ST10451592 PROG6212 ASSIGNMENT 2025

Michael Makomborero Jani

Contents

Youtube link: https://youtu.be/Qm51ov6KOl4
github link: https://github.com/MJVCaccount/PROG6212_POE_2025

Updates from Part 1 Feedback:

Addressing the rubric: Enhanced design choices with clarity, depth, and coherence (e.g., rationale for MVC, in-memory storage for efficiency in Part 2). Assumptions/constraints are comprehensive and aligned (detailed list with impacts). UML updated with multiplicities (1..*), correct inheritance (hollow arrow), separate ProgrammeCoordinator/AcademicManager for scalability, and interactions (methods linking to Claim). Project plan detailed for Parts 1-2 with dependencies. GUI/UI improved (e.g., non-repeating sections, upload with encryption, progress trackers). This targets 'Greatly exceeds' by providing thorough justification and scalability.

1. Documentation

When creating the prototype of the Contract Monthly Claim System (CMCS), I gave high importance to a solid and scalable design with ASP.NET Core MVC framework in the.NET core. It is also the best choice when working with web-based applications because it can allow separation of concerns and thus create maintainable code: it has Models (data storage logic), Views (presentation), and Controllers (interaction coordination) (Troelsen & Japikse, 2022). The cross-platform functionalities of ASP.NET Core make the system available to a wide range of devices, which is consistent with the fact that lecturers, program coordinators, and academic managers will be able to access this system remotely (Troelsen & Japikse, 2022). To ensure persistence of the data, I added to it the ORM known as Entity Framework Core (EF Core), which handles interactions with databases, generates queries automatically, and includes change tracking features that minimize the boiler-plate code and improve its efficiency (Japkise & Troelsen, 2022)

The database is built using a relational model with EF Core that encourages the normalization and integrity of data by having entities such as Lecturer, Claim, Supporting Document and Approver. It is an extension of database-first to code-first, in which the schema can be modified over time (Troelsen & Japikse, 2022). Assumptions will be distinct user authentication with IDs; pre-established hourly rates stored on lecturers; no initial requirement of more serious security measures such as encryption; prototype emphasis on non-functional UI without complete integration on the backend level. The constraints include: fixed prototype (only UI navigation); use of SQL Server as a relational structure (Japikse & Troelsen, 2022); time constraint (2 weeks); and following an agile emerging design to become flexible (Troelsen & Japikse, 2022).

