**A logo of a company

Description automatically generated**

Ryan Millard

ST10383326

## Advanced Databases

ADDB7311

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

NWEG5122

Assignment 1

5 September 2024

Contents

[Documentation: Design Choices and Structure 3](#_Toc176718801)

[Overall System Architecture 3](#_Toc176718802)

[Database Design 4](#_Toc176718803)

[GUI Layout 7](#_Toc176718804)

[Security Considerations 8](#_Toc176718805)

[Extensibility and Maintainability 9](#_Toc176718806)

[Assumptions and Constraints 10](#_Toc176718807)

# Design Choices and Structure

**The system follows a layered architecture:**

* **Presentation Layer:** Razor Pages, Bootstrap
  + Excellent performance, scalability, and reusability.
    - Through partials, SSR, and other Razor features
  + Cleaner architecture and concern separation
  + Simplifies maintenance
  + **Layout:**
    - Top navigation panel
    - Pages
      * Home
      * Dashboard
        + Lecturers:

Submit claim form

List of submitted claims table

* + - * + Managers/Program Co-ordinators:

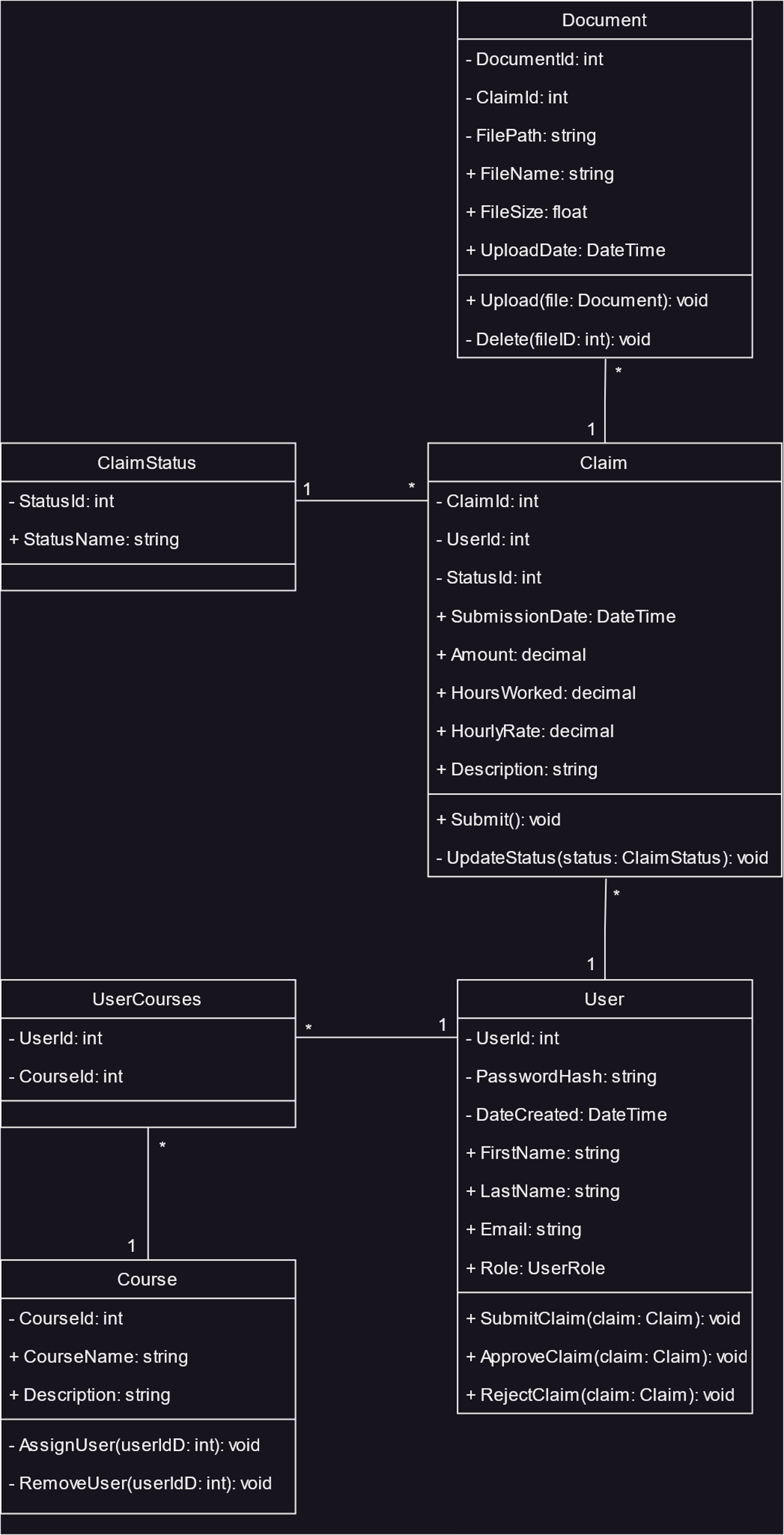
List of pending/approved claims

* + - * Login
    - Simple footer
* **Business Logic Layer:** Services and domain models
  + Claim processing, etc.
  + Dependency injection decouples logic and UI
* **Data Access Layer:** Entity Framework Core
  + Safe against attacks
  + Reusable queries
* **Database:** SQL Server
  + Integrates well with ASP.NET and Entity Framework
  + **Entities:**
    - **Users:** Information about **lecturers** (independent contractors), **coordinators** (programme co-ordinators), and **managers** (academic managers).
    - **Claims:** Claim details submitted by lecturers.
    - **ClaimStatus:** Approval status of each claim.
    - **Documents:** Metadata about uploaded supporting documents.
    - **Courses:** Courses lecturers teach.
  + **Relationships:**
    - One-to-many between **Users** and **Claims**
    - One-to-many between **Claims** and **Documents**
    - Many-to-many between **Users** and **Courses**
  + **Key design decisions:**
    - Separate **ClaimStatus** table allows for easy future extensions of status types.
    - Storing document metadata separately from **Claims** allows for multiple documents per claim.
    - A join table for **Users** and **Courses**, named **UserCourses**, enables tracking which courses each lecturer teaches.

# Assumptions and Constraints

* **User volume:** Up to 1000 active users, primarily lecturers, is assumed.
