Hajee Mohammad Danesh Science and Technology University Dinajpur-5200

Course Code: CSE 305

Course title: Software Engineering

Project title

University Management System with Agile SDLC Model

Submitted By:

Name: Khandakar Istiak Muntasir

Student ID: 21020010

Session: 2021

Level: 03 **Semester: I**

Department: Computer Science and Engineering

Submitted To:

Pankaj Bhowmik

Lecturer

Department of Computer Science and Engineering

Hajee Mohammad Danesh Science and Technology University

Date of Submission: November 24, 2024

Abstract

The University Management System (UMS) is a comprehensive solution aimed at automating and simplifying various academic and administrative processes within a university, eliminating the need for paper-based operations. This project focuses on providing an efficient platform for students, teachers, and administrators to manage academic records, assignments, attendance, exam schedules, and other essential tasks in a digital format.The system allows students to track their attendance, quiz and midterm marks, class routines, and academic progress, as well as manage assignments and communicate with faculty. Teachers can generate class routines, track student attendance and progress, manage assignments, and handle exam duties. Administrators have access to all student information, generate required certificates, and handle necessary paperwork, ensuring smooth operations across the university without the use of physical documents.By transitioning to a paper-free environment, the University Management System reduces manual work, enhances accuracy, and improves communication among students, faculty, and administrative staff, contributing to a more organized, efficient, and environmentally friendly university setting.

Introduction

The University Management System (UMS) is designed to address the complexities of managing a university's academic and administrative functions. As universities grow, the manual processes of handling student records, attendance, class schedules, assignments, exams, and documentation become increasingly challenging and inefficient. The UMS provides an integrated digital platform to streamline these processes, ensuring a more efficient, transparent, and paper-free university environment.The system is tailored to meet the needs of students, faculty, and administrators by offering intuitive dashboards, seamless communication tools, and automated workflows. Students can access their academic progress, track assignments, and receive timely notifications, while teachers can manage attendance, assign tasks, and generate class routines. Administrators can easily manage student information, handle academic paperwork, and ensure smooth university operations.

Objective:

The objective of this project is to develop a user-friendly, paper-free University Management System that simplifies the management of academic and administrative activities. The system aims to:

1. *Automate Academic Tasks*: Streamline attendance tracking, assignment management, and exam scheduling.
2. *Enhance Communication:* Facilitate communication between students, teachers, and administrative staff through digital platforms.
3. *Improve Data Management*: Provide easy access to student records and academic data, eliminating the need for paper-based processes.
4. *Increase Efficiency*: Reduce manual effort and ensure timely notifications for students and faculty.
5. *Promote Paper-Free Operations*: Transition the university to a completely digital environment, reducing the reliance on physical documents.

This system will provide a centralized platform for managing all university-related activities, fostering a more organized, efficient, and eco-friendly university environment.

SDLC model

The Agile model was selected for the development of the University Management System (UMS) due to its dynamic nature and iterative approach, which aligns perfectly with the project's needs. The system requires frequent interaction with the database to track real-time data, such as attendance, assignment submissions, grades, and student records. Given the need for ongoing updates and the likelihood of changes during development based on user feedback, the Agile model offers significant advantages:

1. Dynamic Database Communication: The UMS will involve continuous updates to the database (e.g., real-time attendance tracking, assignment submissions, grade updates). The Agile model allows for flexible and adaptive development, ensuring that database interactions can be refined and tested as the system evolves.
2. Frequent Feedback and Iteration: Agile emphasizes collaboration with stakeholders (students, teachers, and administrators). This ensures that the system is developed according to real user needs, and adjustments can be made quickly. For example, as new features like assignment management or exam scheduling are developed, feedback can be gathered in each sprint to improve database performance and system functionality.

3. Flexibility to Change: In a dynamic academic environment, requirements can evolve quickly. Agile allows the project to adapt to changing requirements, whether it’s the need for additional features, adjustments in the database schema, or updates to existing functionality, ensuring the system remains relevant and responsive.

4. Faster Time-to-Market: With Agile, the system can be released in phases, starting with core features such as student attendance or class routines. This incremental release allows the university to begin using parts of the system sooner while additional features (e.g., exam management, document generation) are being developed in parallel.

Agile Model Description for UMS

The Agile model for this project follows an iterative development cycle consisting of several sprints, each focusing on a specific aspect of the system. Here's how the Agile model will be applied:

1. **Sprint Planning**: The project is broken down into smaller tasks, such as developing student dashboards, teacher functionalities, and administrative features. Each sprint will focus on completing specific features and integrating them with the database to ensure dynamic communication.

