PROG6212

ST10259093

POE part1

Boikemiso

Project plan

Introduction and context

The Contract Monthly Claim System (CMCS) is a web application development proposal with the purpose of modernizing and facilitating the monthly payment claim procedure for contract lecturers in an educational institution. The present manual paper-based nature of the system is vulnerable to delays, errors, and non-transparency. This prototype leads to the digitalization which lecturers could use to declare claims, coordinators of the program could check details, and academic managers could approve payments all done efficiently. The purpose of digitization via CMCS is to promote accuracy, reduce delays, and provide visibility to the audit trail to all stakeholders, resulting in a shift in industries from manual downward administration systems.

Project plan

Task name	Duration of task	Dependencies
Project startup and documentaion	1-3 hours	None
Designing UML class diagram	2 hours	None
GUI prototype - Login and dashboard views	3 hours	
GUI prototype – Claim submission view	2 hours	View made
GUI prototype – Admin approval views	2 hours	View made
Finished documentation and approval	1 hour	Tasks complete
Finishing review and submission	1 hour	Review and submission done

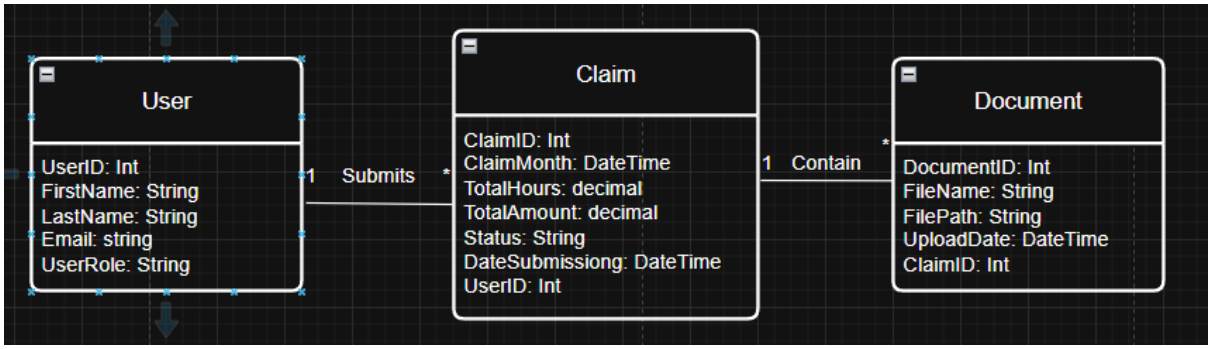
An iteration-based Agile method was chosen since it fits well with the nature of prototyping and evolving design decisions. The project was divided into key stages, such as initial setup and planning, UML database design, GUI prototype development, and final documentation. Values and dependencies were established, for example, the database design was completed before the GUI can even begin to be rationalized against that data structure. Resources include Visual Studio 2022, .NET 8 framework, and Draw.io for drawing diagrams. Another main risk was the scope creep that could affect the prototype, which was avoided through strict agreement to the non-functional requirement and focus on the major features.

Sytsem Design Choices

ASP.NET Core MVC is the framework chosen for this application because it follows a well-structured model-view-controller pattern that is suitable for scalable web applications. It separates the business logic from the data logic and the user interface, making development and maintenance simple. For data management, a relational database model was chosen to enforce integrity of data and efficiently handle relationships defined between user, claim, as well as supporting documents. The design supports the claim submission process from a lecturer's intuitive dashboard while coordinators and managers maintain separate interface views supporting their respective verification and approval workflows, thus ensuring one consistent user experience.

Database design

Three main entities define the database structure being User, Claim, and Document. The User entity holds all user details and the role of the user (Lecturer, coordinator, or manager). The Claim entity serves as the focal object, holding all monthly claim details along with its status. The Document entity, on the other hand manages storage of file uploads that are attached to each claim. In terms of relationships, one User submits many Claims, and in turn, one Claim contains many Documents.



Assumptions and Constraints

It is assumed that one user will be assigned only one system role, and that a lecturer can submit one claim per month. The claim process happens linearly: submitted, verified, and approved. The big limitation of Part 1 is that the application is only a non-functional prototype; therefore, no real data processing takes place, no files are uploaded, and no authentication logic is built in. Instead, it is a mere static wireframe for the GUI. Design limitations include the absence of real-time notifications and a blatant oversimplification of the approval process, with complex business rules and conditional workflows.

Conclusion

Part 1 successfully delivers the project plan with all its necessaries, the UML class diagram showing in detail the database schema, and a non-functioning GUI prototype designed using ASP.NET Core MVC. With these, a firm base for developing the system has been laid out. Planning and design have made an effective and clear plan of the structure and flow of the application. Hence, implementation of the functionality, integration of a live database, and coding of business logic for submission, verification, and approval will form the focus of Part 2.

(585 words)

Reference list

Satzinger, J.W., Jackson, R.B. and Burd, S.D. (2016). *Systems Analysis and Design in a Changing World*. 7th ed. Boston, Ma: Cengage Learning.

Project Management Institute (2021). *A guide to the project management body of knowledge: (PMBOK guide)*. 7th ed. Newtown Square, Pa: Project Management Institute.