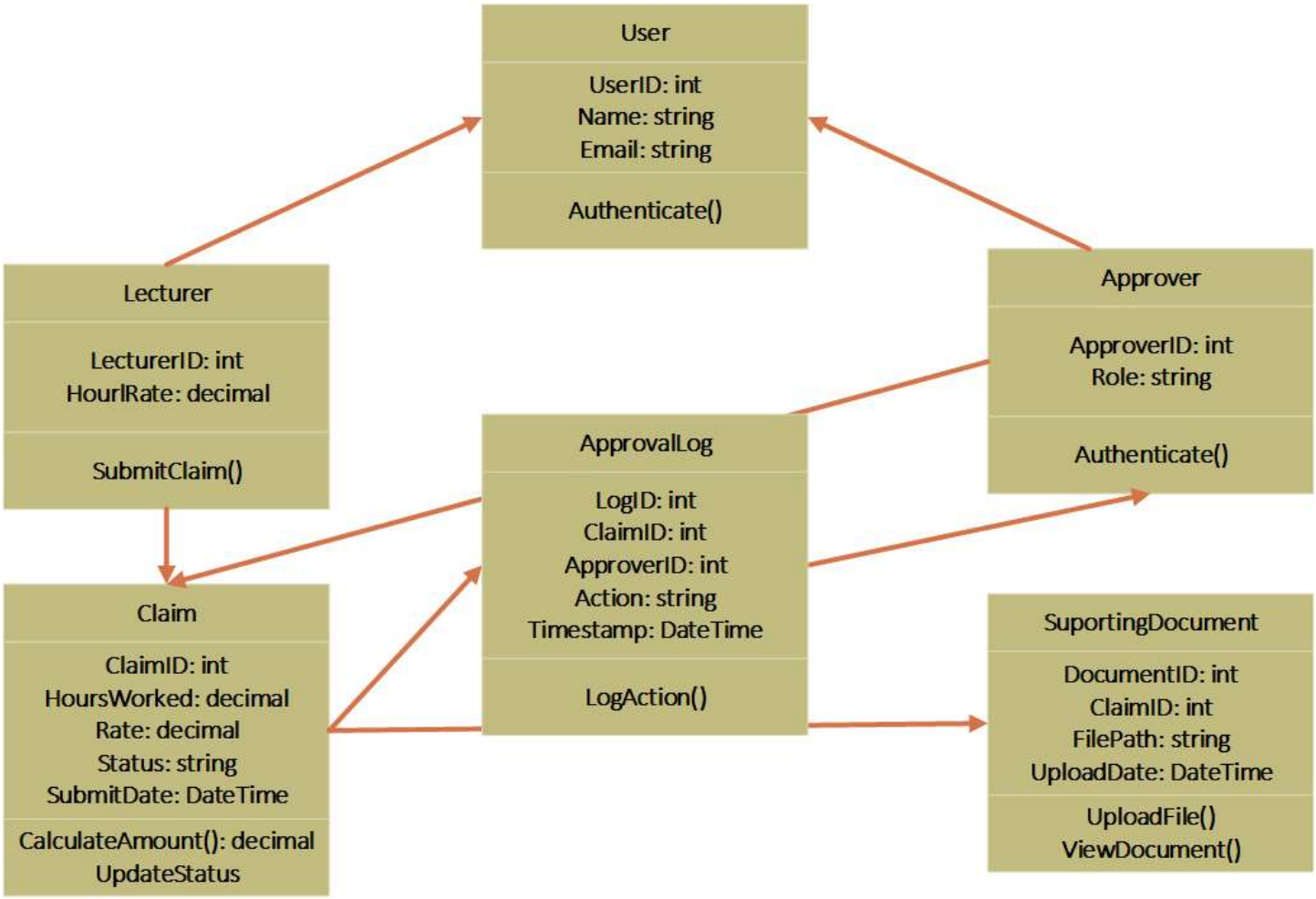
The GUI is designed to be user-friendly by combining Bootstrap with responsive layouts, which enable ease of navigation and interactions with forms (Japikse & Troelsen, 2022). This counters any possible mistakes in claims submitted and approved, increasing the overall system user-friendliness and satisfaction.

The final paragraphs include a summary of the research findings, which are absent from this section. The report has summarized the research findings in final paragraphs that have not been included in this section.

Footnote: Grok was only consulted on brainstorming and structuring ideas on preliminary topics.

