**Question 1:**

In this circumstance(case), I recommend using Agile methodology for some(several) following reasons:

* *In terms of requirements characteristics:*

+ Reliability: The requirements are defined quite clearly but they are not enough and still have some missing points that need to discover later in the development process. So, when applying Agile, after each sprint, the requirements will be more detailed thanks to the collaboration between customers and development team.

+ How often the requirements can change: As mentioned above, the requirements are not 100% clear, fixed and still missing in some points. It is the reason why we are not sure that the requirements will remain unchanged throughout the development process. With the help of the organization, using Agile is a good option because one of the principles of Agile is "Welcome changes", so we can adapt easily with changes in user requirements.

(As said before, the requirements aren't totally clear, fixed, and there are still some unknowns. That's why we're not sure if they'll stay the same during development. Using Agile with the organization's help is a good idea because Agile is okay with changes, making it easier for us to adapt to any shifts in user needs.)

+ Can the requirements be defined at an early stage: the requirements are defined quite clear, but still have some vague (unclear) points needed to be discovered by the collaboration between the team and customer.

(The requirements are pretty clear, but there are still some unclear points that the team and customer need to work together to figure out.)

* *In terms of development team:*

+ Team size: From this situation, our team includes 6 developers and 2 QA. Thus, it is very suitable to apply Agile methods because the number of members for a team working in Agile methods is quite small. And the appropriate number of members for Agile method is between 3 and 9 members, so 8 members in this case (6 developers and 2 QA) will help the team operate in the most effective way.

+  Level of understanding of user requirements by the developers: From this circumstance, our developers are expected to have a quite good understanding of requirements. This makes the development process become easier. And with the help of customer to clarify the requirements and their needs, after each sprint, customer requirements become clearer and more detailed, developers' understanding of the requirements improves.

(In this case, our developers need to really get what the requirements are about. It makes things easier for development. Working with the customer to clear up any confusion helps the requirements get clearer and more detailed after each sprint. This way, our developers keep getting better at understanding what's needed.)

* *In terms of user involvement in the project:*

+ In this case, the organization is expected to contact with our company to provide additional resources. It is a good condition to choose Agile because in Agile, the customer participation will be throughout the product development process, helping to provide feedbacks as well as additional resources to help to improve the quality of product.

(In this situation, the organization is expected to get in touch with us for more help. Choosing Agile is good because it keeps the customer involved in the whole product-making process. They can give feedback and provide extra resources to make the product better.)

* + The situation mentioned “*The organization had contracted with a local company to provide additional resources when needed*.”
  + We have a contact with organization to communicate and give the feedback about the project.
  + Our project about Web abb has many customers to join in and use

1. => The user involvement is highly.

-> Our requirements are sort of clear, but they might tweak a bit down the road. Agile fits the bill because it's cool with changes. With 6 devs and 2 QA on board, our team size aligns nicely with what Agile likes. Having the customer in the loop all the way helps us get feedback and extra resources to jazz up the product. To sum it up, going Agile seems like the smart move for staying flexible, working as a team, and making improvements as we roll along.

(In summary, based on the characteristics mentioned in the context of the software development project, it can be concluded that the **Agile methodology** is the best approach to use. The benefit of Agile over other method, for example like Waterfall is the ability to change dynamically to the customer’s want and needs. The Agile model is not only iterative and incremental, prioritizing flexibility, collaboration but also allow users to feedback at every stage of development. From that, the development team can have an opportunity to apply user’s feedback into future iterations of the product. This customer-centric approach ensures that the final product meets all the requirements of its users. Overall, the Agile/Scrum methodology is well-suited for this mobile application and will likely result in a high-quality end product.)

**Question 1:**

1. Requirements characteristics
   * Reliability:
     + It was stated clearly above that there is a problem that needs to be overcomed, and an application is required to overcome this problem.
     + The project requirements are well-defined and possible.
     + It can immedate run while the project finished.

=> The project is expected to be highly reliable.

* + Types and number of requirement:
    - The software requirements include both *functional requirements* and *non-functional requirements*. All of them defined clearly and not confusing.
    - There are more than 8 requirements that was listed above. It include 6 functional requirements and 2 non-functional requirements.

