**Avi System Software Design and Architecture Document**

**Version 1.3**

**Revision History**

|  |  |  |
| --- | --- | --- |
| **Date** | **Version** | **Description** |
| 25-09-2018 | 1.0. | First Draft |
| 01-10-2018 | 1.1. | Added Sequence Diagrams and Activity Diagrams |
| 05-10-2018 | 1.2. | Added Package Diagram and Deployment Diagram |
| **07-10-2018** | **1.3.** | **Fixed spelling and grammatical errors** |

**1. Introduction**

This section provides an overview of this document. It includes the purpose, scope, definitions and acronyms, references and overview of the system.

**1.1. Purpose**

The main purpose of this document is to provide the design and architectural overview of the Avi System. It presents the architectural decisions made in designing and developing the system. This document can be used by the project stakeholders (developers and client) to better understand the problem being solved and how the Avi System will represent the solution.

**1.2. Scope**

Avi is a web application that will recommend elective courses to prospective postgraduate students and further predict the grades the students are likely to get in the recommended courses based on their grades in previously completed courses.

A student will start by creating an account which includes specifying their first name, surname, student number, highest completed level of study and their login password.

Once a student has created an account they can only be able to get recommendations once they have added the courses they have done and the grade they obtained in the courses. This information can be updated by the student for cases where they may have made a mistake or their mark has changed.

The student can now request to get recommendations. After this the system will present to them the recommended courses along with the predicted grade for each course.

**1.3. Definitions, Acronyms and Abbreviations**

PC: Personal computer

Client: University of Witwatersrand

See the Glossary Document for system definitions

**1.4. References**

Sources: <http://www.michael-richardson.com/processes/rup_for_sqa/core.base_rup/guidances/guidelines/software_architecture_document_F4C93435.html>

Source: <https://en.wikipedia.org/wiki/4%2B1_architectural_view_model>

Source: <https://djangobook.com/model-view-controller-design-pattern/>

Source: <http://www.ece.uvic.ca/~itraore/seng422-05/notes/arch05-5.pdf>

The Glossary Document

The System Requirements Specification Document

**1.5. Overview**

The next section, section 2, of this document describes the goals and constraints of designing the system’s architecture. Section 3 describes the architectural representation of the system. Section 4 describes the 5 different ways in which the system architecture can be viewed. describes all the software requirements. Section 5 describes the system’s size and performance and Section 6 talks about the system’s quality concerns.

**2.** **Architectural Goals and Constraints**

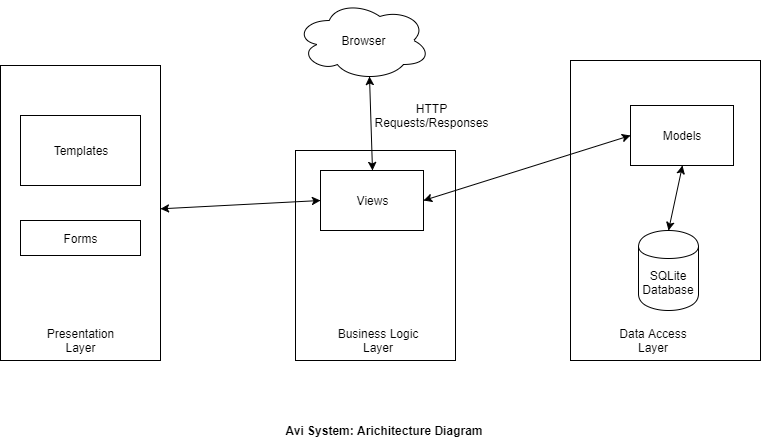
The architecture design of the Avi System was influenced by the requirements specified in the System Requirements Specification document and it was constrained by the Django framework architecture.

**3. Architectural Representation**

**3.1. Architectural Design Pattern and Architectural Style**

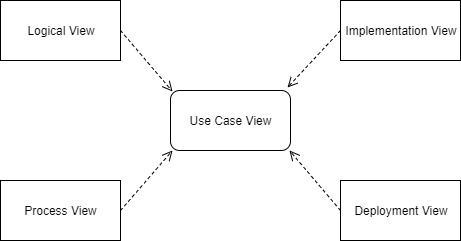
The Avi System follows the Model-Template-View (MTV) design pattern. The model is the data access layer, it is a representation of the database and contains everything about how to access the data. The template is the presentation layer, it is what the user sees and interacts with. The view is the business logic layer, it controls the flow of information between the models and templates and is responsible for any processing.

