Jade Heeger

ST10445517

PROG6212

POE Part 1

Table of Contents

[Table Of Contents 1](#_Toc209037661)

[Documentation 2](#_Toc209037662)

[UML Class Diagram 3](#_Toc209037663)

[Project Plan 4](#_Toc209037664)

[GUI/UI 5](#_Toc209037665)

# Documentation



Figure \_ System logo designed with Canva

#### Overview

NHEXIT is a web-based Contract Monthly Claim System (CMCS) that allows Independent Contractor Lecturers to submit and keep track of claims for the hours they’ve worked and allows administrators to review and approve those claims. The goal is to streamline manual claim processes, improve data accuracy and allow both parties to access claim history securely and efficiently.

#### User Interface

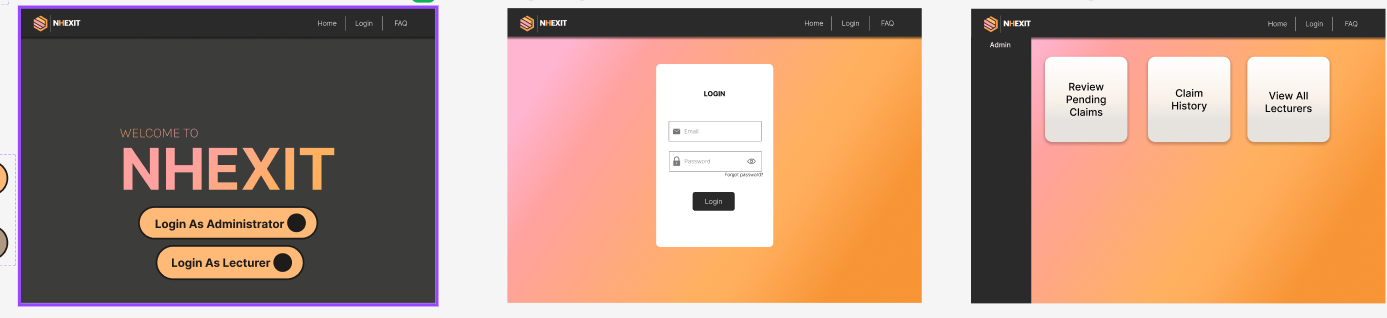


Figure \_ GUI Design created with Figma

Four main screens will be designed in Figma to illustrate the system flow:

* Home Screen: A home page with the option of logging in as an Administrator or Lecturer.
* Login Page: A login form for entering email and password, with the option of clicking forget password.
* Lecturer Main Page: After login, lecturers can create a new claim, view their claim history and update their user profiles.
* Admin Main Page: Administrators can review pending claims, view all lecturers’ details and access claim history for auditing.

All the pages will be consistent with a gradient background, top navigation bar, and side navigation bar to improve usability and visibility. The color scheme for this UI is pink, salmon and orange. Blue, black, orange and white buttons will be designed with effects for contrast to create a consistent and appealing look and feel. All buttons will have hover effects to indicate to the user they are clickable.

