Task 1: Use Case Development

Use Case Title	Schedule a Zoom Meeting
Actors	Primary: Meeting Organizer Secondary: Meeting Participants
Preconditions	1. The Meeting Organizer has a Zoom account 2. The Meeting Organizer is logged into their Zoom account 3. The Meeting Organizer has access to a device with internet connection
Main Flow	1. Meeting Organiser selects "Schedule" option in Zoom application or website 2. System displays scheduling form 3. Meeting Organizer enters meeting details (topic, date, time, duration) 4. Meeting Organizer selects meeting options (video, audio) 5. Meeting Organizer adds participants' email addresses 6. Meeting Organizer clicks "Schedule" button 7. System creates the meeting and generates a unique meeting ID and link 8. System sends meeting invitations to participants via email 9. System confirms successful meeting creation to the Meeting Organizer
Alternate Flows	A1. At step 3, if the selected date/time conflicts with another scheduled meeting: 1. System notifies Meeting Organizer of the conflict 2. Meeting Organiser selects a different date/time 3. Flow returns to step 3 A2. At step 5, if Meeting Organizer wants to use a contact list: 1. Meeting Organizer selects "Choose from Contacts" option 2. System displays contact list 3. Meeting Organizer selects desired contacts 4. Flow returns to step 5
Postconditions	1. A new Zoom meeting is created and scheduled 2. Meeting details are saved in the Meeting Organizer's Zoom account 3. Meeting invitations are sent to all added participants 4. Meeting Organizer receives a confirmation of the scheduled meeting

Task 2: Software Development Methodologies

Overview of CI/CD

From what I've learned, CI/CD is a methodology focused on automating the process of integrating and deploying code.

Continuous Integration means that developers frequently merge their code into a shared repository, where automated tests are run to check if anything breaks. This allows bugs to be caught early, and saves time.

On the other hand, **Continuous Deployment** takes things a step further by automatically deploying code to production if all the tests pass. This way, features or fixes can reach the users quickly without manual intervention. This seems really efficient, especially for projects where speed and reliability are important.

Overview of Scrum

Scrum is another software development methodology, and it's part of the broader Agile framework. The main idea behind Scrum is to break down the development process into small, manageable pieces called **sprints**, which usually last 2 to 4 weeks depending on the type of work.

In Scrum, there's a **Product Owner** who decides what tasks need to be done, a **Scrum Master** who makes sure everything runs smoothly, and the **Development Team** who works on the tasks. At the end of each sprint, the team delivers a small part of the product that's ready for feedback. This iterative process means the team can adjust to changes in requirements, which is great for projects where things aren't fully defined at the start.

Comparison of CI/CD and Scrum

After reading about both methodologies, here's how I see the differences:

Aspect	CI/CD	Scrum
Goal	Automate testing and deployment	Deliver product increments in sprints
Time Frame	Continuous with frequent releases	2-4 week sprints (fixed iterations)
Focus	Automation of the development pipeline	Team collaboration and process adaptation
Feedback	Automated feedback from tests	Human feedback through reviews and demos
Best for	Frequent updates and fast changes	Projects with evolving or unclear requirements

Scenarios for Preference

From my understanding, CI/CD is better when you need fast, frequent updates. For example, if you're working on a web app like **PrioritiQ**, a smart calendar I've been thinking about, CI/CD would be really helpful. Since it would need quick updates based on user feedback, I could use CI/CD to push those updates faster without having to manually deploy each time. Automated tests would ensure that everything works before the new features are sent out. Also it would be really useful for early stage startups who want to build MVP and make changes on the go.

On the other hand, Scrum would be better when the requirements aren't fully clear, or the project is still evolving. For example, in the early stages of developing **PrioritiQ**, Scrum would allow me to focus on small parts of the app, like building the task prioritization feature, and then get feedback at the end of each sprint. This would help me adapt to user feedback and make sure the app is growing in the right direction.