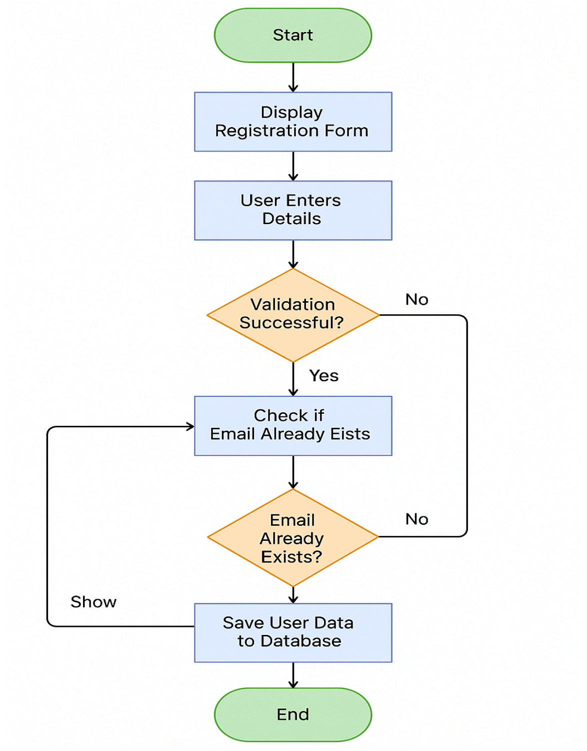
Unit 1: Introduction to Software Development

Exercise Long Answer Questions.

1. **Design a flowchart for a user registration process in a software application. Outline its key steps.**
Flowchart: User Registration Process
Key Steps:

1. **Start**
2. **Display Registration Form**
3. **User Enters Details**
(e.g., Name, Email, Password)
4. **Validate Input Fields**
- Are all fields filled?
- Is email format correct?
- Is password strong enough?
5. **If Validation Fails → Show Error → Go Back to Form**
6. **Check if Email Already Exists**
- If **yes** → Show error: "Email already registered" → Go back to form
- If **no** → Proceed
7. **Save User Data to Database**
8. **Show Success Message**
9. **End / Redirect to Login Page**



2. **Imagine you are managing a project to develop a simple mobile application. Describe how you would use the Agile Methodology to handle this project.**
Agile methodology helps manage projects by dividing the work into small, manageable parts called **sprints**, allowing teams to develop, test, and improve features step by step. It focuses on **continuous delivery**, **team**

collaboration, and **regular feedback** to ensure the app meets user needs effectively.

Steps:

1. **Define App Goal** – e.g., ride booking app.
2. **Create Product Backlog** – list features like login, booking, payment.
3. **Plan First Sprint** – choose small tasks (e.g., login screen).
4. **Develop & Test** – build and test selected features.
5. **Review & Get Feedback** – improve based on feedback.
6. **Next Sprint** – repeat with new features until the app is complete.

Agile builds the app step-by-step, with testing and feedback at every stage. Using Agile makes mobile app development more organized, efficient, and user-focused. It allows the team to deliver working features quickly, make changes when needed, and build a better product through regular feedback and collaboration.

3. **You are working on a project that requires extensive documentation and has very specific regulatory requirements. Discuss why the Scrum methodology might not be suitable for this project and suggest an alternative methodology.**

Scrum is a popular Agile methodology that works well for projects where **requirements can change frequently**, and **quick development with team collaboration** is needed. However, in projects that demand **extensive documentation** and must follow **strict regulatory or legal standards**, Scrum may not be the best fit.

Limitations of Scrum in This Case:

- Low emphasis on documentation – Scrum values working software over comprehensive documents, which may not meet regulatory standards.
- Frequent changes – Scrum encourages changes during sprints, but regulated projects often require fixed, approved plans.
- Less formal process – Scrum’s flexible style may not align with strict review, approval, or audit trails.
- Lack of early detailed planning – Scrum starts with rough planning, while regulated projects need detailed specifications from the beginning.