#### Database Design

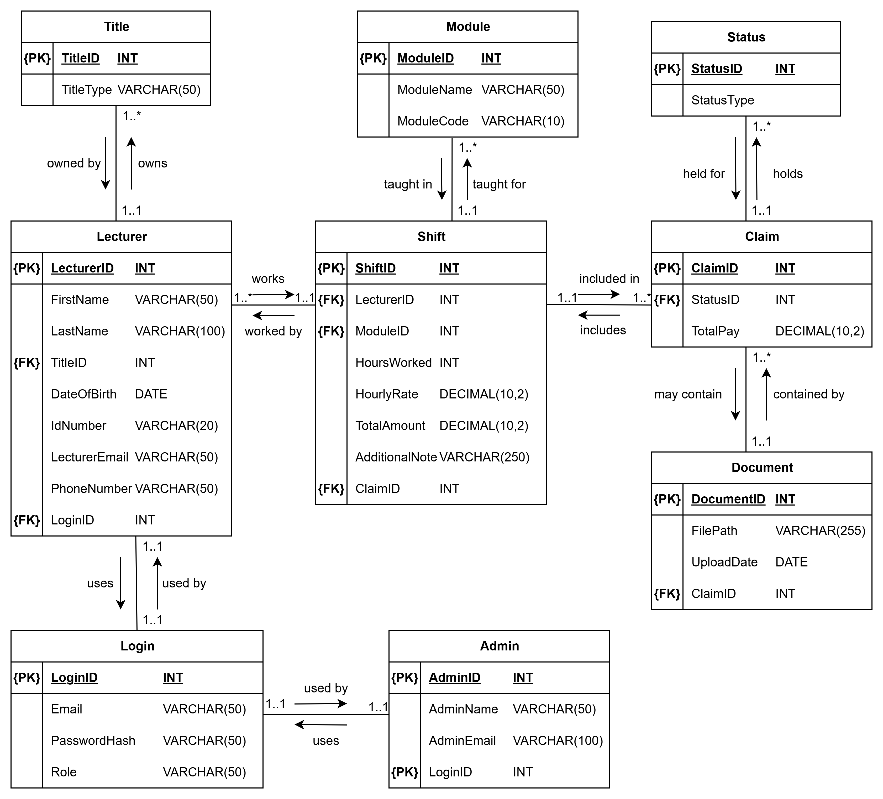


Figure \_ UML ERD designed with draw.io

The database is normalized into nine entities (Lecturer, Title, Login, Module, Shift, Claim, Status, Claim, Status, Document, Admin), as shown in the UML diagram above.

* Lecturer stores personal and contact information linked to a title table to reduce duplication.
* Login holds credentials and roles; it links to Lecturer pr Admin for access control.
* Module lists the modules actively being taught.
* Shift records hours and rate per module for a lecturer and each shift links to a Claim.
* Claim is the claim header that contains the claim date, total pay and foreign key to Status.
* Status stores the states in which the claims stand (Pending, Approved, Rejected).
* Document stores multiple supporting file paths per claim.
* Admin represents administrative users who approve the claims.

This schema was designed to ensure that it supports multiple shifts per claim, no repetitive data and easy data retrieval methods later in the development process. Monetary fields use decimal data type for more accuracy and file paths are limited to 255 characters.

#### Assumptions and Constraints

* Each lecturer has one login but can submit multiple claims
* A claim can contain multiple shifts and multiple supporting documents
* Only administrators can approve or change the claim status
* Files are stored by path; actual files will be secured in an allocated space

#### Implementation Plan

*Phase 1*: Design database schema

*Phase 2*: Design lecturer UI (edit profile, create and view claims)

*Phase 3*: Design admin UI (review and approve claims)

*Phase 4* Implement designs, test and deploy

# UML Class Diagram

# A diagram of a computer program AI-generated content may be incorrect.

# Project Plan

# Contract Monthly Claims System (CMCS)

#### Project Overview

* Project Goal: To develop a .NET web-based application to streamline the monthly claims process for independent contractor lecturers.
* Scope: The project includes the design, development and implementation of a web application with specific user interfaces for lecturers, programme coordinators, academic managers and HR.

#### Deliverables

* Part 1: Detailed documentation, UML Class Diagram, Project Plan, and non-functional GUI mockup prototype.
* Part 2: Functional web application with core features (claim submission, approval, document upload, status tracking and unit testing).
* Part 3: Enhanced application with automated features and a final PowerPoint presentation.

#### Assumptions

* All necessary software tools (.NET Core, Visual Studio, Git) are available to the developer.
* User requirements stated in the POE document are complete and final.

#### Constraints

* The project must be completed withing the allocated timeline.
* The final submission requires a GitHub repository and a PowerPoint presentation.

#### Project Milestones and timeline

Phase 1: Planning and Mockups (Week 1-2)

* Week 1 (3-10 September):
  + Task 1.1: Requirements analysis and review
  + Task 1.2: Database design and creation of the UML Class Diagram.
  + Task 1.3: Development of the project plan documentation
* Week 2 (10-17 September):
  + Task 1.4: GUI/UI design and creation of a non-functional prototype using a wireframing tool
  + Task 1.5: Finalize documentation and commit to GitHub

Phase 2: Implementation- (Week 3-5)

* Week 3 (22-29 September):
  + Task 2.1: Set up ASP.NET Core MVC project and basic file structure
  + Task 2.2: Implement lecturer claim submission form and basic data model
* Week 4 (29 Sept-6 October):
  + Task 2.3: Implement a view for coordinators/managers to see pending claims.
  + Task 2.4: Implement document upload functionality
  + Task 2.5: Commit changes to GitHub.
* Week 5 (6-13 October):
  + Task 2.6: Implement claim status tracking
  + Task 2.7: Develop and run unit tests for functionalities
  + Task 2.8: Implement error handling and data validation
  + Task 2.9: Commit final changes for this part to GitHub