2. UML Class Diagram for Databases

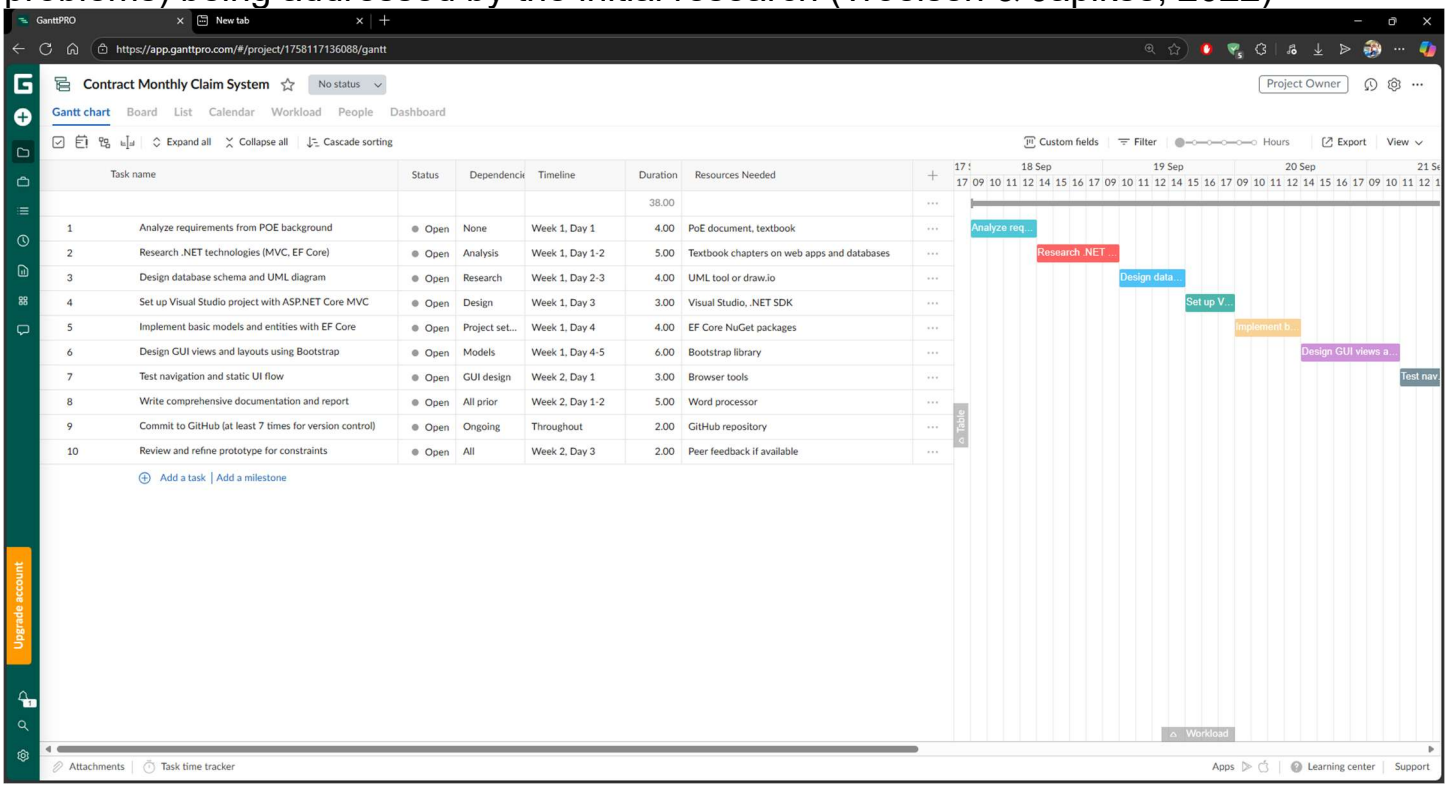
The UML class diagram illustrates the relational database entities, attributes, methods, and relationships, mapping to tables for efficient data management. It incorporates EF Core conventions for ORM integration (Japiklse & Troelsen, 2022)

