1. Stakeholder Analysis

Key Stakeholders:

1. **Students**:
    - **Needs**: Ability to browse courses, enroll, take quizzes, and track progress.
    - **Expectations**: Simple and intuitive interface, fast performance, and reliable system.
2. **Admins**:
    - **Needs**: Ability to manage courses, users, and quizzes.
    - **Expectations**: Easy-to-use admin panel, secure access, and efficient management tools.
3. **Developers**:
    - **Needs**: Clear requirements, scalable architecture, and maintainable code.
    - **Expectations**: Well-defined scope, realistic timelines, and minimal scope creep.
4. **Project Manager**:
    - **Needs**: Clear milestones, deliverables, and risk management.
    - **Expectations**: On-time delivery, and meeting stakeholder expectations.

2. User Stories & Use Cases

User stories and use cases describe how users will interact with the system. They help in understanding the system's functionality from the user's perspective.

User Stories:

1. **As a Student**, I want to **browse available courses** so that I can find courses I'm interested in.
2. **As a Student**, I want to **enroll in a course** so that I can start learning.
3. **As a Student**, I want to **take quizzes** for the courses I'm enrolled in so that I can test my knowledge.
4. **As a Student**, I want to **view my progress** (e.g., completed quizzes, scores) so that I can track my learning.
5. **As an Admin**, I want to **add, edit, or delete courses** so that I can manage the course catalog.
6. **As an Admin**, I want to **manage users** (e.g., add, remove, assign roles) so that I can control access to the system.

Use Cases:

1. **Browse Courses**:
    - **Actor**: Student
    - **Description**: The student views a list of available courses with details like title, description, and price.
    - **Precondition**: The student is logged in.
    - **Postcondition**: The student can select a course to enroll in.
2. **Enroll in a Course**:
    - **Actor**: Student
    - **Description**: The student selects a course and clicks "Enroll" to add it to their profile.
    - **Precondition**: The student is logged in and the course is available.
    - **Postcondition**: The course is added to the student's enrolled courses.
3. **Take a Quiz**:
    - **Actor**: Student
    - **Description**: The student takes a quiz for an enrolled course and submits answers.
    - **Precondition**: The student is enrolled in the course and the quiz is available.
    - **Postcondition**: The quiz results are saved, and the student can view their score.

4. **View Progress**:
    - **Actor**: Student
    - **Description**: The student views their progress, including completed quizzes and scores.
    - **Precondition**: The student is logged in and has enrolled in at least one course.
    - **Postcondition**: The student can see their progress.
5. **Manage Courses**:
    - **Actor**: Admin
    - **Description**: The admin adds, edits, or deletes courses in the system.
    - **Precondition**: The admin is logged in and has the necessary permissions.
    - **Postcondition**: The course catalog is updated.
6. **Manage Users**:
    - **Actor**: Admin
    - **Description**: The admin adds, removes, or assigns roles to users.
    - **Precondition**: The admin is logged in and has the necessary permissions.
    - **Postcondition**: The user list is updated.

3. Functional Requirements

Functional requirements describe the features and functionalities the system must have.

Functional Requirements:

1. **User Authentication**:
    - Users can register, log in, and log out.
    - Users have roles (Student, Admin).
2. **Course Management**:
    - Admins can add, edit, or delete courses.
    - Students can browse and enroll in courses.
3. **Quiz Management**:
    - Admins can add quizzes to courses.
    - Students can take quizzes and view results.
4. **Progress Tracking**:
    - Students can view their progress (e.g., completed quizzes, scores).
5. **Admin Panel**:
    - Admins can manage courses, quizzes, and users.

4. Non-Functional Requirements

Non-functional requirements describe the system's performance, security, usability, and reliability.

Non-Functional Requirements:

1. **Performance**:
    - The system should respond to user requests within **2 seconds**.
    - The system should handle up to **1,000 concurrent users**.
2. **Security**:
    - User passwords should be hashed using a secure algorithm (e.g., bcrypt).
    - Admin actions should require authentication and authorization.
3. **Usability**:
    - The system should have an intuitive and responsive UI.
    - The system should be accessible on both desktop and mobile devices.
4. **Reliability**:
    - The system should have **99.9% uptime**.
    - The system should automatically recover from minor errors (e.g., database connection issues).
5. **Scalability**:
    - The system should be designed to allow for future expansion (e.g., adding payment integration, forums).