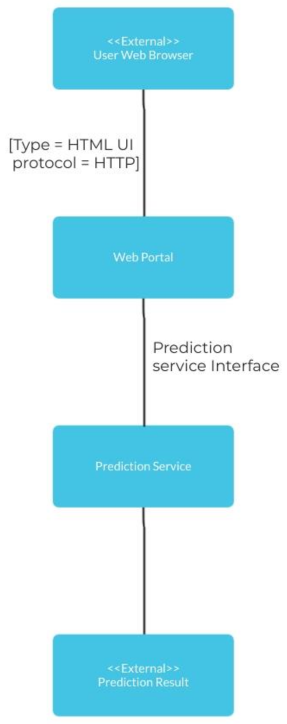


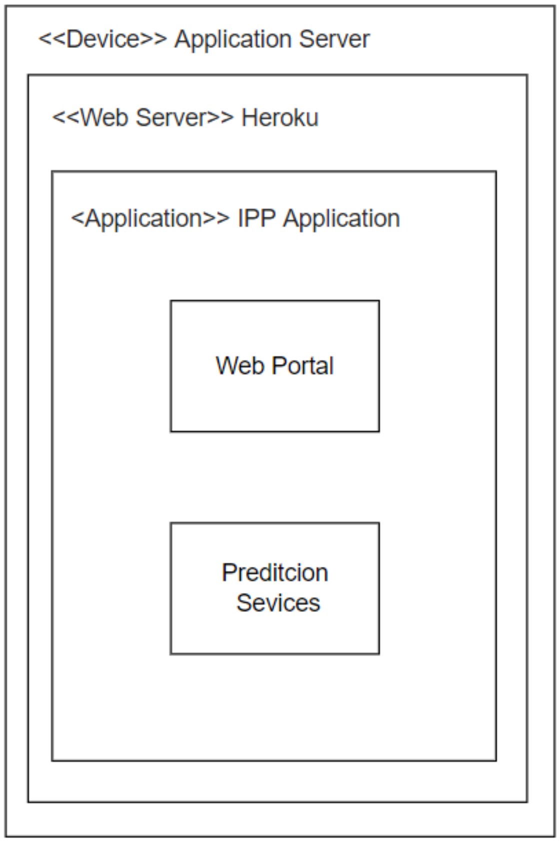
Table 5.1 Element Responsibilities

Element	**Responsibilities**
Web Portal	• Present users with an HTML-based user interface accessible through a web browser. • Interact with other components in the system to allow users to authenticate with Assembla, choose an Assembla project for analysis, and analyze the chosen project.
Prediction Service	• Provide an interface to get a prediction for a given Assembla project. • Provide an interface for providing training data to Google Predictive.

6. Deployment View

The web application will be hosted on a single physical server. A Heroku cloud platform will be used to serve the application pages.
The application's deployment specifics can be seen below.

Figure 6.1 Deployment View Diagram


<<Device>> Application Server
<<Web Server>> Heroku
<Application>> IPP Application
Web Portal
Preditcion Sevices