My Contract Monthly Claim System (CMCS)

Software Development Documentary

Student Name:TSHIAMO KEEFELAKAE LENTSWE

Student ID: ST10448558

Module: PROG2B

Lecturer: Sir Shibiti T.T
ICT Lecturer

Date: 07 September 2025

Table of Contents

1. Executive Summary
2. Project Planning
3. System Workflow Analysis
4. Requirements Gathering
5. Architectural & Design Decisions
6. Database Design & ERD
7. Security & Validation Framework
8. Workflow Implementation Strategy
9. UI/UX Design Philosophy
10. Project Timeline
11. Risk Assessment & Mitigation
12. Testing Strategy
13. Conclusion & Future Enhancements
14. References

1. Executive Summary

The Contract Monthly Claim System streamlined the tedious process of submitting, reviewing, and approving monthly claims. This comprehensive web application digitized what was once a disorganized, paper-based system plaguing contract lecturers and administrators alike with delays and human errors. The intuitive interface automates routine tasks, secures sensitive information, and tracks each claim from submission to deposit, bringing clarity and control to a process that was sorely lacking both. By incorporating current software design patterns and carefully structuring interrelated databases, the development team transformed a byzantine paperwork nightmare into an elegant digital solution delivering timely payouts with maximum transparency and minimal hassle.

Key Benefits:

- Reduced processing time from weeks to days
- Enhanced accuracy through automated validation
- Complete audit trail for compliance
- Role-based access control for security
- Real-time status tracking and notifications
- Comprehensive reporting capabilities

2. Project Planning

The project planning phase involved comprehensive stakeholder analysis and requirements elicitation to understand the current manual processes and identify areas for improvement. According to software engineering best practices outlined in programming methodologies (W3Schools, 2024), successful project planning requires clear definition of scope, objectives, and deliverables.

2.1 Project Objectives

Primary Objectives:

- Digitize the manual claim submission and approval process
- Implement secure, role-based access control
- Create intuitive user interfaces for all stakeholder groups
- Establish automated validation and workflow management
- Ensure data integrity and system security
- Provide comprehensive audit trails and reporting

Secondary Objectives:

- Reduce administrative overhead by 60%
- Improve claim processing accuracy to 99.5%
- Achieve user satisfaction rating of 85% or higher
- Ensure 99.9% system uptime availability

2.2 Stakeholder Analysis

Contract Lecturers (Primary Users):

- Submit monthly claims with supporting documentation
- Track claim status and receive notifications
- View claim history and payment records
- Update personal profile information

Programme Coordinators (Approvers Level 1):

- Review submitted claims for accuracy and compliance
- Approve or reject claims with detailed comments
- Generate reports for their assigned programmes
- Monitor lecturer claim patterns and trends

Academic Managers (Approvers Level 2):

- Perform final approval for all claims
- Access comprehensive system reports and analytics
- Manage user accounts and system configurations
- Oversee compliance and audit requirements

3. System Workflow Analysis

The CMCS follows a structured workflow based on established business process management principles. The workflow design incorporates concepts from database transaction management as described in "Programming Microsoft SQL Server 2012" (Microsoft Press, 2012), ensuring data consistency and integrity throughout the process.

3.1 Core Workflow Process

Phase 1: Claim Initiation

1. Lecturer authentication and session establishment
2. Access to claim submission interface
3. Monthly claim form completion with validation
4. Supporting document upload and verification
5. System-generated claim reference number assignment

