PROG6212: DCUMENTATION

COMPLETE DOCUMENTATION OF PROG6212 PART 1

NAOYUKI CHRISTOPHER HIGAKI

Table of Contents

Contract Monthly Claim System - Prototype Documentation

GITHUB LINK: https://github.com/HChristopherNaoyuki/contract-monthly-claim-system-cs.git

1.0: Project Overview

The Contract Monthly Claim System (CMCS) is a comprehensive web-based application designed to streamline the monthly claim submission and approval process for independent contractor lecturers. This prototype represents the initial development phase, focusing on core functionality while maintaining a user-centric design approach. The system addresses the complex administrative challenges of claim management through an intuitive interface that serves three distinct user roles: lecturers, program coordinators, and academic managers.

2.0: System Architecture

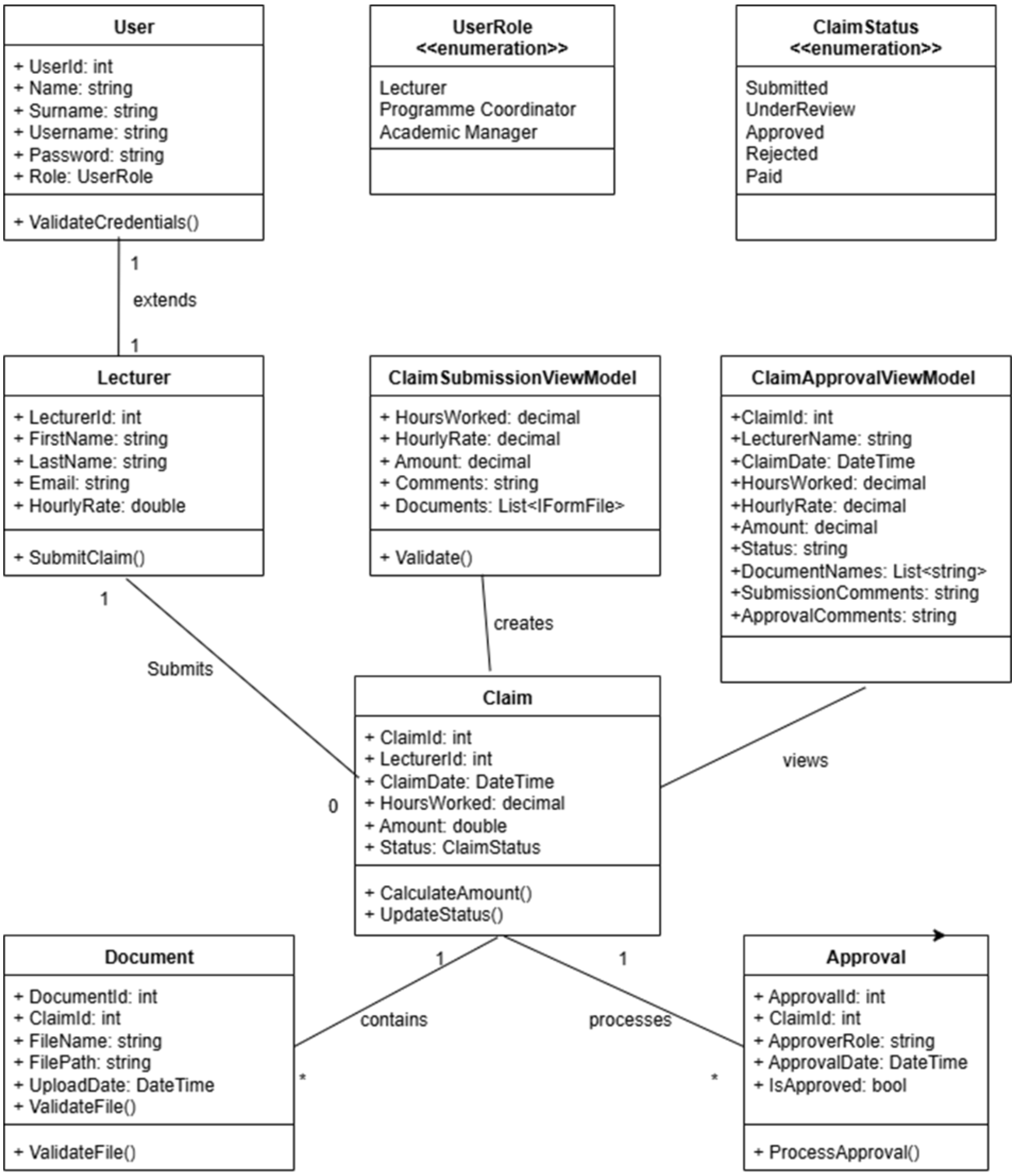
The prototype is built on ASP.NET Core MVC framework using C# 7.0, following industry-standard development practices. Architecture employs the Model-View-Controller pattern, ensuring clear separation of concerns and maintainable code structure. Key technical components include session-based authentication, client-side validation, and in-memory data storage for the prototype phase. The frontend utilizes modern CSS features including Grid and Flexbox for responsive design, ensuring optimal user experience across various devices.