2. **Sprint Execution**: Development occurs in short, time-boxed iterations (usually 2-4 weeks). During each sprint, the team will focus on implementing and testing features like student attendance tracking, assignment submissions, and real-time updates for grades. The features will be built incrementally, starting with core functionality and adding complexity over time.

3. **Continuous Database Integration**: As database communication is central to the system’s functionality, database interactions will be integrated and tested continuously throughout the development process. For instance, as features such as attendance or grade tracking are developed, they will be integrated with the backend database, ensuring real-time data updates and seamless user experiences.

4. **Daily Standups and Collaboration**: The project team will hold daily standup meetings to ensure clear communication and quick issue resolution. This promotes teamwork and allows for fast problem-solving when issues arise with database queries or feature integration.

5. **User Feedback and Testing**: At the end of each sprint, the system's features will be tested and reviewed by stakeholders (students, teachers, and administrators). Feedback will be gathered to refine features, improve database handling, and ensure that the system meets the end users' needs.
6. **Iteration and Refinement**: After gathering feedback, adjustments are made, and features are improved in subsequent sprints. For example, if database queries are not optimized or if users encounter issues with real-time data updates, these can be addressed in the next sprint.
7. **Release and Deployment**: As the system progresses through sprints, parts of the system can be deployed and used by university staff. By using Agile, the university can begin using functional portions of the system early on, such as the student dashboard for attendance and class routines, while other features are still being developed.

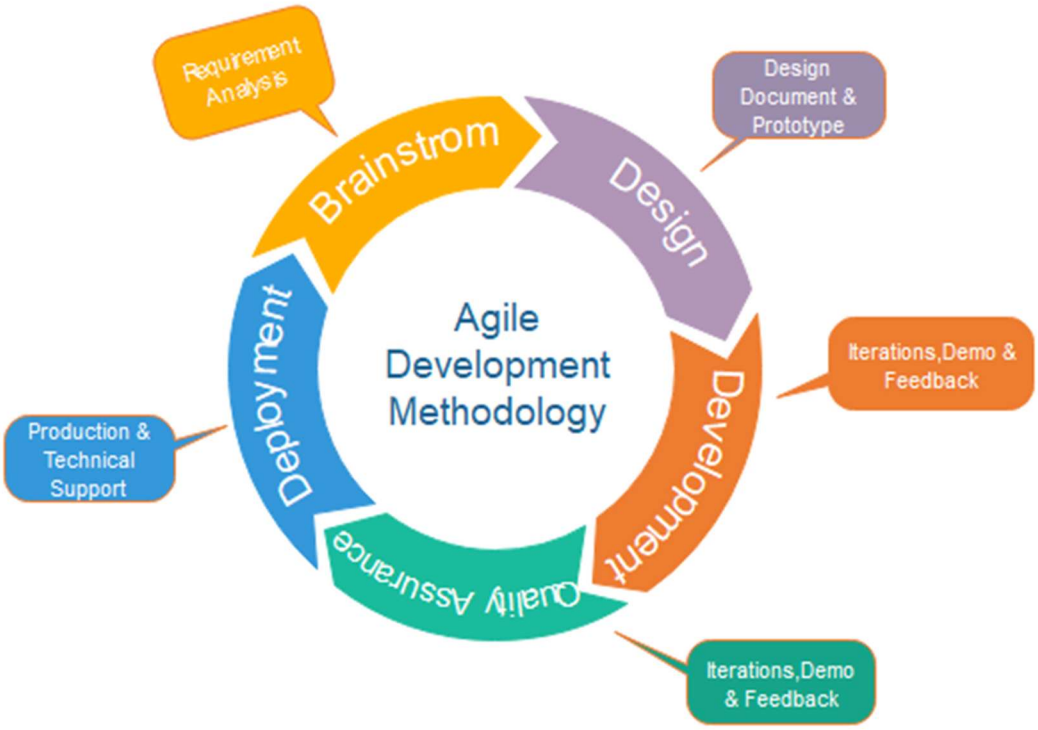

Fig. Agile Model

Software Requirements Specification (SRS)

1. Introduction

1.1 Purpose

The purpose of this document is to provide a detailed description of the software requirements for the University Management System (UMS). This system aims to automate and streamline various academic and administrative processes within a university, making them more efficient and paper-free. This document defines the functional and non-functional requirements for the system, which will serve as a guide for developers, testers, and stakeholders throughout the project.

