## ST10407732

## CHRISTINAH CHITOMBI

**DOCUMENTATION**

**Chosen Technologies:**

Framework: .NET Core MVC

Programming Language: C#

Database: SQL Server

**Design choices**

The selection of.NET Core MVC was based on its great support for modern web development, cross-platform compatibility, and robustness. In complete compliance with the requirements of the Contract Monthly Claim System, this framework offers a strong, adaptable foundation for developing web applications with a distinct division of responsibilities.

**WHY MVC?**

**Cross-Platform Reach for Our Business:** By using ASP.NET Core, our Contract Monthly Claim System can run smoothly on any platform, whether it's Windows, Android, Linux, or iOS. This flexibility means our lecturers, coordinators, and managers can access the system from any device they prefer, making the process of submitting and approving claims easier and more accessible. Additionally, the optimized performance of ASP.NET Core ensures that our system runs faster, enhancing the user experience and allowing us to handle tasks more efficiently, which is crucial for our business operations (Yukti Solutions, n.d.).

**Security:** The built-in security capabilities of.NET Core, such as data protection, authentication, and authorization, are crucial for handling sensitive data, such as confidential personal information for lectures and financial claims (Yukti Solutions, n.d.).

**Why SQL Server?**

Relational database has a strong support for complex queries, transactions, and relationships, SQL Server is chosen to handle the CMCS's data requirements. For managing the complex data processes involved in claim processing, SQL Server is the best option because of its support for stored procedures and its capacity to ensure data integrity through constraints. Since it makes it possible to define relationships between entities like Lecturers, Claims, and Documents in a clear and understandable way, a need for effectively presenting business logic, the relational model fits the needs of the system quite well (Troelsen and Japikse, 2020).

**User Roles**: The system incorporates three main user roles Lecturers, Programme Coordinators, and Academic Managers to ensure a structured and secure approach to managing claims.

- **Lecturers**: Can submit claims, upload supporting documents, and track the status of their submissions.
- **Programme Coordinators**: Review and verify claims, ensuring that submitted hours and rates align with agreed-upon terms.
- **Academic Managers**: Provide final approval of claims, adding an additional layer of oversight to the process. The implementation of distinct permissions for each role is critical in safeguarding sensitive information and preventing unauthorized actions, thereby maintaining the integrity of the claim process.

**Database Structure**:

- The database will comprise several key tables, each designed to encapsulate specific data relevant to the CMCS:

  **Lecturers Table**: Stores lecturer information, including LecturerID (primary key), Name, ContactDetails, and Role.

  **Claims Table**: Central to the system, this table includes fields such as ClaimID (primary key), LecturerID (foreign key), HoursWorked, HourlyRate, TotalAmount (calculated), SubmissionDate, Status, and ApprovalDates. This structure allows for detailed tracking of each claim from submission to final approval.

  **Support Documents Table**: Contains DocumentID (primary key), ClaimID (foreign key), FilePath, and UploadDate, linking supporting files to specific claims.

  **Coordinators and Managers Tables**: These tables store details of users in review roles, ensuring traceability of actions such as verifications and approvals through user-specific logging.

  **Users Table**: Stores general user information, including UserID (primary key), UserName, and RoleID (foreign key). This table facilitates the assignment of different roles to users.

  **Roles Table:** Defines various roles within the system, such as Lecturer, Coordinator, and Manager. It includes fields like RoleID (primary key) and RoleName.

  **Relationships**:

  **One-to-Many Relationship**

  Exists between Lecturers and Claims, as each lecturer can submit multiple claims over time.

  Exists between Claims and Support Documents, allowing for multiple supporting documents per claim.

  Users and Roles: Each user can have one role, and each role can be assigned to multiple users. This relationship is represented by a User table with a RoleID foreign key linking to the Roles table.

**GUI Layout**:

- The user interface is designed with a focus on clarity, ease of navigation, and intuitiveness, ensuring that all users from lecturers to managers can interact with the system effectively.

  **Lecturer Dashboard**: Features buttons for submitting new claims, uploading supporting documents, and viewing the status of past submissions. A streamlined form for entering claim details ensures quick and accurate data entry, with validations to minimize errors.

  **Coordinator Dashboard**: Provides a list of pending claims requiring verification, and action buttons for approving or rejecting claims. Coordinators can view detailed claim data, including supporting documents, before making decisions.

  **Manager Dashboard**: Displays verified claims awaiting final approval. Managers have access to comprehensive claim histories and can approve claims with a single click, expediting the process.

- The layout adheres to the MVC architecture, which separates concerns into three interconnected components(Troelsen and Japikse, 2020).

  **model**: Manages data and business logic, such as calculating the total claim amount based on hours worked and hourly rates.

  **View**: Handles the presentation layer, focusing on user interface elements and ensuring they are visually appealing and functional.

  **Controller**: Acts as an intermediary between the Model and View, processing user inputs and returning the appropriate responses (Troelsen and Japikse, 2020).

## Assumptions and Constraints:

**1. Assumptions:**

• **System accessibility:** The application is web-based, it is assumed that users (lecturers, program coordinators, and academic managers) have dependable internet connectivity in order to engage with the system.

• **User Ability:** In order to file claims, upload documents, and check statuses, users are expected to have a basic understanding of computers and be able to use a standard online interface.

• **Accuracy of Data:** The system is predicated on the lecturers' provision of correct data, including hourly rates and hours worked. There aren't any automated validation checks with outside systems.

• **User Roles:** Every user shall rigorously adhere to the responsibilities given to them. For example, Lecturers shall only submit claims and shall not use coordinator or management functions.

• **Document forms**: Supporting documents that are uploaded will be in forms that are accepted (such as PDF and JPEG), and it is assumed that these files are pertinent to the claim and free of viruses.

## 2. Constraints:

• **Technical Restrictions**: The application's development cannot use other technologies or frameworks because it must be created with.NET Core MVC and SQL Server in accordance with project specifications.

• **Data Privacy:** The system must abide by data privacy laws to guarantee the safe processing and storage of sensitive data, such as credit card and personal information.

• **Performance:** The system may need to take scalability into account for larger user bases or higher usage because it is built to manage a moderate number of users and claims submissions.

• **Deployment Environment:** The system will be set up in a web hosting environment that allows for the use of SQL Server and.NET Core, which may restrict the available platforms and services for deployment.

•**Security:** All user interactions, particularly claim submissions and document uploads, must occur over secure connections (HTTPS) in order to safeguard data.

• **Controlling Concurrency:**When managers and coordinators, for example, try to update the same claim at the same time, provide positive concurrency control to handle the situation.

**REFERENCES**

Troelsen, A. and Japikse, P., 2020. *Pro C# 9 with .NET 5: Foundational principles and practices in programming*. Tenth ed. Apress.

Yukti Solutions, n.d. *9 advantages of using .NET Core for developing your software*. Available at: https://yuktisolutions.com/blog/benefits-of-asp-net-core-for-web-application-development [Accessed (n.d)].