2024

PROGRAMMING 2B

PART 1 (PROTOTYPE PLANNING)

Thato NGWAKO Menetje
st10221273

1. Documentation for Contract Monthly Claim System Prototype

Design Choices:

- **Technology Stack:**
    - **WPF (Windows Presentation Foundation)**: WPF was chosen for its flexibility in building complex desktop applications with a modern UI. It allows for a clean separation of logic (via MVVM pattern) and design (using XAML), providing the ability to create a professional-looking and responsive UI.
    - **C# with .NET Core**: C# was selected due to its strong integration with WPF and .NET Core. It is a widely used language for enterprise-grade desktop applications, which makes it ideal for handling the logic behind the system.
    - **XAML for UI**: XAML (Extensible Application Markup Language) is used to define the GUI layout. It allows the separation of user interface design from application logic and provides powerful binding, styling, and templating features.

User Interface (UI) Design:

- **Color Scheme**:
    - A professional, modern color palette of purples (#4A148C and #7E57C2) and pinks (#D81B60) was chosen to enhance readability, engagement, and visual appeal while maintaining a formal tone.
    - Accent colors such as white and light shades were used to create contrast and highlight key elements like buttons, text fields, and borders.

- **Layout & Structure**:

- ❖ **Grid System**: The UI primarily uses a grid-based layout to organize content systematically. This ensures the layout is adaptable to various screen sizes and resolutions while keeping it intuitive for users.
- ❖ **TextBox and Border Elements**: Input fields (such as for lecturer IDs, claim dates, and descriptions) were enclosed within Border elements to improve visual structure. This also provides a clear visual separation between labels and input areas.
- ❖ **Buttons**: The buttons use ControlTemplate for customization, ensuring a uniform design with rounded corners and consistent color schemes. This adds to the polished appearance of the application.
- ❖ **Data Display**: Data grids are used in forms like ClaimHistory.xaml and ProgrammeCoordinator.xaml for displaying claims. The grids are configured with custom headers for fields such as Lecturer Name, Claim Date, Claim Amount, and Status, allowing users to easily browse and interact with the data.
- ❖ **StackPanel**: A vertical StackPanel was used for the form layout (in the SubmitClaim.xaml file) to maintain a neat, linear structure for each input field.

- **Modern User Experience (UX) Features**:

  - ❖ **Rounded Corners**: Applied to input fields, buttons, and containers to give the interface a softer, more modern look.
  - ❖ **Uniform Typography**: Text blocks and labels use a consistent font size and weight to enhance clarity, with important elements such as section headers emphasized by increased font size and bold text.
  - ❖ **Search and Tabs**: The Programme Coordinator window incorporates a search section to allow quick filtering of claims. A TabControl is used to switch between different categories of claims (Pending, Approved, Rejected).

Database Structure:

The system uses a database to store claim records, user information, and statuses.

Database Tables:

1. **Lecturers**:

    - **LecturerID** (Primary Key): Unique identifier for each lecturer.
    - **FirstName**: First name of the lecturer.
    - **LastName**: Last name of the lecturer.
    - **Email**: Lecturer's email for correspondence.
    - **Department**: The department the lecturer belongs to.

2. **Claims**:

    - **ClaimID** (Primary Key): Unique identifier for each claim.
    - **LecturerID** (Foreign Key): Reference to the lecturer who submitted the claim.
    - **ClaimDate**: Date when the claim was submitted.
    - **Description**: Detailed description of the claim.
    - **Amount**: The amount being claimed.
    - **Status**: Status of the claim (Pending, Approved, Rejected).

3. **ClaimActions**:

    - **ActionID** (Primary Key): Unique identifier for each action.
    - **ClaimID** (Foreign Key): Reference to the related claim.
    - **ActionType**: Type of action performed (Approve, Reject).
    - **ActionDate**: Date when the action was performed.
    - **PerformedBy**: The user (admin or coordinator) who took the action.

