C# Project Requirements

Examination Management System

Project Overview:
The Examination Management System is a desktop application designed to automate the exam process for organizations. The system allows users to record and manage questions, exams, and answers in a SQL Server database and generate reports efficiently.

Key Features:

User Types:

- **Students**: Login, take exams (practice or final).
- **Teachers**: Login, add questions and answers.
- **Admin**: Add students and teachers.

User Authentication:

- Users must log in with valid credentials to access the system.
- Different user roles (Admin, Teacher, Student) with corresponding permissions.

Question Recording:

- Teachers can add questions, which are stored in the SQL Server database.
- Students can answer exams, and their scores are recorded in the database.

SQL Server Database Storage:

- Question and exam data are stored in a SQL Server database.
- Student answers are securely saved and associated with their records.

Data Validation:

- Ensure recorded questions adhere to predefined constraints and relationships.
- Implement foreign key relationships to maintain data integrity.

Exam Structure:

- Design a class to represent the **Question** object (Body, Marks, Header, etc.).
- Support different question types: True/False, Choose One, Choose All.
- Implement a **Base Question** class with specific inherited classes for each type.
- Create a **Question List** class that inherits from `List<>`, overriding the `Add` method to log questions.
- Implement an **Answer List** class and associate it with Question objects.
- Define an **Exam** base class with attributes like Time, Number of Questions, and a Question-Answer Dictionary.
- Implement **Practice Exam** (shows correct answers after submission) and **Final Exam** (only shows questions and answers).

Exam Management:

- Every **Exam** object is associated with a **Subject** object.
- Implement modes for exams: **Starting, Queued, Finished**.
- Notify students when an exam starts using event handling and delegates.
- Allow users to select exam type and display accordingly.

Exam Reporting:

- Generate result reports for individual students.
- Reports can be viewed within the application or exported to external formats (PDF, Excel).

User Interface (UI):

- Clear and intuitive UI for ease of use.
- Separate interfaces for Admin, Teacher, and Student with role-specific functionalities.

Error Handling:

- Implement robust error handling for scenarios such as invalid login credentials, data input errors, etc.
- Provide informative error messages to guide users.

Data Backup and Security:

- Implement automatic backup of question and exam data at regular intervals.
- Ensure data security through role-based access control and encryption.

Preferences:

- Allow customization of preferences such as date formats, language, etc.

Documentation:

- User manual explaining system usage and features.
- Include a README file with installation instructions and any prerequisites.

Testing:

- Comprehensive testing plan covering unit tests, integration tests, and user acceptance tests.
- Generate sample exam data for testing different scenarios.

Deployment:

- Create an installer for easy deployment on Windows systems.

Technologies Used:

- C# (Windows Forms)
- SQL Server
- ADO.NET for database interaction

Conclusion:
The Examination Management System aims to streamline the exam process for educational institutions. By adhering to these specifications, the system should offer a reliable and user-friendly solution for managing exams, questions, and student performance through a structured SQL Server database.