**Adaptability:**

**I suggest using Agile for these reasons:**

* Requirements:

+ **Flexibility**: Requirement not clearly to defined, might change during development.

+ **Adaptability**: Need to adjust base on user feedback and market trends.

+ **Thoroughness**: Many to requirement need to early validation.

* Development Team:

+ **Team constructure**: 4-6 developer experienced, perfect scrum

+ **Strong collaboration**: Good communication within team.

+ **Role clarity**: well-defined role, example Scrum master, developers, ensure roles to clearly for tasks and managed effectively(Scum master, developers)

+ **Adaptability**: Can adjust quickly to changes.

+ **Time constrains**: Fits well with sprint cycles.

* Conclusion: Scrum is the best this project due to requirement not clearly, can change to user needs and market trend and faster test, found error to solve

**I suggest using the Prototype model for these reasons:**

* **Requirements:**
  + **Clarification:** Ideal for projects where requirements are unclear or incomplete, as prototypes help refine user needs.
  + **Flexibility:** Allows iterative refinement of requirements based on user feedback on the prototype.
  + **Early visualization:** Provides a tangible representation of the product early in the process to identify gaps and ambiguities.
* **Development Team:**
  + **Team structure:** Requires a team skilled in rapid development techniques to create functional prototypes efficiently.
  + **Collaboration:** Encourages close interaction between stakeholders and developers to refine the prototype iteratively.
  + **Role clarity:** Clear roles for designers, developers, and analysts to focus on prototype creation and feedback integration.
  + **Adaptability:** Prototypes can be quickly adjusted based on stakeholder feedback and evolving requirements.
  + **Validation focus:** Helps the team validate design decisions and functionalities before committing to full-scale development.
* **Conclusion:**  
  The Prototype model is ideal for projects where user requirements are not fully defined, as it enables early visualization of the product, incorporates user feedback efficiently, and minimizes misunderstandings or costly changes later in the development cycle.

**I suggest using the Kanban model for these reasons:**

* **Requirements:**
  + **Continuous flow:** Suitable for projects with evolving and continuously changing requirements.
  + **Flexibility:** Requirements can be added, updated, or reprioritized at any time, ensuring adaptability to business needs.
  + **Incremental delivery:** Focuses on delivering smaller, manageable features or tasks without needing a fully defined scope upfront.
* **Development Team:**
  + **Team structure:** Works well with teams of any size, emphasizing individual task ownership and responsibility.
  + **Collaboration:** Promotes real-time communication and teamwork to resolve bottlenecks and ensure smooth task flow.
  + **Role clarity:** While roles can be flexible, team members are responsible for managing their tasks on the Kanban board.
  + **Focus on efficiency:** Encourages the team to optimize workflow by identifying and resolving bottlenecks promptly.
  + **Adaptability:** Easily adjusts to changing priorities and supports a continuous improvement mindset.
* **Conclusion:**  
  The Kanban model is ideal for projects requiring continuous delivery and adaptability, as it provides a visual workflow for task management, allows for dynamic prioritization, and promotes team efficiency and collaboration.

**I suggest using the Extreme Programming (XP) model for these reasons:**

* **Requirements:**
  + **Customer-driven:** Continuous involvement of customers ensures requirements are refined and aligned with user needs.
  + **Flexibility:** Accommodates frequent changes in requirements throughout the development cycle.
  + **Iterative approach:** Delivers functional software in small, incremental releases to incorporate feedback effectively.
* **Development Team:**
  + **Team structure:** Requires a small, skilled team with strong communication and collaboration skills.
  + **Collaboration:** Encourages close teamwork, often involving pair programming for higher code quality and knowledge sharing.
  + **Role clarity:** Clear roles for developers, customers, and testers to ensure effective coordination and rapid delivery.
  + **Adaptability:** The team adapts quickly to changing requirements through continuous integration and testing.
  + **Focus on quality:** Practices like test-driven development (TDD), frequent code reviews, and refactoring ensure high-quality output.
* **Conclusion:**  
  Extreme Programming is best suited for projects with dynamic and frequently changing requirements, where close collaboration, high-quality deliverables, and rapid iteration are essential for success.

**I suggest using the Incremental & Iterative model for these reasons:**

* **Requirements:**
  + **Progressive development:** Requirements can evolve over time, allowing for updates and refinements in subsequent iterations.
  + **Flexibility:** Early iterations can focus on high-priority or core functionalities, with less critical features added later.
  + **User feedback:** Provides opportunities for regular validation and adjustment based on user input at the end of each iteration.
* **Development Team:**
  + **Team structure:** Suitable for teams of various sizes, with the ability to focus on smaller, manageable increments of the project.
  + **Collaboration:** Encourages regular communication between stakeholders and the development team.
  + **Role clarity:** Defined roles such as iteration planners, developers, and testers ensure efficient task management for each increment.
  + **Adaptability:** Teams can adjust the development approach based on feedback or emerging priorities after each iteration.
  + **Deliverables:** Each iteration delivers a working product, enabling faster time-to-market for key features.
* **Conclusion:**  
  The Incremental & Iterative model is best for projects with evolving requirements, as it enables the delivery of functional components early, integrates user feedback effectively, and provides flexibility to adjust in response to changes.