Architecture Diagram:



**3.2. Architectural Views**

This section describes the system from different perspectives of the project stakeholders. The image below illustrates the views considered.



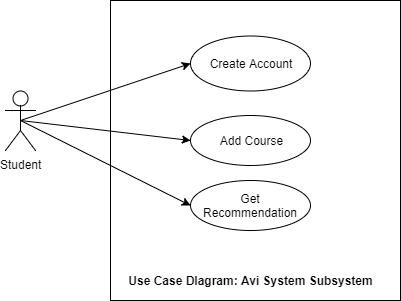
**4.** **Architectural View Decomposition**

**4.1. Use Case View**

The use case view depicts the system from the users’ perspective. In this section we have focused on three use cases which we believe are crucial to the running of the system, these use cases are Create Account, Add Course and Get Recommendation.

**Use Case Diagram**

This Use Case Diagram is a subset of the final Use Case Diagram found in the System Requirements Specification document.

****

**Use Case Descriptions**

1. Create Account

**Main Success Scenario:**

The use case starts when the user requests to create an account. The system will prompt the user to enter their details (i.e. their first name, surname, student number, highest completed level of study and their login password). The user will enter these details and commit the details by pressing the ‘Create Account’ button. The system will check that the password and confirmation of password match. If they match, the system will create the account and the user will be redirected to the login page.

**Alternative Scenario:**

S1: If the password and confirmation of password do not match the use case will be restarted

2. Add Course

**Main Success Scenario:**

The use case starts when the user requests to add a course. The system will prompt for a course code and the grade obtained in the course. The user will enter these details and commit the details by pressing the ‘Add Course’ button. The system will validate (i.e. Check if the course code exists and if the entered grade is a number between 0 and 100) the details and if they are valid it will then add the course and the added course will appear under the ‘Your Courses’ section of the page.

**Alternative Scenario:**

S1: If the details are invalid the use case will be restarted

3. Get Recommendation

**Main Success Scenario:**

The use case is started when the user requests to get a recommendations. The system will check if the user has added courses to their profile. If they have, the system will generate the recommendations and they will appear under the ‘Your Recommendations’ section of the page.

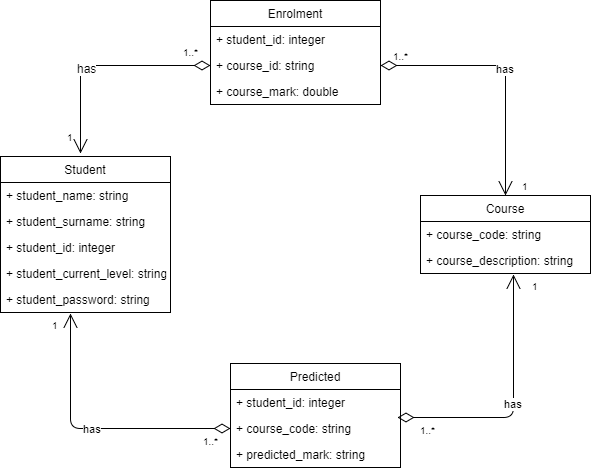
**Alternative Scenario:**

S1: If the user does not have any courses added to their profile, the use case will halt and the user will be advised by the system to add their courses before requesting a recommendation.

**4.2. Logical View**

This view is concerned with depicting the functionality provided to the users by the system. This is done using class and state diagrams.

Class Diagram



State Diagram

The AVI System does not contain any state changes.