=> Types and number of requirements defined this project is not too complex for our team.

* + Frequency of requirement may change:
    - The requirements that was mentioned above, is just some features of this appication.
    - In the process of project, some of features can be modified and changed to meet with the requirements of customer.

=> The requirements may change regular.

* + Determination of requirements at an early stage
    - Some of requirements was defined above but it isn’t enough to build a completed system.
    - The organization can be added or removed some of features in the process of project.

=> It is well-defined but not enough.

1. Development team
   * Team size:
     + The situation above metioned our team have 9 people.
     + It is *6 developers, 2 QA* and a project owner who is me.

=> It is a average team size and enough to build a project that was not too complex.

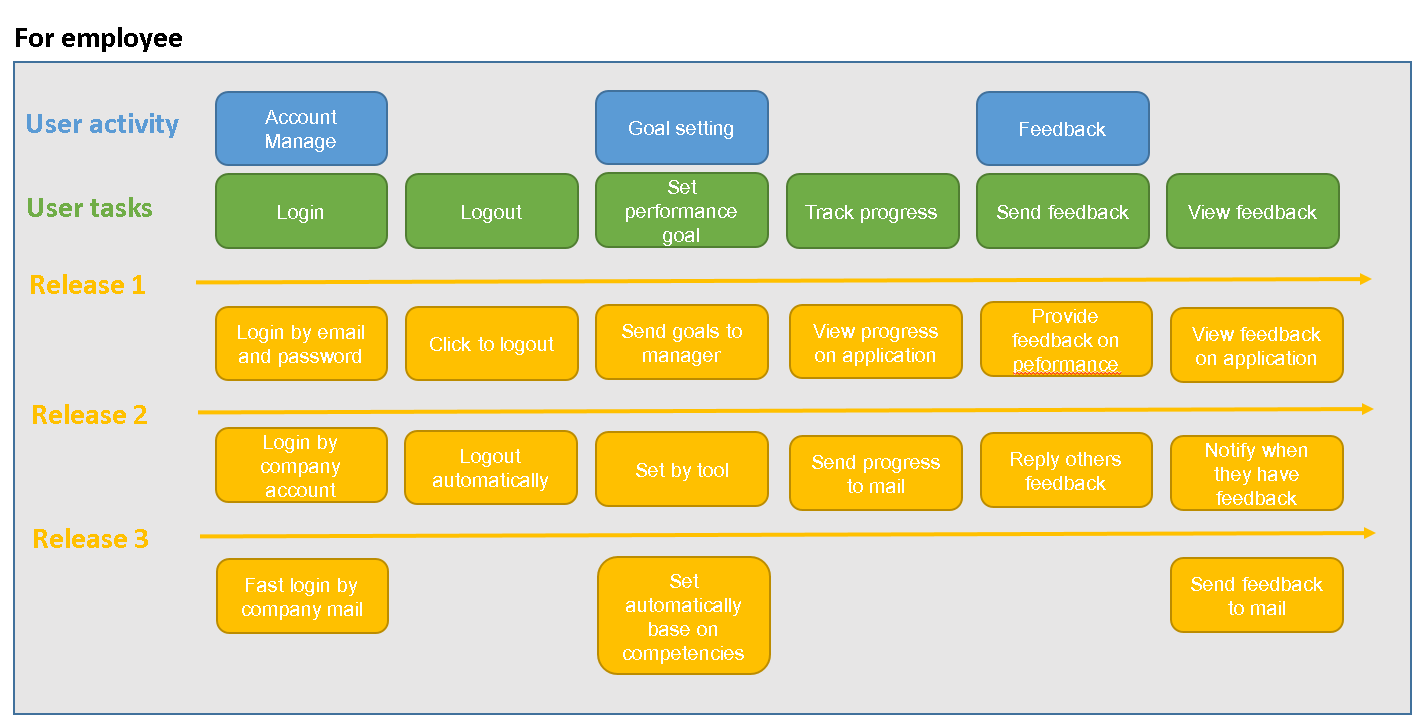
* + Level of understanding of user requirements by the developers:
    - All of requirements defined clearly above and our member can understand.
    - The organization can provide additional resources and information when needed.