2. **Users** (Admin, Programme Coordinator):
    - **UserID** (Primary Key): Unique identifier for the user.

- **Username**: Username for login.
- **Password**: Hashed password for security.
- **Role**: Role of the user (Admin, Programme Coordinator).

3. **Database Relationships:**

- **One-to-Many** between Lecturers and Claims: Each lecturer can submit multiple claims.
- **One-to-Many** between Claims and ClaimActions: Each claim can have multiple actions (Approve, Reject).
- **One-to-One** between Users and actions in ClaimActions: Each action on a claim (approval or rejection) is performed by a user (admin or coordinator).

GUI Layout Summary:

1. **MainWindow**:
    - Acts as the dashboard for navigation between different windows such as SubmitClaim, ProgrammeCoordinator, and ClaimHistory.
    - Contains buttons to open various windows.
2. **SubmitClaim.xaml**:
    - A form-based layout allows lecturers to submit claims with fields for Lecturer ID, claim date, description, and amount.
    - Includes a button for uploading supporting documents and action buttons for submitting or resetting the form.
3. **ProgrammeCoordinator.xaml**:
    - Displays claims that are Pending, Approved, or Rejected using a tab-based interface.
    - Each tab contains a DataGrid displaying the relevant claims and allows for actions like approving or rejecting claims.
4. **ClaimHistory.xaml**:

- ❖ Shows the history of claims for a lecturer, using a DataGrid to display details such as Claim Date, Amount, and Status.

Assumptions and Constraints:

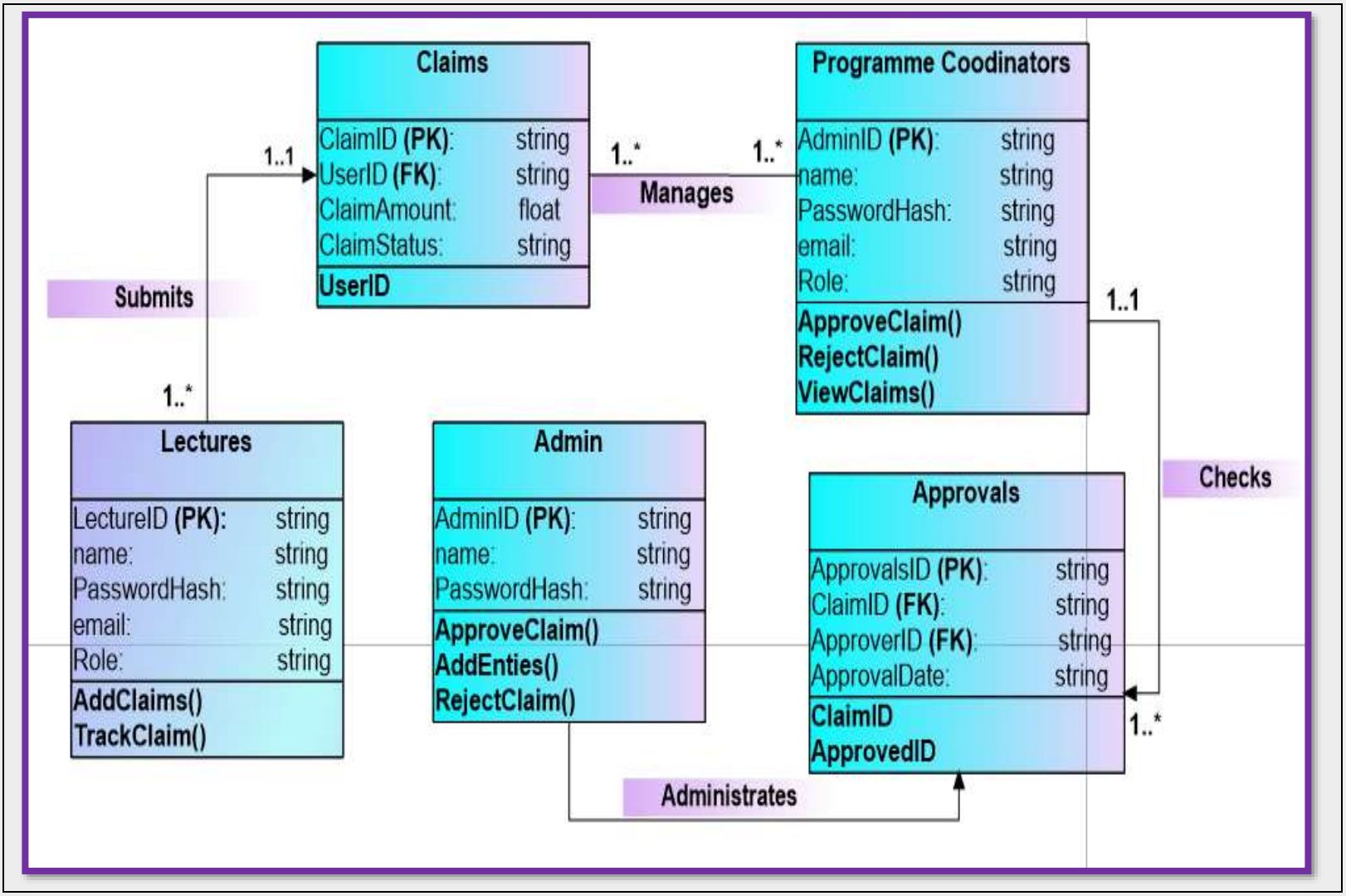
1. **Assumptions**:

