

PROG6212\_pOE\_PART1

ST10439052



**Report**

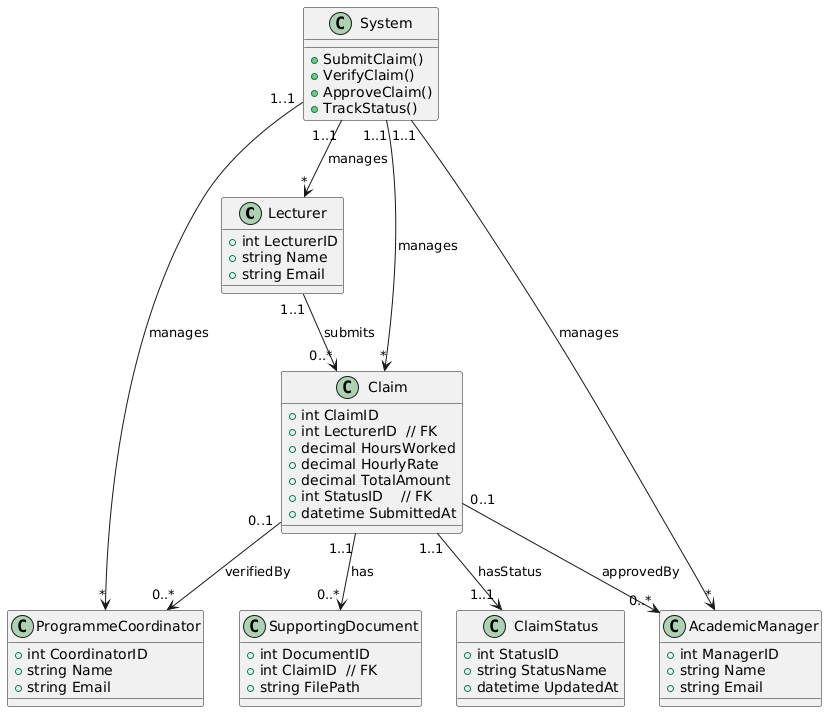
This Report presents the planning and prototype design for the Contract Monthly Claim System (CMCS). The system is a .NET web-based application designed to simplify the monthly claim submission and approval process for independent contractor lecturers. At this prototype stage the focus is on planning, designing the database model, and producing a front-end MVC prototype—functionality will be added in later parts.

**Design Choices and Assumptions**  
I selected the MSV CORE because it makes it easier to create and edit models, views, and controllers and is also good at web deployment. The GUI at this stage is a visual front-end prototype only there is no backend logic yet or data persistence is implemented yet.Main assumnption lecturers submit claims monthly, each claim must be verified by a Programme Coordinator before approval by an Academic Manager; supporting documents can be uploaded per claim.

**UML Class Diagram**  
entities: *Lecturer, Claim SupportingDocument, ClaimStatus, ProgrammeCoordinator,* and *AcademicManager*.

Relationships reflect real-world workflows:

Each lecturer may submit multiple claims (Lecturer 1..1 → Claim 0..*)  
Each claim can have multiple supporting documents (Claim 1..1 → SupportingDocument 0..*);  
Each claim holds one current status record (Claim 1..1 → ClaimStatus 1..1). Claims may be verified and approved by staff (Claim 0..1 → ProgrammeCoordinator 0..\* and Claim 0..1 → AcademicManager 0..\*). A System controller is included to represent submission, verification, approval and status-tracking operations.



**GUI Prototype**  
The prototype contains three role-based views:

* **Lecturer View:** Submit Claim button, a form for hours and rate, file upload area for supporting documents, and a Track Claim Status panel.
* **Programme Coordinator View:** pending claims list and Verify Claim action.
* **Academic Manager View:** approved/declined controls and claim history.  
  Bootstrap is used for consistent styling. All UI elements are placeholders to show how flows will work when backend logic is implemented.

**Project Plan & Version Control**

| **Task ID** | **Task Description** | **Dependencies** | **Timeline** | **Deliverable** |
| --- | --- | --- | --- | --- |
| 1 | Gather requirements and draft documentation | None | Week 1 | Draft report section (documentation) |
| 2 | Create UML class diagram | Task 1 | Week 1 | UML diagram (image + explanation) |
| 3 | Design GUI prototype screens (Lecturer, Coordinator, Manager) | Task 1 | Week 2 | GUI description (front-end only) |
| 4 | Set up GitHub repo and commit initial project structure | Task 1 | Week 2 | GitHub repo created |
| 5 | Commit GUI prototype scaffolds | Task 3, Task 4 | Week 3 | GUI prototype pushed to repo |
| 6 | Finalize project plan, UML, and report | Tasks 1–5 | Week 4 | Complete Word report (400–500 words) |
| 7 | Push final documentation to GitHub (5 commits total) | Task 6 | Week 4 | Final GitHub repo + report |

[Github Link:](https://github.com/SmitJaco/PROG6212_POE_ST10439052)

https://github.com/SmitJaco/PROG6212\_POE\_ST10439052

## Screenshots of prototype

A screen shot of a computer

AI-generated content may be incorrect.

A close-up of a white box

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.