**Exam date Sheet Scheduler**

**Software Requirements Specification**

Version 1.0



**Group Id: S2402FBB6E (BC210208098)**

**Supervisor Name:** **Dr. Hasnain Ahmed**

**Revision History**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date (dd/mm/yyyy)** | **Version** | **Description** | **Author** |
| 10/05/2024 | 1.0 | My project is **“Exam Date Sheet Scheduler”.**  The "Exam Date Sheet Scheduler" is a web application that automates exam scheduling in educational institutes, managing courses, examination halls, and superintendents, while ensuring fair and conflict-free schedules. It includes user management, reporting, scalability, security, and integration features. | BC210208098 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**Table of Contents**

[Scope of Project 4](#_Toc167094500)

[Functional Requirements Non Functional Requirements 5](#_Toc167094501)

[Use Case Diagram 7](#_Toc167094502)

[Usage Scenarios 8](#_Toc167094503)

[Adopted Methodology 13](#_Toc167094504)

[Work Plan 17](#_Toc167094505)

**SRS Document**

# Scope of Project

The Exam Date Sheet Scheduler web application is designed to simplify the complex task of creating exam schedules in educational institutions. This system will handle course management, including adding, updating, and deleting courses, and managing student enrollment to ensure data integrity and accuracy. Examination hall management will include functionalities for adding, updating, and deleting halls, as well as managing their capacities and availability. Superintendent management will allow the creation, updating, and deletion of superintendent records, along with their assignment to specific halls and time slots. The automated scheduling feature will generate exam date sheets based on predefined constraints, ensuring no overlapping exams for students and optimizing schedules to minimize the number of exams per day.

User management with role-based access control will be a core aspect, providing secure login and authentication processes, along with password recovery and reset functionalities. The application will feature intuitive user interfaces tailored for admins, students, superintendents, and exam coordinators, allowing them to view and manage schedules, courses, and halls efficiently. Reporting capabilities will include generating exam schedules, hall usage, and superintendent assignments, accompanied by notifications for schedule updates and important information. This project aims to enhance fairness, efficiency, and resource optimization in exam scheduling, adhering to data privacy and security regulations while being scalable for future growth.

# Functional Requirements Non Functional Requirements

**Functional Requirements:**

* **Course Management**:
  + Add, update, and delete courses.
  + Record and manage student enrollment in each course.
* **Examination Hall Management**:
  + Add, update, and delete examination halls.
  + Record and manage the capacity of each examination hall.
* **Superintendent Management**:
  + Add, update, and delete superintendent records.
  + Assign superintendents to examination halls and time slots.
* **Exam Scheduling**:
  + Automatically generate an exam date sheet based on course enrollment, hall capacity, and superintendent availability.
  + Ensure no student has overlapping exams.
  + Optimize scheduling to minimize the number of exams a student has on the same day.
* **User Management**:
  + Manage user roles (admin, student, superintendent, exam coordinator).
  + Provide role-based access control to different features of the application.
* **Login Management:**
  + Provide secure login functionality for users.
  + Implement password recovery and reset functionality.
  + Ensure secure authentication and authorization processes.

Top of Form

**Non-Functional Requirement:**

A Non-Functional Requirement (NFR) defines the quality attribute of a software system. They judge the software system based on Performance, Usability, Maintainability, Portability and other non-functional standards that are critical to the success of the software system. These are also called quality attributes.

* **Performance:** Performance requirements concern the speed of operation of a system. The non-functional requirements should identify those software functions that have constraint on their performance.
* **Maintainability:** Maintainability requirements may cover diverse levels of documentation, such as system documentation, as well as test documentation, e.g. which test cases and test plans will accompany the system.
* **Usability:** It is an easy way for a user to learn to operate. It is well-formed GUI.
* **Portability:** Portability specifies the ease with which the software can be installed on all necessary platforms, and the platforms on which it is expected to run.
* **Reliability:** Reliability is the ability of a system to perform its required functions under specific period of time.
* **Time:** The project speed is very high. Every task takes low time for completion.
* **User-Friendly Interfaces:** All interfaces should have intuitive design and easy navigation, ensuring accessibility for all users (admin, student, superintendent, exam coordinator).
* **Validation Checks:** Implement validation checks across all entry, update, delete, and search interfaces to ensure data integrity and accuracy.