Suggested Alternative: Waterfall Model

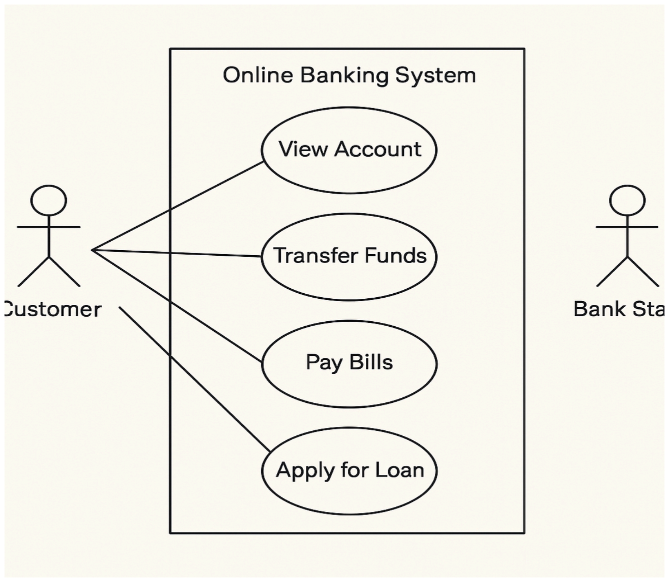
The Waterfall Model is a better choice for such projects. It follows a linear and structured approach where each phase is completed before moving to the next. This makes it ideal for situations requiring well-documented processes and approvals.

Why Waterfall Works Better:

- Strong focus on documentation – Each phase includes thorough records and reports.
- Clear requirements up front – Everything is defined early, which suits regulation-heavy work.
- Easy to manage approvals – Each stage ends with a review or sign-off, ideal for audits.
- Predictable and controlled – Less risk of unexpected changes disrupting compliance.

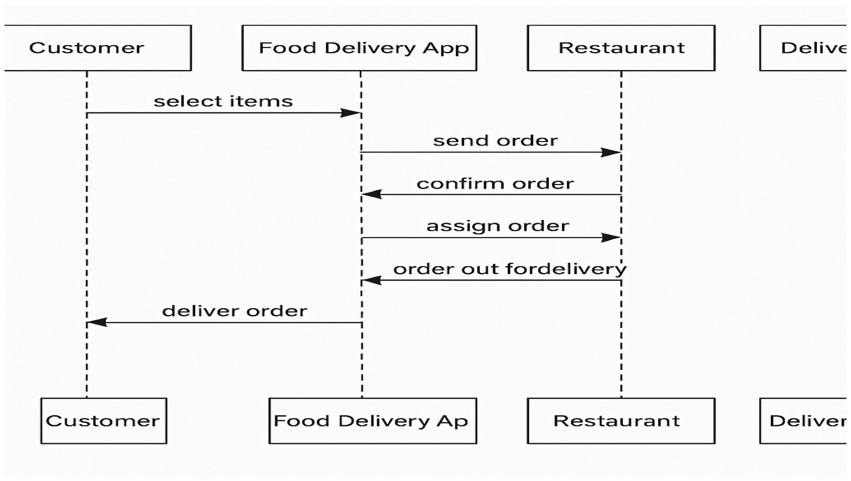
In projects with strict documentation and regulatory requirements, **Scrum's flexible nature can become a weakness**. The **Waterfall model** is a better choice because it provides **clear structure, full documentation, and strong control**, all of which are necessary for meeting legal and industry standards.