   - ❖ Lecturers are responsible for submitting their claims via the provided interface.
   - ❖ Programme Coordinators and Admins have separate roles with distinct permissions (e.g., Coordinators approve/reject claims, while Admins manage the system).
   - ❖ Claim statuses are limited to "Pending", "Approved", and "Rejected" for simplicity.
   - ❖ The database is relational and will store essential information about lecturers, claims, and user actions.
   - ❖ WPF and .NET Core will handle the entire desktop application, and no web-based interface is required at this stage.

2. **Constraints**:

   - ❖ The design assumes a desktop environment, meaning mobile support or web-based features are not included in this prototype.
   - ❖ The prototype does not include complex workflows like email notifications or automatic claim validation, which may be part of a more complete system.
   - ❖ For security reasons, user authentication is assumed to be basic with hashed passwords but without more advanced measures like OAuth or SSO.
   - ❖ The GUI does not include advanced reporting or analytics features, as it is a simple interface focused on claim submission and management.

2.UML Class Diagram



3.Project planner

Project Plan for Contract Monthly Claim System Prototype

1. Project Overview

- **Objective**: Develop a prototype of the Contract Monthly Claim System, including GUI design, database structure, and essential functionalities.

- **Duration**: 6 weeks
- **Team**: 1 Developer

2. Tasks and Timeline

Task	Description	Start Date	End Date	Dependencies	Status
1. Requirements Analysis	Define detailed requirements and scope of the prototype.	Sep 9, 2024	Sep 10, 2024	None	Not Started
2. Database Design	Design database schema and relationships.	Sep 11, 2024	Sep 12, 2024	Requirements Analysis	Not Started
3. UI/UX Design	Create wireframes and design mockups for the application's UI.	Sep 13, 2024	Sep 15, 2024	Requirements Analysis	Not Started
4. Setup Development Environment	Install and configure development tools and environment.	Sep 16, 2024	Sep 17, 2024	None	Not Started
5. Main Window Implementation	Develop and implement the MainWindow with navigation buttons.	Sep 18, 2024	Sep 20, 2024	UI/UX Design	Not Started
6. Submit Claim Window	Implement the Submit Claim form with validation and functionality.	Sep 21, 2024	Sep 23, 2024	UI/UX Design, Main Window	Not Started