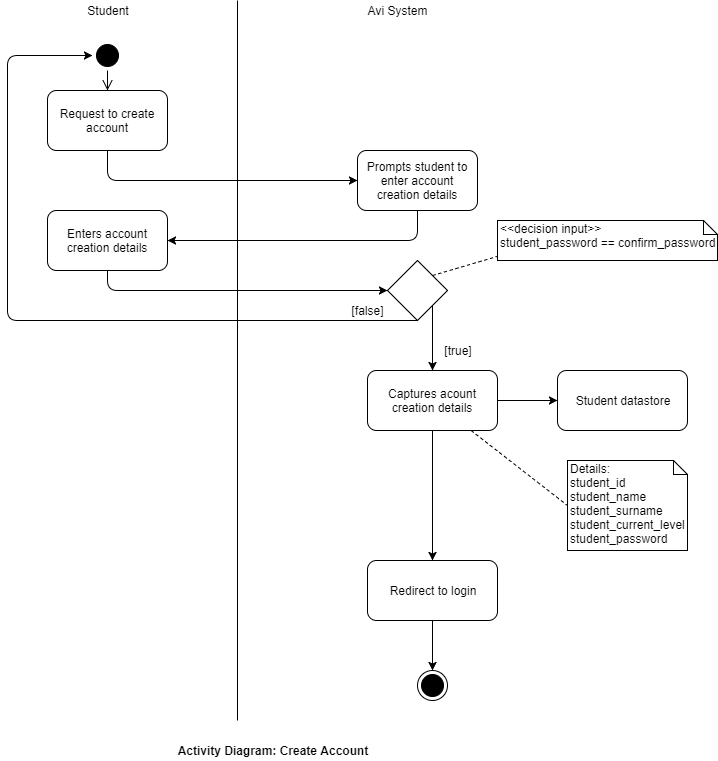
**4.3. Process View**

This view demonstrates the processes that form part of the system and how they interact with each other. It focusses largely on concurrency and the sequence of processes.

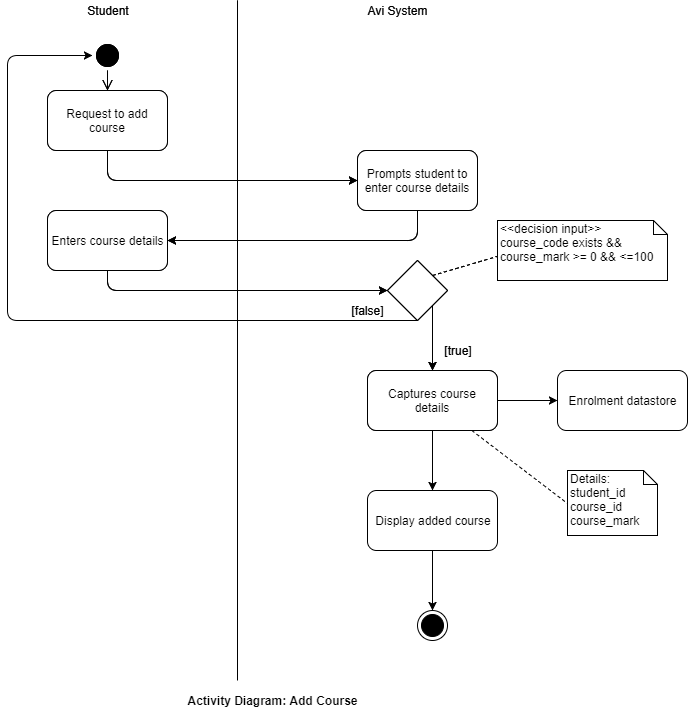
Activity Diagrams

These diagrams are a graphical representation of multiple business processes, specifically, the Create Account, Add Course and Get Recommendation business processes. They show the data flow and the control flow of the business processes.