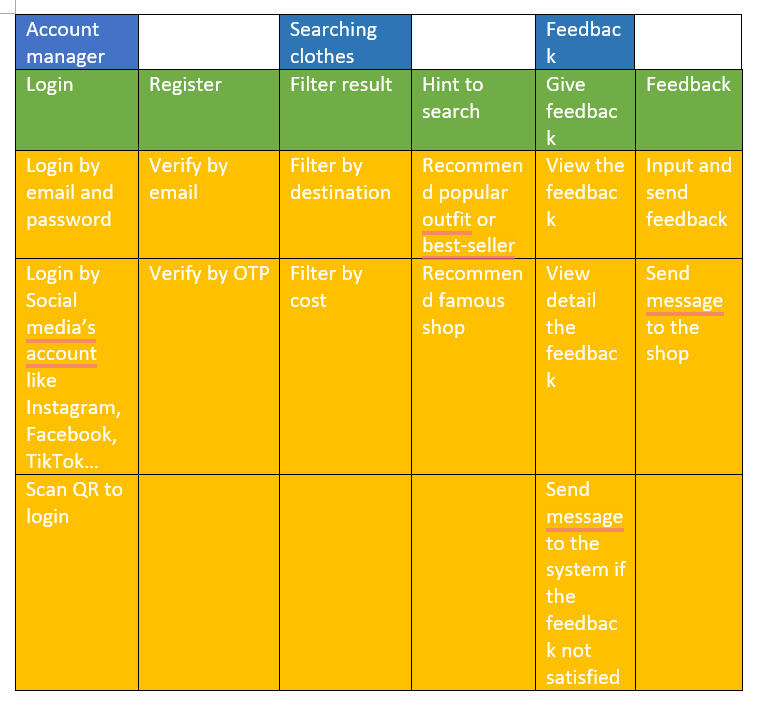
=> Our team can easy to understand and build a appication that meets the requirements.

1. User involvement
   * The situation mentioned “*The organization had contracted with a local company to provide additional resources when needed*.”
   * We have a contact with organization to communicate and give the feedback about the project.

=> The user involvement is highly.

Based on the characteristics mentioned in the context of the software development project, it can be concluded that the **Agile/Scrum methodology** is the best approach to use. The Agile/Scrum model is iterative and incremental, prioritizing flexibility, collaboration, and customer feedback at every stage of development. It also allows for customers to release the product early and gather feedback from users, which gives the development team an opportunity to apply user feedback into future iterations of the product. This customer-centric approach ensures that the final product meets the requirements of its users. Overall, the Agile/Scrum methodology is well-suited for this software development project and will likely result in a high-quality end product.





TESTCASE

Test authentication

Case: user can log in the control app of admin

1. **Login Test:**
   * **Case:** Entering incorrect username and password.
   * **Expected Output:** User should not be able to log in, and an error message should be displayed.
2. **UI/UX Test:**
   * **Case:** Opening the app on a device with a small screen.
   * **Expected Output:** Ensure that the app layout remains readable and functional, avoiding any cut-off text or overlapping elements.
3. **Functionality Test:**
   * **Case:** Testing file upload with a large file.
   * **Expected Output:** The app should handle large file uploads gracefully without crashing, and the upload progress should be smooth.
4. **Network Test:**
   * **Case:** Simulating poor network conditions during a transaction.
   * **Expected Output:** The app should display a clear error message indicating the issue and should not crash.
5. **Performance Test:**
   * **Case:** Opening a page with a heavy load of data.
   * **Expected Output:** The app should load the page within an acceptable time frame, and scrolling or interactions should not be sluggish.
6. **Compatibility Test:**
   * **Case:** Testing the app on an older version of the operating system.
   * **Expected Output:** The app should function correctly, taking into account any compatibility issues with older OS versions.
7. **Security Test:**
   * **Case:** Attempting to input malicious code into a text field.
   * **Expected Output:** The app should detect and prevent any injection attempts, displaying an error message without compromising security.
8. **Update Test:**
   * **Case:** Updating the app from a significantly older version.
   * **Expected Output:** The app should smoothly update without losing user data, and all new features should work correctly.

