# ST10438801 – Yamika Govender

# Prog6212 part one

Github link:

<https://github.com/Yamika1/Programming2B_partOne/>.

The Model-View-Controller pattern, MVC, was utilized in the development of the extensive web application, the contract Monthly Claim System. The various essential features of the system, such as contractors, monthly claims, payments, payment making section, registration, and claim submission, facilitate an easy contract management process.  
  
The users are greeted with a friendly home page that contains a sidebar, which makes it convenient to navigate to different parts of the system on the first run of the application. A background image and a dark sky-blue color, which is used throughout the application, add value to the looks of the home page.  
  
Users are able to view current contractors under the Contractor tab and view their information and supporting documentation, i.e., contract agreements and ID copies. Users can add comments for current contractors, promoting good communication and working relationships.  
The Payment Section enables one to see all the payments made to the system on a per-claim basis, with details of the payees who have made payments. The documentary evidence, such as the bank statements, may be seen, hence the payment process is clear and traceable.  
The Registration Section includes the facility to add new profiles, selecting their bank, entering branch codes, account numbers and CVC's. Users are able to view information of created profiles, for accurate record maintenance.  
The Submit Claims Section enables users to submit new claims by entering details such as claim ID, claimant's name, and other details. The users are able to view the details of the submitted claims, status traced, and progress.

The key features of the system include:

* It’s easy navigation and simple design
* Review and management of contractors, monthly claims, payments, registrations and submitted claims
* Supporting of uploading and reviewing documents for contractors, payments and monthly claims.
* Can leave comments for contractors
* Payment tracking and review
* User profile creation and management

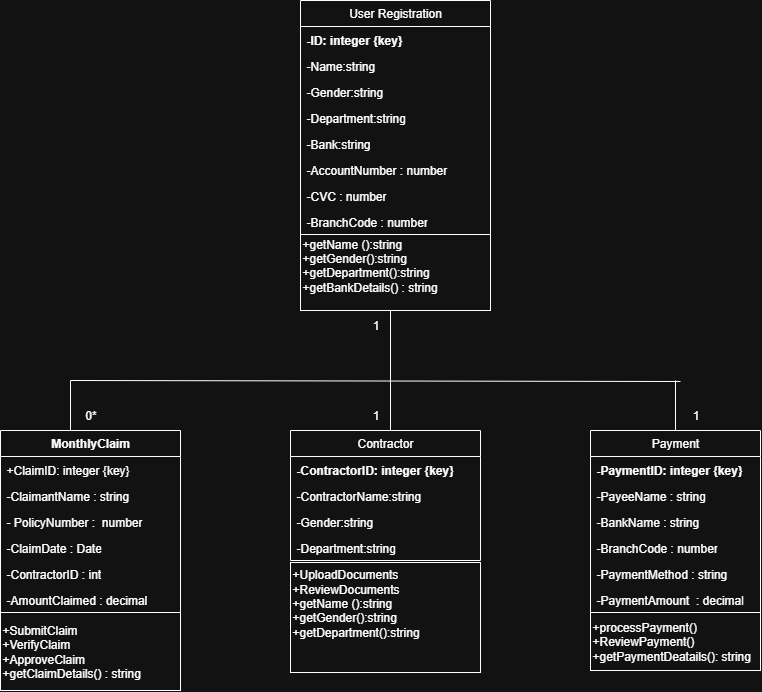
The system is user-friendly and straightforward to administer, and it enables the user to manage contracts, claims, and payments effectively. The fact that it provides a central point of monitoring and document analysis strengthens transparency, accountability, and proper communication among parties. The simplicity and ease of use make it an effective solution for managing complex contract processes.  
Part 2 includes core functionalities like building a model for all entities with full CRUD capability, file upload and display the name and its details, file size cap, and status monitoring system. The third part has automations like auto-computing the payment based on the number of hours worked and hourly rate, generating invoices or reports based on approved claims, and automatically checking submitted claims against a given criterion.

The Contract Monthly Claim System is plagued with scalability, security risks, integration issues, user acceptability, and regulatory requirements issues. It is plagued with working issues in document upload administration, compliance verification and error correction. Flexibility towards adapting to changing business rules and criteria needs to be ensured for the systems long-term success.