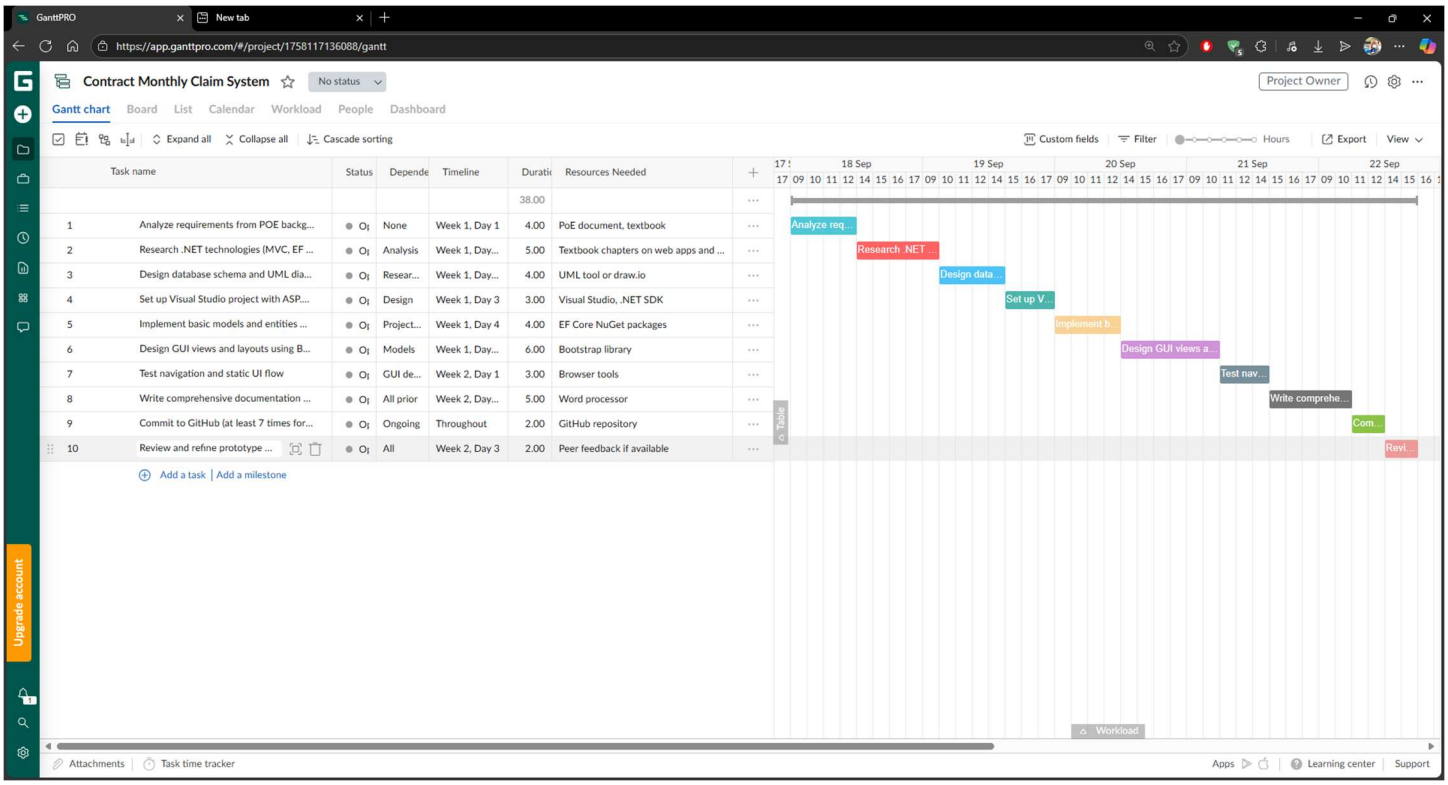


This diagram depicts one-to-many relationships: a Lecturer submits multiple Claims, a Claim links to multiple Supporting Documents, and an Approver handles multiple Claims. Methods like CalculateAmount() support business logic, such as total claim computation (hours * rate), while ensuring normalization to prevent redundancy (Japikse & Troelsen, 2022). Attributes facilitate tracking and auditing, with EF Core handling concurrency via timestamps (Japikse & Troelsen, 2022).

3. Project Plan

The project plan gives a systematic 2-week period of the prototype of Contract Monthly Claim System development, as the development is guided by the agile principles of the iterative development (Troelsen & Japikse, 2022)It has the dependencies, milestones, and risk mitigation plans and approximates a total effort of about 38 hours, which is comparable with the 45 hours required under the POE (The Independent Institute of Education, 2025, p. 1). The Gantt chart presented below is a plan that presupposes the sequential implementation of the tasks and the version control with the help of GitHub commits with the buffers (which will allow dealing with possible risks like tool setup problems) being addressed by the initial research (Troelsen & Japikse, 2022)





The Gantt chart shows the allocation of tasks in the course of Week 1, beginning with the date of September 17, 2025, and closing with the date of September 22, 2025. It involves, among other things, studying the POE background requirements, researching on the .NET technologies, creating the database schema and UML diagram, creating the Visual Studio project with ASP.NET core, implementing the basic models with EF core, designing the GUI views with Bootstrap, testing the navigation and static UI flow, writing a comprehensive documentation, and committing to GitHub at least 7 times, and reviewing the prototype to identify constraints. Tasks have allocated durations, resources and timelines and dependencies help in maintaining logical flow such as research is done before design and set up is done before implementation. The implicit milestones are established at the conclusion of Week 1 (schema and setup is finished) and Week 2 (prototype and report have been done).

Example: The Gantt chart is a graphical representation of the bars of the tasks with the activities overlapping in the Week 1 (e.g. research and design) and the phases separate in the Week 2 (e.g. testing and review), which proves the feasibility of the schedule with sufficient time margins to debug and optimize the tasks.

