|  |  |
| --- | --- |
|  |  |
| Programming Part 1 |  |
|  |  |
|  | 17 September 2025Prog6212 |
|  | Lebohang SelematselaST10446504 |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  | Documentation **Introduction**  Universities and colleges rely on part-time or contract lecturers most of the time who submit monthly claims for their teaching hours and other activities they do. Traditionally this process was always paper based which required the lecturers to fill out forms, attach the required documents and submit the forms and documents to the department. After this whole process the manager had to review the claims manually, which was time-consuming, which led to errors, and it had a lack of transparency. The Contract Monthly Claim System (CMCS) has been proposed as a digital solution to streamline the claim process. The current stage of this development is to focus on the design and the presentation of a graphical user interface (GUI) which currently is only a prototype. This system will allow lecturers to create and submit claims online, allow managers to review and approve the claims efficiently and both parties are going to benefit from a transparent and paperless workflow.  **Objectives**  Main objectives:  **.**  Provide lecturers with an easy-to-use interface for submitting monthly claims.  **.**  Ensures supporting documents are uploaded easily and associated with the claims.  **.**  Gives managers the ability to review and approve/reject claims digitally.  **.**  It maintains transparency by allowing lecturers to track their claim status.  **.**  It reduces paper usage and moves towards a fully electronic approval process.  **Assumptions and Constraints**  **.** This prototype is just a GUI meaning it has no back end or database functionality.  **.** Only registered contact lecturers can access this claim submission feature.  **.** Managers and administrative staff have distant roles with the appropriate permissions.  **.** This project should be completed within 5 weeks, focusing on design deliverables.  **UML Class Diagram**  UML class diagram main entities and methods:  **. Lecturer**: Represents the contract lecturer who submits the claims.  Attributes: LecturerID, Name, Department and Email  Methods: submitClaim() and updateClaim()  **. Claim**: Represents each submitted claim.  Attributes: ClaimID, LectureID, DateSubmitted, ClaimAmount, Status and Description  Methods: createClaim(), updateClaim() and cancelClaim()  **. SupportingDocuments**: Represents the files uploaded to justify a claim.  Attributes: DocumentID, ClaimID, FileName, FilePath and Description  Methods: uploadDocument() and removeDocument()  **. Manager**: Represents the user responsible for reviewing the claims.  Attributes: ManagerID, Name, Email and Role  Methods: reviewClaim() and fowardClaim()  **. Approval**: Records the manager’s decision and links a claim with a manager.  Attributes: ApprovalID(), ClaimID, ManagerID, ApprovalDate, Decision and Comments  Methods: approveClaim() and rejectClaim()  **Project Plan**  This project should be completed within a period of 5 weeks.  **Week 1**: Requirements and Initial Design  Identify the system’s scope, constraints and requirements. Draft an early documentation.  **Week 2**: UML and Project Plan  Develop a UML diagram and create a table/Gantt chart for the project scheduling.  **Week 3**: GUI Wireframes and Prototype  Design a key screen which includes the lecture dashboard, claim submission form and a Manager Dashboard.  **Week 4**: Review and Refine GUI + Documentation  Conduct peer review, refine layouts and update the documentation.  **Week 5**: Report and Git Flow  Finalize the report and make sure that the diagrams are embedded and finally commit all the files with proper version control.  **Wireframes**  The 3 main wireframes:  **. Lecturer Dashboard**: This provides navigation and quick access to create claims, upload documents and view past submissions.  **. Claim Submission Form**: This contains fields for the date, amount, description and file uploads.  **. Manager Dashboard**: This dashboard displays pending claims requiring review, approved claims and rejected claims.  **Version Control Plan**  The 5 commits:  **1.** Added a ReadMe file  **2.** Added my controllers  **3.** Added my models  **4.** Added my views  **5.** Decorated my project made it look professional  **Conclusion**  This CMCS prototype provides a GUI model of how the contract lecturer claim system is going to function. The documentation, UML diagram, project plan, wireframes and version control strategy together shows a clear pathway for a full system development in later phases. Project Plan **Methodology**  This project follows an Iterative development methodology. Each week represents a cycle of design, review and refinement. Using this approach allows  Incremental progress, early feedback and continuous improvement without needing a fully finished product before evaluation.  **Tasks, Dependencies and Timeline**  **Week 1**: Requirements and Initial Design  Define system scope and constraints.  **Week 2**: UML and Project Plan  Create a UML diagram and table/Gantt chart.  **Week 3**: GUI Wireframes and prototype  Design and implement initial front-end screens.  **Week 4**: Review and Refine GUI + Documentation  Review deliverables and apply feedback.  **Week 5**: Report + Git Flow  Finalize documentation and ensure proper version control.  **Resources**  **Tools**: Draw.io (UML class diagram) and LucidChart (Table)  **Technologies**: Visual Studio (MVC)  **Version Control**: GitHub  **Risks, Assumptions and Constraints**  **Risks**: Time limitations, inconsistent feedback and potential design misalignment.  **Assumptions**: Lecturers and managers use web-based access.  **Constraints**: Project limited to 5 weeks and prototype is restricted to GUI only (no back-end functionality)   System Design Choices **Design Justification**  The system was designed with simplicity and usability as its primary goals. Since the lecturers and managers are the main users the interface prioritizes clarity, input fields and step-by -step navigation. This interface separates the lecturer and manager dashboards which reduces complexity and ensures that each user only interacts with the features that are relevant to their role. The system adopts an MVC (Model-View-Controller) architecture in the prototype. This approach adopts an MVC architecture in the prototype. This approach separates concerns which make it easier to maintain the interface, improve scalability and eventually connect to a database back end.  **Database Structure**  Even though this stage only focuses on GUI design, the system’s UML class diagram reflects an underlying database schema.  **Tables**:  **. Lecturer**: LecturerID, Name, Department, Email  **. Claim**: ClaimID, LecturerID, DateSubmitted, ClaimAmount, Status, Description  **.** **SupportingDocumen**t: DocumentID, ClaimID, FilePath, Description  **. Manager**: ManagerID, Name, Role, Email  **. Approva**l: ApprovalID, ClaimID, ManagerID, ApprovalDate, Decision, Comments  The primary keys are used to identify records while the foreign keys are used to link entities. Using this structure ensures traceability and enforces relationships between claims, documents and approvals.  **Support for Claim Workflow**  **. Claim Submission**: A lecturer logs in, enters the claim details into the claim form provided and uploads supporting documents which the claim table stores the record and the supporting documents table links the files to the claim.  **. Verification**: Once the claim is submitted the claim status is set to “Pending” then the managers can view the pending claims through their dashboard which retrieves data from the claim and the lecturer tables.  **. Approval**: A manager records the decision in the approval table which is linked to both the claim and the manager then the claim status is updated to “Approved” or “Rejected” and the lecturer can tract the status from their status.  This design ensures that every claim can be traced from the submission all the way through to the decision (Approval or rejection) with supporting evidence stored and accessible for verification. The database structure also allows future scalability such as adding audit logs or integrating with the payroll systems. Database Design **UML Class Diagram**  This UML class diagram represents the data requirements of the Contact Monthly Claim System (CMCS). It models the entities and their relationships. The diagram shows how the data flows between users and the claim verification process.  **.** Lecturer(1..) **→** (1..\*) Claim  **.** Claim(1..) **→** (0..\*)SupportingDocument  **.** Manager(1..) **→** (0..)Approval  **.** Claim (1..1) **→** (0..\*)Approval  **Entities and Relationships**  **Lecturer**  **.** Attributes: LecturerID (PK), Name, Department, Email  **.** Each Lecturer can submit multiple claims.  **Claim**  **.** Attributes: ClaimID (PK), LecturerID (FK), DateSubmitted, ClaimAmount, Status, Description  **.** Each Claim belongs to one Lecturer and may have multiple SupportingDocuments.  **SupportingDocument**  **.** Attributes: DocumentID (PK), ClaimID (FK), FilePath, Description  **.** Each document is tied to one Claim.  **Manager**  **.** Attributes: ManagerID (PK), Name, Role, Email  **.** Each Manager can review multiple claims.  **Approval**  **.** Attributes: ApprovalID (PK), ClaimID (FK), ManagerID (FK), ApprovalDate, Decision, Comments  **.** Each Approval record connects a Claim with a Manager’s decision.  **Data Attributes and Constraints**  **. Primary Keys (PK)**: Guarantees unique identification of records.  **. Foreign Keys (FK)**: Enforces referential integrity between tables.  **. Status Field**: Constrained to values such as pending, approved, rejected to standardize claim tracking.  **. Decision Field in Approval**: Restricted to approved or rejected values.  **. Claim Amount**: Must be a positive number.  **. Data Fields**: Must store valid calendar dates.  **Rationale for Data Model**  **. Data integrity** through primary and foreign keys.  **. Flexibility** to handle one-to-many relationships.  **. Scalability** for future integration with payroll systems or audit logs.  **. Clarity** by organizing information into distinct and normalizing entities that reduce redundancy.  This database design makes sure that the system can support the full claim cycle.   Graphical User Interface **GUI Layout and Technology**  This prototype for the Contract Monthly Claim System (CMCS) is implemented using an MVC. Using this approach allows clean separation design, logic and data representation. If the system were extended in the future the MVC structure would also support integration.  **Key Screens**   1. **Lecturer Dashboard**: This provides an overview of submitted claims, statuses and an option to create a new claim. 2. **Claim Submission Form**: This allows lecturers to enter claim details and upload supporting documents. 3. **Document Upload Page**: This enables attaching multiple files to a claim. 4. **Manager Dashboard**: Displays pending claims requiring review with filters to show approval and rejected claims 5. **Claim Tracking Page**: This allows lecturers to view the progress and status of the submitted claims.   **Usability, Consistency and Accessibility Principles**  **. Usability**: Forms use clear labels, mandatory field indicators and validation feedback which is used to minimize input errors and navigation is simplified with a sidebar menu.  **. Consistency**: Uniform layouts, consistent button styles and recurring icons make sure that the user can quickly recognize functions across different screens.  **. Accessibility**: High-contrast colors and readable fronts were chosen to ensure legibility. Input fields and buttons follow accessibility standards for keyboard navigation and screen readers.  **Wireframes and Screenshots**  **1. Dashboard**    **2. Lecturers**  **A screenshot of a computer  AI-generated content may be incorrect.**  **A screenshot of a computer  AI-generated content may be incorrect.**  **3. Claims**    **A screenshot of a computer  AI-generated content may be incorrect.**  **4. Managers**  **A screenshot of a computer  AI-generated content may be incorrect.**  **A screenshot of a computer**  **5. Approvals**    **A screenshot of a computer  AI-generated content may be incorrect.**  **GitHub**  **1**. **ReadMe File**    **A screenshot of a computer  AI-generated content may be incorrect.**  **2. Controllers**    **A screenshot of a computer  AI-generated content may be incorrect.**  **3. Models**    **A screenshot of a computer  AI-generated content may be incorrect.**  **4. Views**    **A screenshot of a computer  AI-generated content may be incorrect.**  **5. Display**    **A screenshot of a computer  AI-generated content may be incorrect.** Assumptions and Constraints **Assumptions**  **. User Roles**: There are only 2 primary roles in the system which are Lecturers (Submit claims) and Managers (Review and approve claims)  **. Claim Process**: Claims can be submitted monthly which could include teaching-related description and supporting documents.  **. Supporting Documents**: All claims submitted must include at least 1 valid document and documents should be in a digital format.  **. System Access**: Users access the system through the website.  **. Workflow**: Every claim must go through submission, verification and approval steps before being finalized.  **Constraints**  **. Prototype Scope**: This non-functional interface includes only GUI mock-ups, diagrams and documentation with no back end or database integration.  **. Timeframe**: The project is running on a 5-week schedule which does restrict the depth of the development.  **. Technology Choice**: This prototype is limited to a front end MVC structure with static data.  **Design Limitations**  **.** There is no live integration with payroll or institutional databases has been implemented.  **.** Authentication and authorization are not fully modeled but instead user roles are stimulated in the wireframes.  **.** Error handling, file validation and security checks assumed but not included in this stage of the prototype.  **.** Scalability and performance will be discussed in the later stages. Conclusion Part 1 of the Contract Monthly Claim System (CMCS) shows the foundation of a structured, paperless claim process for contact lecturers. This report has shown a clear problem statement, objectives and system documentation which is followed by a project plan which outlines the tasks, dependencies, methodology, resources and risks. A UML class diagram and database design which illustrates the underlying data requirements while the GUI prototype showcased how lecturers and managers will interact with the system through dashboards, submission forms and approval screens. Wireframes, screenshots and design justifications that the interface is developed with usability, consistency and accessibility in mind. Assumptions and constraints were documented to set realistic expectations, which ensures that the scope remained focused on a GUI-only prototype that meets the 5-week timeline. These deliverables form a robust foundation for future development. In the later phases this system can be extended and have a fully functional back end, database integration and live approval workflows. Once this stage is completed the team will be able to establish groundwork for a scalable and reliable system that can eventually support institutional claim processing at a production level. Reference List SPARX SYSTEMS (n.d.). *Database Modeling with UML | Sparx Systems*. [online] sparxsystems.com. Available at: https://sparxsystems.com/resources/tutorials/uml/datamodel.html.  ‌ Malsam, W. (2025). *8 Project Plan Examples (Templates Included)*. [online] ProjectManager. Available at: https://www.projectmanager.com/blog/project-plan-examples.  ‌ | |  |
|  |  |  |  |

|  |  |
| --- | --- |
|  |  |