**Non-funtional requirement**

Performance:

Requirement: The system should load the homepage within 3 seconds under normal operating conditions.

Rationale: Ensures a responsive and efficient user experience, preventing frustration due to slow loading times.

Scalability:

Requirement: The product should handle a 20% increase in user traffic without a significant decrease in performance.

Rationale: Ensures the product can accommodate growth in user base without compromising on speed and responsiveness.

Reliability:

Requirement: The system should have a 99.9% uptime, allowing for scheduled maintenance windows of no more than 1 hour per month.

Rationale: Ensures the product is consistently available, minimizing disruptions for users.

Security:

Requirement: All user data must be encrypted during transmission using SSL/TLS protocols.

Rationale: Enhances the security posture of the product by safeguarding sensitive user information during communication.

Usability:

Requirement: The product should adhere to WCAG 2.0 accessibility standards, ensuring compatibility with screen readers and other assistive technologies.

Rationale: Guarantees inclusivity, making the product accessible to users with disabilities.

Scalability:

Requirement: The system should support up to 10,000 concurrent users without a degradation in performance.

Rationale: Ensures the product can handle high user loads during peak times without impacting user experience.

Maintainability:

Requirement: Code changes should follow a version control system, and documentation should be updated with each release.

Rationale: Facilitates ongoing development and maintenance by ensuring a systematic approach to code changes and comprehensive documentation.

Non funtional next

------------------------------------------------------------------------------------------------------------------------------------

Performance or Scalability:

Requirement: The system should work smoothly even when lots of people use it at the same time, like during busy hours. It should stay fast, with no more than a small slowdown.

Multi-platform, Multi-language:

Requirement: The app should work well on different web browsers (like Chrome or Firefox) and on phones (both iPhones and Androids). It should also be available in different languages.

UI/UX:

Requirement: The app should look nice and be easy to use. When new people try it, 9 out of 10 should be able to figure out how to use it for basic stuff in the first 5 minutes.

Security:

Requirement: To keep things safe, everyone logging in should use two-step verification. The app should also get checked for security issues twice a year to make sure it stays secure.

+ The module should be able to load and display large number of tasks without significant delay.

+ The app's user interface should be responsive and optimized for various devices and screen sizes.

STORY

As a Customer, I want to view my order history so that I can track my previous purchases and stay informed about my buying patterns.

As a Project Manager, I need to generate detailed reports on team productivity so that I can analyze performance trends and make informed decisions for project improvement.

As a Student, I want to set personalized study reminders so that I can manage my study schedule effectively and avoid missing important deadlines.

As a Sales Representative, I need to access real-time inventory levels so that I can provide accurate information to customers and prevent overselling of products.

As a Traveler, I want to receive notifications about flight delays or cancellations so that I can adjust my plans accordingly and minimize inconvenience during my journey.

1. **As a Social Media Manager, I want to schedule posts in advance so that I can maintain a consistent and timely presence on different platforms without manual intervention.**
2. **As a HR Manager, I need to conduct efficient online interviews so that I can assess candidates' qualifications and interpersonal skills, making informed hiring decisions.**
3. **As a Fitness Enthusiast, I want to track my daily calorie intake and workouts so that I can monitor my progress and make adjustments to achieve my health goals.**
4. **As a Customer Support Agent, I want to access customer interaction history so that I can provide personalized assistance and address their queries more effectively.**
5. **As a Parent, I need to set up parental controls on my child's device so that I can ensure age-appropriate content and limit screen time.**

As a Customer Support Representative, I want to have access to a customer feedback dashboard so that I can quickly identify areas for improvement and address any concerns raised by customers.

As a Sales Manager, I need a CRM system that allows me to track customer interactions and sales leads, enabling better management of customer relationships and improved sales strategies.

As a Retail Store Manager, I want an inventory management feature so that I can keep track of product availability, manage stock levels, and avoid stockouts or overstock situations.

As a Restaurant Owner, I need an online reservation system so that I can efficiently manage table bookings, provide a better customer experience, and optimize table turnover.

As a Marketing Manager, I want to analyze customer demographics and behavior through analytics tools so that I can create targeted marketing campaigns and enhance customer engagement.

