***St10372889***

|  |
| --- |
| ***Julie*** |
| ***Kalumba*** |
| ***Part 1*** |
| ***Poe prog*** |
|  |

|  |
| --- |
| **Document** |
| **Design Choices**  **Project Overview:** |

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

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

Objective: To design and prototype a Contract Monthly Claim System with a GUI that allows lecturers to submit claims, academic managers to approve them, and supports document uploads.

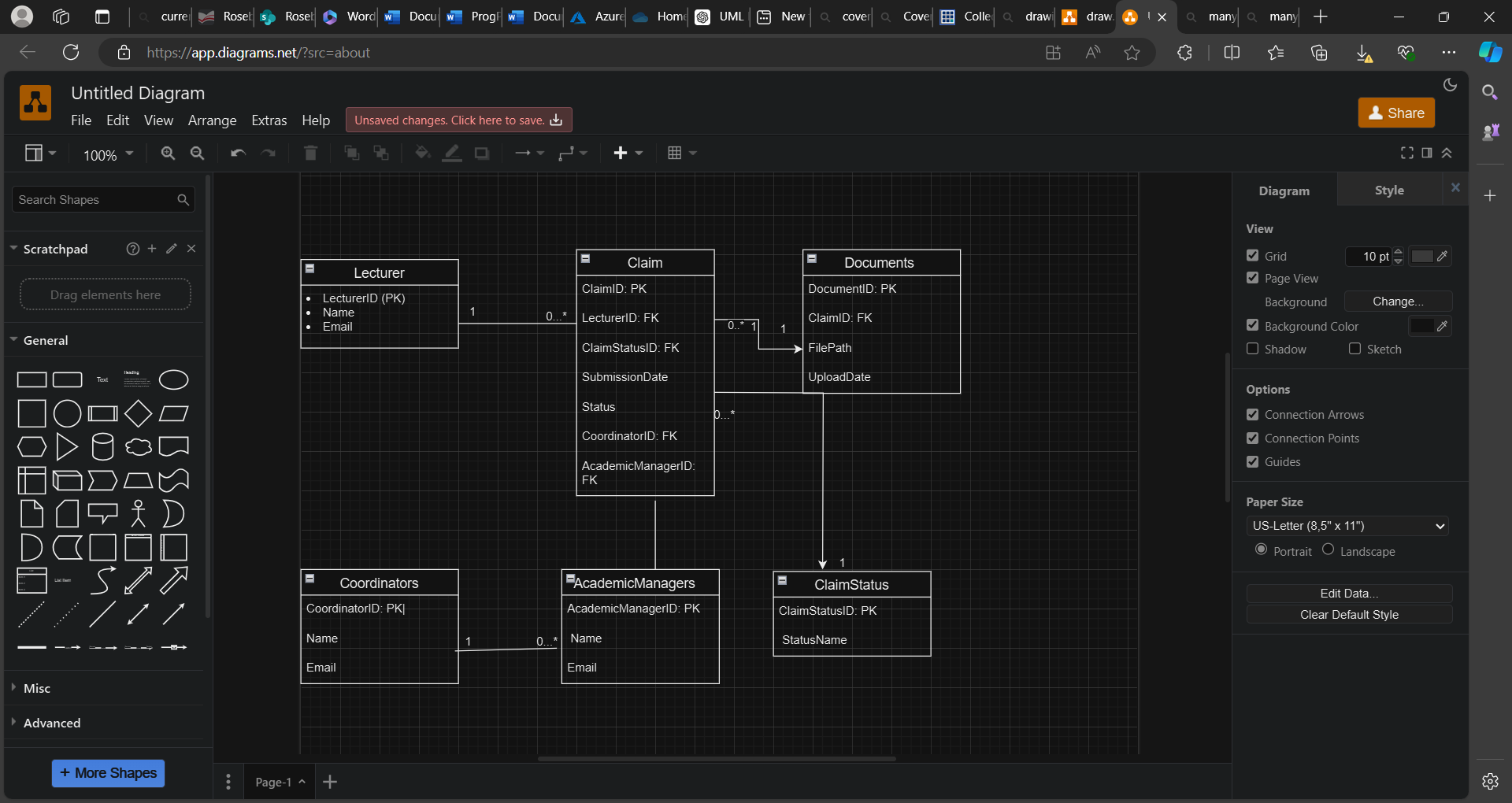
-Technologies: .NET Core for GUI, Windows Presentation Forms or MVC, UML for database structure, GitHub for version control.

The system will use Windows Presentation Forms (WPF) to create the graphical user interface (GUI). This provides a modern and responsive interface for the users (lecturers, coordinators, academic managers).