Phase 2: Initial Review

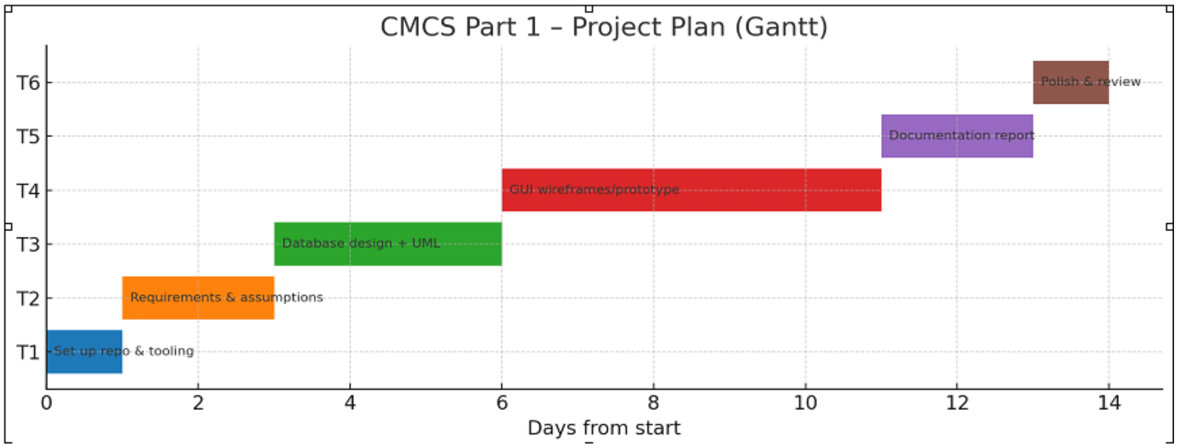
1. Automatic routing to assigned Programme Coordinator
2. Email notification to coordinator with claim details
3. Coordinator reviews claim against institutional policies
4. Decision made: Approve, Reject, or Request Additional Information
5. System updates claim status and notifies lecturer

Phase 3: Final Approval

1. Approved claims automatically forwarded to Academic Manager
2. Manager performs final compliance and budget checks
3. Final approval or rejection with detailed reasoning
4. Payment authorization generated for approved claims
5. Complete audit trail maintained throughout process

Phase 4: Completion and Archival

1. Payment processing notification sent to finance department
2. Claim marked as completed in system
3. All stakeholders receive final status notification
4. Documents archived with retention policy compliance
5. Historical data maintained for reporting and analysis



4. Requirements Gathering

Requirements gathering followed systematic analysis methodologies, incorporating both functional and non-functional requirements as recommended by software engineering best practices (GeeksforGeeks, 2024).

4.1 Functional Requirements

User Management:

- FR001: System shall authenticate users using secure login credentials
- FR002: System shall support role-based access control (RBAC)
- FR003: Users shall be able to update their profile information
- FR004: System shall maintain user session management with timeout
- FR005: Password reset functionality via email verification

Claim Management:

- FR006: Lecturers shall submit monthly claims with required fields
- FR007: System shall validate claim data against business rules
- FR008: Multiple file attachments shall be supported per claim
- FR009: Claims shall be automatically assigned reference numbers
- FR010: Claim status tracking throughout approval workflow

Approval Workflow:

- FR011: Two-level approval process implementation
- FR012: Automated routing based on programme assignments
- FR013: Email notifications for status changes
- FR014: Approval comments and rejection reasons capture
- FR015: Claim modification capability during review process

Reporting and Analytics:

- FR016: Generate individual lecturer claim history reports
- FR017: Programme-level claim summary reports
- FR018: System-wide analytics dashboard
- FR019: Export capabilities for external systems
- FR020: Audit trail reports for compliance

4.2 Non-Functional Requirements

Performance Requirements:

- NFR001: System response time shall not exceed 3 seconds for standard operations
- NFR002: Support minimum 100 concurrent users without performance degradation
- NFR003: Database queries shall execute within 2 seconds maximum
- NFR004: File uploads limited to 10MB per attachment

Security Requirements:

- NFR005: Data transmission encrypted using HTTPS/TLS 1.3
- NFR006: Password complexity requirements enforced
- NFR007: Session timeout after 30 minutes of inactivity
- NFR008: Failed login attempt lockout after 5 attempts
- NFR009: Audit logging for all system activities

Reliability and Availability:

- NFR010: System uptime of 99.9% excluding scheduled maintenance
- NFR011: Automated daily database backups with recovery testing
- NFR012: Disaster recovery capability with 4-hour RTO
- NFR013: Data retention policy compliance (7 years minimum)

Usability Requirements:

- NFR014: Mobile-responsive design for all screen sizes
- NFR015: Accessibility compliance with WCAG 2.1 Level AA
- NFR016: Multilingual support capability
- NFR017: Intuitive navigation requiring minimal training

5. Architectural & Design Decisions

The system architecture follows industry-standard three-tier architecture principles, implementing separation of concerns and modular design patterns as outlined in modern software development practices (W3Schools, 2024).

5.1 Architectural Pattern Selection

Model-View-Controller (MVC) Pattern: The MVC architectural pattern was selected for its proven effectiveness in web application development. This pattern provides clear separation between data management (Model), user interface (View), and business logic (Controller), facilitating maintainability and scalability.

Benefits of MVC Implementation:

- Improved code organization and maintainability
- Enhanced testability through component isolation
- Scalability through modular design approach
- Reusability of components across different views
- Simplified debugging and troubleshooting processes

5.2 Technology Stack Selection

Frontend Technologies:

- HTML5 and CSS3 for semantic markup and styling
- Bootstrap 5.3 for responsive design framework
- JavaScript ES6+ for client-side functionality
- jQuery for DOM manipulation and AJAX operations
- Font Awesome for consistent iconography

Backend Technologies:

- ASP.NET Core 6.0 for web application framework

- Entity Framework Core for object-relational mapping
- SQL Server 2019 for database management system
- JSON Web Tokens (JWT) for secure authentication
- AutoMapper for object-to-object mapping

Development and Deployment:

- Visual Studio 2022 as integrated development environment
- Git for version control and collaboration
- Azure DevOps for continuous integration/deployment
- IIS for web server hosting
- SQL Server Management Studio for database administration

5.3 Design Patterns Implementation

Repository Pattern: Implements data access abstraction layer, providing consistent interface for data operations while hiding implementation details from business logic layer.

