9/10/2024

Sashiel Moonsamy

ST10028058

pROg6212 POE PArt 1

Sashiel Moonsamy

ST10028058

PROG6212

POE Part 1

**1: Documentation**

**1.1 Design Choices Explanation**

The design of the **Contract Monthly Claim System (CMCS)** was driven by the need to simplify and streamline the process of claim submissions and approvals for Independent Contractor (IC) lecturers. Each design decision was made with careful consideration of scalability, security, and ease of use, ensuring that the system meets real-world business needs effectively.

**System Architecture**

I chose the **Model-View-Controller (MVC)** architecture for this system because it enforces a clear separation of concerns, making the application more modular, easier to maintain, and scalable. Here's why:

* **Model:** Manages the business logic and application data. It ensures that all operations, such as claim calculations, document uploads, and user authentication, are handled efficiently. I chose to separate this from the view and controller to make future adjustments (like adding new types of users) easy to implement without affecting the entire system.
* **View:** Manages the user interface, where I opted for simplicity to enhance user experience. Since lecturers, coordinators, and managers will use the system, the design must cater to users with varying technical skills. I focused on creating an intuitive, easy-to-navigate interface.
* **Controller:** Handles requests from the user, communicates with the model, and updates the view. This decoupling from the model ensures that user input (e.g., claim submissions) is processed without compromising the system's core business logic.

**Why MVC?**  
The MVC architecture ensures that the CMCS can evolve and scale as more lecturers, claims, and functionalities are added. By separating concerns, it becomes easier to update individual components, reducing the risk of bugs and improving maintainability.

**Scalability and Flexibility**

In a system like CMCS, scalability is a crucial factor. As the number of lecturers, claims, and document uploads grows, the system must handle increasing data loads without performance degradation. The MVC architecture naturally supports scalability because it separates the core components:

* **Model**: The business logic and database interactions are independent of the user interface, which means the system can be scaled at the backend (e.g., adding more servers or optimizing databases) without requiring changes to the frontend.
* **View**: The user interface (UI) can be scaled or changed independently to meet growing demands, for instance, by adding new features like reports or dashboards, without affecting the backend.

**Security Considerations**

**Why Security Is Essential?**  
Since the system handles sensitive data, including financial claims and lecturer details, it is essential to implement security measures such as password hashing and encrypted communication. For this reason, I opted for:

* **Hashed Passwords**: All passwords are securely stored using a hashing algorithm to prevent unauthorized access in case of a data breach.
* **Data Validation and Sanitization**: To ensure that all inputs are safe, reducing the risk of SQL injections or malicious data entries.

**1.2 Database Structure Description**

**The database structure for the Contract Monthly Claim System (CMCS) is designed to handle the submission, approval, and processing of monthly claims for independent contractor lecturers. This structure includes key entities and relationships to support various stakeholders such as Lecturers, Programme Coordinators, Academic Managers, and Human Resources. Each of these roles interacts with claims in different ways to ensure that the claims are efficiently managed from submission to approval.**

**Tables and Key Fields:**

1. **Lecturer**
   * **Key Fields:**
     + **LecturerID: Primary Key (PK) uniquely identifying each lecturer.**
     + **FirstName, LastName, Email, Password: Personal and login information.**
     + **Department: The department to which the lecturer belongs.**
     + **Date\_Joined: The date the lecturer joined.**
   * **Description: This table stores lecturer-specific information. Each lecturer can submit claims and view the status of their claims.**
2. **ProgrammeCoordinator**
   * **Key Fields:**
     + **CoordinatorID: Primary Key (PK) for each Programme Coordinator.**
     + **FirstName, LastName, Email, Password: Personal and login information.**
   * **Description: Programme Coordinators manage claims from multiple lecturers. They review claims before forwarding them to the Academic Manager or Human Resources.**
3. **AcademicManager**
   * **Key Fields:**
     + **ManagerID: Primary Key (PK) identifying each Academic Manager.**
     + **FirstName, LastName, Email, Password: Personal and login information.**
   * **Description: Academic Managers oversee Programme Coordinators and provide additional review of claims submitted by lecturers.**
4. **HumanResources (HR)**
   * **Key Fields:**
     + **HR\_ID: Primary Key (PK) for each Human Resources staff member.**
     + **FirstName, LastName, Email, Password: Personal and login information.**
   * **Description: The Human Resources department is responsible for processing claims, ensuring they are ready for payment, and maintaining claim records.**
5. **Claim**
   * **Key Fields:**
     + **ClaimID: Primary Key (PK) for each claim.**
     + **LecturerID: Foreign Key (FK) linking the claim to the lecturer who submitted it.**
     + **CoordinatorID: Foreign Key (FK) linking the claim to the Programme Coordinator overseeing the lecturer.**
     + **ManagerID: Foreign Key (FK) linking the claim to the Academic Manager reviewing it.**
     + **HR\_ID: Foreign Key (FK) linking the claim to the HR department for processing.**
     + **HoursWorked, HourlyRate, TotalAmount: Key data for the calculation of the claim.**
     + **Status: The current status of the claim (e.g., submitted, approved, rejected).**
     + **DateSubmitted: The date the claim was submitted.**
   * **Description: This table records each claim submitted by a lecturer. The claim moves through different statuses as it is reviewed and approved by coordinators, managers, and HR.**
6. **SupportingDocument**
   * **Key Fields:**
     + **DocumentID: Primary Key (PK) for each supporting document.**
     + **ClaimID: Foreign Key (FK) linking the document to its associated claim.**
     + **DocumentName: The name of the document.**
     + **DateUploaded: The date the document was uploaded.**
   * **Description: This table stores documents associated with claims. These documents can include evidence of work completed, contracts, or other relevant paperwork.**

