Non-Functional Requirements

1 Elicitation Techniques Used

- Interviews
- Surveys
- Brainstorming among group members

1. **Performance**
   (a) **Minimize page load times** :-
   The system should load pages quickly to provide a smooth user experience.
   (b) **Deliver AI responses quickly** :-
   AI-generated responses must appear without noticeable delays for better interactivity.
   (c) **Start IDE code execution quickly** :-
   The coding environment should execute programs with minimal time.
2. **Availability**
   (a) Maintain high uptime :-
   The system should be available and accessible most of the time.
   (b) **Handle database failover** :-
   Backup databases must automatically take over in case of primary database failures.
3. **Scalability**
   (a) **Support more concurrent users without performance loss** :-
   The system should scale efficiently to handle increasing user loads.
4. **Security**
   (a) **Hash all passwords** :-
   Passwords must be stored securely using bcrypt hashing with salting
   (b) I**mplement HTTPS with auto-redirect** :-
   All communications should be encrypted and automatically redirected to secure HTTPS connections.
   (c) **Secure OAuth 2.0 logins; and make sure to never expose tokens** :-
   Authentication should use OAuth 2.0 while keeping tokens confidential.
5. **Data Validation**
   (a) **Validate data against schemas** :-
   Ensures input matches a predefined structure and prevents incorrect formats.
   (b) **Implement input rules** (fields, types, formats, length):-
   Enforces restrictions like correct field types, lengths, and valid formats.

(c) **Apply server-side validation with clear error messages**:-
Adds an extra security layer and helps users fix mistakes easily.

6. **Usability**

(a) **Ensure intuitive and user friendly design**:-
The UI will be responsive, adapting seamlessly to mobile, tablet, and desktop screens

(b) **Add real-time form validation**:-
Instantly shows errors while filling forms to save time and effort.

(c) **Keep workflows minimal (signup, setup, quizzes, etc.)**:-
Reduces unnecessary steps to make tasks quicker and smoother.

(d) **Use user-friendly, actionable error messages**:-
Provides clear guidance so users can correct errors immediately.

(e) **Support dark mode toggle**:-
Offers visual comfort and accessibility for users preferring dark themes.

7. **Maintainability**

(a) **Follow coding standards and naming conventions**:-
Improves readability and consistency across the project.

(b) **Use modular architecture for easy updates**:-
Allows independent updates and better scalability of features.

(c) **Separate and document config/env files**:-
Makes project setup clearer and reduces errors in different environments.

8. **Compliance**

(a) **Follow data protection laws (e.g., GDPR):-**
Ensures user privacy and legal compliance in handling personal data.

(b) **Store explicit user consent:-**
Records clear user approval before collecting or using their data.

(c) **Use licensed, compliant third-party content:-**
Prevents copyright issues and guarantees safe, legal use of resources.