# Use Case Diagram



**Figure 1**

# Usage Scenarios

|  |  |
| --- | --- |
| **Use Case Title** | **Course Management** |
| **Use Case Id** | 1 |
| **Actors** | **Admin, Teacher, Student** |
| **Actions** | 1. Admin adds a new course.  2. Admin updates an existing course.  3. Admin deletes a course.  4. Admin records student enrollment in courses.  5. Student enrolls in a course.  6. Teacher views the list of enrolled students in a course. |
| **Description** | This use case describes the process of managing courses within the school management system. It includes adding new courses, updating course details, deleting courses, and managing student enrollment in each course. The system allows admins to perform these actions and provides teachers and students with relevant course information. |
| **Alternative Paths** | 1. If the course already exists when attempting to add a new course, the system notifies the admin and does not add the duplicate course.  2. If the course to be updated does not exist, the system notifies the admin and does not perform the update.  3. If the course to be deleted has enrolled students, the system prompts the admin to confirm the deletion, ensuring no accidental data loss.  4. If a student tries to enroll in a course that is full, the system notifies the student and prevents the enrollment. |
| **Pre-Conditions** | 1. The admin is logged into the system.  2. The admin has the necessary permissions to manage courses.  3. The student is logged into the system and has the necessary permissions to enroll in courses. |
| **Post Conditions** | 1. New courses are added to the system.  2. Existing courses are updated with new details.  3. Courses are deleted from the system.  4. Student enrollments are recorded and managed in the system. |
| **Author** | BC210208098 |
| **Exceptions** | 1. Database connection failure while adding, updating, or deleting a course.  2. Admin does not have the necessary permissions to perform course management actions.  3. System error while recording student enrollment. |

|  |  |
| --- | --- |
| **Use Case Title** | **Examination Hall Management** |
| **Use Case Id** | 2 |
| **Actors** | **Admin, Exam Coordinator** |
| **Actions** | 1. Admin adds a new examination hall.  2. Admin updates details of an existing examination hall.  3. Admin deletes an examination hall.  4. Admin records and manages the capacity of each examination hall.  5. Exam Coordinator views the list of examination halls and their capacities. |
| **Description** | This use case describes the process of managing examination halls within the school management system. It includes adding new examination halls, updating hall details, deleting examination halls, and managing the capacity of each hall. The system allows admins to perform these actions and provides exam coordinators with relevant examination hall information. |
| **Alternative Paths** | 1. If the examination hall already exists when attempting to add a new hall, the system notifies the admin and does not add the duplicate hall.  2. If the examination hall to be updated does not exist, the system notifies the admin and does not perform the update.  3. If the examination hall to be deleted has scheduled exams, the system prompts the admin to confirm the deletion, ensuring no accidental data loss. |
| **Pre-Conditions** | 1. The admin is logged into the system.  2. The admin has the necessary permissions to manage examination halls.  3. The exam coordinator is logged into the system. |
| **Post Conditions** | 1. New examination halls are added to the system.  2. Existing examination halls are updated with new details.  3. Examination halls are deleted from the system.  4. Examination hall capacities are recorded and managed in the system. |
| **Author** | BC210208098 |
| **Exceptions** | 1. Database connection failure while adding, updating, or deleting an examination hall.  2. Admin does not have the necessary permissions to perform examination hall management actions.  3. System error while recording or managing the capacity of examination halls. |

|  |  |
| --- | --- |
| **Use Case Title** | Superintendent Management |
| **Use Case Id** | 3 |
| **Actors** | Admin, Exam Coordinator |
| **Actions** | 1. Admin adds a new superintendent record.  2. Admin updates details of an existing superintendent.  3. Admin deletes a superintendent record.  4. Exam Coordinator assigns superintendents to examination halls.  5. Exam Coordinator assigns superintendents to time slots. |
| **Description** | This use case describes the process of managing superintendents within the school management system. It includes adding new superintendent records, updating superintendent details, deleting superintendent records, and assigning superintendents to examination halls and time slots. The system allows admins and exam coordinators to perform these actions to ensure proper examination management. |
| **Alternative Paths** | 1. If the superintendent record already exists when attempting to add a new record, the system notifies the admin and does not add the duplicate record.  2. If the superintendent record to be updated does not exist, the system notifies the admin and does not perform the update.  3. If the superintendent record to be deleted is currently assigned to an examination hall or time slot, the system prompts the admin to confirm the deletion, ensuring no accidental data loss. |
| **Pre-Conditions** | 1. The admin is logged into the system.  2. The admin has the necessary permissions to manage superintendent records.  3. The exam coordinator is logged into the system.  4. The exam coordinator has the necessary permissions to assign superintendents. |
| **Post Conditions** | 1. New superintendent records are added to the system.  2. Existing superintendent records are updated with new details.  3. Superintendent records are deleted from the system.  4. Superintendents are assigned to specific examination halls.  5. Superintendents are assigned to specific time slots. |
| **Author** | BC210208098 |
| **Exceptions** | 1. Database connection failure while adding, updating, or deleting a superintendent record.  2. Admin or exam coordinator does not have the necessary permissions to perform superintendent management actions.  3. System error while assigning superintendents to examination halls or time slots. |