7. Claim History Window	Develop Claim History window with DataGrid and data binding.	Sep 24, 2024	Sep 26, 2024	UI/UX Design, Main Window	Not Started
8. Programme Coordinator Window	Implement Programme Coordinator window with tab control and DataGrid.	Sep 27, 2024	Sep 30, 2024	UI/UX Design, Main Window	Not Started
9. Backend Integration	Integrate database with application for data retrieval and submission.	Oct 1, 2024	Oct 3, 2024	Database Design, UI Windows	Not Started
10. Testing and QA	Perform unit testing, integration testing, and user acceptance testing.	Oct 4, 2024	Oct 7, 2024	All Previous Tasks	Not Started
11. Documentation	Prepare detailed project documentation, including user manuals and code comments.	Oct 8, 2024	Oct 10, 2024	Testing and QA	Not Started
12. Final Review and Adjustments	Review project, make final adjustments, and prepare for presentation.	Oct 11, 2024	Oct 13, 2024	Documentation	Not Started

13. Deployment	Deploy the prototype and conduct final walkthrough.	Oct 14, 2024	Oct 15, 2024	Final Review and Adjustments	Not Started

3. Key Milestones

- **Requirements Analysis Complete**: Sep 10, 2024
- **Database Design Complete**: Sep 12, 2024
- **UI/UX Design Approved**: Sep 15, 2024
- **Development Environment Ready**: Sep 17, 2024
- **Main Window Implementation Complete**: Sep 20, 2024
- **Submit Claim Window Complete**: Sep 23, 2024
- **Claim History Window Complete**: Sep 26, 2024
- **Programme Coordinator Window Complete**: Sep 30, 2024
- **Backend Integration Complete**: Oct 3, 2024
- **Testing and QA Complete**: Oct 7, 2024
- **Documentation Complete**: Oct 10, 2024
- **Final Review and Adjustments Complete**: Oct 13, 2024
- **Deployment and Final Walkthrough**: Oct 15, 2024