The chosen database structure is **relational**, using entities and relationships that clearly represent the real-world relationships between **lecturers**, **claims**, and **documents**. A relational database supports **data integrity**, **complex querying**, and ensures scalability as the system grows.

The structure consists of entities that are easy to manage and **normalize**, reducing data redundancy and ensuring efficient storage of multiple claims and related documents.

|  |
| --- |
| **Database Structure** |



The system will have six main tables, corresponding to the key entities:

Lecturer Table: Stores lecturer details.

Claim Table: Stores claim details submitted by lecturers.

Document Table: Stores paths to supporting documents uploaded for each claim.

Coordinator Table: Stores coordinator details, responsible for initial claim review.

AcademicManager Table: Stores details of the academic managers, responsible for final approval.

ClaimStatus Table: Tracks the status of each claim (e.g., "Pending", "Approved", "Rejected").

Relationships between tables will be set up to ensure that claims can have multiple supporting documents, lecturers can submit multiple claims, and each claim has to go through both a coordinator and an academic manager for approval.

|  |
| --- |
| **GUI Layout** |

Rationale Behind UI Elements:

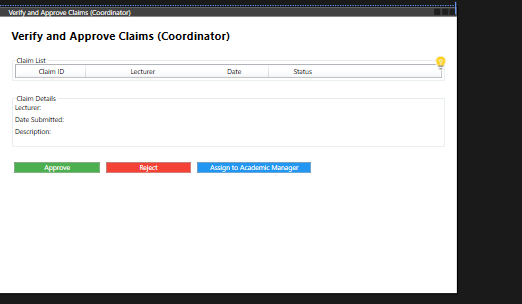
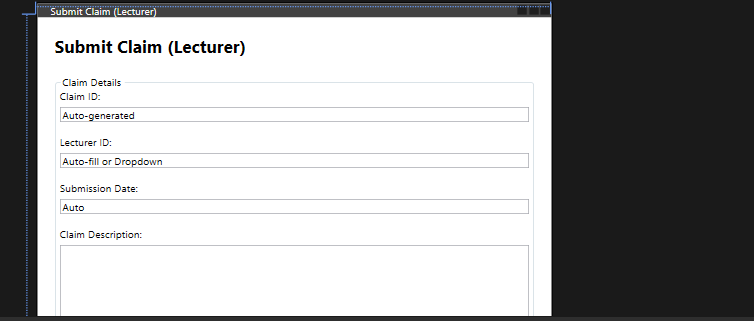
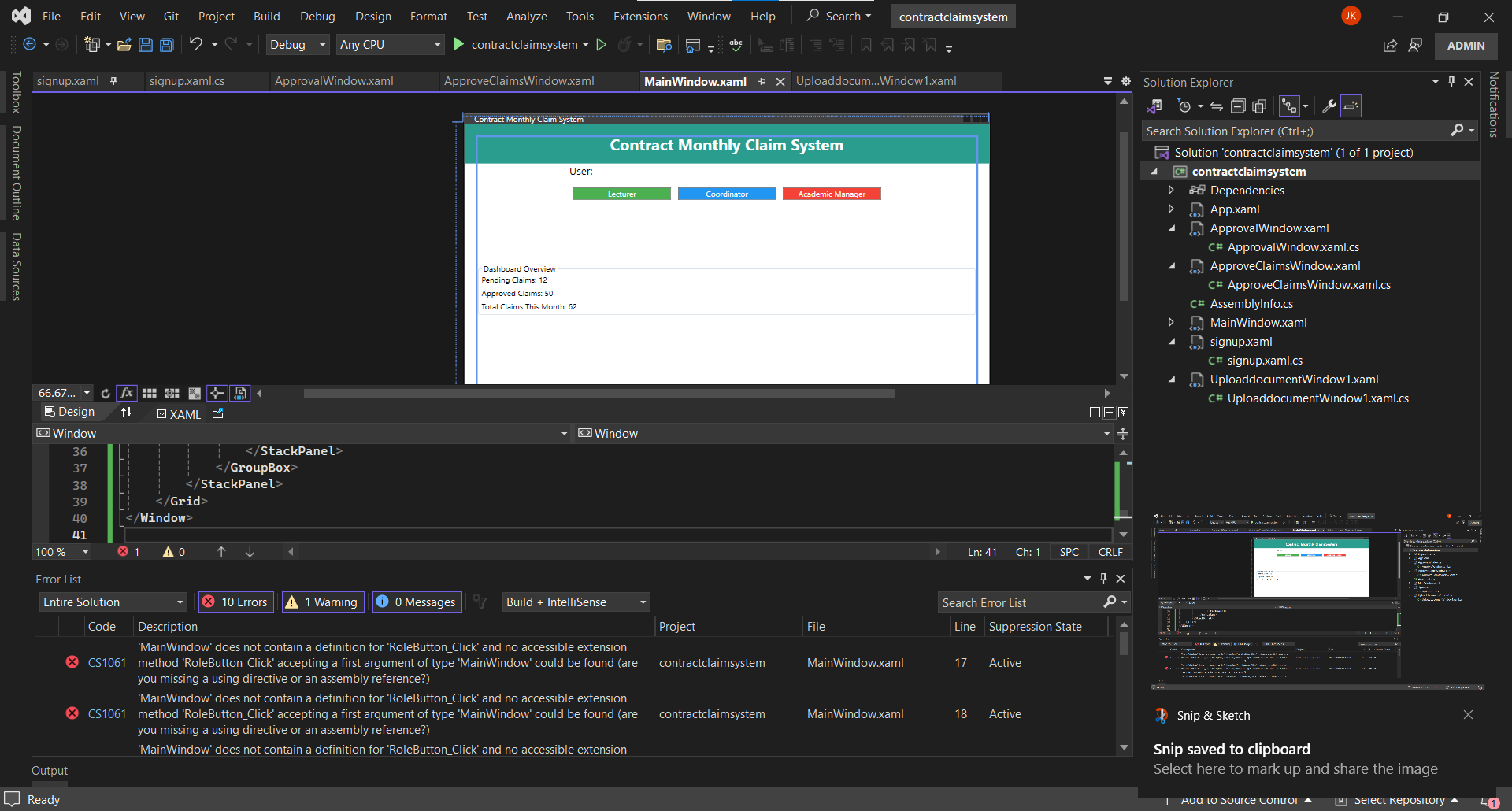
Buttons: Used for actions and navigation. Buttons are chosen for their clear visual distinction and intuitive functionality, making it easy for users to interact with the system.

