**PROG 6212 POE**

DOCUMENTATION

The platform will be designed by the **Model-View-Controller (MVC)** architecture to ensure scalability, maintainability and separation of concerns. This design divides the application into three distinct components: Model, View and Home Controller which perform the following actions:

**Model**: Represents the data and business logic where we have our getters and setters

**View:** Responsible for displaying the data provided by the model

**Controller**: Manages the interaction between the model and the view. It handles user input, updates the model and renders the appropriate view.

**Layout**

Claim Verification Table: A simple table that displays the claims and provides two action buttons for "Verify" and "Approve". The goal is to ensure that program coordinators or managers can easily access and process claims.

Claim Form: A form that allows users to submit claims, specifying details such as the date range, claim ID, program, module name/code, hours worked, hourly rate and supporting documents. This form dynamically allows users to add multiple entries using the "Add More" button and calculates totals automatically.

Lecture Form: Allows users to enter lecture-related details, such as lecture name, ID, and email. This form is intended for users submitting specific information about a lecture or session related to their claim.

Track: Allows the Lectures to track their application by entering the Claim ID

**Structure of Database**

#### For now there is no database present because the current version doesn't require it. Instead, data can be hardcoded for testing purposes as in the sample table of claims but for future purposes this is an example of how the database will structured using a Lecture table :

Stores information about lectures related to claims.

* Lecture ID (Primary Key, INT, Auto-increment): Unique identifier for each lecture.
* Lecture Name (VARCHAR): Name of the lecture.
* Email (VARCHAR): Email address associated with the lecture

**Assumptions**

User Experience:

* Lectures are expected to fill out and submit forms for claims and lectures suggesting that a straightforward and user-friendly interface is essential.
* The form should include input validation to ensure that required fields are completed before submission.

System Assumption:

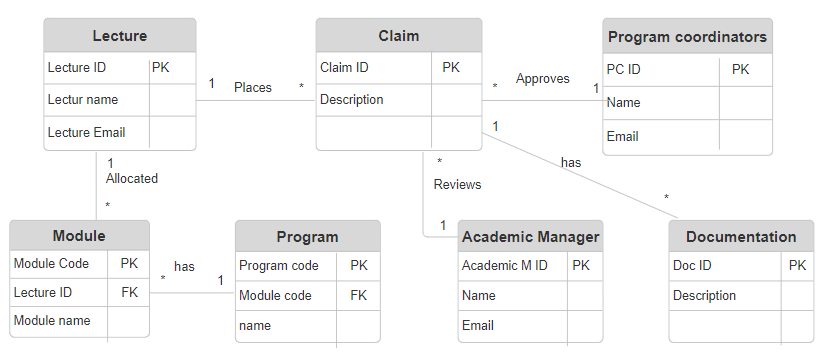
* Program coordinators and academic managers will manually verify and approve claims in the current version.

**Constraint**

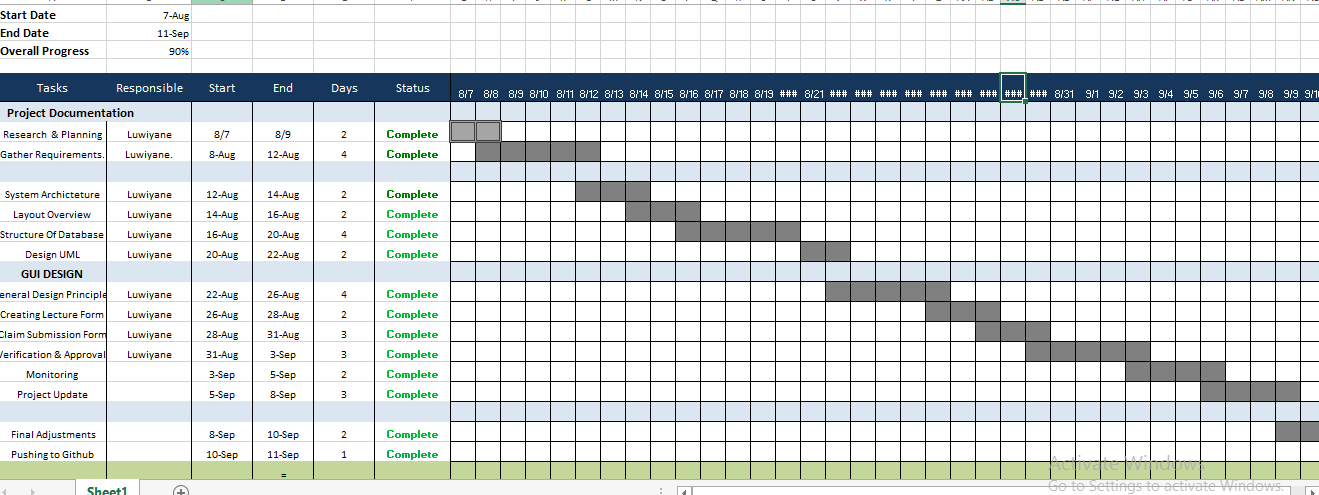
Data Storage:

* Since the database is not functional currently there will be a plan for how form data will be stored or processed temporarily.