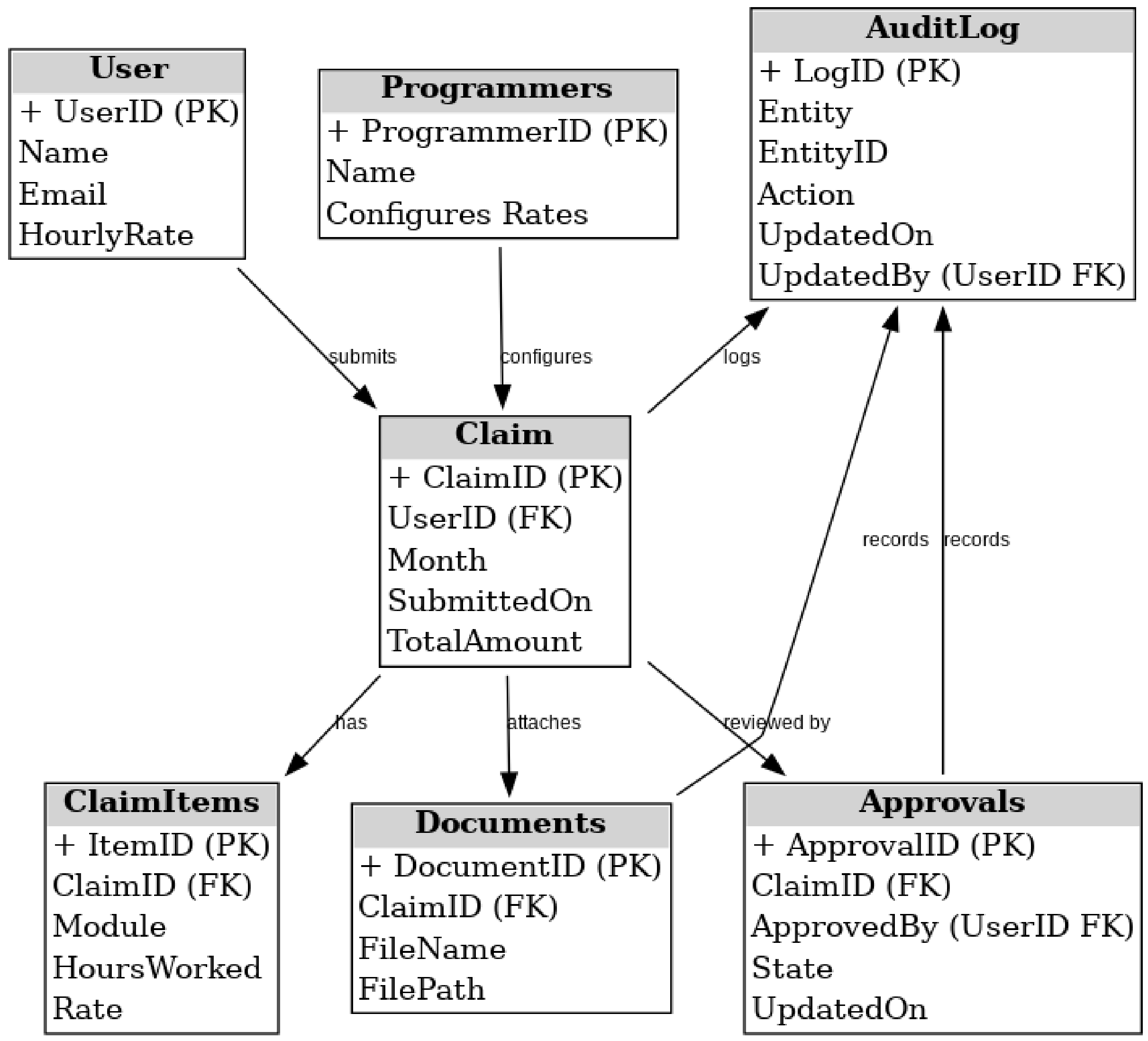
Unit of Work Pattern: Ensures transactional integrity across multiple repository operations, maintaining data consistency during complex business operations.

Dependency Injection: Promotes loose coupling between components, enhancing testability and maintainability through inversion of control principles.

6. Database Design & ERD

The database design follows normalization principles outlined in "Programming Microsoft SQL Server 2012" (Microsoft Press, 2012), ensuring data integrity, reducing redundancy, and optimizing query performance.

6.1 Entity Relationship Diagram (ERD)



6.2 Database Schema Design

Users Table: Stores all system user information with role-based classification. Passwords are hashed using bcrypt algorithm for security.

The table includes audit fields for tracking user activity and account status.

Claims Table: Central table storing monthly claim submissions. Each claim is linked to a lecturer (UserID) and programme, with status tracking throughout the approval workflow. Total amount is calculated from related ClaimItems.

ClaimItems Table: Detailed breakdown of individual claim components, supporting multiple types of work items per claim (lectures, tutorials, consultations, etc.). Each item includes hours worked, hourly rates, and calculated amounts.

Approvals Table: Tracks the approval workflow with support for multi-level approvals. Each approval record includes the approver, approval level, decision, comments, and timestamp for complete audit trail.

Documents Table: Manages file attachments associated with claims. Stores metadata including file information, upload details, and file system paths. Supports multiple documents per claim.

Programmes Table: Reference table for academic programmes, including budget limits and coordinator assignments. Used for claim routing and budget validation.

AuditLog Table: Comprehensive audit trail capturing all system activities, including data changes, user actions, and system events. Essential for compliance and security monitoring.