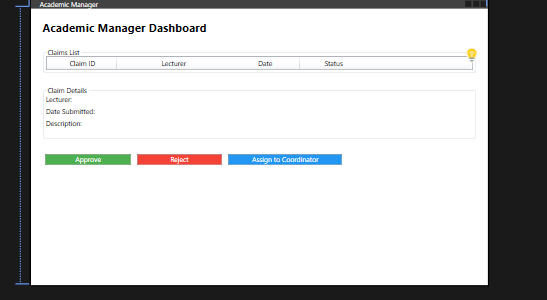
TextBoxes: Gather user input or display static information. TextBoxes are used for user-entered data or displaying read-only information such as Claim ID.

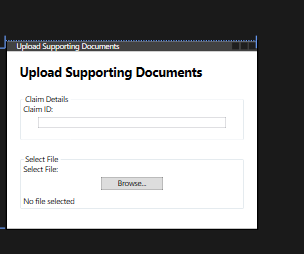
-ListViews: Display lists of items in a structured format. ListViews with GridViews are used to show tabular data, which is ideal for displaying lists of claims.

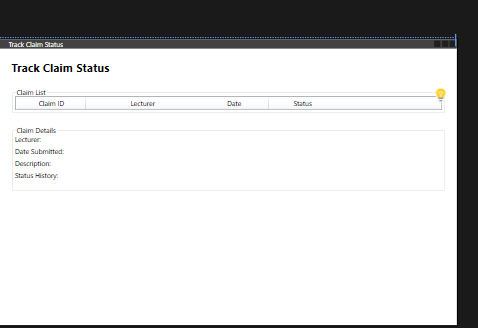
TextBlocks: Provide non-editable text and status updates. TextBlocks are used for displaying labels, status messages, and detailed information.

This layout aims to balance functionality with usability, ensuring that users can easily navigate the system, perform their tasks, and access relevant information without unnecessary clutter.









GUI Layout Description:

1. Main Window: System Dashboard

- Header:

- TextBlock: Displays the system's name and serves as a prominent title for the application.

- Purpose: Provides a clear and engaging header to immediately convey the application's purpose.

- User Role Selection:

- Buttons: `Lecturer`, `Coordinator`, and `Academic Manager` buttons are used for role selection.

- Purpose: Allows users to select their role directly, making the navigation intuitive and straightforward. Buttons are visually distinct and are designed to be clearly labeled.

- Navigation Menu:

- Buttons:

- `Submit Claim`: Leads to the claim submission form for lecturers.

- `Upload Documents`: Directs to the document upload form for lecturers.