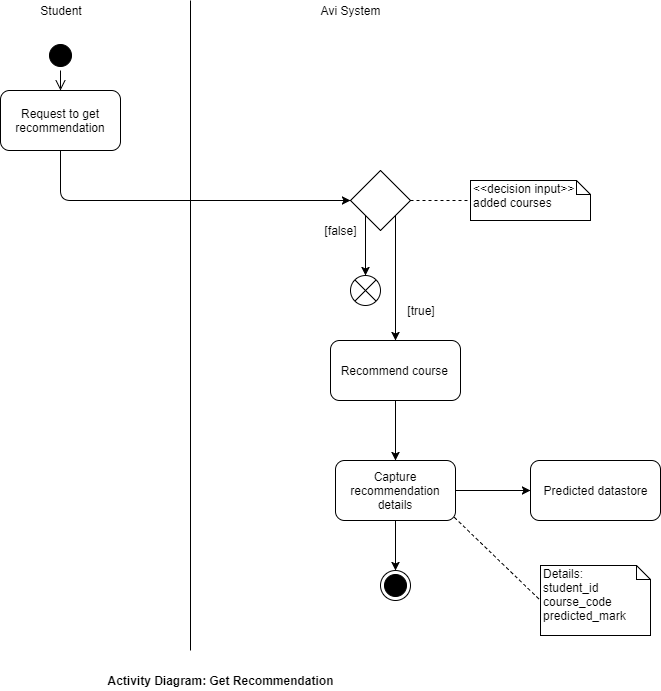
Create Account



Add Course



Get Recommendation

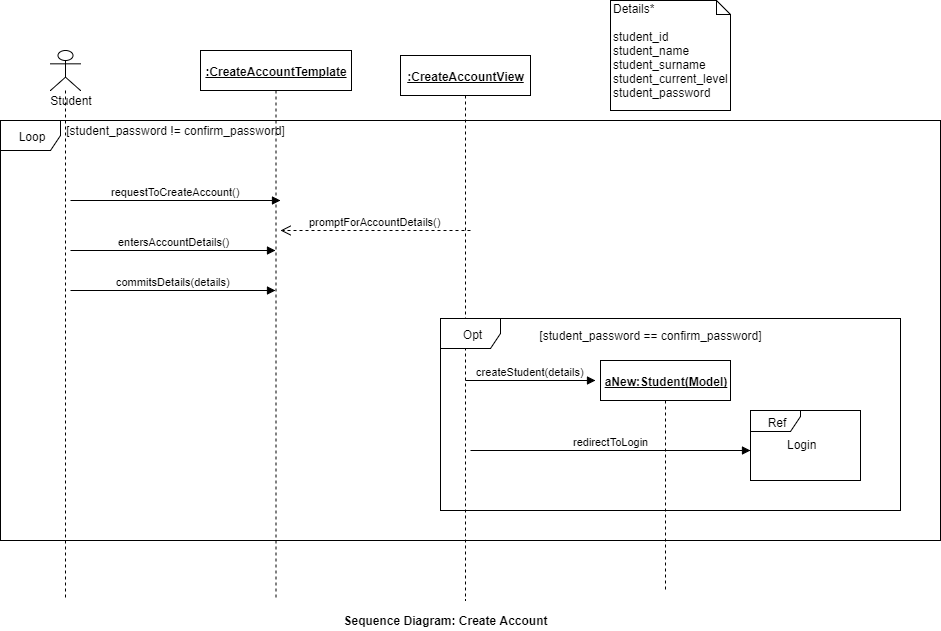


Sequence Diagrams

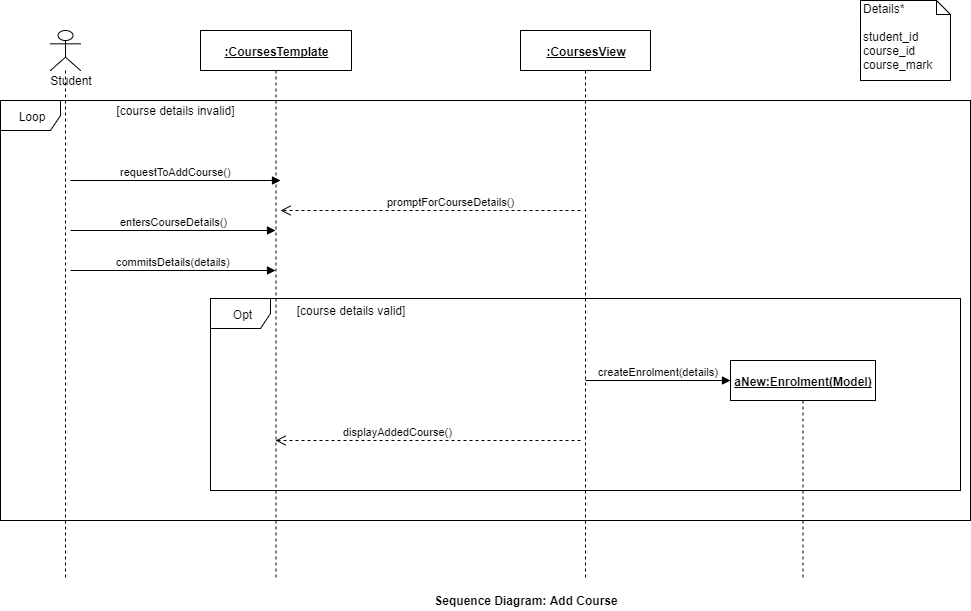
The purpose of these diagrams is to illustrate the communication between the actor and