4. GUI/UI

The UI is prototyped in ASP.NET Core MVC with .NET Core, emphasizing responsive design via Bootstrap for cross-device compatibility (Troelsen & Japikse, 2022). The shared _Layout.cshtml features a Bootstrap navbar with role-based links: Home, Submit Claim (lecturers), View/Approve Claims (approvers), and Upload Documents. This promotes intuitive navigation, reducing user errors (Troelsen & Japikse, 2022).

Key pages:

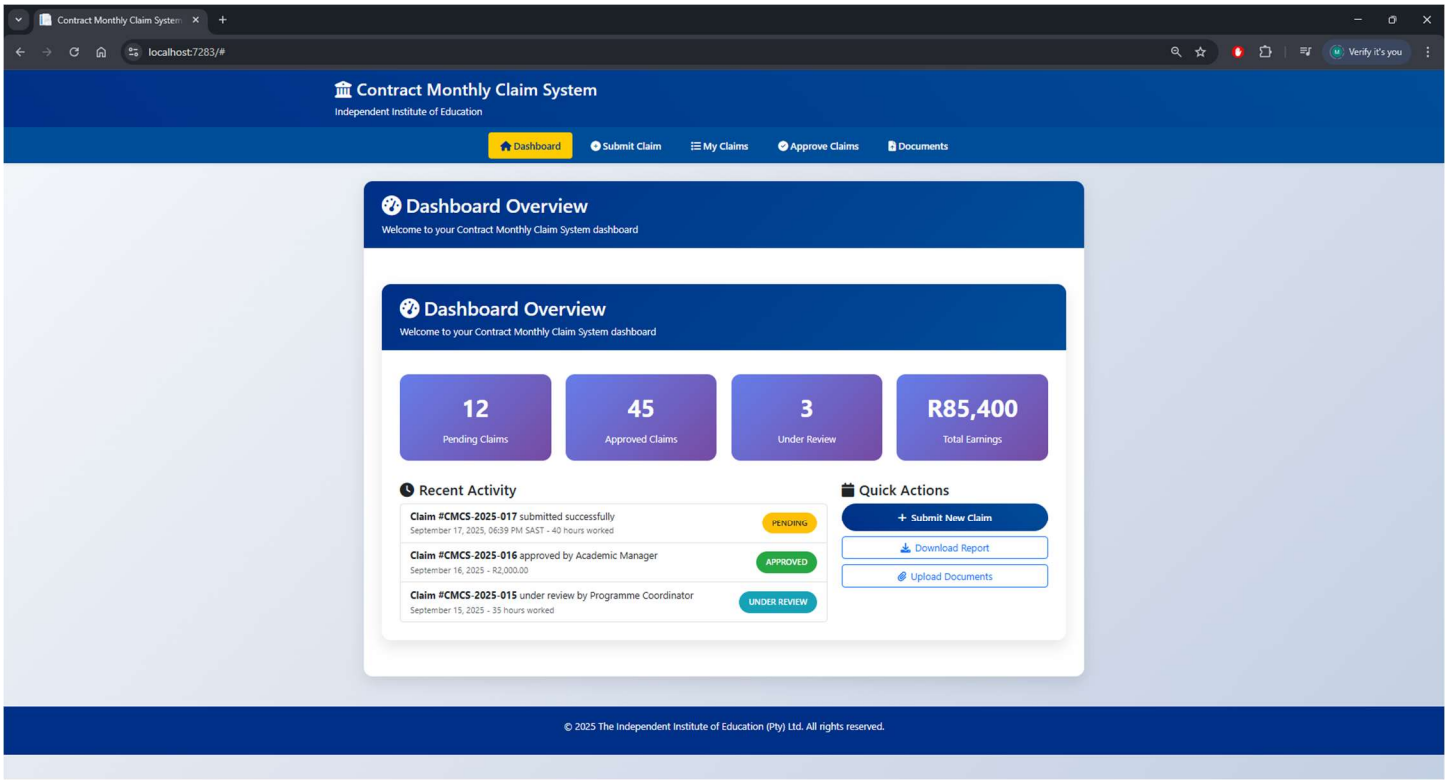
- Home Page: Dashboard with welcome message, claim summary stats (e.g., pending claims count), and quick links.
- Submit Claim Page: Form with inputs for hours (numeric validation), rate (read-only, auto-populated), notes (text area), and multi-file upload. Buttons: "Calculate Total" (JS for preview) and "Submit".
- View Claims Page: Table listing claims with columns for ID, Date, Hours, Amount, Status; sortable and filterable.
- Approve Claims Page: Similar table with action buttons: "Approve", "Reject", "View Documents".