- `Track Claim Status`: Opens a view where users can track the status of their claims.

- `Verify Claims`: This button is conditionally visible based on the user role, specifically for Coordinators and Academic Managers to verify and approve claims.

- Purpose: Provides easy access to core functionalities based on user roles. The visibility of buttons is managed dynamically to ensure users only see relevant options.

- Dashboard Overview:

- GroupBox with TextBlocks: Displays summary statistics like Pending Claims, Approved Claims, and Total Claims This Month.

- \*\*Purpose\*\*: Offers a quick overview of the system's current status, helping users stay informed about the overall claim metrics.

2. Claim Submission Form (Lecturers)

- Claim Details:

- TextBox:

- Claim ID: Auto-generated and displayed as read-only.

- Lecturer ID: Either auto-filled or selected from a dropdown.

- Submission Date: Auto-filled to reflect the current date.

- Claim Description: A text box for users to describe the claim.

- Purpose: Collects necessary details for claim submission in a structured manner.

- Submit Button:

- Button: `Submit Claim`

- Purpose: Triggers the claim submission process. Its action is clearly indicated by its label.

- Status Message:

- TextBlock: Displays a confirmation message once the claim is successfully submitted.

- Purpose: Provides immediate feedback to users, enhancing the user experience and ensuring clarity regarding the status of their submission.

3. Verify and Approve Claims Form (Coordinators/Academic Managers)

- Claim List:

- ListView with GridView:

- Columns: Claim ID, Lecturer, Date, and Status.

- Purpose: Presents a tabular view of claims, allowing users to easily browse through and select claims for detailed review.

- Claim Details:

- TextBlocks: Displays details such as Lecturer, Date Submitted, Description, and Status History.

- Purpose: Provides a detailed view of the selected claim, including historical data, which is crucial for the verification and approval process.

- Action Buttons:

- Buttons: `Approve`, `Reject`, `Assign to Academic Manager`

- Purpose: Facilitates actions on selected claims, enabling Coordinators and Academic Managers to manage claims effectively.

4. Upload Supporting Documents Form (Lecturers)

- Claim ID:

- TextBox: Displaying the claim ID to which documents will be attached.

- Purpose: Ensures that documents are associated with the correct claim.

- File Selection:

- Button: `Browse...`

- TextBlock\*\*: Shows the selected file path.

- \*\*Purpose\*\*: Allows users to select and view the file they intend to upload.

- Upload Button:

- Button: `Upload Document`

- Purpose: Executes the document upload action, providing a clear call-to-action for users.

- Status Message:

- TextBlock: Displays feedback on the upload status.

- Purpose: Gives users confirmation about the upload process and success.

Track Claim Status Form (All Users)

- Claim List:

- ListView with GridView:

- Columns: Claim ID, Lecturer, Date, and Status.

- Purpose: Provides a comprehensive view of claims with their current status, enabling users to track and manage claims easily.

- Claim Details:

- TextBlocks: Displays detailed information about the selected claim, including status history.

-Purpose: Offers transparency into the claim’s lifecycle, including who has handled it and when.

### **Project planning**

Objective: To design and prototype a Contract Monthly Claim System with a GUI that allows lecturers to submit claims, academic managers to approve them, and supports document uploads.

-Technologies: .NET Core for GUI, Windows Presentation Forms or MVC, UML for database structure, GitHub for version control.

### **Planning & Requirement Gathering (5 Days)**

* **Duration**: 4 days
* **Tasks**:
  + Kickoff meeting, resource allocation, requirement gathering, and final approval of requirements.
  + Goal: Ensure clear project understanding, gather system requirements from stakeholders, and allocate resources.

#### **2. Design (9 Days)**

* **Duration**: 9 days
* **Tasks**:
  + Create the database design (ERD, UML Class Diagram), UI wireframes, and back-end system architecture.
  + Goal: Establish the core design and architecture for the database, front-end UI, and back-end logic.

