Contract Monthly Claim System – Prototype Project Documentation

1. Documentation

* Introduction

The Contract Monthly Claim System is a non-functional prototype developed using ASP.NET Core MVC. The purpose of this prototype is to demonstrate the layout, structure, and interface of a claim management system for Coordinators and Lecturers before the system is made fully functional. The prototype provides navigation, sample dashboards, claim tables, and approval/rejection pages without database integration or backend logic.

* Architecture Choice: MVC Pattern

I used the Model-View-Controller (MVC) pattern with ASP.NET Core for several reasons:

* Separation of Concerns: MVC separates the application logic, user interface, and data processing into distinct units.
* Maintainability: Any module can be modified separately without affecting others.
* Scalability: Easy to scale with new features and controllers as it grows.
* Industry Standard: MVC is widely used as a standard web application standard and is easily understood by future developers.
* Technology Stack
* **ASP.NET Core MVC**: For web application framework
* **HTML/CSS**: For frontend presentation
* **Font Awesome**: For consistent iconography
* **Bootstrap**: For responsive design components
* Database Structure Design
* User Management: Lecturers and Coordinators with different roles and permissions.
* Claim Processing: Submission of claims by lecturers for approval.
* Document Management: Handling of supporting documents linked to claims.
* Status Tracking: Monitoring claim statuses such as Pending, Approved, and Rejected.
* GUI Layout Design
* Fixed Sidebar Navigation: Provides consistent access throughout the application
* Role-based Dashboards: Separate interfaces for Lecturers and Coordinators.
* Summary Cards: Visual representations of claim statistics for quick insights.
* Data Tables: Clear presentation of detailed claim information.
* Personalized Touch: Included my profile picture on the navigation bar for better user identification.

Assumptions and Constraints

* Assumptions Made:

One Institution: Only one educational institution is served by the system.  
Monthly Claims: Every month, claims are filed.  
Lecturers have the ability to upload supplementary documentation.  
Two roles for users: The system is only used by lecturers and coordinators.  
Web-based Access: Web browsers are used by users to access the system.  
Authentication is necessary. To access the system, each user needs to log in.

* Technical Constraints:

Absence of Database Implementation The current prototype only uses static data. Authentication: Simple login using modals without backend validation  
Absence of File Storage The ability to upload documents is UI-only.  
Static Data: For demonstration purposes, all shown data is hardcoded.

* Business Constraints:

Permission Process: Coordinator permission is needed for claims.  
Status Monitoring: Throughout their existence, claims must have a clear status.  
Roles of Users: Different permissions for coordinators and lecturers

1. UML Class Diagram for Database

A diagram of a computer

AI-generated content may be incorrect.

**Relationships Shown:**

* **Users → Claim** (1:\*) - Lecturers submit multiple claims
* **Claim → Document** (1:\*) - Claims can have multiple supporting documents
* **Claim → ClaimApprovals** (1:1) - Each claim gets one approval decision
* **Users → ClaimApprovals** (1:\*) - Coordinators can approve multiple claims

Diagram Legend

* User Class: Stores system user information with authentication details
* Claim Class: Monthly claims with status tracking and financial data
* ClaimApproval Class: Approval records with decisions and comments
* Document Class: Supporting files uploaded for claims
* Relationships show cardinality (e.g., "1" to "\*" indicates one-to-many relationship)

1. Project Plan

Project Timeline and Tasks

**PHASE 1: REQUIREMENTS AND DESIGN (Week 1)**

**Duration**: 5 days

**Tasks:**

* Requirements gathering and analysis (1 day)
* Database design and ERD creation (2 days)
* UI/UX wireframe design (1 day)
* Technical architecture planning (1 day)

**Deliverables:**

* Requirements document
* Database schema
* UI mockups
* Architecture diagram

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task Name** | **Duration** | **Start** | **Predecessors** | **Resource Names** |
| **1. Requirements and Design Phase** | 5 days | Day 1 | - | - |
| 1.1 Requirements Gathering and Analysis | 1 day | Day 1 | - | Business Analyst |
| 1.2 Database Design and ERD Creation | 2 days | Day 2 | 1.1 | Database Designer |
| 1.3 UI/UX Wireframe Design | 1 day | Day 4 | 1.1 | UI/UX Designer |
| 1.4 Technical Architecture Planning | 1 day | Day 5 | 1.2,1.3 | System Architect |
| **Milestone: Design Complete** | 0 days | Day 5 | 1.4 | - |