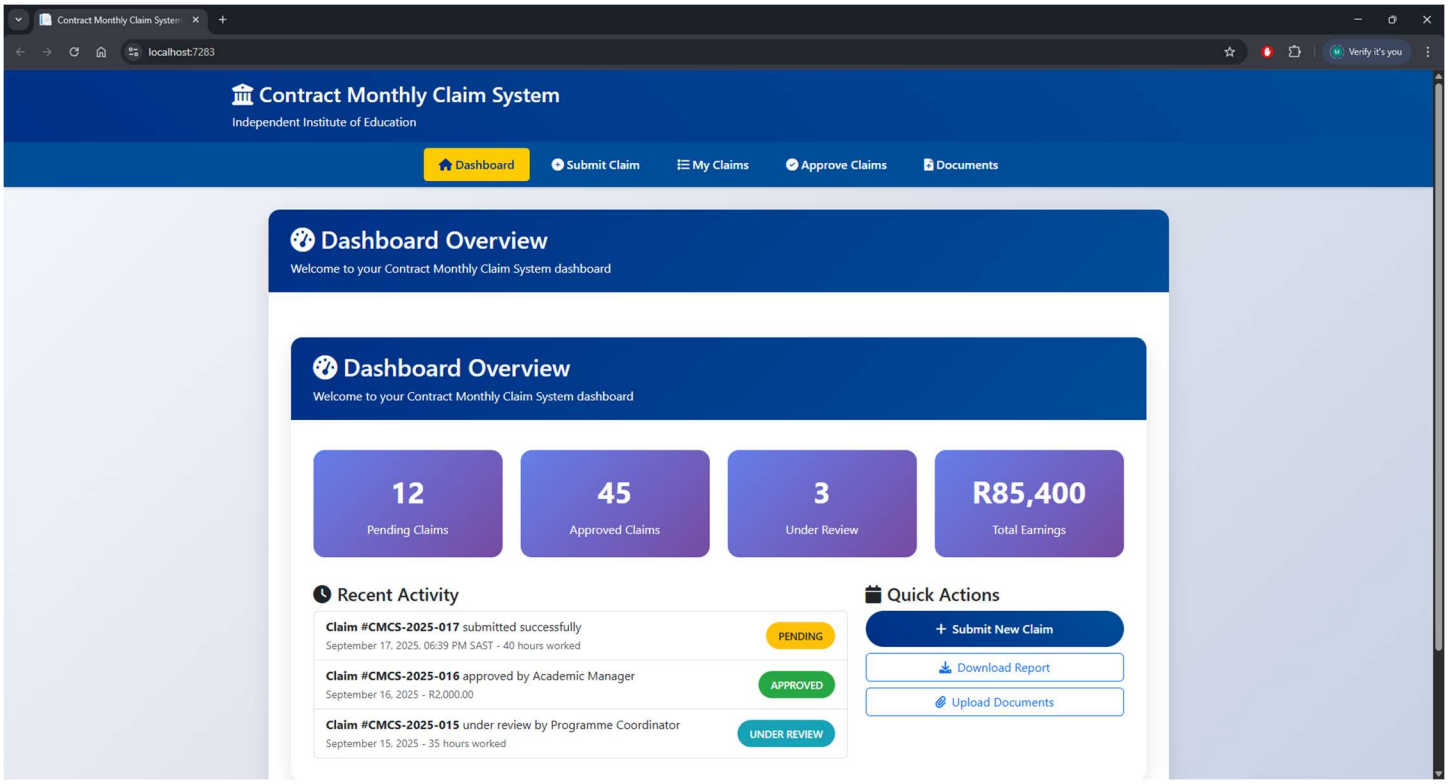
Design principles: Clear labels, error messages for validation, Bootstrap modals for confirmations, and accessibility features like ARIA attributes (Troelsen & Japikse, 2022). Prototype is non-functional, focusing on static views.

