1) What is SDLC ?
   - ⇨ The software development life cycle (SDLC) is the process of planning, writing, modifying, and maintaining software. Developers use the methodology as they design and write modern software for computers, cloud deployment, mobile phones, video games, and more. Adhering to the SDLC methodology helps to optimize the final outcome.

2) What is Software Testing ?
   - ⇨ Software testing is a process used to identify the correctness, completeness and quality of development computer software.

3) What is agile methodology?
   - ⇨ Agile methodology is a project management framework that breaks projects down into several dynamic phases, commonly known as sprints.

4) What is SRS?
   - ⇨ SRS is a complete description of an application which is to be developed.
   - ⇨ SRS contains use case diagram that describes all the interaction user will have with the software application.
5) What is opps?
   - ⇨ Object Oriented Programming way of writing the programs in organized way , Provides security, reduces code redundancy etc.
   - ⇨ Object like a black box where data are hidden.

6) Write basic concept of oops.
   - ⇨ 1) Class
   - ⇨ 2) Object
   - ⇨ 3) Inheritance
   - ⇨ 4) Polymorphism
   - ⇨ 5) Encapsulation
   - ⇨ 6) Abstraction
7) What is Object?

⇨ Object gives the permission to access functionality of class. It can represent a dog, a person, a table, etc.

8) What is Class?

⇨ Class is a collection of Data member and member function.

9) What is Encapsulation ?

⇨ The Process wrapping the data in a single unit, to Secure the data from outside world.

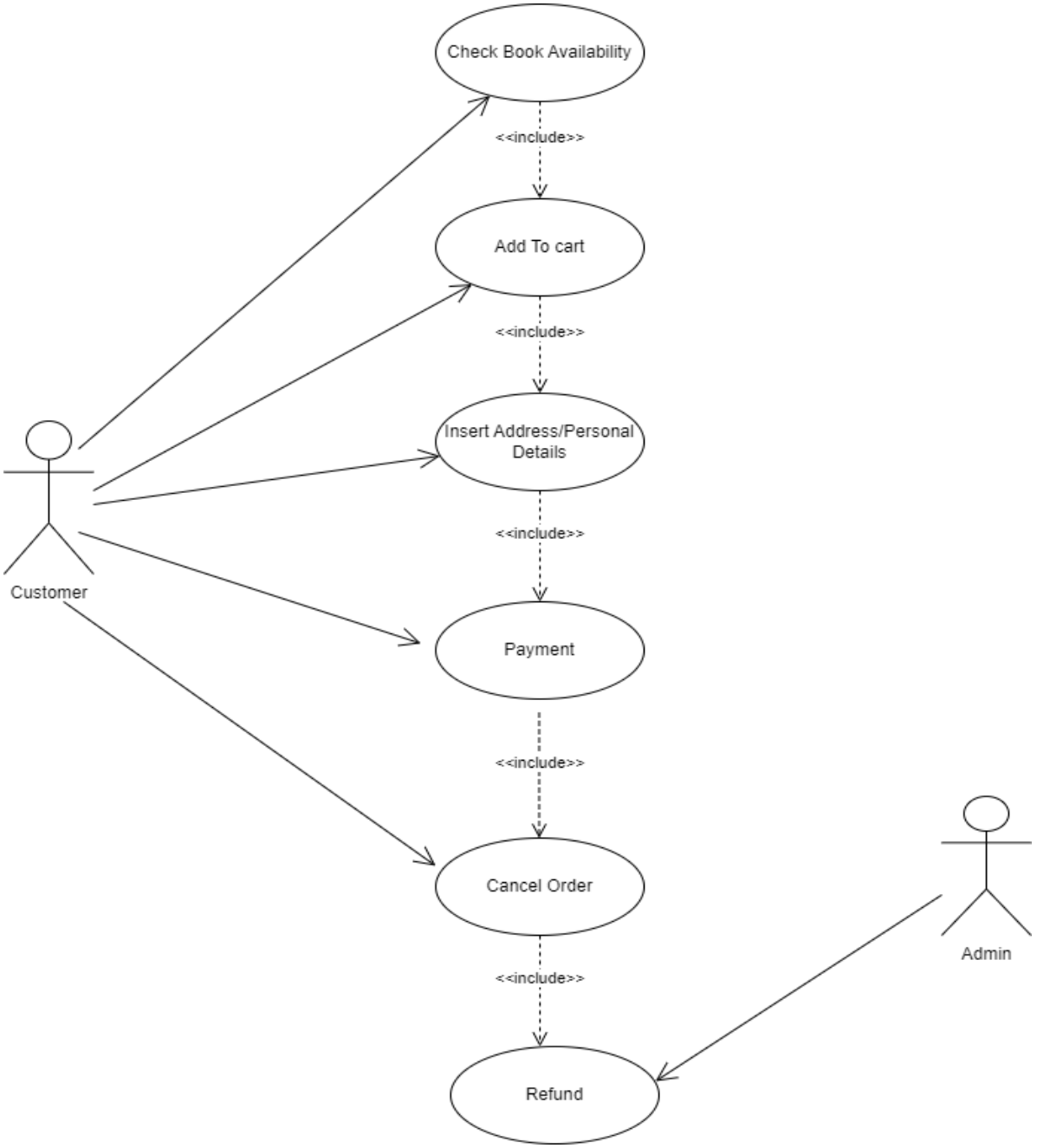
10) What is inheritance ?