|  |  |
| --- | --- |
| **Use Case Title** | **Exam Scheduling** |
| **Use Case Id** | 4 |
| **Actors** | **Admin, Exam Coordinator** |
| **Actions** | 1. System automatically generates an exam date sheet.  2. System ensures no student has overlapping exams.  3. System optimizes the schedule to minimize the number of exams a student has on the same day.  4. Exam Coordinator reviews and finalizes the generated exam date sheet.  5. Exam Coordinator publishes the final exam date sheet. |
| **Description** | This use case describes the process of automatically generating an exam date sheet based on course enrollment, hall capacity, and superintendent availability. The system ensures that no student has overlapping exams and optimizes the schedule to minimize the number of exams a student has on the same day. The generated schedule is reviewed and finalized by the exam coordinator. |
| **Alternative Paths** | 1. If the system cannot generate a conflict-free schedule, it prompts the exam coordinator to manually resolve conflicts.  2. If a superintendent becomes unavailable after scheduling, the system prompts the exam coordinator to reassign the affected exams. |
| **Pre-Conditions** | 1. Course enrollment data is available and up-to-date.  2. Examination hall capacities are recorded in the system.  3. Superintendent availability is recorded in the system.  4. The exam coordinator is logged into the system. |
| **Post Conditions** | 1. An exam date sheet is automatically generated.  2. Students do not have overlapping exams.  3. The exam schedule minimizes the number of exams a student has on the same day  4. The final exam date sheet is reviewed and published by the exam coordinator. |
| **Author** | BC210208098 |
| **Exceptions** | 1. Database connection failure during the generation of the exam date sheet.  2. Incomplete or incorrect data on course enrollment, hall capacity, or superintendent availability.  3. System errors that prevent automatic scheduling. |

|  |  |
| --- | --- |
| **Use Case Title** | **User Management** |
| **Use Case Id** | 5 |
| **Actors** | **Admin** |
| **Actions** | 1. Admin adds a new user and assigns a role.  2. Admin updates user details and roles.  3. Admin deletes a user.  4. System provides role-based access control to different features of the application. |
| **Description** | This use case describes the process of managing user roles within the school management system. It includes adding new users, updating user details and roles, deleting users, and providing role-based access control to ensure that users can only access features relevant to their roles (admin, student, superintendent, exam coordinator). |
| **Alternative Paths** | 1. If the user to be added already exists, the system notifies the admin and does not add the duplicate user.  2. If the user to be updated does not exist, the system notifies the admin and does not perform the update.  3. If the user to be deleted is currently active in an ongoing process (e.g., an ongoing exam for a superintendent), the system prompts the admin to confirm the deletion, ensuring no disruption. |
| **Pre-Conditions** | 1. The admin is logged into the system.  2. The admin has the necessary permissions to manage users. |
| **Post Conditions** | 1. New users are added to the system with appropriate roles.  2. User details and roles are updated.  3. Users are deleted from the system.  4. Users have access to features based on their assigned roles. |
| **Author** | BC210208098 |
| **Exceptions** | 1. Database connection failure while adding, updating, or deleting a user.  2. Admin does not have the necessary permissions to perform user management actions.  3. System errors that prevent role-based access control from functioning correctly. |

|  |  |
| --- | --- |
| **Use Case Title** | **Login Management** |
| **Use Case Id** | 6 |
| **Actors** | **User (Admin, Student, Superintendent, Exam Coordinator)** |
| **Actions** | 1. User provides credentials to log into the system.  2. System authenticates user credentials.  3. System authorizes user access based on role.  4. User requests password recovery.  5. System sends password recovery instructions.  6. User resets password using the provided instructions.  7. System confirms password reset and allows user to log in. |
| **Description** | This use case describes the process of providing secure login functionality for users, implementing password recovery and reset functionality, and ensuring secure authentication and authorization processes within the school management system. |
| **Alternative Paths** | 1. If login credentials are incorrect, the system notifies the user and prompts for re-entry.  2. If a user fails to authenticate after multiple attempts, the system temporarily locks the account and notifies the user.  3. If the password recovery request is invalid or the email does not exist, the system notifies the user and does not send recovery instructions. |
| **Pre-Conditions** | 1. User has an existing account in the system.  2. The system has access to the user's registered email for password recovery. |
| **Post Conditions** | 1. Users are securely logged into the system.  2. Password recovery instructions are sent to the user.  3. Users are able to reset their passwords securely.  4. User access is authorized based on their role. |
| **Author** | BC210208098 |
| **Exceptions** | 1. Database connection failure during the login process.  2. System error during authentication or authorization.  3. System error while sending password recovery instructions. |

# Adopted Methodology

Water Fall Model and Spiral Model are two models which are combined to each other and make VU Process Model. It is also called hybrid approach of system development. This Process model takes advantages and minimizes the drawbacks of both approaches. This is a linear model to solve many problems.