4. **Consider an online banking system. Create a Use Case Diagram to show the interactions between customers, bank staff, and the system.**



The use case diagram represents how different users interact with the **online banking system**. There are two main actors: the **Customer** and the **Bank Staff**. The **Customer** can log in to the system to view their account balance and transaction history, transfer money to other accounts, pay utility bills like electricity or internet, and apply for loans by filling out an online form. The **Bank Staff** accesses the system to review and approve loan requests, monitor transactions for security, and update customer information when needed. The system processes customer inputs, validates them, and then performs the requested operations, such as updating account balances or generating transaction receipts. This use case diagram helps developers understand **who uses the system and what actions they perform**, which is useful for planning the system's features and user interface.

5. **You are developing a food delivery application. Create a Sequence Diagram to show the process of placing an order, from the customer selecting items to the delivery of the order.**



This sequence diagram shows the step-by-step interaction between the **Customer**, **Food Delivery App**,

Restaurant, and **Delivery Person** during the process of placing a food order.

1. **Customer → Food Delivery App**: The process begins when the **customer selects food items** in the app and places the order.
2. **Food Delivery App → Restaurant**: The app **sends the order details** to the restaurant for preparation.
3. **Restaurant → Food Delivery App**: Once the restaurant receives the order, it **confirms the order**, indicating that it is being prepared.
4. **Food Delivery App → Delivery Person**: After confirmation, the app **assigns a delivery person** who will pick up and deliver the order.
5. **Restaurant → Food Delivery App**: When the order is ready, the restaurant informs the app that the **order is out for delivery**.
6. **Delivery Person → Customer**: Finally, the **delivery person delivers the order** to the customer.

6. Discuss the importance of software development tools in the software development process.

a) **Explain the role of language editors, translators, and debuggers in creating and maintaining software.**

b) **Provide examples of each tool and describe how they contribute to the efficiency and accuracy of software development.**

Importance of Software Development Tools:

Software development tools play a critical role in writing, testing, and maintaining programs efficiently. They help reduce errors, improve productivity, and make code easier to manage. These tools help developers write code, find and fix errors, and manage software projects effectively. Software development tools are programs or applications that assist in various stages of software creation. They are used to write, edit, test, debug, and manage code, ensuring that software functions correctly and efficiently.

a) Explain the role of language editors, translators, and debuggers in creating and maintaining software.

Language Editors:

Language editors, also known as code editors, are tools that help developers write and edit code in different programming languages. The purpose of language editors is to provide a user-friendly interface for writing code. Examples include:

- **Notepad++:** A simple yet powerful code editor.
- **Sublime Text:** Known for its speed and ease of use.
- **VS Code:** A popular editor with many extensions.

Translators:

Translators :are tools that convert code written in one programming language into another language that the computer can understand. Translators convert high-level programming languages (like Python) into machine language (binary code) that computers can execute.

- **Interpreters:** Translate code line-by-line (e.g., Python interpreter).
- **Compilers:** Translate the entire code at once (e.g., GCC for C/C++).

Compilers:

Compilers are a type of translator that converts entire programs from high-level languages into machine code before execution. The purpose of compilers is to optimize and convert code into efficient machine code. Examples include: **GCC:** GNU Compiler Collection for C and C++. **Javac:** Compiler for Java programs.

Debuggers:

Debuggers are tools that help developers find and fix errors (bugs) in their code. The purpose of debuggers is to allow developers to test their code and identify where emors occur. Examples include:
GDB: GNU Debugger for C/C++. **Visual Studio Debugger:** Integrated with Visual Studio IDE.

b) Examples and Their Contributions

- **Language Editor Example:**
  Visual Studio Code (VS Code) – Offers smart suggestions, real-time error detection, and extensions for many languages.
  → *Speeds up development and reduces syntax mistakes.*
- **Translator Example:**
  GCC Compiler (for C/C++) or **Python Interpreter** – Translates code so it can be executed on a computer.
  → *Allows programs to run on different systems after translation.*
- **Debugger Example:**
  GDB (GNU Debugger) or **Python Debugger (pdb)** – Lets programmers pause and examine code execution.
  → *Helps locate logic or runtime errors efficiently.*