⇨ Making a class from an existing class. Driving the attribute of some parent class. The class which inherits the features is known as the child class, and the class whose features it inherited is called the parent class. For example, Class Vehicle is the parent class, and Class Bus, Car, and Bike are child classes.

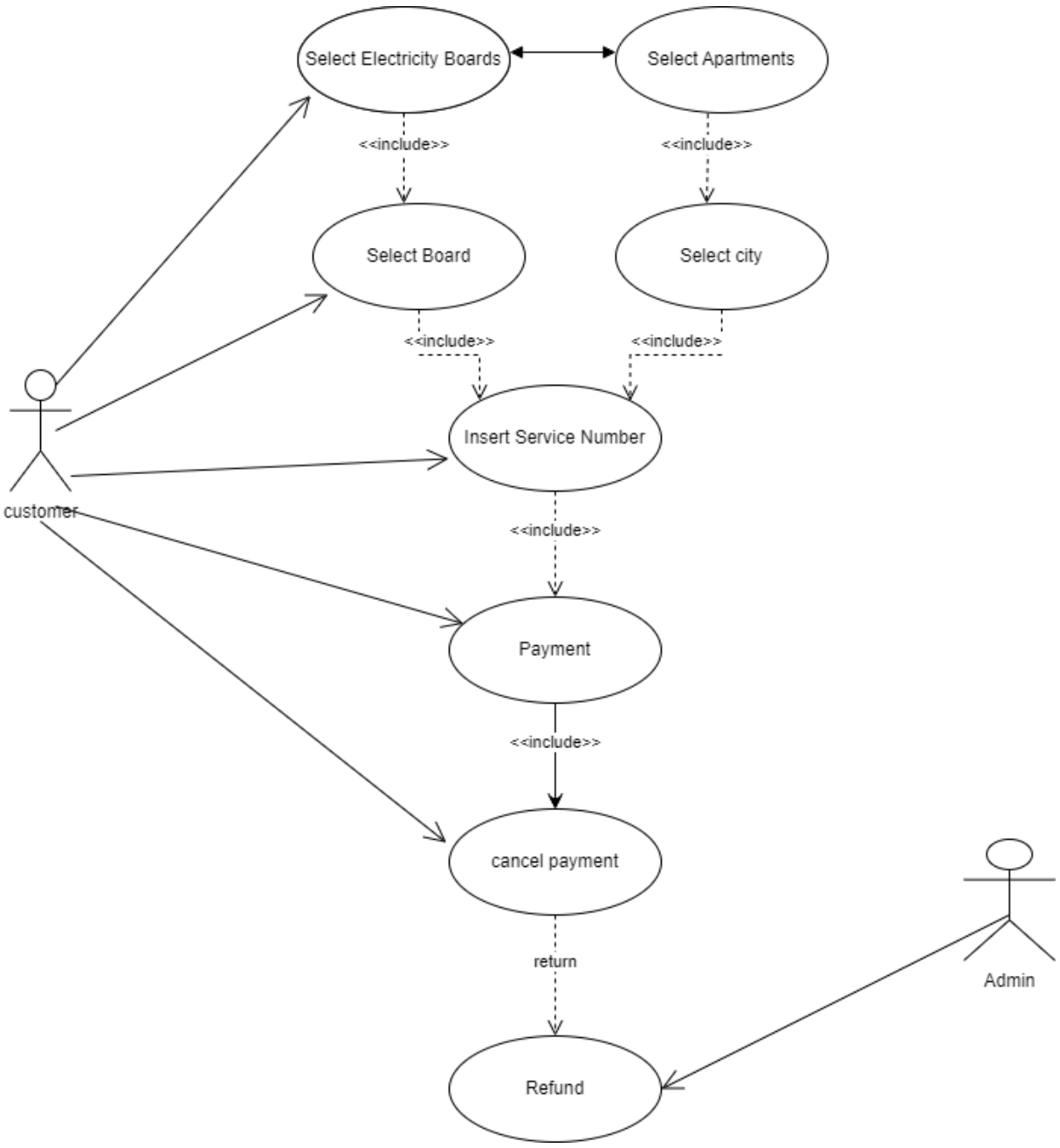
11) What is Polymorphism

⇨ Polymorphism means many forms. It is the ability to take more than one form. It is a feature that provides a function or an operator with more than one definition. It can be implemented using function overloading, operator overload, function overriding, virtual function.

12) Draw Usecase on online book shopping?



13)Draw Usecase on online bill payment system(paytm).



14) Write SDLC Phases with basic introduction ?