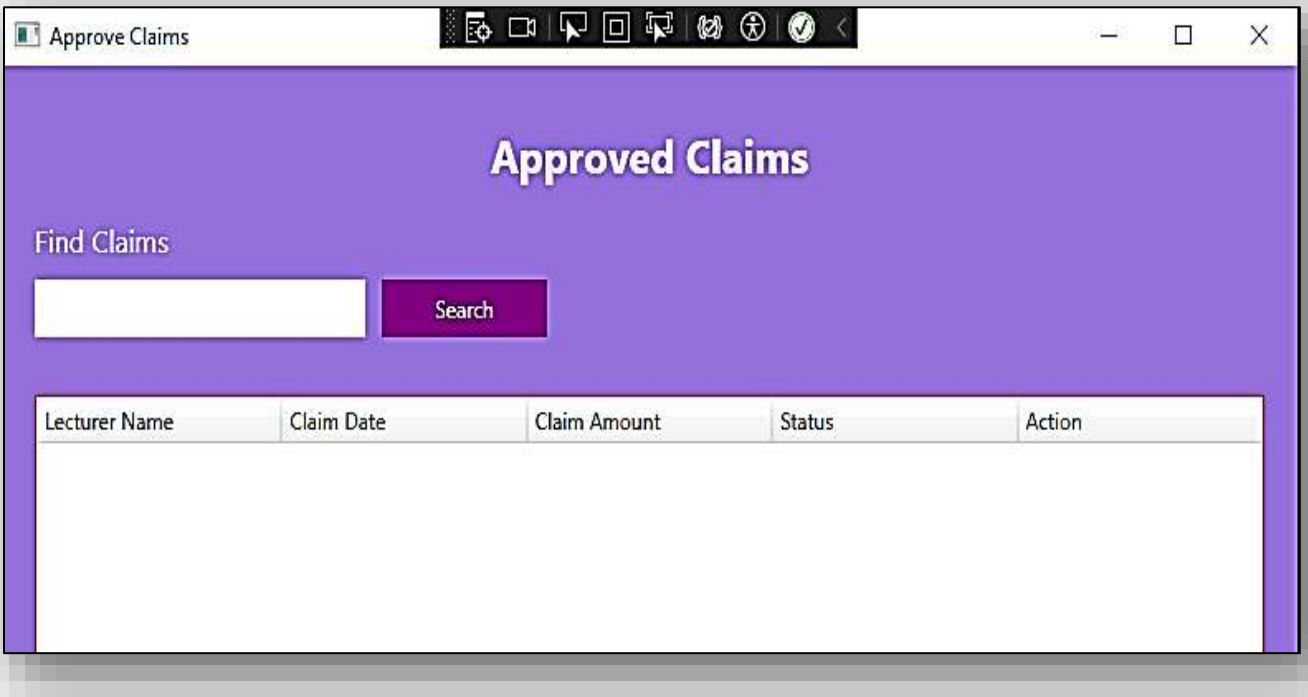
4. Dependencies

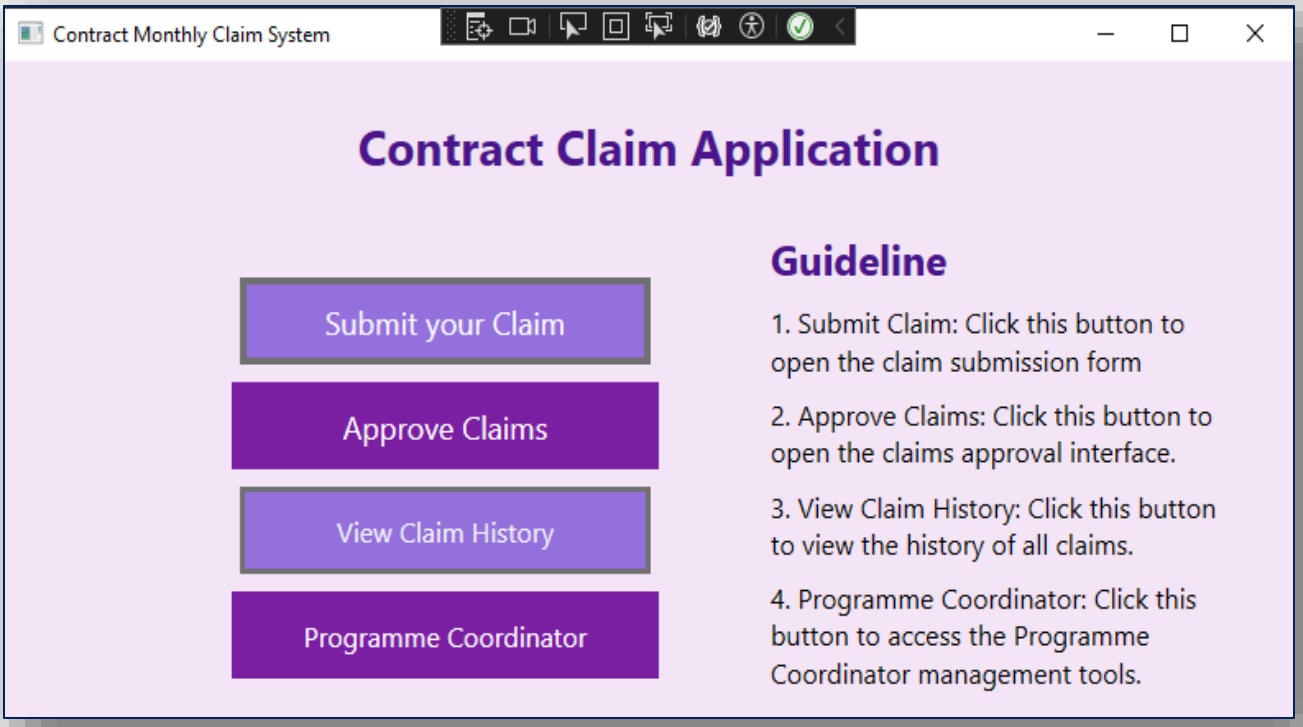
- **Requirements Analysis**: This must be completed before database design, UI/UX design, and development environment setup.
- **Database Design**: Required before backend integration and data binding in the UI.
- **UI/UX Design**: Essential for implementing all the windows (MainWindow, SubmitClaim, ClaimHistory, ProgrammeCoordinator).
- **Backend Integration**: Necessary for linking the UI with the database.
- **Testing and QA**: Must follow all development tasks to ensure functionality and quality.
- **Documentation**: To be completed after testing to ensure all features and functionalities are documented.