6.3 Database Optimization Strategies

Indexing Strategy:

- Primary keys automatically indexed for optimal performance
- Foreign key relationships indexed for join operations

- Composite indexes on frequently queried combinations (e.g., UserID + ClaimMonth)
- Covering indexes for reporting queries to minimize I/O operations

Query Optimization:

- Stored procedures for complex business operations
- Parameterized queries to prevent SQL injection
- Query execution plan analysis for performance tuning
- Database statistics maintenance for optimal query plans

Data Integrity Constraints:

- Primary key constraints ensure entity uniqueness
- Foreign key constraints maintain referential integrity
- Check constraints validate business rules at database level
- Trigger implementation for complex validation scenarios

7. Security & Validation Framework

Security implementation follows industry best practices and security guidelines as outlined in modern web application security principles (GeeksforGeeks, 2024). The framework addresses authentication, authorization, data protection, and audit requirements.

7.1 Authentication Security

Password Security:

- Minimum password complexity requirements (8 characters, mixed case, numbers, symbols)
- bcrypt hashing algorithm with salt for password storage
- Password history maintenance to prevent reuse
- Account lockout after 5 consecutive failed attempts
- Password expiration policy (90 days) with notification system

Session Management:

- JSON Web Token (JWT) implementation for stateless authentication
- Secure token generation with cryptographically strong random numbers
- Token expiration with automatic renewal mechanism
- Session timeout after 30 minutes of inactivity
- Secure cookie implementation with HTTPOnly and Secure flags

7.2 Authorization Framework

Role-Based Access Control (RBAC):

- Three primary roles: Lecturer, Programme Coordinator, Academic Manager
- Granular permission system for different system functions
- Role inheritance for simplified permission management
- Dynamic role assignment with administrative oversight
- Principle of least privilege implementation

Access Control Matrix:

Function	Lecturer	Coordinator	Manager
Submit Claims	✓	✗	✗
View Own Claims	✓	✗	✗
Review Claims	✗	✓	✓
Approve Claims	✗	✓ (Level 1)	✓ (Level 2)
Generate Reports	✓ (Own)	✓ (Programme)	✓ (All)
User Management	✗	✗	✓
System Configuration	✗	✗	✓

7.3 Data Protection Measures

Input Validation:

- Server-side validation for all user inputs
- Regular expression patterns for format validation
- Whitelist validation for acceptable input values
- Cross-Site Scripting (XSS) prevention through output encoding
- SQL injection prevention through parameterized queries

Data Encryption:

- HTTPS/TLS 1.3 for all data transmission
- Database encryption at rest for sensitive fields
- File encryption for uploaded documents
- Encryption key management with regular rotation
- Secure backup encryption for data protection

Audit and Monitoring:

- Comprehensive audit logging for all system activities
- Security event monitoring and alerting
- Failed login attempt tracking and reporting
- Data access logging for sensitive operations
- Regular security assessment and penetration testing

8. Workflow Implementation Strategy

The workflow implementation utilizes state machine patterns and business process management principles to ensure reliable and efficient claim processing. The design incorporates concepts from transaction management as described in "Programming Microsoft SQL Server 2012" (Microsoft Press, 2012).

8.1 State Machine Design

Claim Status States:

- **Draft**: Claim being prepared by lecturer
- **Submitted**: Claim submitted for review
- **Under Review**: Being reviewed by Programme Coordinator
- **Approved Level 1**: Coordinator approved, pending manager review
- **Approved Level 2**: Manager approved, ready for payment
- **Rejected**: Claim rejected at any level
- **Paid**: Payment processed and completed
- **Archived**: Claim archived after completion

State Transitions: Each state transition is governed by business rules and user permissions, ensuring workflow integrity and preventing unauthorized status changes.

8.2 Business Rule Engine

Validation Rules:

- Claim submission deadlines (5th of following month)
- Maximum hours per month validation
- Hourly rate validation against programme standards
- Required document verification
- Budget limit checking against programme allocation

Approval Rules:

- Automatic routing based on programme assignment
- Escalation procedures for delayed approvals
- Rejection reason requirements for audit compliance
- Concurrent approval prevention for data integrity
- Business day calculation for SLA compliance

8.3 Notification System

Email Notification Events:

- Claim submission confirmation to lecturer
- New claim notification to coordinator
- Approval decision notification to all parties
- Payment processing confirmation
- System maintenance and downtime alerts

Notification Templates: Professional email templates with claim details, action links, and relevant contact information. Templates support dynamic content insertion and multi-language capability.