* **Claim frequency:** Most claims are expected to be submitted monthly.
* **Document storage:** 2-3 documents per claim assumed, with a max file size of 10MB each.
* **Approval workflow:** Two-step approval process (coordinator then manager).
* **Browser support:** Modern web browsers (Chrome, Firefox, Safari, Edge).

## Database Design



# Project Plan

**Phases and Milestones**

* **Planning (Week 1)**:
  + Define project scope and specifications.
  + Finalize design choices and database structure.
* **Design & Prototyping (Week 2-3)**:
  + Create mock-ups for key views (Dashboard, Claim Form, Approvals).
  + Develop initial Razor Page layouts using Bootstrap.
* **Development (Week 4-8)**:
  + Set up database schema with Entity Framework Core.
  + Implement role-based Razor Pages (Lecturer, Coordinator, Manager).
  + Implement Claim submission and approval workflows.
* **Testing & Debugging (Week 9-10)**:
  + Unit testing for business logic.
  + Integration testing for Razor Pages and database interactions.
  + User Acceptance Testing (UAT) for role-based functionality.
* **Finalization & Submission (Week 11)**:
  + Final bug fixes.
  + Documentation.
  + Deployment (local or cloud).

**Dependencies**

* **Tech Stack**:
  + SQL Server, ASP.NET Core, Entity Framework Core, Razor Pages, Bootstrap.
* **Dependencies**:
  + Ensure environment setup (SQL Server, ASP.NET Core).
  + Database migration tools (EF Core Migrations).
  + Browser testing (modern browsers).

**Tasks**:

* **Week 1**: Define detailed scope and database schema (ERD creation).
* **Week 2**: Build mock-ups.
* **Week 3**: Bootstrap implementation for page layouts (home, dashboard, claims).
* **Week 4-6**: Razor Pages implementation with role-based views.
* **Week 7-8**: Backend services for claim processing, document handling.
* **Week 9-10**: Test all functionalities, fix bugs.
* **Week 11**: Final submission preparations.

## GUI Layout

* **Role-based views**
  + **Lecturer View:**
    - Dashboard showing claim history and status.
    - Claim submission form with clear input validation.
    - Document upload area with drag-and-drop functionality.
  + **Coordinator/Manager View:**
    - Overview of pending claims requiring approval.
    - Detailed view of individual claims with approve/reject options.
    - Reporting tools for analysing claim patterns.
* **Consistent navigation:**
  + Minimal top navigation bar.
* **Clean, modern and consistent aesthetic:**
  + [Bootstrap](https://getbootstrap.com/) for consistent styling and layout.
  + Razor Pages for reusable code.