**REASONS FOR CHOSEN METHODOLOGY**

Following are the basic reasons for choosing VU Process Model for our project.

1. Testing is inherent to every phase
2. It is an enforced disciplined approach because to having stage development cycle and every phase has defined start and end date.
3. It is documentation driven which means documentation is produced at every stage.
4. Emphasis on requirement and design before writing code ensure minimal wastage of time and efforts.
5. Our project is divided into different phases which we have to complete in sequence and submit to our supervisor.
6. If some mistakes are found than he suggests us to correct those mistakes and improve our project.
7. This process will be adopted due to its spiral nature.

**VU Process Model**

VU PROCESS MODEL The adopted methodology is VU Process Model which is a combination of Waterfall and Spiral Model.



**Figure 2**

**STRUCTURE OF VU PROCESS MODEL**

It has four phases and which are Gathering and analyzing requirement, planning, analysis and design, development and final report

**1. Gathering & Analyzing Requirement:** In this phase we gathered information regarding functional and non- functional requirements of our Project and also prepared Use case diagrams to understand Actors and Uses cases of project and develop usages scenario diagrams to understand all those situations which will come across while using this application

**2. Planning Phase:** In this phase we adopted methodology to complete the project different Phase also prepared the Schedule using Gant chart to know which phase will be completed in how many days.

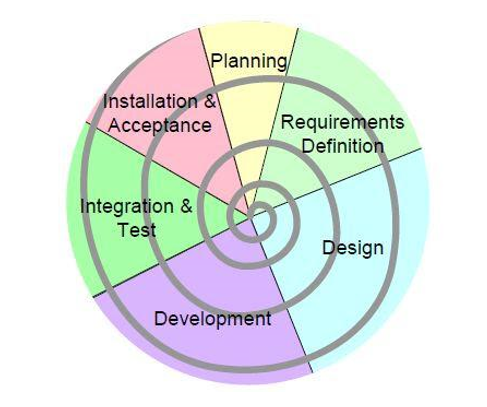
**3. Analysis and Design Phase:** In this phase consist of two design documents. In Design document 1, we will develop the Activity Diagram, Sequence diagrams, Collaboration Diagram, Architecture Design Diagram and DFD diagram. And Design document II we develop the Object model, Class diagram, interface design, database design, test cases, Deployment diagram and Component diagram.

**4. Development & Final Project Report:** In this phase we will write necessary code for implementation of our application. Coding will be completed in iterations.

**Spiral Model:**

The spiral model is a risk-driven [process model](https://en.wikipedia.org/wiki/Software_development_process) generator for software projects. Based on the unique risk patterns of a given project, the spiral model guides a team to adopt elements of one or more process models, such as  [waterfall](https://en.wikipedia.org/wiki/Waterfall_model). It combines the idea of iterative development with the systematic, controlled aspects of the waterfall model. It is a combination of iterative development process model and sequential linear development model i.e. the waterfall model with a very high emphasis on risk analysis. It allows incremental releases of the product or incremental refinement through each iteration around the spiral.

**Diagram of Spiral Model**



**Figure 3**

**Waterfall Model**

The Waterfall Model was the first Process Model to be introduced. It is also referred to as a linear-sequential life cycle model. It illustrates the software development process in a linear sequential flow. It is very simple to understand and use. In a waterfall model, each phase must be completed before the next phase can begin and there is no overlapping in the phases. The Waterfall model is the earliest SDLC approach that was used for software development.

This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap.

The waterfall model is a [sequential](https://en.wikipedia.org/wiki/Sequence) (non-iterative) [design](https://en.wikipedia.org/wiki/Design) process, used in [software development processes](https://en.wikipedia.org/wiki/Software_development_process), in which progress is seen as flowing steadily downwards (like a [waterfall](https://en.wikipedia.org/wiki/Waterfall)) through the phases of conception, initiation, [analysis](https://en.wikipedia.org/wiki/Analysis), [design](https://en.wikipedia.org/wiki/Software_design), construction, [testing](https://en.wikipedia.org/wiki/Software_testing), [production/implementation](https://en.wikipedia.org/wiki/Implementation) and [maintenance](https://en.wikipedia.org/wiki/Software_maintenance). Despite the development of new software development process models, the Waterfall method is still the dominant process model with over a third of software developers still using it.

This model is depicted in the following diagram.



**Figure 4**

# Work Plan

The work plan is a document that consulting firms use to organize a project. It outlines the plan by which the company plans to complete a quality project within a given amount of time and in compliance with a set budget.

Gantt charts are useful tools for planning and scheduling projects. They allow you to assess how long a project should take, determine the resources needed and layout the order in which tasks need to be carried out. They are useful in managing the dependencies between tasks.



**Figure 5**