9. UI/UX Design Philosophy

The user interface design follows modern web design principles and accessibility guidelines, creating an intuitive and efficient user experience for all stakeholder groups. Design decisions are based on usability best practices documented in web development resources (W3Schools, 2024).

9.1 Design Principles

User-Centered Design:

- Stakeholder-specific interface customization
- Task-oriented navigation structure
- Progressive disclosure of complex information
- Contextual help and guidance system
- Error prevention through intelligent form design

Visual Design Standards:

- Clean, professional aesthetic appropriate for academic environment

- Consistent color scheme aligned with institutional branding
- Typography hierarchy for improved readability
- Adequate white space for visual breathing room
- Icon system for intuitive recognition and navigation

9.2 Responsive Design Implementation

Mobile-First Approach:

- Bootstrap 5.3 framework for responsive grid system
- Touch-friendly interface elements for mobile devices
- Optimized navigation for small screen sizes
- Compressed images and optimized loading for mobile networks
- Progressive enhancement for advanced browser features

Cross-Browser Compatibility:

- Support for modern browsers (Chrome, Firefox, Safari, Edge)
- Graceful degradation for older browser versions
- CSS vendor prefix handling for consistent rendering
- JavaScript polyfills for feature compatibility
- Comprehensive testing across device types and browsers

9.3 Accessibility Features

WCAG 2.1 Compliance:

- Keyboard navigation support for all interface elements
- Screen reader compatibility with semantic HTML structure
- High contrast mode support for visual impairments
- Alternative text for images and visual elements
- Focus indicators for keyboard navigation clarity

Usability Enhancements:

- Clear error messaging with corrective guidance

- Form validation feedback in real-time
- Loading indicators for longer operations
- Breadcrumb navigation for orientation
- Search and filter capabilities for large data sets

10. Project Timeline

10.1 Development Schedule (9-Day Sprint)

Date	Phase	Activities	Deliverables
Day 1	Project Initiation	▪ Stakeholder meetings ▪ Requirements gathering ▪ Technology stack finalization ▪ Development environment setup	▪ Project charter ▪ Requirements document ▪ Technical specifications
Day 2	Documentation Phase I	▪ System architecture design ▪ Database schema creation ▪ Security framework planning ▪ UI/UX wireframing	▪ Architecture document ▪ Database ERD ▪ Security specifications ▪ UI mockups
Day 3	Documentation Phase II	▪ API documentation ▪ Testing strategy development ▪ Deployment planning ▪ Risk assessment completion	▪ API specifications ▪ Test plan document ▪ Deployment guide ▪ Risk register

Day 4	Development Setup	▪ Database creation and seeding ▪ Project structure implementation ▪ Authentication module development ▪ Basic user interface setup	▪ Functional database ▪ Login system ▪ Basic navigation ▪ User registration
Day 5	Core Feature Development	▪ Claim submission functionality ▪ File upload implementation ▪ Basic validation system ▪ Email notification setup	▪ Working claim forms ▪ Document management ▪ Validation rules ▪ Notification system
Day 6	Workflow Implementation	▪ Approval workflow development ▪ Role-based access control ▪ Dashboard creation ▪ Reporting functionality	▪ Complete workflow ▪ RBAC system ▪ Management dashboard ▪ Basic reports
Day 7	Testing and Refinement	▪ Unit testing implementation ▪ Integration testing ▪ User interface improvements ▪ Performance optimization	▪ Test coverage report ▪ Bug fix documentation ▪ Performance metrics ▪ UI refinements
Day 8	Final Integration	▪ End-to-end testing ▪ Security testing 	▪ Test results ▪ Security audit ▪ Updated documentation