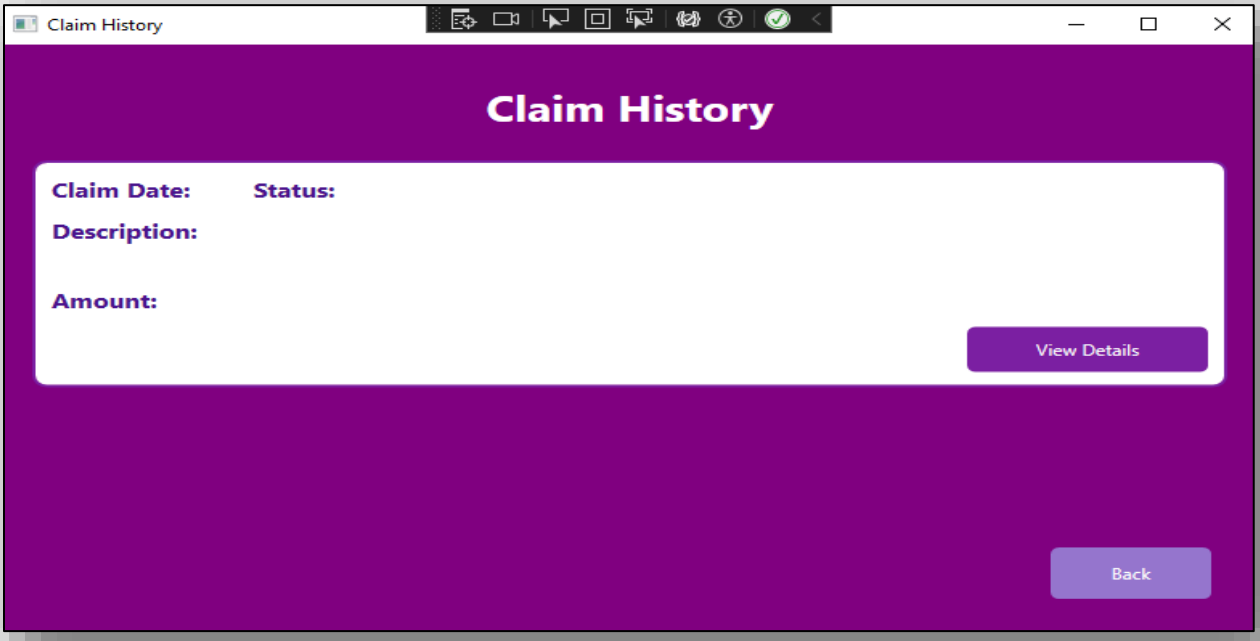
5. Assumptions and Constraints

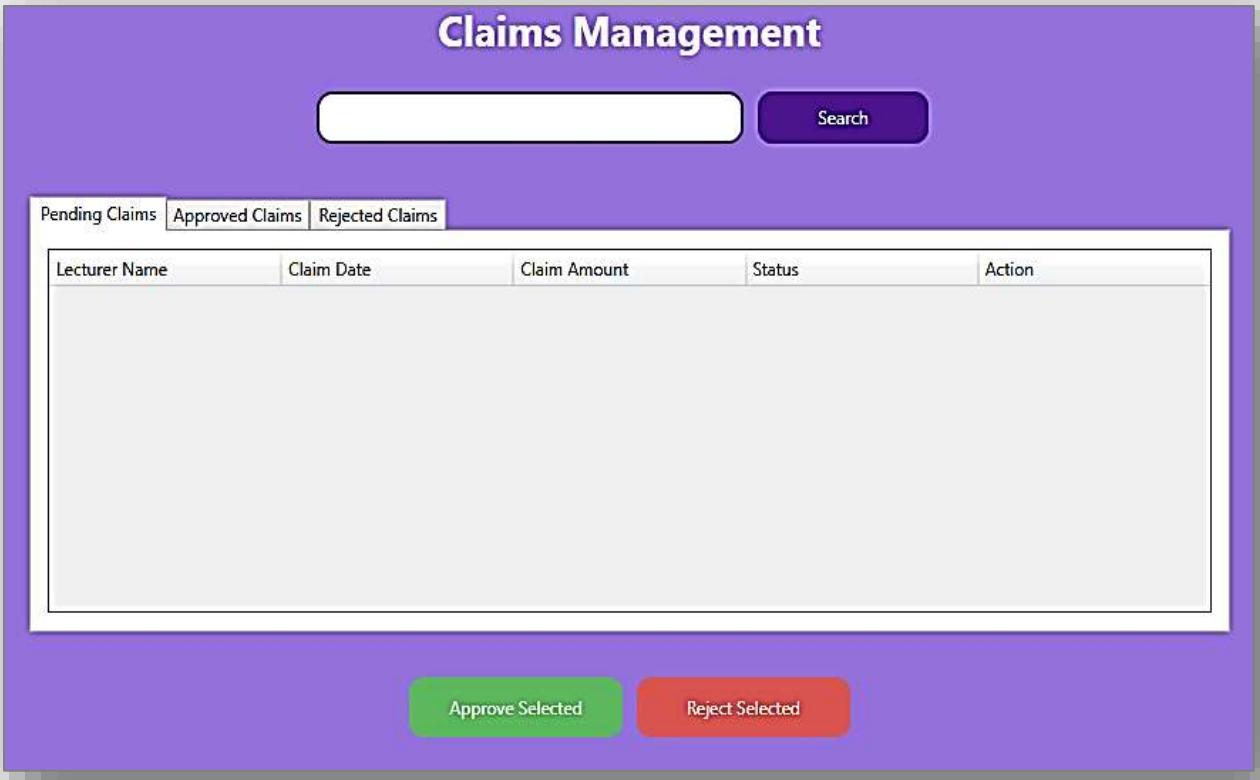
- **Assumptions**:
    - ❖ The development will be carried out with no major changes in requirements.
    - ❖ All dependencies and resources will be available as planned.
    - ❖ Testing will cover typical use cases and edge cases relevant to the prototype.
- **Constraints**:
    - ❖ Time constraints may impact the depth of testing and documentation.
    - ❖ Limited scope to prototype functionality means some advanced features may not be included.