**PHASE 2: DATABASE IMPLEMENTATION (Week 2)**

**Duration**: 5 days

**Tasks:**

* Database creation and table setup (2 days)
* Stored procedures development (1 day)
* Database testing and validation (1 day)
* Sample data insertion (1 day)

**Deliverables:**

* Working database
* Test data
* Database documentation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task Name** | **Duration** | **Start** | **Predecessors** | **Resource Names** |
| **2. Database Implementation Phase** | 5 days | Day 6 | 1.4 | - |
| 2.1 Database Creation and Table Setup | 2 days | Day 6 | 1.4 | Database Developer |
| 2.2 Stored Procedures Development | 1 day | Day 8 | 2.1 | Database Developer |
| 2.3 Database Testing and Validation | 1 day | Day 9 | 2.2 | QA Tester |
| 2.4 Sample Data Insertion | 1 day | Day 10 | 2.3 | Database Developer |
| **Milestone: Database Ready** | 0 days | Day 10 | 2.4 | - |

**PHASE 3: BACKEND DEVELOPMENT (Week 3-4)**

**Duration**: 10 days

**Tasks:**

* User authentication system (3 days)
* Claim management controllers (3 days)
* File upload functionality (2 days)
* Email notification system (2 days)

**Dependencies:**

* Requires completed database schema
* UI mockups for reference

**Deliverables:**

* Authentication system
* Core business logic
* API endpoints

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task Name** | **Duration** | **Start** | **Predecessors** | **Resource Names** |
| **3. Backend Development Phase** | 10 days | Day 11 | 2.4 | - |
| 3.1 User Authentication System | 3 days | Day 11 | 2.4 | Backend Developer |
| 3.2 Claim Management Controllers | 3 days | Day 14 | 3.1 | Backend Developer |
| 3.3 File Upload Functionality | 2 days | Day 17 | 3.2 | Backend Developer |
| 3.4 Email Notification System | 2 days | Day 19 | 3.2 | Backend Developer |
| **Milestone: Backend Complete** | 0 days | Day 20 | 3.3,3.4 | - |

**PHASE 4: FRONTEND DEVELOPMENT (Week 5-6)**

**Duration**: 10 days

**Tasks:**

* Lecturer dashboard implementation (3 days)
* Coordinator dashboard implementation (3 days)
* Claim submission forms (2 days)
* Document upload interface (2 days)

**Dependencies:**

* Backend APIs must be complete
* Database integration required

**Deliverables:**

* Complete user interfaces
* Responsive design
* Form validation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task Name** | **Duration** | **Start** | **Predecessors** | **Resource Names** |
| **4. Frontend Development Phase** | 10 days | Day 21 | 3.3,3.4 | - |
| 4.1 Lecturer Dashboard Implementation | 3 days | Day 21 | 3.3,3.4 | Frontend Developer |
| 4.2 Coordinator Dashboard Implementation | 3 days | Day 24 | 4.1 | Frontend Developer |
| 4.3 Claim Submission Forms | 2 days | Day 27 | 4.2 | Frontend Developer |
| 4.4 Document Upload Interface | 2 days | Day 29 | 4.3 | Frontend Developer |
| **Milestone: UI Complete** | 0 days | Day 30 | 4.4 | - |

**PHASE 5: INTEGRATION AND TESTING (Week 7)**

**Duration**: 5 days

**Tasks:**

* System integration testing (2 days)
* User acceptance testing (2 days)
* Bug fixes and refinements (1 day)