* + As a user, I want to send feedback to shop and share my friends about product so that shop can adjust or improve product in the future and I want my friends can take a correct decision before buy product.
  + As a manager, I want to send emails to all customers so that I can notify all new products, shipment updates and discounts to help customers give more selection.
* **Student:**
  + As a student, I want to view the list of course with the information such as professor, department, and prerequisites to decide register for courses for the coming semester.
  + As a student, I want to personally view my electronic report at the end of the semester.
* **Professor:**
  + As a professor, I want to indicate which courses will be teaching, and see which students register for these courses.
  + As a professor, I want to record and edit the grades of the students in each class I’ve taught.

Story map

1. **User Activity: Account Management**
   * *User Tasks:*
     + Create a new account
     + Log in and log out
     + Update account information
     + Reset password
2. **User Activity: Task Management**
   * *User Tasks:*
     + Create a new task
     + Edit and delete tasks
     + Mark tasks as completed
     + Set task priorities
3. **User Activity: Reporting**
   * *User Tasks:*
     + Generate and download task reports
     + View task completion analytics
     + Filter and sort tasks based on different criteria
4. **Release 1: Basic Account Operations**
   * *Includes:*
     + User can create a new account
     + User can log in and log out
     + User can update account information
5. **Release 2: Enhanced Task Management**
   * *Includes:*
     + User can create, edit, and delete tasks
     + User can mark tasks as completed
     + User can set task priorities
6. **Release 3: Advanced Reporting**
   * *Includes:*
     + User can generate and download task reports
     + User can view task completion analytics
     + User can filter and sort tasks based on different criteria

NÊXT

1. **User Activity: Onboarding and Profile Setup**
   * *User Tasks:*
     + Download and install the app
     + Sign up for a new account
     + Set up user profile and preferences
     + Complete onboarding tutorial
2. **User Activity: Social Connectivity**
   * *User Tasks:*
     + Connect with friends or contacts
     + Share updates or content
     + Comment and like posts
     + Receive and respond to friend requests
3. **User Activity: Content Discovery**
   * *User Tasks:*
     + Explore and discover new content
     + Search for specific content or users
     + Follow or subscribe to content creators
     + Save favorite content for later
4. **Release 1: Basic User Setup and Onboarding**
   * *Includes:*
     + User can download and install the app
     + User can sign up for a new account
     + User can set up their profile and preferences
5. **Release 2: Social Connectivity Features**
   * *Includes:*
     + User can connect with friends or contacts
     + User can share updates or content
     + User can comment and like posts
     + User can receive and respond to friend requests
6. **Release 3: Enhanced Content Discovery**
   * *Includes:*
     + User can explore and discover new content
     + User can search for specific content or users
     + User can follow or subscribe to content creators
     + User can save favorite content for later

Moblie app

Low Impact, Low Probability

Situation: Inaccurate Billing Information

Cause: Rare occurrences of errors in billing data entry, often stemming from user mistakes or technical glitches. The likelihood of this happening is minimal. The consequences are minor, as any necessary corrections can be implemented through customer service, and the financial impact is negligible.

High Impact, Low Probability

Situation: Data Corruption in Reports

Cause: Infrequent instances of data corruption or formatting problems in generated reports, with a low chance of occurrence. The substantial impact results from the potential misinterpretation of crucial information by users, leading to erroneous decisions and actions.

High Impact, High Probability

Situation: Login Server Downtime

Cause: Frequent short-term outages of the login server, making the likelihood of such events high. The significant impact stems from user dissatisfaction and disrupted workflows due to the inability to access the system during these downtime periods.

Tác động thấp, xác suất thấp

Tình huống: Thông tin thanh toán không chính xác

Nguyên nhân: Hiếm khi xảy ra lỗi nhập dữ liệu hóa đơn, thường xuất phát từ sai sót của người dùng hoặc trục trặc kỹ thuật. Khả năng điều này xảy ra là tối thiểu. Hậu quả là nhỏ, vì bất kỳ sự điều chỉnh cần thiết nào cũng có thể được thực hiện thông qua dịch vụ khách hàng và tác động tài chính là không đáng kể.