system during the Create Account, Add Course and Get Recommendations processes

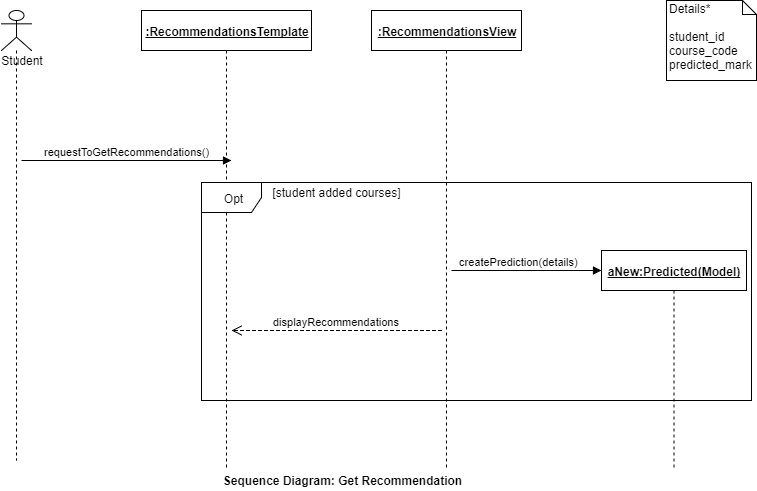
Create Account



Add Course



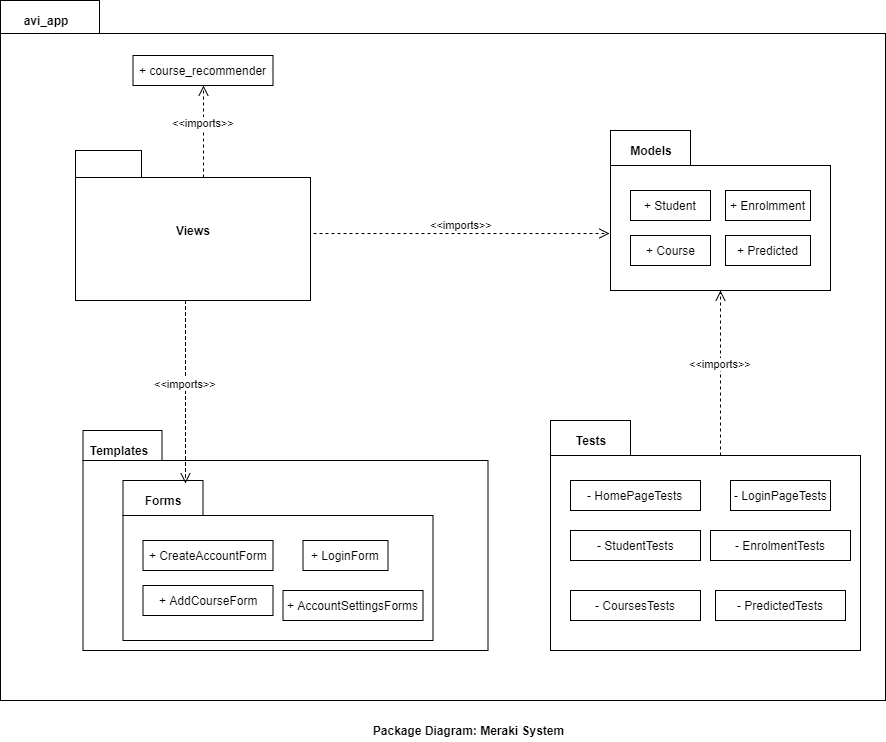
Get Recommendation



**4.4. Implementation View**

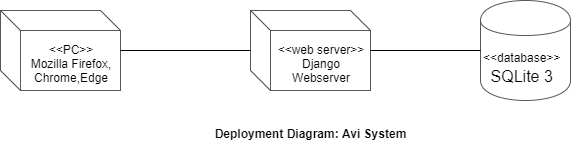
This view depicts the system from the developer’s perspective. It describes the system’s modules in terms of packaging, layering and configuration

Package Diagram



**4.5. Deployment View**

This view depicts the physical components on which the system runs on. The diagram below only depicts the system’s deployment diagram in the development environment since we did not move it to a production environment



**5. Size and Performance**

Below is a summary of the system’s size and performance

* The size of the system (classes, packages etc.) is approximately 2.2 MB
* The size above does not include the external libraries and software that need to be installed

**6. Quality**

This system is not entirely reliable mainly because this was a prototype to demonstrate how the final system would operate.

* Password encryption was not implemented
* It depends on the operating system and database security features
* Difference user’s data is protected i.e. No user has access to another user’s data