**Dependencies:**

* All frontend and backend components complete

**Deliverables:**

* Tested system
* Bug report and fixes
* User feedback incorporation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task Name** | **Duration** | **Start** | **Predecessors** | **Resource Names** |
| **5. Integration and Testing Phase** | 5 days | Day 31 | 4.4 | - |
| 5.1 System Integration Testing | 2 days | Day 31 | 4.4 | QA Tester |
| 5.2 User Acceptance Testing | 2 days | Day 33 | 5.1 | End Users, QA Tester |
| 5.3 Bug Fixes and Refinements | 1 day | Day 35 | 5.2 | Full Stack Developer |
| **Milestone: Testing Complete** | 0 days | Day 35 | 5.3 | - |

**PHASE 6: DEPLOYMENT AND DOCUMENTATION (Week 8)**

**Duration**: 5 days

**Tasks:**

* Production environment setup (1 day)
* System deployment (1 day)
* User training materials (2 days)
* Final documentation (1 day)

**Dependencies:**

* Tested and approved system

**Deliverables:**

* Live system
* User manuals
* Technical documentation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task Name** | **Duration** | **Start** | **Predecessors** | **Resource Names** |
| **6. Deployment and Documentation** | 5 days | Day 36 | 5.3 | - |
| 6.1 Production Environment Setup | 1 day | Day 36 | 5.3 | DevOps Engineer |
| 6.2 System Deployment | 1 day | Day 37 | 6.1 | DevOps Engineer |
| 6.3 User Training Materials | 2 days | Day 38 | 6.2 | Technical Writer |
| 6.4 Final Documentation | 1 day | Day 40 | 6.3 | Technical Writer |
| **Milestone: Project Complete** | 0 days | Day 40 | 6.4 | - |

Resources Requirements:

* 1 Full-stack Developer
* 1 Database Administrator
* 1 UI/UX Designer (part-time)
* Development environment and tools
* Testing environment
* Production hosting

Risk Mitigation:

* **Technical Risks**: Regular code reviews and testing
* **Timeline Risks**: Buffer time built into each phase
* **Resource Risks**: Cross-training team members
* **Requirement Changes**: Change control process

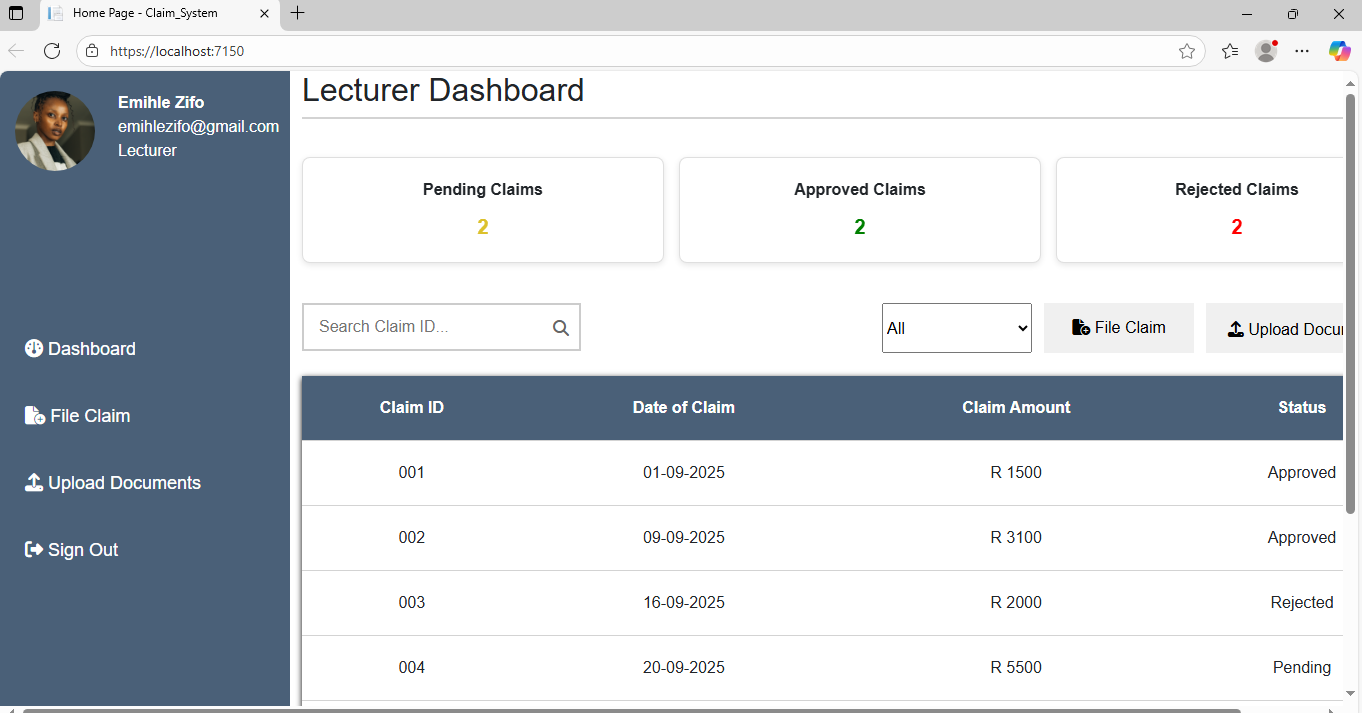
Success Metrics:

