Software Engineering

✓ Question: 1

i. Product Backlog for the User Stories:

User Story 1:

1. Design the login page UI.
2. Implement the frontend form for login (username & password input).
3. Develop backend authentication logic.
4. Implement password hashing and encryption.
5. Write unit tests for authentication logic.
6. Implement error handling.
7. Connect the login system with the user database.

User Story 2:

1. Design the search UI.
2. Implement the category filter dropdown.
3. Develop backend logic for retrieving products by category.
4. Optimize database queries for efficient searching.
5. Write unit tests for search functionality.
6. Handle error cases (e.g., no results found).

ii. Prioritizing User Stories in Sprint Planning:

During the Sprint Planning Meeting, the development team will prioritize these user stories based on two key factors:

1. Value to the Customer:
2. Technical Feasibility:

Priority Order:
✅ Sprint 1: Focus on implementing the Login feature first because it's essential for user access.
✅ Sprint 2: Work on Search by Category, ensuring efficient database queries and UI usability.

iii. Tracking Tasks Using a Scrum Board:

- To Do (Not Started):
  - Design login page UI
  - Develop login form

    - Design search UI
    - Write unit tests for search
- In Progress (Being Worked On):
    - Implement backend authentication
    - Implement category filter
    - Optimize search queries
- Done (Completed):
    - UI design completed
    - Authentication logic tested
    - Search logic implemented

Question-2:

The best approach in this scenario would be to use Agile methodologies, specifically frameworks like Scrum to handle uncertainty, manage iterative changes, and address high-risk components. check—->khata

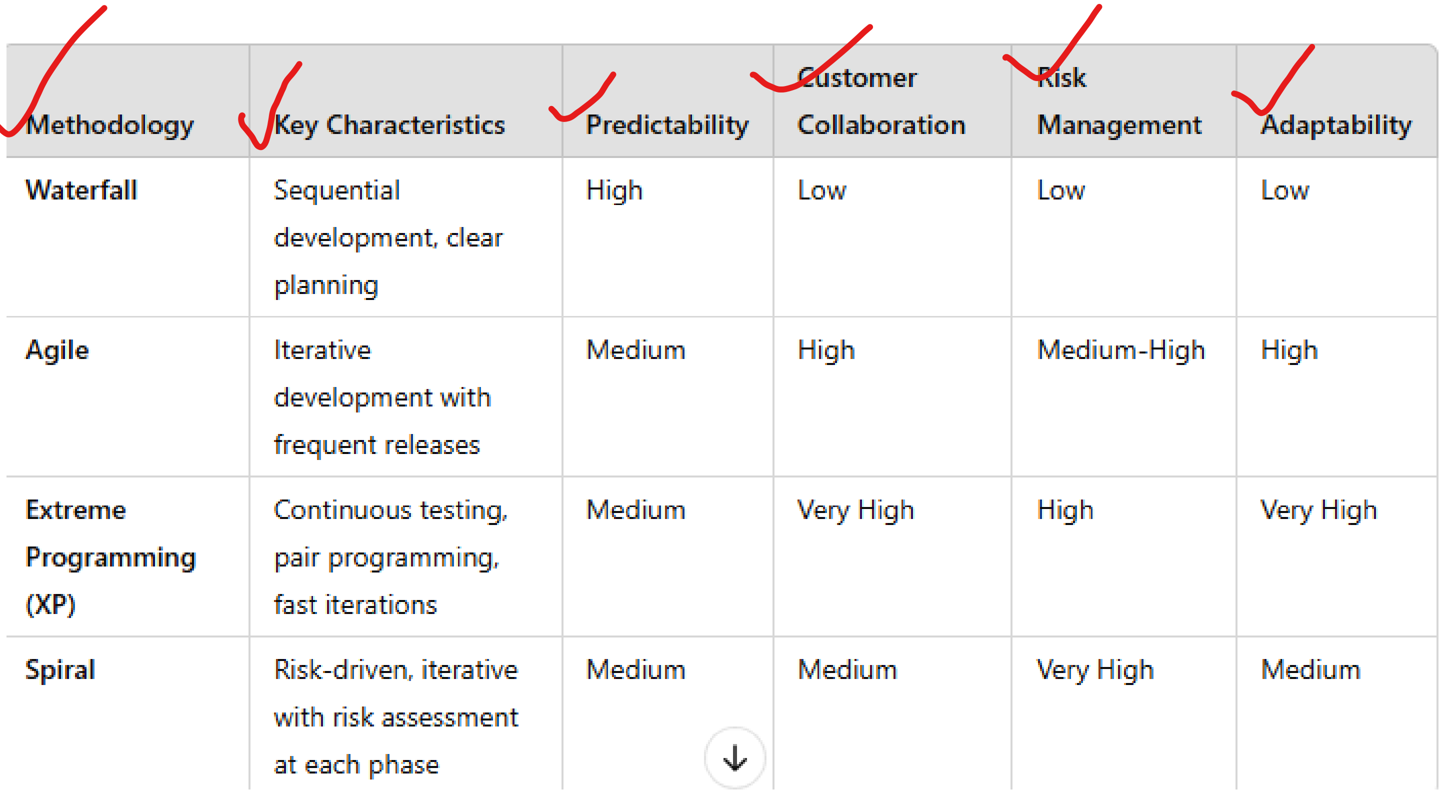
Question-3:

1. Project A (Well-defined requirements, strict deadline)

    ✔ Best methodology: Waterfall

2. Project B (Evolving requirements, uncertain timeline, continuous customer feedback)

    ✔ Best methodology: Agile or Extreme Programming (XP)



Methodology	Key Characteristics	Predictability	Customer Collaboration	Risk Management	Adaptability
Waterfall	Sequential development, clear planning	High	Low	Low	Low
Agile	Iterative development with frequent releases	Medium	High	Medium-High	High
Extreme Programming (XP)	Continuous testing, pair programming, fast iterations	Medium	Very High	High	Very High
Spiral	Risk-driven, iterative with risk assessment at each phase	Medium	Medium	Very High	Medium

4.Principles of Software Engineering Ethics and Professional Responsibility and ACM/IEEE Code of Ethics and Ethical Decision-Making:

- **Public Welfare**
- **Honesty**
- **Fairness**
- **Privacy and Security**
- **Professionalism**
- **Public Interest**
- **Product Quality**
- **Continuous Learning**

5.Functional Requirements:

- **User Authentication & Authorization**
- **Flight Search & Booking**
- **Payment Processing & Confirmation**
- **Ticket Modification & Cancellation**
- **Check-in**

Non-Functional Requirements:

- Performance & Response Time
- Scalability
- Security & Data Protection
- Usability & Accessibility
- System Maintenance & Availability

8. process improvement cycle:

- Process Assessment (Measurement & Analysis)
- Goal Setting & Planning
- Process Improvement (Implementation)
- Measurement & Monitoring (Data Collection)
- Evaluation & Continuous Improvement

9. Software Engineering Institute Capability Maturity Model (SEI CMM):

- Level 1: Initial
- Level 2: Managed
- Level 3: Defined
- Level 4: Quantitatively Managed
- Level 5: Optimizing

13.

Testing: Testing is the process of checking a system or software to ensure it works as expected, identifies bugs or issues, and confirms it meets the required standards.

Difference:https://www.geeksforgeeks.org/differences-between-verification-and-validation/

14.Layered Architecture Model:

The system is divided into four main layers:

1. Presentation Layer (User Interface)
2. Application Layer (Business Logic)
3. Service Layer (Core Services)
4. Data Layer (Persistence)

1. Presentation Layer (User Interface)

This layer is responsible for interacting with the end user. It includes:

- Web Interface: A responsive web application for users to view problems, submit code, and see results.
- Mobile Interface (optional): A mobile app for users to access the system on the go.
- Admin Interface: A separate interface for administrators to manage problems, test cases, and user accounts.

2. Application Layer (Business Logic)

This layer handles the core functionality of the system, including:

- User Management: Registration, authentication, and authorization.
- Problem Management: Adding, updating, and deleting problems.
- Submission Handling: Receiving code submissions and managing the evaluation process.
- Leaderboard and Statistics: Calculating and displaying user rankings and problem statistics.

---

3. Service Layer (Core Services)

This layer provides the backend services required to support the application layer. It includes:

- Code Execution Service: Executes user-submitted code in a secure and isolated environment (e.g., Docker containers).
- Evaluation Service: Compares the output of the user's code against predefined test cases.
- Notification Service: Sends notifications to users about submission results.
- Scoring Service: Calculates scores based on the correctness and efficiency of the code.

4. Data Layer (Persistence)

This layer is responsible for storing and retrieving data. It includes:

- User Data: Stores user profiles, credentials, and activity logs.
- Problem Data: Stores problem statements, test cases, and constraints.
- Submission Data: Stores code submissions, results, and execution logs.
- Leaderboard Data: Stores user rankings and statistics.

18.**QA** হল গুণমান নিশ্চিত করার জন্য ব্যবহৃত একটি সামগ্রিক প্রক্রিয়া। এটি ত্রুটি প্রতিরোধের উপর জোর দেয় এবং সফটওয়্যার তৈরির শুরু থেকে শেষ পর্যন্ত সমস্ত কার্যক্রমের সাথে জড়িত। অন্যদিকে, **QC** হল তৈরি করা পণ্য

পরীক্ষা করে ত্রুটি খুঁজে বের করার একটি প্রক্রিয়া। এটি **QA** এর একটি অংশ। **QC** ত্রুটি সনাক্ত করে এবং তা সংশোধনের উপর জোর দেয়।