✔ 1. Improved Home UI Design

Feedback: The home screen needs to look more professional and intuitive.
Implementation:

- Redesigned the Home page with a hero banner, feature cards, and clear role-based navigation.
- Updated typography, spacing, and layout using Bootstrap.

✔ 2. Improved Claim Submission Form

Feedback: Make the form easier to use and ensure accurate calculations.
Implementation:

- Added **automatic total payment calculation** using JavaScript.
- Structured form inputs with Bootstrap for better usability.
- Added field labels, placeholders, and validation messages.
- Added a supporting document upload field.

✔ 3. Added Front-end + Back-end Validation

Feedback: Ensure data integrity and prevent invalid submissions.
Implementation:

- Used Data Annotations such as `Required`, `Range`, and `StringLength`.
- Added client-side validation alerts.
- Added file type & file size checks.

✔ 4. Suspicious Claim Detection Logic

Feedback: Coordinator should identify suspicious claims.
Implementation:

- Implemented detection rules for:
    - Hours > 160
    - Hourly Rate > 1000
    - Missing document
    - Missing notes
- Highlighted suspicious claims visually in yellow (`table-warning`).
- Added a ⚠ **Suspicious** badge.

✔ 5. Coordinator & Manager Workflow Fixes

Feedback: Ensure correct workflow and prevent invalid actions.
Implementation:

- Coordinator sees **Pending** claims only.
- Manager sees **VerifiedByCoordinator** claims only.
- Added guard clauses to prevent:
    - approving already approved claims
    - rejecting claims that weren't verified
- Added success/error alerts using TempData + global alert section in `_Layout.cshtml`.

✔ 6. Global Error Handling

Feedback: Show meaningful feedback when something goes wrong.
Implementation:

- Created a custom `Error.cshtml` page.
- Wrapped controller actions in `try-catch`.
- Implemented global exception handling in `Program.cs`.
- Displayed user-friendly messages.

✔ 7. SQL Server Migration & Script

Feedback: Use SQL Server (not MySQL/SQLite) and provide a script.
Implementation:

- Converted the project to **SQL Server Express** with correct connection string.
- Ran EF Core migrations to generate schema.
- Added precision to decimal fields (`decimal(10,2)`) to prevent truncation warnings.
- Generated a **full SQL Script (schema + data)** in SSMS for submission.

✔ 8. Version Control (GitHub)

Feedback: Commit regularly with clear messages.
Implementation:

- Pushed all source code to GitHub.
- Used meaningful commit messages throughout Part 1–3.
- Included SQL script, documentation, and PPT.

📌 Evidence of Improvements

Screenshots included in the repo show the implemented features:

- Lecturer Claim Submission
- Auto total calculation
- File upload validation
- Suspicious claim detection
- Coordinator approval dashboard
- Manager approval dashboard
- Global success/error alerts
- Error page
- SQL database in SSMS

These demonstrate the successful application of the lecturer's recommendations.

📌 Conclusion

The lecturer's feedback significantly improved the clarity, usability, and robustness of the CMCS prototype.
All recommended enhancements were implemented, resulting in:

- a cleaner UI
- accurate workflows
- safer processing
- stronger validation
- proper database integration

The system now aligns fully with POE requirements and industry-level best practices for ASP.NET Core MVC applications.