[Type the document title]

[Type the document subtitle]



**LOGO** - [ Company Name ]

CHAPTER ONE

# 1 . Requirement Analysis (Use Cases)

**Requirement Analysis** is the process of identifying, gathering, analyzing, and documenting the needs and expectations of stakeholders for a new or modified system. It involves understanding, documenting, and managing the needs and expectations of stakeholders.

As 3rd-year Software Engineering students at Wachemo University, we began our Online Examination System project by analyzing system requirements based on stakeholder expectations including students, instructors, and administrators.We conducted our requirement analysis through interviews with department head staff, document analysis, and observation.These methods helped us gather real-world insights into the system’s requirements, user behaviors, and existing manual processes.

Functional Requirements

1. **Login**: Users (Admin, Instructor, and Student) must authenticate using login credentials to access the system.

2. **Logou**t: Allows users to safely exit their session, ensuring data protection and access control.

3. **Update Password**: Enables all users to change their passwords for enhanced account security.

4.**View Admin Dashboard**: Admin can view a centralized dashboard showing system status and management options.

5. **Assign Instructors**: Admin assigns instructors to specific roles or courses.

6. **Assign Students**: Admin enrolls students into the system or into specific

courses.

7. **Manage Instructors**: Admin adds, updates, or deletes instructor profiles.

8.**Manage Students**: Admin handles student account creation, editing, or deletion.

9**.Add Course**: Admin can create new courses for exams to be associated with.

10.**Manage Courses**: Admin edits or deletes existing course information.

11. **Schedule Exam**: Admin schedules new exams by setting time, date, and assigning to courses.

12. **Manage Schedule**: Admin updates or removes scheduled exams.

13.**All Feedbacks**: Admin reviews all feedback given by students.

14. V**iew Instructor Dashboard**: Instructor accesses a personalized interface with teaching and exam tools.

15.**Create Exam**: Instructor can create new exams for assigned courses.

16. **Import Exam**: Instructor imports exam content from external sources or previous versions.

17. **Edit Exam**: Instructor modifies exam details and questions.

18.**Delete Exam**: Instructor deletes an unwanted or outdated exam.

19**.Manage Exam**: Broad functionality for instructors to control exam creation, editing, and publishing.

21. **Generate Exam Report**: Instructor generates analytical reports on performance.

22. **Feedbacks**: Instructor can view and respond to feedback submitted by students.

23. **View Student Result**: Instructor checks student results for performance evaluation.

24. **Available Exam**: Student views exams currently open for participation.

25. **View Student Dashboard**: Student views a summary of available exams, results, and feedback options.

26. **View Exam Schedule**: Student checks upcoming exam dates and times.

27.**Taken Exams**: Student can view a history of completed exams.

28.**View Result**: Student sees the results of completed exams.

29.**Add Feedback**: Student submits feedback after viewing their exam results.

# Non-Functional Requirements

1. **Security:** The system should ensure only authorized users can access specific features, protecting sensitive data.

2. **Performance**: The application should load and respond quickly, even under peak usage conditions.

3. **Usability**: The user interface should be simple and intuitive, requiring minimal learning.

4.**Reliability**: The system must operate consistently, handling errors gracefully and maintaining uptime.

5. **Scalability:** The system should support more users and courses as the university grows.

6. **Maintainability**: Code and system components should be modular and easy to update or fix.

# 4. Use Case Diagram Components

**A use case diagram** is a high-level visual representation of user interactions with a system. Its core components include:

• **Actors:** Represent the external users or systems that interact with the system (e.g., Admin, Instructor, Student).

• **Use Cases:** These are the functionalities provided by the system, represented as ovals with actions inside.

• **System Boundary**: A rectangular box that shows the scope of the system; all use cases fall within this boundary.

• **Relationships**: Connections between actors and use cases, or between use cases themselves. Includes:

* **Association**: A straight line showing direct interaction between an actor and a use case.
* **Include**: A dashed arrow showing that a use case always uses another use case as part of its behavior.
* **Extend**: A dashed arrow indicating optional or conditional behavior added to a base use case.
* **Generalization**: An arrow representing inheritance between actors or use cases (e.g., specialized roles).