1.2 Scope

The University Management System (UMS) is a web-based application designed to manage the university's academic, administrative, library, and hall sectors. The system will provide students, teachers, and administrators with interactive dashboards, real-time updates on attendance, grades, assignments, class routines, and exam schedules. Additionally, it will allow for communication between students and teachers, manage library book checkouts, and handle administrative processes such as document generation.

1.3 Definitions, Acronyms, and Abbreviations

- UMS: University Management System
- SRS: Software Requirements Specification
- Student Dashboard: The interface where students can view their academic information.
- Teacher Dashboard: The interface where teachers manage attendance, assignments, and exams.
- Admin Dashboard: The interface where university administrators manage student records, issue certificates, etc.
- Hall Sector: The part of the system that manages student accommodation and seat assignments.

- Library Sector: The part of the system that tracks student book checkouts and returns.

1.4 References

- Project proposal and design documents
- University policies on student data handling and privacy

2. Overall Description

2.1 Product Perspective

The University Management System is an integrated solution that will automate academic tasks, manage attendance, handle assignments, track grades, and provide communication between students and faculty. It is designed to replace paper-based systems with a digital platform, reducing administrative overhead and ensuring a more efficient workflow. The system will include a centralized database and will be accessible by students, faculty, and administrative staff.

2.2 Product Functions

The key features of the University Management System are as follows:

- Student Dashboard: View academic information, attendance, assignments, grades, and results.
- Teacher Dashboard: Manage student attendance, assignments, generate class routines, track student progress, and communicate with students.
- Admin Dashboard: Manage student records, issue certificates, handle university documentation, and maintain data.
- Library Management: Track books issued to students, manage return dates, and notify students about overdue books.
- Hall Management: Assign and track students' seat numbers for accommodation based on their information from the administrative sector.

2.3 User Characteristics

- Students: Primary users who will interact with the dashboard to manage their academic records, assignments, and communicate with teachers.
- Teachers: Users who will input and track student attendance, assignments, exams, and provide feedback to students.
- Administrators: Users who will manage the university's data, issue certificates, and handle university-related documentation.

2.4 Operating Environment

The system will be web-based, accessible through modern web browsers (Chrome, Firefox, Safari). It will be hosted on a university server or cloud infrastructure with the following requirements:

- Web server: Apache/Nginx
- Database: MySQL or PostgreSQL
- Languages/Frameworks: JavaFX (for desktop components), HTML/CSS, JavaScript (for web interface), and PHP/Python (for backend logic)

3. System Features

3.1 Student Dashboard

- Description: The Student Dashboard will allow students to view their academic progress, including attendance, assignment submissions, quiz and midterm marks, and final results.
- Functional Requirements:
    1. Students will be able to view their semester-wise class routine.
    2. Students can track their attendance and assignment submissions.
    3. The system will automatically calculate and display quiz and midterm marks.
    4. Students will have access to their results once published.

5. Students can apply for administrative documents (e.g., certificates) and receive a signed PDF document.

3.2 Teacher Dashboard

- Description: The Teacher Dashboard will provide teachers with tools to manage student attendance, assignments, and exams, as well as communicate with students.
- Functional Requirements:
    1. Teachers will be able to record and track student attendance.
    2. Teachers can create and assign assignments to students and track their submissions.
    3. The system will allow teachers to manage and grade student quizzes and midterm exams.
    4. Teachers can generate and share class routines.
    5. Teachers will have access to a messaging feature for communication with students.

3.3 Admin Dashboard

- Description: The Admin Dashboard will enable university staff to manage student data, generate certificates, and handle administrative tasks.
- Functional Requirements:
    1. Administrators will be able to view and update student records.
    2. The system will allow administrators to issue certificates, transcripts, and other necessary documents.
    3. Administrators will manage user roles and permissions.
    4. The system will allow for real-time updates of administrative tasks and documents.

3.4 Library Management

- Description: The Library Management feature will track books issued to students and manage their return dates.
- Functional Requirements:
    1. The system will track which books are issued to which students.
    2. It will monitor how long a student has had a book and notify them if it is due for return.
    3. Notifications will be sent to students when books need to be returned within 20 days.
    4. Admins can view current book status and availability.

3.5 Hall Management

- Description: The Hall Management feature will manage the seating allocation of students based on data provided by the administrative sector.
- Functional Requirements:
    1. The system will assign seat numbers to students upon their admission.
    2. The Hall sector will fetch student information from the administrative sector for seating allocation.