**I suggest using the Spiral model for these reasons:**

* **Requirements:**
  + **Flexibility:** Accommodates changing or unclear requirements through iterative refinement.
  + **Risk management:** Identifies and mitigates risks early by analyzing and addressing them in each spiral cycle.
  + **Thoroughness:** Combines iterative refinement with detailed requirement analysis and validation at each phase.
* **Development Team:**
  + **Team structure:** Requires a skilled team adept at risk analysis, planning, and iterative development.
  + **Collaboration:** Encourages close collaboration with stakeholders to refine requirements and resolve uncertainties at each spiral loop.
  + **Role clarity:** Clear roles for risk analysts, planners, developers, and testers to ensure every stage is addressed effectively.
  + **Adaptability:** Capable of adjusting plans based on feedback and risks identified during each cycle.
  + **Focus on quality:** Emphasizes iterative testing and validation to ensure continuous improvement.
* **Conclusion:**  
  The Spiral model is ideal for complex and high-risk projects, as it integrates risk management, iterative development, and flexibility, ensuring that changing requirements and potential challenges are addressed throughout the development process.

**Predictability:**

**I propose adopting Waterfall methodology based on the following reasons:**

* Requirements:

+ **Clear and Stable**: Well-defined the requirements, unlikely to change significantly thoughout development project.

+ **Structure Approach**: Step by step process is necessary to ensure thoroughness and can predictability during development.

+ **Detail Planning**: Complex requirements need to well-defined and comprehensive in the doccument, upfont plan prepare thorough avoid ambiguites and risks.

* Development Team:

+ **Team Structure**: Team need experienced and clealy role to each task, functions effectively in the environment structured.

+ **Specific role**: Clearly separation and responsibility.

+ **Documentation focus**: Emphasizes strong documentation to practice, provide detail records to referenced and good communication.

+ **Predictability**: Well-defined timeline and milestons, ansuring steady progress and minimal risks.

* Conclusion: Waterfall is best due to clear requirements, structured roles, and need for documentation.

**I suggest using the V-Model for these reasons:**

* **Requirements:**
  + **Clarity:** Requirements must be clearly defined and well-documented from the start.
  + **Stability:** Limited scope for changes during the development phase, making it suitable for projects with stable requirements.
  + **Validation:** Emphasizes thorough validation of requirements at each phase.
* **Development Team:**
  + **Team structure:** Typically requires a well-defined team with specific expertise in testing and documentation.
  + **Sequential approach:** Clear, step-by-step workflow where each phase is completed before the next begins.
  + **Role clarity:** Each role, such as analysts, developers, and testers, has well-defined responsibilities to ensure smooth transitions between phases.
  + **Focus on quality:** Testing is integrated at every stage, ensuring fewer errors in the final product.
  + **Documentation focus:** Detailed documentation supports the process, making it easier to track progress and changes.
* **Conclusion:**  
  The V-Model is ideal for projects with well-defined, stable requirements where a systematic and rigorous testing process is critical to ensure high-quality deliverables.

**I suggest using the Unified Process (UP) for these reasons:**

* **Requirements:**
  + **Phased development:** Requirements are gathered and refined progressively across phases (Inception, Elaboration, Construction, Transition).
  + **Flexibility:** Supports evolving requirements, especially during the early phases, while stabilizing them in later stages.
  + **User-driven:** Focuses on addressing user needs through iterative feedback and prioritization of use cases.
* **Development Team:**
  + **Team structure:** Requires a team familiar with iterative processes and capable of working across multiple disciplines (analysis, design, implementation, testing).
  + **Collaboration:** Encourages collaboration among stakeholders, ensuring that requirements and solutions align with business goals.
  + **Role clarity:** Clear roles for project managers, architects, developers, and testers, with responsibilities evolving across phases.
  + **Adaptability:** Teams can adapt their focus during different iterations, addressing risks, design challenges, and feature implementation progressively.
  + **Documentation:** Balances agility with the need for structured documentation, ensuring traceability and clarity throughout the project.
* **Conclusion:**  
  The Unified Process is well-suited for medium to large-scale projects with evolving requirements and complex architectures, as it combines iterative development with a phased approach, ensuring that risks are mitigated early, and quality is maintained throughout the lifecycle.
* Ambiguities : mơ hồ
* Steady: ổn định
* Latent: tiềm ẩn

**1. Agile (Scrum)**

* **Linh hoạt:** Phù hợp với yêu cầu không rõ ràng hoặc thường xuyên thay đổi.
* **Phát triển theo vòng lặp ngắn:** (Sprint) từ 1-4 tuần, giúp nhanh chóng phản hồi và điều chỉnh.
* **Tập trung vào người dùng:** Phản hồi của khách hàng là trung tâm của quá trình phát triển.
* **Đội nhóm:** Nhóm nhỏ (4-6 người), làm việc chặt chẽ với vai trò rõ ràng như Scrum Master, Product Owner.

