**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)

# Documentation: Design Choices and Structure

## Overall System Architecture

* **Scalability, performance:** ASP.NET Core provides excellent performance scalability.
* **Compatibility:** Compatible with any server and any device regardless of their OSs.
* **Modern development practices:**
  + ASP.NET Core encourages clean architecture and concern separation.
  + Razor pages increase code reusability and simplifies maintenance.
* **Rich ecosystem:** .NET ecosystem has powerful libraries and tools.

**The system follows a layered architecture:**

* **Presentation Layer:** Razor Pages
* **Business Logic Layer:** Services and domain models
* **Data Access Layer:** Entity Framework Core
* **Database:** SQL Server

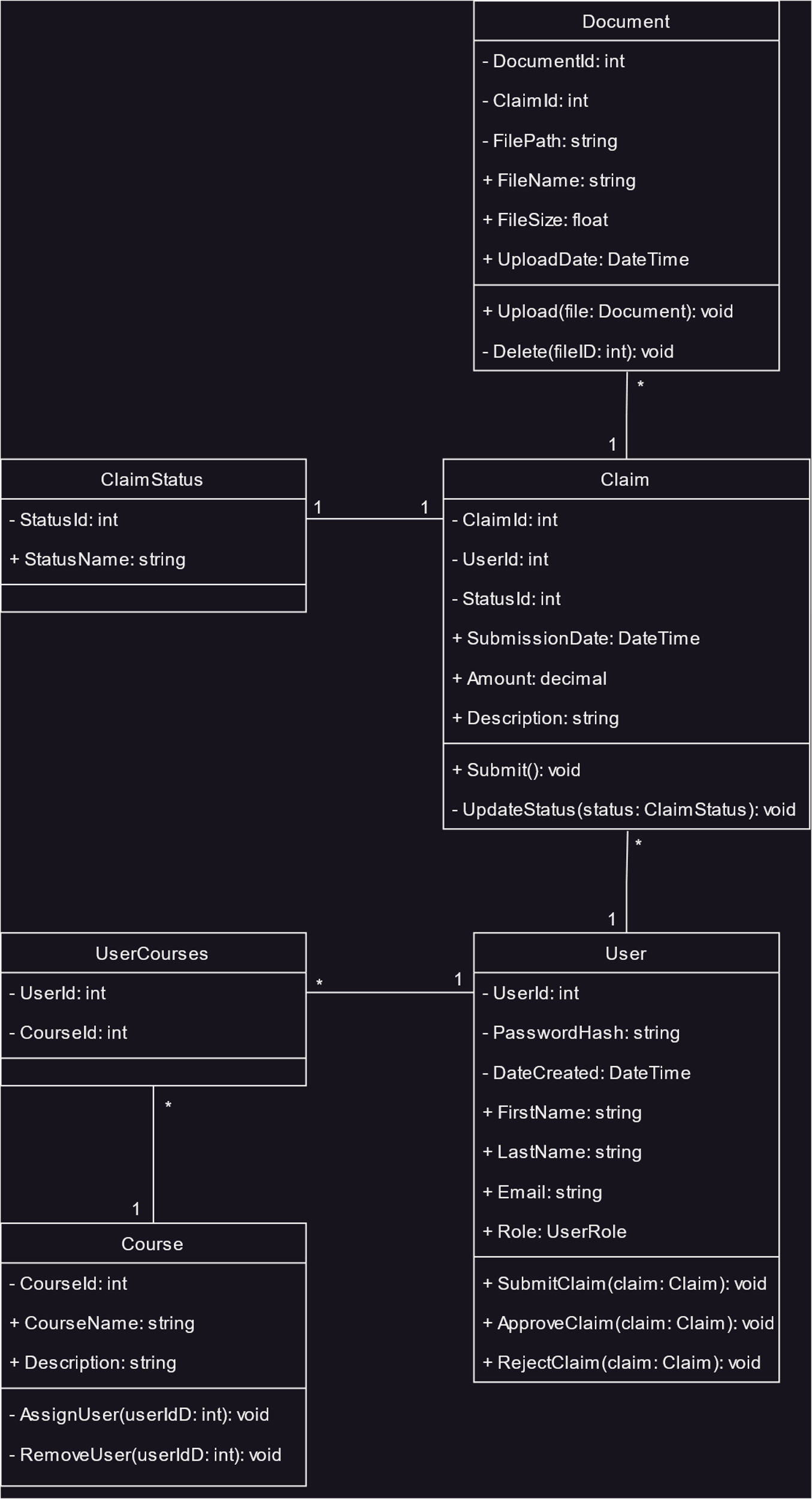
## Database Design

**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**
  + Each **User** can have 0 to an unlimited number of **Claims**.
  + Each **Claim** can only be linked to a single **User** (the one who submitted it).
* One-to-many between **Claims** and **Documents**
  + Each **Claim** can have 0 to 5 **Documents**.
  + Each **Document** can only be linked to one **Claim**.
* Many-to-many between **Users** and **Courses**
  + Each **User** (with the lecturer role) can teach 1 to an unspecified number of **Courses**.
  + Each **Course** can be taught by one to an unlimited number of **Users** (with the lecturer role).



**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.

## 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.

# Security Considerations

* **Authentication**: ASP.NET Core Identity for secure user authentication.
* **Authorization**: Role-based access control restricts features based on user roles.
* **Data** **protection**: Sensitive data is encrypted at rest and in transit.
* **Input** **validation**: Server-side validation prevents injection attacks.

# Extensibility and Maintainability

* **Modular architecture:** New modules can be added with minimal impact on existing code.
* **Use of interfaces:** Key components are defined by interfaces, allowing for easy substitution or mocking in tests.
* **Configuration-driven:** Many aspects of the system can be adjusted through configuration files rather than code changes.

# Assumptions and Constraints

* **User volume:** The initial design assumes up to 1000 active users, primarily lecturers.
* **Claim frequency:** Most claims are expected to be submitted monthly.
* **Document storage:** Assuming an average of 2-3 documents per claim, with a max file size of 10MB each.
* **Approval workflow:** Two-step approval process (coordinator then manager) is sufficient for most use cases.
* **Browser support:** Targeting modern web browsers (Chrome, Firefox, Safari, Edge).