4. External Interface Requirements

4.1 User Interfaces

The user interfaces will be designed to be simple, intuitive, and easy to navigate. Each dashboard (Student, Teacher, Admin) will have a separate login page, with a unique set of permissions. The design will include:

- Responsive UI design for both web and mobile platforms.
- Access to all key features from a single page/dashboard.
- A common notification system for updates (assignments, attendance, results).

4.2 Hardware Interfaces

- The system will be hosted on university servers or a cloud infrastructure, with adequate resources to handle multiple concurrent users.
- End-users (students, faculty, admins) will access the system using desktops, laptops, or mobile devices with modern web browsers.

4.3 Software Interfaces

- The system will interface with a relational database (e.g., MySQL or PostgreSQL) for storing user and academic data.
- The application will integrate with email systems to send notifications about assignments, results, and library book due dates.

5. Non-Functional Requirements

5.1 Performance Requirements

- The system should be capable of handling concurrent user requests from hundreds of students, teachers, and administrative staff without performance degradation.
- Response time for accessing dashboard data (e.g., attendance, assignments) should not exceed 5 seconds.

5.2 Security Requirements

- The system must implement secure authentication and authorization mechanisms for students, teachers, and administrators.
- User data must be encrypted, and sensitive information (such as grades, student records) must be stored securely.
- The system should provide audit logs for administrative actions like document generation and record updates.

5.3 Usability Requirements

- The system should be user-friendly and accessible to non-technical users, with clear instructions and error handling.

- Training materials and help documentation should be available for users.

5.4 Scalability

- The system should be designed to scale as the number of users grows, with the ability to support additional users, courses, and data storage.

6. SDLC Phases for University Management System

The development of the University Management System (UMS) will be based on the Agile SDLC model, where the system will be built iteratively, with regular feedback from users and stakeholders. Below is the detailed breakdown of each SDLC phase:

6.1 Requirement Gathering & Analysis

- Objective: To gather all functional and non-functional requirements from stakeholders (students, teachers, administrators, and library staff).
- Activities:
    - Conduct meetings with stakeholders to understand their requirements.
    - Create user stories for key features such as attendance tracking, class routine generation, assignment management, and certificate issuing.
    - Prioritize features and prepare for the first sprint.
- Deliverables:
    - SRS Document with complete functional and non-functional requirements.
    - User Stories for system features.

6.2 Design

- Objective: To design the system architecture, database schema, and user interfaces.
- Activities:
    - Define the database structure for students, courses, grades, attendance, and assignments.
    - Design the UI/UX for each dashboard (Student, Teacher, Admin).
    - Plan API integration for notifications and external system interactions.

- Deliverables:
    - System Design Document, including UI wireframes, database schema, and system architecture.
    - Prototypes of key system interfaces.

6.3 Implementation (Development)

- Objective: Develop the system iteratively in sprints, starting with core features.
- Activities:
    - Implement front-end and back-end functionality.
    - Develop the initial version of the Student and Teacher dashboards.
    - Set up database integration and ensure real-time updates for attendance and grades.
    - Conduct unit testing for each feature.
- Deliverables:
    - Working Codebase: Each sprint will deliver a working module (e.g., Student Dashboard, Teacher Dashboard).

6.4 Testing

- Objective: To ensure the system functions correctly and meets all requirements.
- Activities:
    - Perform functional testing (unit testing, integration testing, and system testing).
    - Conduct user acceptance testing (UAT) to validate features against user stories.
    - Perform load testing to ensure the system can handle high user traffic.
- Deliverables:
    - Test Reports documenting issues and resolutions.
    - Bug Fixes applied to the system based on feedback.

6.5 Deployment

- Objective: To deploy the system in a live production environment.
- Activities:
    - Deploy the system on university servers or cloud infrastructure.
    - Conduct final checks and integration testing.
    - Make the system accessible to students, teachers, and administrators.
- Deliverables:
    - Live System: The final working version of the system.
    - Deployment Report: Documenting the deployment process and any challenges faced.

6.6 Maintenance & Support

- Objective: To maintain and improve the system after deployment.
- Activities:
    - Monitor the system for issues and user feedback.
    - Apply patches and updates to the system.
    - Provide support for any technical problems encountered by users.
- Deliverables:
    - Maintenance Logs and User Support Documentation.

7. Conclusion

This Software Requirements Specification (SRS) document defines the requirements for the University Management System (UMS). It includes detailed functional and non-functional requirements, ensuring the system meets the needs of students, teachers, and administrative staff, while also supporting a paper-free, efficient university environment.