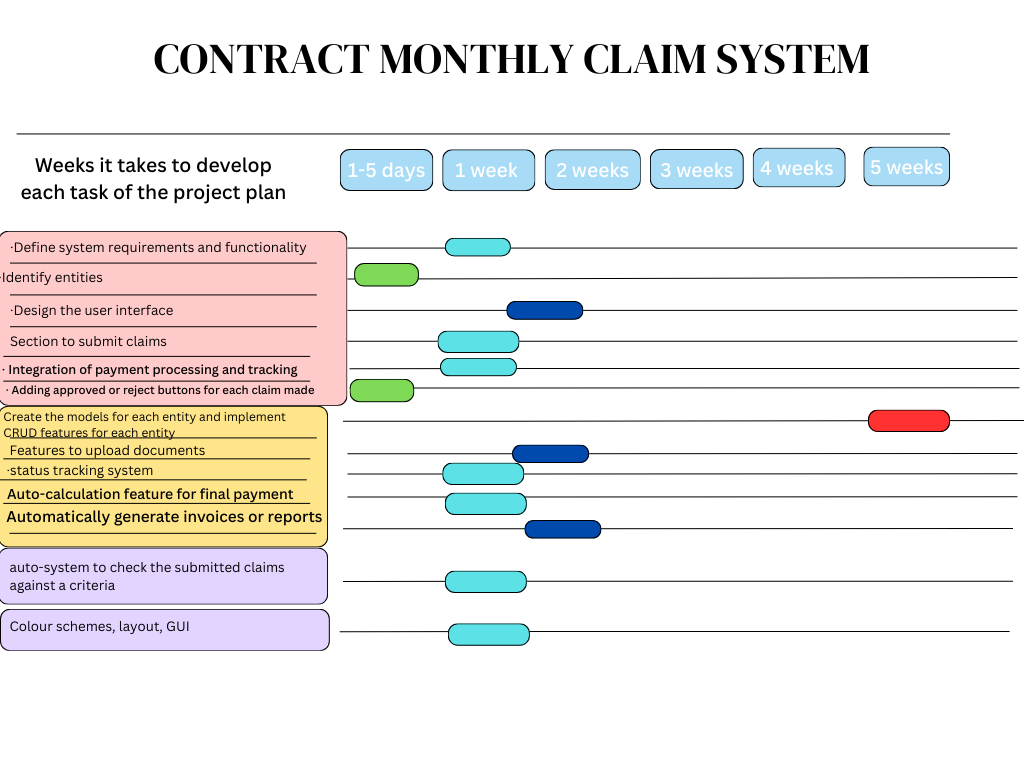
**Tasks needed to create a contract monthly claim system and the timelines for them:**

* Dependencies:
* Finish-to-start (FS)
* Start-to-start (SS)
* Part one, includes front-end development only (weeks 1 – 6)
* Define system requirements and functionality (1 week) – (FS)
* Identify entities (contractors, claims, payments, registration, claims submission) - (2-3 days) – (FS)
* Design the user interface (1.5 weeks) – (FS)
* Section to submit claims (form UI and validation) - (1 week) – (FS)
* Integration of payment processing and tracking (1 week) – (FS)
* Adding approved or reject buttons for each claim made (4-5 days) (FS)
* Colour schemes, layout, GUI (1 week) – (SS)
* Part 2 will involve adding functionalities. (weeks 7 – 13)
* Create the models for each entity and implement CRUD features for each entity (5 weeks) – (FS)
* Features to upload documents to be stored, and displaying the name of the file (1.5 weeks) – (FS)
* Feature to implement a file size limit (2-3 days) – (FS)
* Implementation of a status tracking system (1 week) – (FS)
* Part 3: Automations (Weeks 14 – 16)
* The final part of the POE deals with automations such as the auto-calculation feature to compute the final payment based on the hours worked and hourly rate inputted by the lecturer (1 week) – (FS)
* developing functionality to automatically generate invoices or reports summarising approved claims for payment processing. (1.5 weeks) – (FS)
* Feature to automatically check the submitted claims against a specified criterion. (1 week) – (FS)

UML Class Diagram



Gnatt chart



6 commits on github