**2. V-Model**

* **Quy trình tuần tự:** Mỗi giai đoạn có đầu vào và đầu ra rõ ràng, không quay lại bước trước.
* **Tập trung vào kiểm thử:** Song song với mỗi giai đoạn phát triển, có giai đoạn kiểm thử tương ứng.
* **Thích hợp với yêu cầu ổn định:** Đòi hỏi tài liệu chi tiết và rõ ràng ngay từ đầu.
* **Chất lượng cao:** Nhấn mạnh kiểm thử kỹ lưỡng để đảm bảo ít lỗi.

**3. Incremental & Iterative**

* **Phát triển theo từng phần:** Sản phẩm được phát hành qua nhiều bản nhỏ, tăng dần chức năng.
* **Phản hồi liên tục:** Có thể nhận góp ý từ người dùng sau mỗi lần phát hành.
* **Linh hoạt:** Dễ thay đổi hoặc bổ sung tính năng trong các vòng lặp sau.
* **Tiến độ rõ ràng:** Dễ dàng theo dõi tiến độ qua các phần hoàn thành.

**4. Spiral Model**

* **Quản lý rủi ro:** Tích hợp phân tích và xử lý rủi ro trong mỗi vòng lặp.
* **Phát triển theo chu kỳ:** Chia thành nhiều vòng xoắn, mỗi vòng giải quyết một vấn đề hoặc rủi ro.
* **Linh hoạt:** Dễ dàng thích nghi với các thay đổi trong yêu cầu.
* **Thích hợp cho dự án lớn:** Đặc biệt hiệu quả khi phát triển các hệ thống phức tạp.

**5. Prototype Model**

* **Tập trung vào nguyên mẫu:** Nhanh chóng xây dựng mô hình để minh họa (illustration) ý tưởng và yêu cầu.
* **Phản hồi nhanh:** Dễ dàng nhận phản hồi và điều chỉnh để phù hợp với kỳ vọng của khách hàng.
* **Giảm rủi ro:** Tránh hiểu sai yêu cầu thông qua nguyên mẫu.
* **Thích hợp với yêu cầu chưa rõ ràng:** Hỗ trợ khách hàng hình dung sản phẩm trước khi phát triển hoàn chỉnh.

**6. Unified Process (UP)**

* **Phát triển theo pha:** (Inception, Elaboration, Construction, Transition) phù hợp cho dự án có quy mô vừa và lớn.
* **Phân tích và thiết kế chi tiết:** Tập trung vào tài liệu hóa và mô hình hóa (Use Case).
* **Lặp và gia tăng:** Cho phép phát triển tính năng qua nhiều vòng lặp.
* **Kiểm soát tốt:** Kết hợp tài liệu hóa và phản hồi để giữ chất lượng cao.

**7. Kanban**

* **Tập trung vào dòng công việc:** Hiển thị công việc qua bảng Kanban, trực quan hóa tiến độ.
* **Linh hoạt:** Dễ dàng thay đổi ưu tiên công việc bất kỳ lúc nào.
* **Không giới hạn chu kỳ:** Không cố định thời gian phát hành, tập trung vào luồng công việc liên tục.
* **Tối ưu hiệu suất:** Giảm tắc nghẽn và cải thiện hiệu quả đội nhóm.

**8. Extreme Programming (XP)**

* **Phát hành nhanh:** Chu kỳ ngắn (1-2 tuần) với các bản cập nhật nhỏ.
* **Tập trung vào chất lượng:** Áp dụng Test-Driven Development (TDD), code review, và refactoring.
* **Hợp tác chặt chẽ:** Thường xuyên làm việc với khách hàng và thực hiện lập trình đôi (Pair Programming).
* **Linh hoạt:** Dễ dàng thay đổi yêu cầu bất kỳ lúc nào.

**9. Waterfall Model**

* **Quy trình tuần tự:** Mỗi giai đoạn (Yêu cầu, Thiết kế, Triển khai, Kiểm thử) phải hoàn thành trước khi chuyển sang giai đoạn tiếp theo.
* **Tài liệu chi tiết:** Đòi hỏi ghi chép toàn bộ yêu cầu và thiết kế trước khi bắt đầu phát triển.
* **Thích hợp với dự án nhỏ:** Hoặc các dự án có yêu cầu rõ ràng, ổn định từ đầu.