Tác động cao, xác suất thấp

Tình huống: Sai dữ liệu trong báo cáo

Nguyên nhân: Các trường hợp hỏng dữ liệu hoặc sự cố định dạng không thường xuyên trong các báo cáo được tạo, với khả năng xảy ra thấp. Tác động đáng kể là kết quả của việc người dùng có thể hiểu sai thông tin quan trọng, dẫn đến các quyết định và hành động sai lầm.

Tác động cao, xác suất cao

Tình huống: Thời gian ngừng hoạt động của máy chủ đăng nhập

Nguyên nhân: Máy chủ đăng nhập thường xuyên ngừng hoạt động trong thời gian ngắn khiến khả năng xảy ra các sự kiện như vậy cao. Tác động đáng kể bắt nguồn từ sự không hài lòng của người dùng và quy trình công việc bị gián đoạn do không thể truy cập hệ thống trong những khoảng thời gian ngừng hoạt động này.

Low Impact, Low Probability

Situation: Intermittent User Interface Glitches

Cause: Infrequent incidents of minor glitches within the user interface, such as temporary visual discrepancies or graphical anomalies, with a low likelihood of occurrence. The effect is minor as these glitches are short-lived and do not impede users' capacity to complete tasks. Users can smoothly continue utilizing the software without substantial interruptions.

Giờ cao điểm

High Impact, Low Probability

Situation: Unexpected Breakdown in Integration

Cause: Infrequent cases of integration breakdowns between software components, usually triggered by factors such as version disparities, with a slim chance of occurrence. The notable consequence results from interrupted data exchange and inter-system communication, potentially leading to delays and inaccuracies.

Lỗi version

High Impact, High Probability

Situation: Sluggish System Performance During Busiest Periods

Cause: Regular occurrences of system slowdowns and delayed responses encountered by users during peak usage hours, resulting in a strong likelihood of this situation happening. The notable impact arises from reduced efficiency and user discontent caused by extended waiting periods.

Kind of test

User

In this case, I suggest using a combination of manual testing and automated testing. Because there are some things that cannot be tested with automated testing such as performance, as well as whether the application interface is user-friendly, ...

With manual testing, there will be two types: White box, Blackbox.

- White box: Used to test all errors in the code and the flow of the code so that the next software maintenance time the development team can easily find and fix the errors encountered.

- Blackbox: Used to test whether the application meets all the functional and non-functional requirements of the customer or not as well as to check if the user experience related issues such as user-friendly interface, performance, etc. good function, ...

With Automated Test: Automated testing will help reduce costs, reduce time in the testing process as well as increase the efficiency of finding errors because it is automatic and is deployed regularly.

Trong trường hợp này, tôi khuyên bạn nên sử dụng kết hợp thử nghiệm thủ công và thử nghiệm tự động. Bởi vì có một số thứ không thể kiểm thử bằng kiểm thử tự động như hiệu năng, cũng như giao diện ứng dụng có thân thiện với người dùng hay không,...

Với kiểm thử thủ công sẽ có 2 loại: Hộp trắng, Hộp đen.

- Hộp trắng: Dùng để kiểm tra tất cả các lỗi trong code và flow của code để lần bảo trì phần mềm tiếp theo nhóm phát triển có thể dễ dàng tìm và sửa các lỗi gặp phải.

- Blackbox: Dùng để kiểm tra xem ứng dụng có đáp ứng đầy đủ các yêu cầu chức năng và phi chức năng của khách hàng hay không cũng như kiểm tra xem người dùng có gặp phải các vấn đề liên quan như giao diện thân thiện, hiệu suất, v.v. chức năng tốt hay không,.. .

Với Automated Test: Automated testing sẽ giúp giảm chi phí, giảm thời gian trong quá trình kiểm thử cũng như tăng hiệu quả tìm lỗi vì nó được thực hiện tự động và được triển khai thường xuyên.

Because you have both developers and testers on your team, I recommend using white-box testing for this project. This approach allows for collaboration and knowledge sharing between the two groups. Developers can catch possible issues and make improvements, while testers give helpful feedback on how users will experience the interface. By working together, they make sure the testing covers both how the product works and the technical side, making the final product more solid and dependable. It also helps save time and money on testing overall.