Phase 3: Automation and Presentation - (Week 6-8)

* Week 6 (14-21 October):
  + Task 3.1: Implement auto-calculation of total pay in the lecturer's view using JavaScript
  + Task 3.2: Implement automated validation checks for submitted claims
* Week 7 (21 Sept-28 October):
  + Task 3.3: Develop automated approval workflows for coordinators and managers
  + Task 3.4: Create a dedicated view for HR with data management capabilities.
* Week 8 (28 October- 6 November):
  + Task 3.5: Prepare and finalize the PowerPoint presentation
  + Task 3.6: Final review of the entire project and documentation
  + Task 3.7: Final push of all source code and documentation to the GitHub repository

#### Risk Management

* Technical Challenges: The complexity of implementing certain features (e.g., file uploads, real-time status updates) may cause delays.
  + Mitigation: Conduct thorough research on a feature before beginning implementation. Utilize online documentation for troubleshooting.
* Time Management: Falling behind schedule on a specific task could impact the entire project timeline.
  + Mitigation: Break down tasks into smaller, more manageable sub-tasks. Regularly review progress against the timeline and adjust accordingly. Prioritize core functionalities over non-essential enhancements.
* Version Control Issues: Incorrectly using Git could lead to lost work or conflicts.
  + Mitigation: Regularly commit changes with clear and descriptive messages. Back up the project locally before major changes.

#### Communication

Communication will mostly be between the student and the lecturer. The GitHub commit messages will serve as a continuous progress log. Any major issues will be communicated via email or during scheduled lectures.

# GUI/UI

A screenshot of a computer

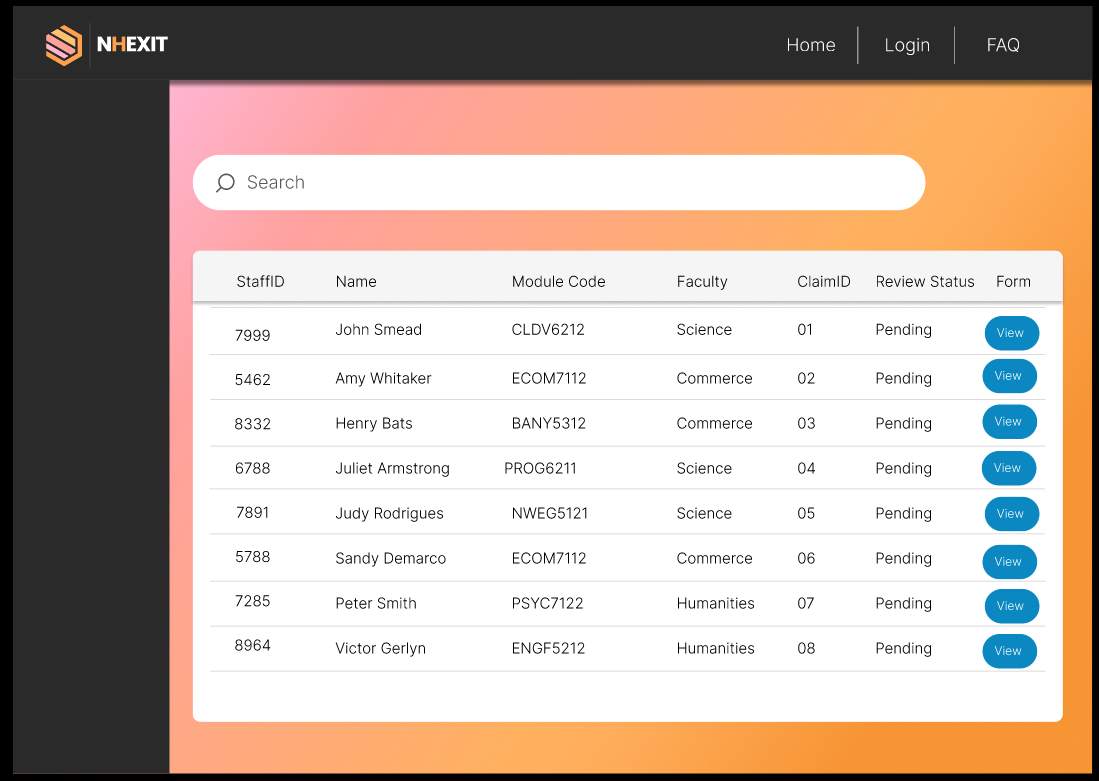
AI-generated content may be incorrect.

A screenshot of a login screen

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.



A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.