**UML**

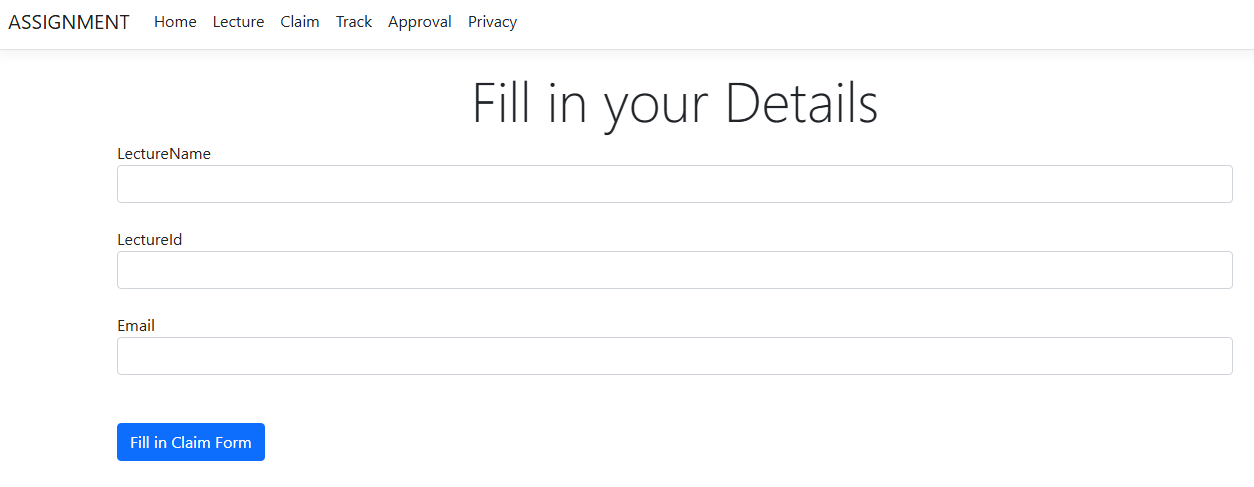


**Project Plan**

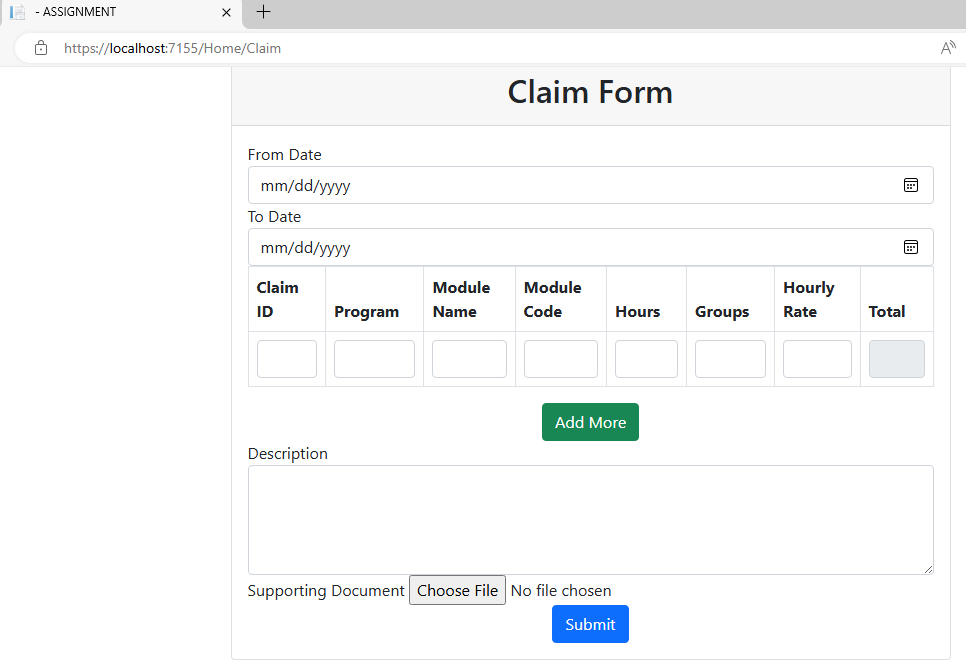


**Gui Design**

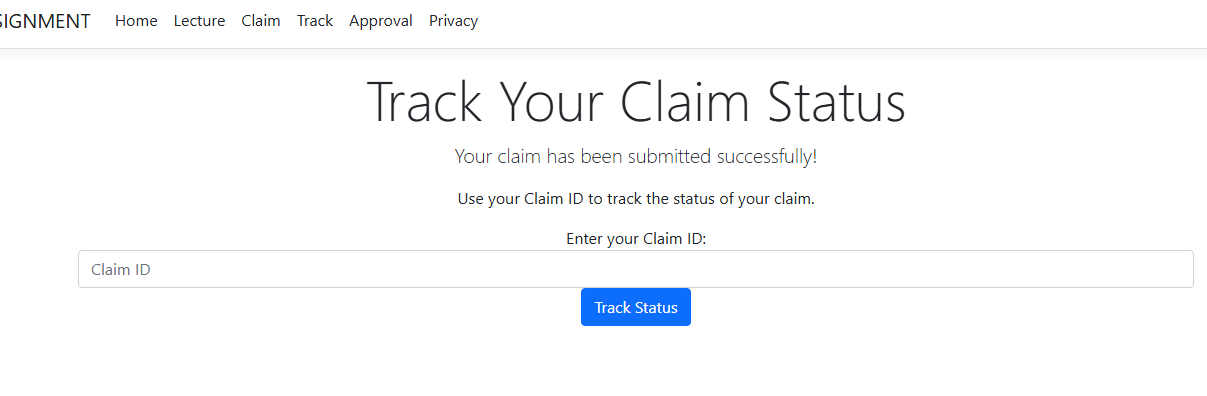
Lecture Form



Claim Form



Track



Verify and Approval

\*Hard coded with no database just a sample of how it will look like