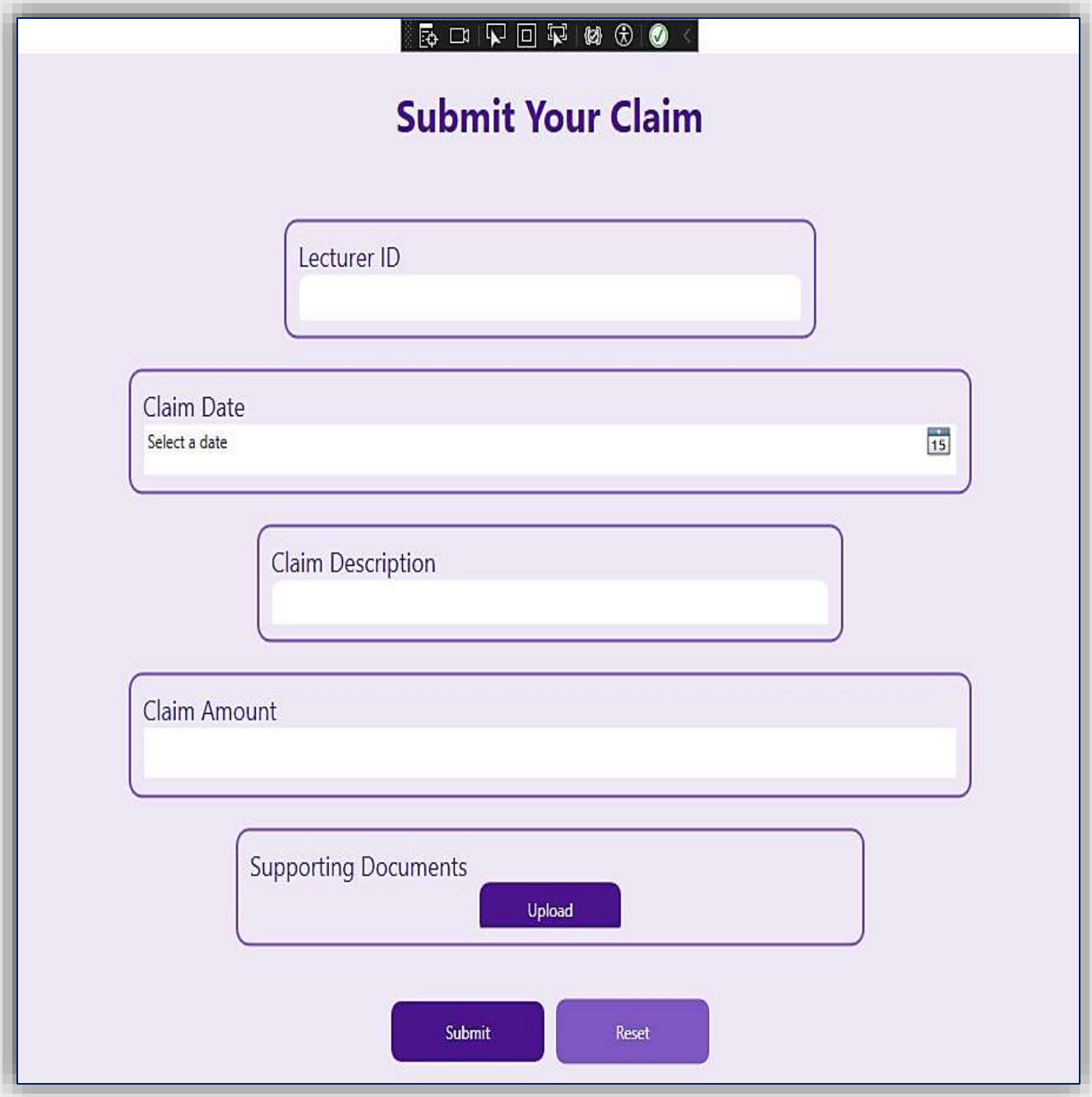
Prototype Screenshots




Contract Monthly Claim System
Contract Claim Application
Submit your Claim
Approve Claims
View Claim History
Programme Coordinator
Guideline
1. Submit Claim: Click this button to open the claim submission form
2. Approve Claims: Click this button to open the claims approval interface.
3. View Claim History: Click this button to view the history of all claims.
4. Programme Coordinator: Click this button to access the Programme Coordinator management tools.


Claim History
Claim History
Claim Date:
Status:
Description:
Amount:
View Details
Back


Claims Management
Search
Pending Claims
Approved Claims
Rejected Claims
Lecturer Name
Claim Date
Claim Amount
Status
Action
Approve Selected
Reject Selected


Submit Your Claim
Lecturer ID
Claim Date
Select a date
Claim Description
Claim Amount
Supporting Documents
Upload
Submit
Reset