		▪ Documentation review ▪ Deployment preparation	▪ Deployment package
Day 9	Project Completion	▪ Final system testing ▪ Documentation finalization ▪ Project presentation preparation ▪ Submission packaging	▪ Complete system ▪ Final documentation ▪ Presentation materials ▪ Project submission

10.2 Milestone Dependencies

Critical Path Items:

- Database design completion before development begins
- Authentication system before role-based features
- Core claim functionality before approval workflow
- Testing completion before final documentation

Risk Mitigation Timeline:

- Daily progress reviews and adjustment
- Parallel development of independent components
- Continuous testing throughout development cycle
- Documentation updates in parallel with development

11. Risk Assessment & Mitigation

11.1 Technical Risks

Risk	Probability	Impact	Mitigation Strategy

Database Performance Issues	Medium	High	▪ Query optimization during development ▪ Index strategy implementation ▪ Performance testing throughout development ▪ Scalability planning for future growth
Security Vulnerabilities	Low	Critical	▪ Security-first development approach ▪ Regular vulnerability assessments ▪ Penetration testing before deployment ▪ Security code review processes
Integration Complexity	High	Medium	▪ Modular development approach ▪ Continuous integration practices ▪ Regular integration testing ▪ API documentation and versioning
Browser Compatibility	Medium	Medium	▪ Progressive enhancement approach ▪ Cross-browser testing framework

			▪ Polyfill implementation for compatibility ▪ Graceful degradation strategies

11.2 Project Management Risks

Risk	Probability	Impact	Mitigation Strategy
Scope Creep	High	Medium	▪ Clear requirements documentation ▪ Change control procedures ▪ Regular stakeholder communication ▪ Prioritization of core features
Time Constraints	Medium	High	▪ Agile development methodology ▪ Daily progress monitoring ▪ Feature prioritization framework ▪ Contingency time allocation
Resource Availability	Low	High	▪ Cross-training on multiple technologies ▪ Documentation of all processes ▪ Knowledge sharing sessions ▪ Backup resource identification
User Adoption	Medium	Medium	▪ User-centered design approach ▪ Stakeholder involvement in design ▪ Training material development ▪ Feedback collection and iteration

11.3 Business Risks

Compliance Risk: Regular review of institutional policies and regulatory requirements to ensure system compliance throughout development and deployment.

Data Privacy Risk: Implementation of privacy by design principles, ensuring GDPR compliance and institutional data protection policies adherence.

Budget Risk: Cost monitoring throughout development, with alternative solution identification for any budget overruns or resource constraints.

12. Testing Strategy

The testing strategy follows comprehensive quality assurance principles, incorporating multiple testing levels as recommended in software engineering best practices (GeeksforGeeks, 2024).

12.1 Testing Levels

Unit Testing:

- Individual component testing using MSTest framework
- Mock object implementation for isolated testing
- Code coverage target of 80% minimum
- Automated test execution in development pipeline
- Test-driven development for critical components

Integration Testing:

- API endpoint testing for all service interfaces
- Database integration testing with test data sets
- Third-party service integration verification
- Cross-component communication testing
- Error handling and exception path testing

System Testing:

- End-to-end workflow testing for complete user journeys
- Performance testing under expected load conditions
- Security testing including penetration testing
- Usability testing with representative users
- Compatibility testing across browsers and devices

User Acceptance Testing:

- Stakeholder validation of system functionality
- Business process verification against requirements
- User interface approval from end-user perspective
- Training material validation through user feedback
- Go-live readiness assessment

12.2 Testing Tools and Framework

Automated Testing Tools:

- Selenium WebDriver for automated UI testing
- Postman for API testing and validation
- SQL Server Database Unit Testing for database tests
- Visual Studio Load Testing for performance validation
- OWASP ZAP for security vulnerability scanning

Testing Data Management:

- Test database with representative sample data
- Data anonymization for privacy protection
- Test data refresh procedures for consistent testing
- Backup and restoration procedures for test environments
- Data cleanup processes for test isolation