Cau 2:

**Unit testing**: Kiểm tra từng thành phần, kiểm tra từng thành phàn nhỏ của phàn mềm có hoạt động chính xác từ đầu không

* Tester Role: Developer
* Focus:

+ Testing individual software components.

+ Verify correct functionality of each component.

+ Check small components of software working correctly in the beginning.

**Integration testing:** Kiểm tra tích hợp, đảm bảo những phần trong phần mềm hoạt động tốt với nhau, xác nhận luồng dữ liệu

* Tester Role: Developer/ Integration team.
* Focus:

+ Make sure different parts of sofware work well together.

+ Interface testing between modules.

+ Data flow verification.

**System Testing:** Kiểm tra toàn bộ hệ thồng để phát hiện lỗi tiềm ẩn

* Tester Role: Tester/ Dedicate team
* Focus:

+ The whole system is tested and latent error may be found.

**User Acceptant Testing:** Kiểm tra sự chấp nhận phù hợp với nhu cầu thực thế và mong muốn của người dùng

* Tester Role: End-user like..
* Focus: The end-user testing system, interface to ensure real-world needs and their expectations.

**Regression Testing:** Kiểm tra hệ thống hoạt động trơn tru ngay cả khi sửa hoặc thay đổi

* Tester Role: Dedicated/ Tester Team
* Focus: Tester team ensuring project working smooth while update and moddified

**Performance & Loading Testing:** Hiệu xuất phải đảm bảo chất lượng cao ngay cả khi có nhiều người dùng.

* Tester Role: Tester Team, engineer
* Focus:

+ The Perfomance still high quality enven if tons user to using.

+ Ensure reliability and secure user

Câu 3:

* Functional: Liên quan đến truy xuất dữ liệu, như kiểu đụng đến data
* Non-Functional: Hiệu năng, các yêu cầu, reability, perfomance, security, availability, scalability, usability ...

Câu 4:

As a ....I want to ....... So that

Câu 5:

Câu 6:

- Activities and User Tasks:  
1. Import Question List:  
1.1 Login to the app.  
1.2 Import file questions.  
2. Create a New Exam:  
2.1 Create the quiz.  
2.2 Set-up the quiz.  
  
- Release:  
+ Release 1:  
1.1.1 Login by fu's gmail account, enter password, and register.  
1.2.1 Import Excel file questions and the columns in the excel match he question fields.  
2.1.1 Create the simple quiz with "Add", "Delete", "Edit" functions.  
2.2.2 Create the "password" if needed for the quiz.  
  
+Release 2:  
1.1.1 Login with Social media account.  
2.2.2 The quiz can setup time started, stoped by the host.  
  
+Release 3:  
1.1.2 User can see the preview questions in the file.  
2.1.2 The quiz can be private and password.

I propose adopting Scrum/Kaban methodology base on the following reasons:

* Requirements:

+ Flexibility: Requirements not clearly to defined and can change during development software

+ Adaptablility: Can quickly to change of the user requirements and market trends

+ Thoroughness: Many to requirements early to validation.

* Development Team:

+ Structure Team: 4-6 experience developer, functionaly effective in evironment structures.

+ Clearity Role: Need to divide role clearly, each menbber have tasks and responsibility ensuring clarity role to each task, managed effective.

+ Adaptiability: Can quickly change following user needs and market.

+ Collaboration Team: Team ensure connection developers, collaboration, good communication.

+ Time Contrain: Fits well with sprint cycles.

* Conclusion:

1. V-Model

* Sequence Step: Each stage clear input and output, no going back previous step.
* Focus Testing: Parallel development stage, these is a fit testing stage.
* High quality: Emphasizes testing to few error.

1. Incremental and Iteration

* Development in part: The product is released in small version, increasing functional.
* Continuous feedback: Can receive many feedback after release version

1. Spiral Model

* Risk Managerment: Integration analysis and handle risk in each iteration.
* Cycle Development: Divide into mutiple spiral and each spiral solve a problem or risk.

1. Prototype Model

* Focus Prototype: Quick build model to illustration idea and requirement
* Quick feedback: Easy receiver feedback and adjust to match customer expectation.
* Reducing Risk: Avoid misunderstanding through prototype.

1. Kanban

* Visualizing Work: Kanban board display task in column( to do, in progress, done)
* Clearly task in a card
* Limit Work in Progress: work in progress limit each task, prevent overloading, ensuring focus completed before new task

1. Extreme Programming

* Rapid Released: Short cyles (1-2 week) with small update
* Work with customer frequently, perform Pair Programming