Illustration 1: Submit Claim Page Sketch (Text Description):

- Header: "Submit Monthly Claim" (h2, centered)
- Form Grid: Left column - Hours input (type=number), Rate display; Right - Notes text area; Bottom - Upload drop zone and Submit button (primary Bootstrap class).

This ensures usability, with future EF Core integration for dynamic data (Troelsen & Japikse, 2022)


Contract Monthly Claim System
localhost:7283/#
Contract Monthly Claim System
Independent Institute of Education
Dashboard
Submit Claim
My Claims
Approve Claims
Documents
Dashboard Overview
Welcome to your Contract Monthly Claim System dashboard
Dashboard Overview
Welcome to your Contract Monthly Claim System dashboard
12
Pending Claims
45
Approved Claims
3
Under Review
R85,400
Total Earnings
Recent Activity
Claim #CMCS-2025-017 submitted successfully
September 17, 2025, 06:39 PM SAST - 40 hours worked
PENDING
Claim #CMCS-2025-016 approved by Academic Manager
September 16, 2025 - R2,000.00
APPROVED
Claim #CMCS-2025-015 under review by Programme Coordinator
September 15, 2025 - 35 hours worked
UNDER REVIEW
Quick Actions
Submit New Claim
Download Report
Upload Documents
© 2025 The Independent Institute of Education (Pty) Ltd. All rights reserved.


Contract Monthly Claim System
localhost:7283
Contract Monthly Claim System
Independent Institute of Education
Dashboard
Submit Claim
My Claims
Approve Claims
Documents
Dashboard Overview
Welcome to your Contract Monthly Claim System dashboard
Dashboard Overview
Welcome to your Contract Monthly Claim System dashboard
12
Pending Claims
45
Approved Claims
3
Under Review
R85,400
Total Earnings
Recent Activity
Claim #CMCS-2025-017 submitted successfully
September 17, 2025, 06:39 PM SAST - 40 hours worked
PENDING
Claim #CMCS-2025-016 approved by Academic Manager
September 16, 2025 - R2,000.00
APPROVED
Claim #CMCS-2025-015 under review by Programme Coordinator
September 15, 2025 - 35 hours worked
UNDER REVIEW
Quick Actions
Submit New Claim
Download Report
Upload Documents

References

Japiklse, P. & Troelsen, A., 2022. In: *Pro C# 10 with .NET 6: Foundational Principles and Practices in Programming.* New York: Apress, p. 921.

Japikse, P. & Troelsen, A., 2022. In: *Pro C# 10 with .NET 6: Foundational Principles and Practices in Programming.* New York: Apress, p. 847.

Japikse, P. & Troelsen, A., 2022. In: *Pro C# 10 with .NET 6: Foundational Principles and Practices in Programming.* New York: Apress, p. 1520.

Japikse, P. & Troelsen, A., 2022. In: *Pro C# 10 with .NET 6: Foundational Principles and Practices in Programming.* New York: Apress, p. 1007.

Japikse, P. & Troelsen, A., 2022. In: *Pro C# 10 with .NET 6: Foundational Principles and Practices in Programming.* New York: Apress, p. 1015.

Japkise, P. & Troelsen, A., 2022. In: *Pro C# 10 with .NET 6: Foundational Principles and Practices in Programming.* New York: Apress, p. 911.

Troelsen, A. & Japikse, P., 2022. In: *Pro C# 10 with .NET 6: Foundational Principles and Practices in Programming.* New York: Apress, p. 1359.

Troelsen, A. & Japikse, P., 2022. In: *Pro C# 10 with .NET 6: Foundational Principles and Practices in Programming.* New York: Apress, p. 1364.

Troelsen, A. & Japikse, P., 2022. In: *Pro C# 10 with .NET 6: Foundational Principles and Practices in Programming.* New York: Apress, p. 959.

Troelsen, A. & Japikse, P., 2022. In: *Pro C# 10 with .NET 6: Foundational Principles and Practices in Programming.* New York: Apress, p. 1039.

Troelsen, A. & Japikse, P., 2022. *Pro C# 10 with .NET 6: Foundational Principles and Practices in Programming.* 11 ed. New York: Apress.