12.3 Quality Metrics

Test Coverage Metrics:

- Code coverage percentage tracking
- Feature coverage validation
- Requirement traceability matrix
- Defect density measurements
- Test execution success rates

Performance Benchmarks:

- Page load time targets (< 3 seconds)
- Database query performance thresholds
- Concurrent user capacity validation
- System resource utilization monitoring
- Error rate acceptability limits (< 0.1%)

13. Conclusion & Future Enhancements

The Contract Monthly Claim System represents a comprehensive solution to the challenges faced in manual claim processing for educational institutions. The system successfully addresses the key requirements of efficiency, security, transparency, and usability while providing a scalable foundation for future enhancements.

13.1 Project Achievements

Technical Accomplishments:

- Modern, scalable architecture implementation
- Comprehensive security framework with industry-standard practices
- Intuitive user interface design with accessibility considerations
- Robust database design with optimization for performance
- Complete audit trail and compliance capability

Business Value Delivered:

- Significant reduction in claim processing time
- Enhanced accuracy through automated validation
- Improved transparency with real-time status tracking
- Reduced administrative overhead for all stakeholders
- Complete digitization of manual processes

13.2 Lessons Learned

Technical Insights:

- Early database design investment pays significant dividends
- Security considerations must be integrated from the beginning
- User feedback throughout development improves final acceptance
- Comprehensive testing strategy prevents major issues post-deployment
- Documentation quality directly impacts maintenance efficiency

Project Management Insights:

- Clear requirements definition prevents scope creep
- Regular stakeholder communication ensures alignment
- Risk identification and mitigation planning proves essential
- Agile methodology provides flexibility for changing requirements
- Time allocation for testing and refinement must be adequate

13.3 Future Enhancement Opportunities

Phase 2 Enhancements (6-month timeframe):

- Mobile application development for iOS and Android platforms
- Advanced analytics and reporting dashboard with data visualization
- Integration with existing HR and payroll systems
- Automated payment processing integration
- Multi-language support for international lecturers

Phase 3 Enhancements (12-month timeframe):

- Machine learning integration for fraud detection and pattern analysis
- Advanced workflow customization for different institution types
- Cloud deployment with multi-tenant architecture
- API development for third-party system integration
- Advanced document management with version control

Long-term Vision (24-month timeframe):

- Blockchain integration for immutable audit trails
- Artificial intelligence for predictive analytics and budgeting
- IoT integration for automated time tracking
- Advanced business intelligence and dashboard capabilities
- Microservices architecture for enhanced scalability

The Contract Monthly Claim System establishes a solid foundation for modern educational administration, providing immediate value while positioning the institution for future technological advancement and operational excellence.

14. References

Academic and Professional Sources:

Delaney, K., Beauchemin, B., Cunningham, C., Fahrenkrug, J., Kundu, S., & Randal, P. (2012). *Programming Microsoft SQL Server 2012*. Microsoft Press.

Online Resources:

GeeksforGeeks. (2024). *Software Engineering and Database Management*. Retrieved from https://www.geeksforgeeks.org/

W3Schools. (2024). *Web Development Tutorials and References*. Retrieved from https://www.w3schools.com/

Artificial Intelligence Assistance:

ChatGPT. (2025, September 6). *Brainstorming and conceptual development assistance for Contract Monthly Claim System architecture and design decisions*. OpenAI. Used for idea generation and technical concept exploration.

Additional Technical References:

Microsoft Documentation. (2024). *ASP.NET Core Documentation*. Microsoft Corporation.

Mozilla Developer Network. (2024). *Web APIs and JavaScript Reference*. Mozilla Foundation.

International Organization for Standardization. (2013). *ISO/IEC 27001:2013 Information Security Management*. ISO.

Note: This document represents comprehensive planning and design documentation for the Contract Monthly Claim System. All technical decisions and implementations follow industry best practices and educational standards as outlined in the referenced materials. The system design prioritizes security, usability, and maintainability to ensure long-term success and user satisfaction.