* All user stories completed
* System passes all test cases
* User acceptance criteria met
* Documentation complete

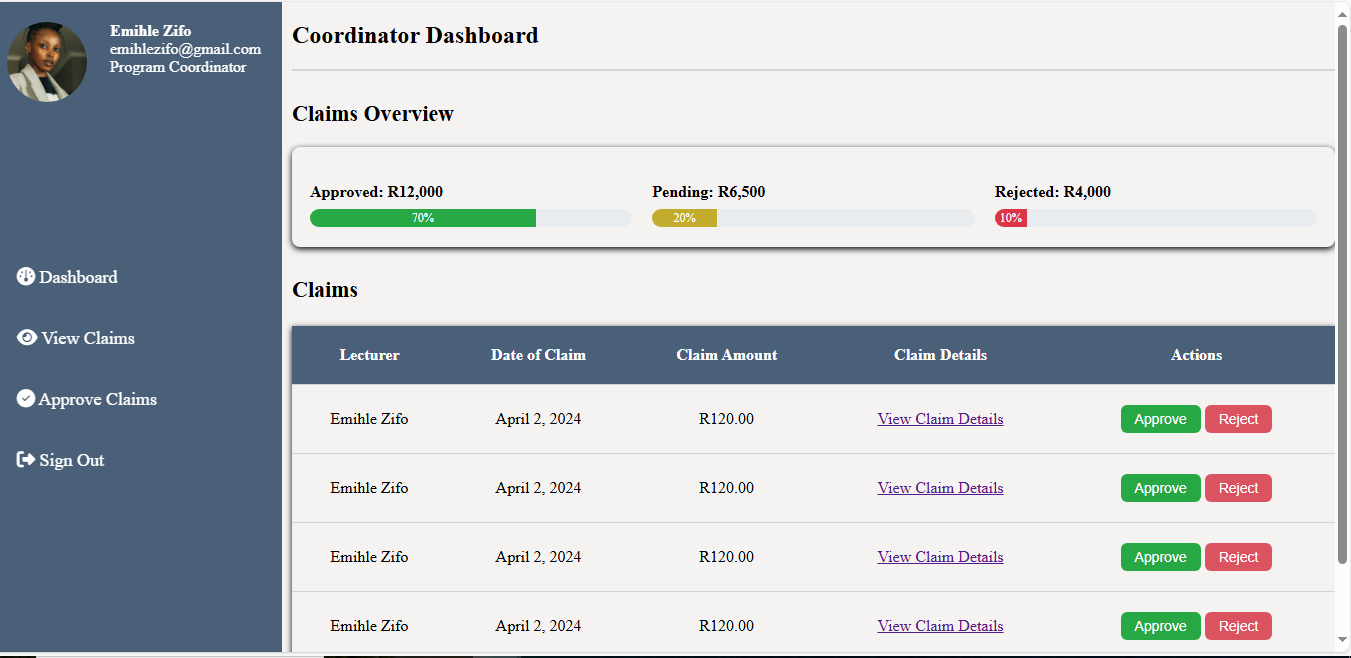
On-time delivery within budget

I have attached an mpp document that contains the same project plan information.

1. This is my Lecture Dashboard



This is my Coordinator Dashboard



1. Github Commits

