**ST10156967**

**DANIEL COLLIN MARINUS**

**PROG6212**

**ASSIGNMENT 1**

**17 SEPTEMBER 2025**

**Table of Contents**

[**Introduction** 3](#_Toc209040268)

[**Documentation** 3](#_Toc209040269)

[**UML Class Diagram** 4](#_Toc209040270)

[**Figure 1** 4](#_Toc209040271)

[**Project Plan** 5](#_Toc209040272)

[**GUI / UI Design** 6](#_Toc209040273)

[**Lecturer Portal** 6](#_Toc209040274)

[**Figure 2** 6](#_Toc209040275)

[**Coordinator Portal** 6](#_Toc209040276)

[**Figure 3** 6](#_Toc209040277)

[**Dashboard** 6](#_Toc209040278)

[**Figure 4** 6](#_Toc209040279)

[**Version Control** 7](#_Toc209040280)

[**Conclusion** 7](#_Toc209040281)

# **Introduction**

The Contract Monthly Claim System (CMCS) is designed to streamline the submission and approval of monthly claims for Independent Contractor (IC) lecturers. This system simplifies the process of submitting hours worked, verifying claim details, and approving payments. Developing this system using C# and .NET Core provides an opportunity to apply object-oriented programming and GUI development principles in a real-world context. The first stage of development focuses on planning and creating a prototype that illustrates the database structure, user interface, and project plan (Troelsen & Japikse, 2021).

# **Documentation**

The design of the CMCS is based on three key requirements: accuracy, efficiency, and user-friendliness. The system must ensure claims are recorded correctly and reviewed by Programme Coordinators and Academic Managers before being approved.

The database structure is built around central entities such as Lecturer, Claim, Supporting Document, and Administrator. Each lecturer can submit multiple claims, each linked to uploaded supporting documents. Claims are verified by administrators and updated with a status (e.g., Pending, Approved, or Rejected).

The GUI layout has been designed with separate interfaces for lecturers and administrators. Lecturers will see a claim submission screen and a document upload function, while administrators will access verification and approval tools.

Assumptions include:

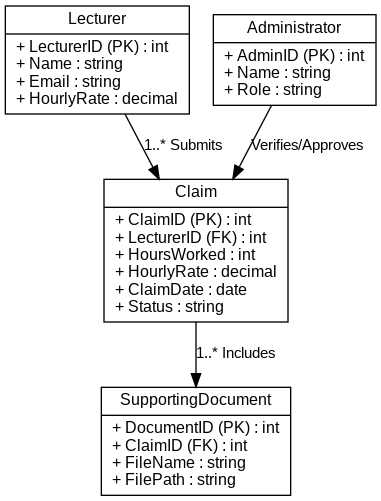
* All lecturers and staff have reliable internet access.
* Lecturers know their hourly rates and can enter hours accurately.

Constraints include:

* Limited budget and time for implementation.
* Compatibility restricted to Windows-based machines using Visual Studio 2022.

# **UML Class Diagram**

The UML diagram below represents the system’s main entities and their relationships.



**Figure 1**: UML Class Diagram showing main entities and relationships

# **Project Plan**

The project plan outlines the sequence of tasks required to deliver the prototype. The table below presents the major phases, dependencies, and estimated timeframes.

| **Task** | **Dependency** | **Duration** |
| --- | --- | --- |
| Requirements gathering | None | 3 days |
| Database design (UML) | Requirements gathering | 4 days |
| GUI mockup design | Database design | 5 days |
| Project plan documentation | Requirements gathering | 2 days |
| Integration of diagrams & report | Database & GUI design | 3 days |
| GitHub commits (5 updates) | Ongoing during all tasks | Throughout |
|  |  |  |

This plan is realistic for a student project while reflecting professional software development practices (Farrell, 2017).

# **GUI / UI Design**

The GUI prototype was created using the principles of Model-View-Controller (MVC) in .NET Core. It includes:

# **Lecturer Portal**

**Figure 2**: Lecturer screen for submitting claims with hours, rates, and supporting documents

# **Coordinator Portal**

**Figure 3**: Coordinator screen for verifying and approving/rejecting claims.

# **Dashboard**

**Figure 4**: Dashboard displaying the status of all claims (Pending, Approved, Rejected).  
The design is user-friendly and intuitive, ensuring lecturers can submit claims in a few clicks, while administrators can easily process approvals (Microsoft, 2020).

# **Version Control**

A GitHub repository will be used for version control. At least five commits will be made with clear commit messages, for example:

* “Initial UML diagram created”
* “Added Lecturer claim submission form mockup”
* “Updated GUI layout for admin dashboard”

This ensures transparency and tracks progress throughout development (Whitaker, 2016).

# **Conclusion**

This prototype planning stage lays the foundation for the Contract Monthly Claim System. By establishing a robust database structure, a clear project plan, and intuitive GUI designs, the system is positioned to improve efficiency and accountability in claim management. The use of UML and structured planning provides a professional approach that can later be extended into a fully functional application (Troelsen & Japikse, 2017).

**Bibliography**

* Farrell, J. (2017). Visual C# 2017: An Introduction to Object-Oriented Programming. 7th ed. Cengage Publishing.
* Sommerville, I. (2016). Software Engineering. 10th ed. Pearson.
* Pressman, R.S. and Maxim, B.R. (2020). Software Engineering: A Practitioner’s Approach*.* 9th ed. McGraw-Hill..
* Sharp, H. and Robinson, H. (2019). Agile Software Development: Current Research and Future Directions*.* Springer.
* Microsoft. (2020). C# programming guide. [Online] Available at: <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/> [Accessed 13 May 2021].
* Troelsen, A. and Japikse, P. (2017). Pro C# 7: With .NET and .NET Core. 8th ed. Apress.
* Troelsen, A. and Japikse, P. (2021). Pro C# 9 with .NET 5: Foundational Principles and Practices in Programming. 10th ed. Apress.
* Whitaker, R.B. (2016). The C# Player’s Guide. Starbound Software.