3.0: UML Class Diagram Structure

The system's data model is represented through a comprehensive UML class diagram featuring several key entities:

- User Class: Manages authentication with attributes for user identification, credentials, and role assignment
- Lecturer Class: Extends user functionality with specific attributes for contract details and hourly rates
- Claim Class: Core entity handling claims submissions with properties for hours worked, rates, and status tracking
- Document Class: Manages supporting file uploads and associations with specific claims
- Approval Class: Tracks review processes with timestamps, decisions, and approver comments
- Relationships between classes include one-to-many associations between lecturers and claims, and claims to documents, ensuring logical data structure and integrity.

UML Diagram:


User
+ UserId: int
+ Name: string
+ Surname: string
+ Username: string
+ Password: string
+ Role: UserRole
+ ValidateCredentials()
UserRole <<enumeration>>
Lecturer
Programme Coordinator
Academic Manager
ClaimStatus <<enumeration>>
Submitted
UnderReview
Approved
Rejected
Paid
1
extends
1
Lecturer
+ LecturerId: int
+ FirstName: string
+ LastName: string
+ Email: string
+ HourlyRate: double
+ SubmitClaim()
ClaimSubmissionViewModel
+ HoursWorked: decimal
+ HourlyRate: decimal
+ Amount: decimal
+ Comments: string
+ Documents: List<IFormFile>
+ Validate()
ClaimApprovalViewModel
+ClaimId: int
+LecturerName: string
+ClaimDate: DateTime
+HoursWorked: decimal
+HourlyRate: decimal
+Amount: decimal
+Status: string
+DocumentNames: List<string>
+SubmissionComments: string
+ApprovalComments: string
1
Submits
creates
0
views
Claim
+ ClaimId: int
+ LecturerId: int
+ ClaimDate: DateTime
+ HoursWorked: decimal
+ Amount: double
+ Status: ClaimStatus
+ CalculateAmount()
+ UpdateStatus()
1
1
contains
processes
*
*
Document
+ DocumentId: int
+ ClaimId: int
+ FileName: string
+ FilePath: string
+ UploadDate: DateTime
+ ValidateFile()
+ ValidateFile()
Approval
+ ApprovalId: int
+ ClaimId: int
+ ApproverRole: string
+ ApprovalDate: DateTime
+ IsApproved: bool
+ ProcessApproval()

4.0: Project Plan & Timeline

The prototype development followed an agile methodology with two weeks sprints:

- Week 1-2: Requirements analysis and UML diagram completion
- Week 3-4: Core framework setup and authentication system implementation
- Week 5-6: Claim submission module with validation and calculation features
- Week 7-8: Review and approval interface development
- Week 9-10: Tracking system and user interface refinement
- Week 11-12: Testing, documentation, and final adjustments

5.0: GUI Design Philosophy

The interface adopts a minimalist aesthetic characterized by:

- Clean, uncluttered layouts with strategic white space utilization.
- Consistent typography using system fonts with proper hierarchy.
- Subtle animations and transitions for enhanced user feedback.
- Intuitive navigation with visually distinct interactive elements.
- Accessibility-focused design including keyboard navigation support.
- Responsive design ensuring functionality across desktop and mobile platforms.

6.0: Core Functionality

The prototype delivers essential features including:

- Role-based authentication and session management.
- Dynamic claim form with real-time amount calculation.
- File upload system with type and size validation.
- Dual-comment system for submission notes and approval feedback.
- Status tracking with visual indicators for different claim states.
- Responsive data tables for efficient information review.

7.0: Compliance & Next Steps

This prototype meets all Part 1 requirements while establishing a solid foundation for future development. The system is prepared for database integration, enhanced security features, and additional functionality in subsequent phases. The modular architecture ensures scalability while maintaining the user-friendly experience that defines the initial design vision.