**Relationships and Multiplicities:**

* **Lecturer to Claim: A lecturer can submit multiple claims, but each claim is associated with only one lecturer. (1..\* Lecturer ⟶ 1 Claim).**
* **ProgrammeCoordinator to Claim: A Programme Coordinator oversees multiple claims, but each claim is overseen by only one Programme Coordinator. (1..\* ProgrammeCoordinator ⟶ 1 Claim).**
* **AcademicManager to Claim: An Academic Manager can review many claims, but each claim is reviewed by only one Academic Manager. (1..\* AcademicManager ⟶ 1 Claim).**
* **HumanResources to Claim: Human Resources can process multiple claims, but each claim is processed by only one HR staff member. (1..\* HumanResources ⟶ 1 Claim).**
* **Claim to SupportingDocument: A claim can have multiple supporting documents, but each document is related to only one claim. (1..\* Claim ⟶ 1 SupportingDocument).**

**How the Structure Supports CMCS Requirements:**

1. **Claim Submission:**
   * **Lecturers can submit claims for their worked hours. The claims are stored in the Claim table, which records key information such as HoursWorked, HourlyRate, and TotalAmount.**
   * **Lecturers can upload supporting documents (e.g., proof of work) through the SupportingDocument table, ensuring claims are backed by evidence.**
2. **Claim Review and Approval:**
   * **Once a claim is submitted, it goes through a review process, starting with the ProgrammeCoordinator, who can approve or reject the claim. If approved, the claim moves to the AcademicManager for further review.**
   * **The final step in the claim's lifecycle is with HumanResources, who process the claim and mark it as completed once ready for payment.**
3. **Multi-level Approval System:**
   * **The relationships between Claim, ProgrammeCoordinator, AcademicManager, and HumanResources ensure that claims are reviewed by the appropriate parties at each stage. This multi-level approval system prevents errors and ensures accountability at each level.**
4. **Document Management:**
   * **The SupportingDocument table ensures that claims can have relevant documentation attached, allowing for more robust claim validation and record-keeping.**
5. **Centralized Data:**
   * **The structure centralizes key data for claims, lecturers, and supporting documents in one relational database, simplifying queries and reporting. This makes it easier for management to review the claim history, track the status of claims, and generate reports.**

**Why This Structure?**

This database structure ensures that the system can manage a large volume of claims, documents, and users efficiently while maintaining clear relationships between different entities. The structure supports scalability, meaning it can handle an increase in data without requiring major changes to the architecture. Additionally, the normalization of data avoids redundancy, improving performance and reducing storage costs.

**1.3 GUI Layout Explanation**

The **Graphical User Interface (GUI)** for the CMCS is designed to prioritize ease of use and functionality. The target users (lecturers, Programme Coordinators, and Academic Managers) vary in technical ability, so simplicity and clarity were paramount in the design choices.

**Layout Features:**

* **Lecturer Dashboard**: The dashboard for lecturers displays their submitted claims, with an option to filter by status (e.g., "Submitted," "Approved"). A prominent “Submit New Claim” button allows easy access to the claim submission form.
* **Claim Submission Form**: The form contains fields for hours worked, hourly rate, and a file upload option for supporting documents. Fields are clearly labeled, and validation is in place to ensure correct data entry.
* **Review Pages for Coordinators and Managers**: Both coordinators and managers have their own dashboards that display claims needing review or approval. The design allows them to easily access claim details and approve/reject claims with a few clicks.
* **Approval Workflow Integration**: The GUI includes clear visual indicators of claim progress, allowing users to track where their claim is in the approval process.

**Why This Layout?**

The GUI layout was designed to streamline the workflow, reduce the number of clicks needed to perform actions, and minimize confusion. The dashboard design gives users an immediate overview of relevant tasks (e.g., claims submitted or awaiting approval). The minimalistic design is intentional to reduce cognitive overload, especially for non-technical users like lecturers and administrators. This ensures a better user experience, fewer errors, and faster processing times.

**1.4 Assumptions and Constraints**

**Assumptions:**

1. **Lecturers Will Submit Claims Correctly**: It is assumed that lecturers will provide accurate data when submitting claims (e.g., correct hours worked and supporting documents). Input validation will ensure data integrity.
2. **Programme Coordinators and Academic Managers Will Follow the Approval Process**: It is assumed that all users responsible for reviewing claims will diligently follow the approval process, ensuring that no claim is left unreviewed.
3. **Supporting Documents Are in PDF Format**: It is assumed that all supporting documents, such as timesheets, will be uploaded as PDFs to maintain consistency and avoid compatibility issues.
4. **System Growth**: The system is designed to scale with the assumption that new lecturers, claims, and documents will be added regularly. It assumes an average increase in data over time without major spikes in usage.

**Constraints:**

1. **Database Performance**: The system’s performance could be affected by the number of claims and supporting documents stored in the database. To mitigate this, optimization techniques such as indexing and regular maintenance will be necessary.
2. **File Size Limits**: The system may impose size limits on uploaded files (e.g., supporting documents) to prevent performance degradation or excessive storage costs.
3. **Security Constraints**: Due to the sensitive nature of the data (financial and personal information), security measures such as encrypted communication and secure password storage must be in place. This adds complexity to the system, but it is a necessary tradeoff for ensuring data integrity.
4. **User Access Levels**: The system assumes distinct roles for users (Lecturers, Programme Coordinators, and Academic Managers), which requires well-defined access control mechanisms to prevent unauthorized actions.

Every design choice made for the CMCS was driven by the need to create a system that is secure, scalable, and easy to use. By implementing a structured database, an intuitive GUI, and adhering to assumptions and constraints, the system will support the submission and approval of claims in a reliable and efficient manner. Each decision is backed by industry best practices, ensuring that the system is not only functional but also adaptable for future enhancements.