#### **3. Development (10 Days)**

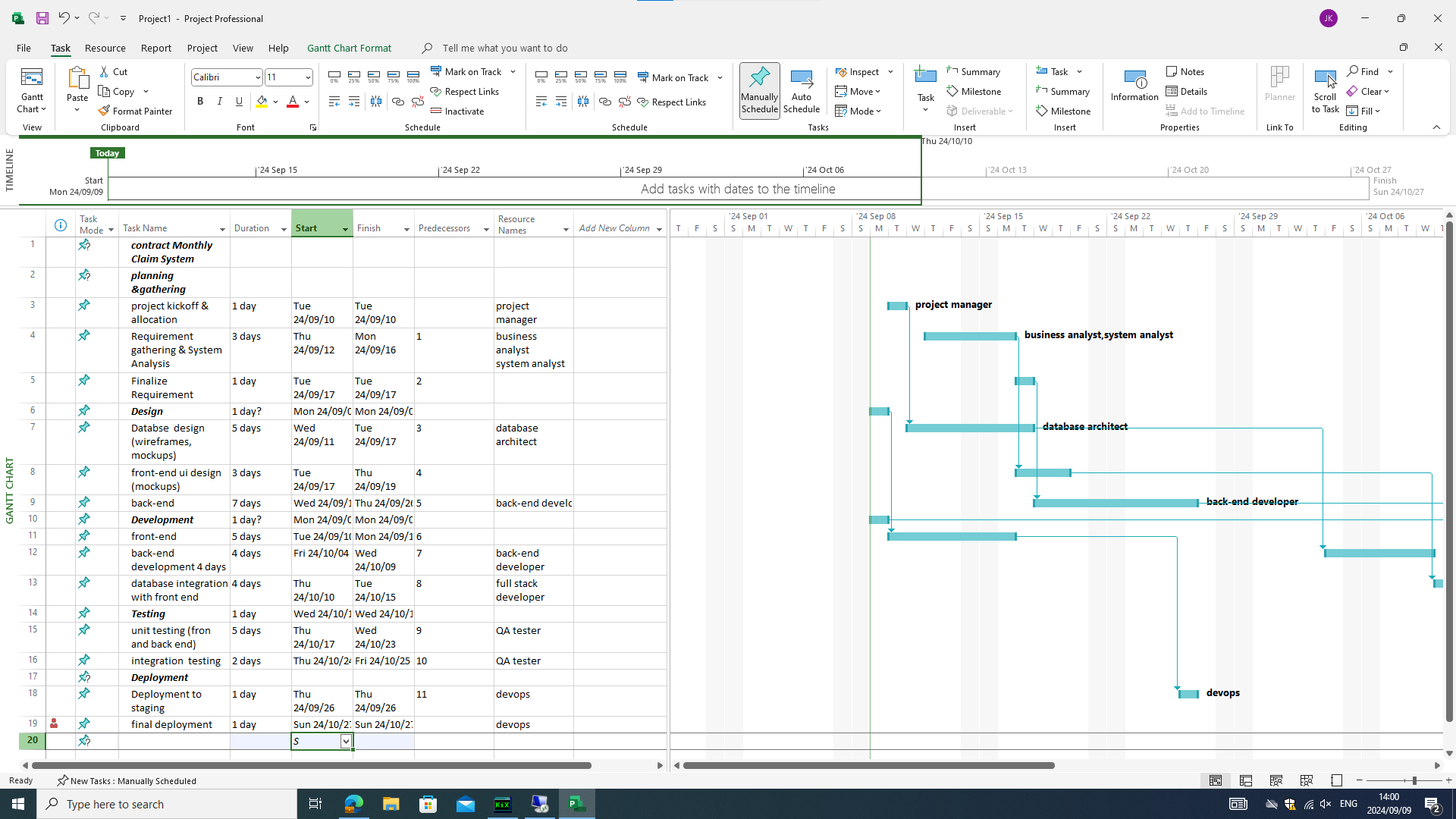
* **Duration**: 10 days
* **Tasks**:
  + Front-end and back-end development, followed by integration with the database.
  + Goal: Develop the front-end user interface, build the back-end functionality, and connect the UI with the database.

#### **4. Testing (5 Days)**

* **Duration**: 5 days
* **Tasks**:
  + Perform unit testing and integration testing to ensure system functionality.
  + Goal: Ensure the developed system is bug-free and functions correctly as a whole.

#### **5. Deployment (2 Days)**

* **Duration**: 2 days
* **Tasks**:
  + Deploy the system to a staging environment for final testing, then deploy it to the production environment.
  + Goal: Move the system to production, making it live and operational for users.



|  |
| --- |
| Constraints |

This prototype is limited to a local environment and doesn't include real-time syncing or cloud-based features.

The system will only be a front-end prototype, not functional beyond the mock interactions defined in the interface.

|  |
| --- |
| Version Control |

The project is managed using Git for version control. You can find the source code and commit history at the following GitHub repository: [Dimercia29/poe1 (github.com)](https://github.com/Dimercia29/poe1)

|  |
| --- |
| References |

Microsoft (2023) *WPF overview*. Available at: <https://learn.microsoft.com/en-us/dotnet/desktop/wpf/overview?view=netdesktop-7.0> (Accessed: 9 September 2024).