- Requirement gathering
- Analysis
- Design
- Implementation
- Testing
- Maintenance

• 1. Requirement gathering:-

Requirements definitions usually consist of natural language, supplemented by (e.g., UML) diagrams and tables. Three types of problems can arise:
Lack of clarity: It is hard to write documents that are both precise and easy-to-read. **Requirements confusion**: Functional and Non-functional requirements tend to be intertwined. **Requirements Amalgamation**: Several different requirements may be expressed together.

• 2.Analysis:-

The analysis phase defines the requirements of the system, independent of how these requirements will be accomplished. This phase defines the problem that the customer is trying to solve.
The deliverable result at the end of this phase is a requirement document. Ideally, this document states in a clear and precise fashion what is to be built. This analysis represents the "**what**" phase.
The requirement documentaries to capture the requirements from the customer's perspective by defining goals.

• 3.Design:-

Design Architecture Document Implementation Plan Critical Priority Analysis Performance Analysis Test Plan The Design team can now expand upon the information established in the requirement document.
The requirement document must guide this decision process. Analyzing the trade-offs of necessary complexity allows for many things to remain simple which, in turn, will eventually lead to a higher quality product.
The architecture team also converts the typical scenarios into a test plan.

• 4.implementation:-

In the implementation phase, the team builds the components either from scratch or by composition.
Given the architecture document from the design phase and the requirement document from the analysis phase, the team should build exactly what has been requested, though there is still room for innovation and flexibility.
For example, a component may be narrowly designed for this particular system, or the component may be made more general to satisfy a reusability guideline.

Implementation - Code Critical Error Removal The implementation phase deals with issues of quality, performance, baselines, libraries, and debugging. The end deliverable is the product itself. There are already many established techniques associated with implementation.

- 5. **Testing phase:-**

Simply stated, quality is very important.
Many companies have not learned that quality is important and deliver more claimed functionality but at a lower quality level.
It is much easier to explain to a customer why there is a missing feature than to explain to a customer why the product lacks quality.
A customer satisfied with the quality of a product will remain loyal and wait for new functionality in the next version.
Quality is a distinguishing attribute of a system indicating the degree of excellence.
Regression Testing
Internal Testing
Unit Testing
Application Testing
Stress Testing

- **6. Maintenance Phase:-**

Software maintenance is one of the activities in software engineering, and is the process of enhancing and optimizing deployed software (software release), as well as fixing defects. Software maintenance is also one of the phases in the System Development Life Cycle (SDLC), as it applies to software development.
The maintenance phase is the phase which comes after deployment of the software into the field.
The developing organization or team will have some mechanism to document and track defects and deficiencies.
configuration and version management reengineering (redesigning and refactoring) updating all analysis, design and user documentation Repeatable, automated tests enable evolution and refactoring.

15) Explain phases of waterfall models.

- The waterfall is unrealistic for many reasons, especially:
- Requirements must be "**frozen**" to early in the life cycle
- Requirements are **validated too late**

Application:

- Requirements are very well documented, clear and fixed.
- Product definition is stable.
- Technology is understood and is not dynamic.
- There are no ambiguous requirements.
- Ample resources with required expertise are available to support the product.

- The project is short.

Pros:

- Simple and easy to understand and use
- Easy to manage due to the rigidity of the model. Each phase has specific deliverables and a review process.
- Phases are processed and completed one at a time.
- Works well for smaller projects where requirements are very well understood.

Cons:

- No working software is produced until late during the life cycle.
- High amounts of risk and uncertainty.
- Not a good model for complex and object-oriented projects.
- Poor model for long and ongoing projects.

16) Write phases of spiral model.

Application:

- Spiral Model is very widely used in the software industry as it is in synch with the natural development process of any product i.e. learning with maturity and also involves minimum risk for the customer as well as the development firms. Following are the typical uses of Spiral model:

- When costs there are a budget constraint and risk evaluation is important.
- For medium to high-risk projects.
- Long-term project commitment because of potential changes to economic priorities as the requirements change with time.
- Customer is not sure of their requirements which are usually the case.

Pros:

- Changing requirements can be accommodated.
- Allows for extensive use of prototypes.
- Development can be divided into smaller parts and more risky parts can be developed earlier which helps better risk management.

17) Write agile manifesto principles.

1) Customer satisfaction through early and continuous software delivery.
2) Accommodate changing requirement throughout the development process.
3) Frequently delivery of working software.
4) Collaboration between the business stakeholders and developers throughout the project.
5) Support, trust ,and motivate the people involved.
6) Enable face-to-face interaction.
7) Working software is the primary measure of progress.
8) Agile process to support a consistent development pace.

9) Attention to technical detail and design enhance agility.
10) Simplicity
11) Self-organization teams encourage great architecture, requirements and designs.
12) Regular reflection on how to become more effective.

18) Explain working methodology of agile model and also write pros and cons.

- Agile model believes that every project needs to be handled differently and the existing methods need to be tailored to best suit the project requirements. In agile the tasks are divided to time boxes (small time frames) to deliver specific features for a release.

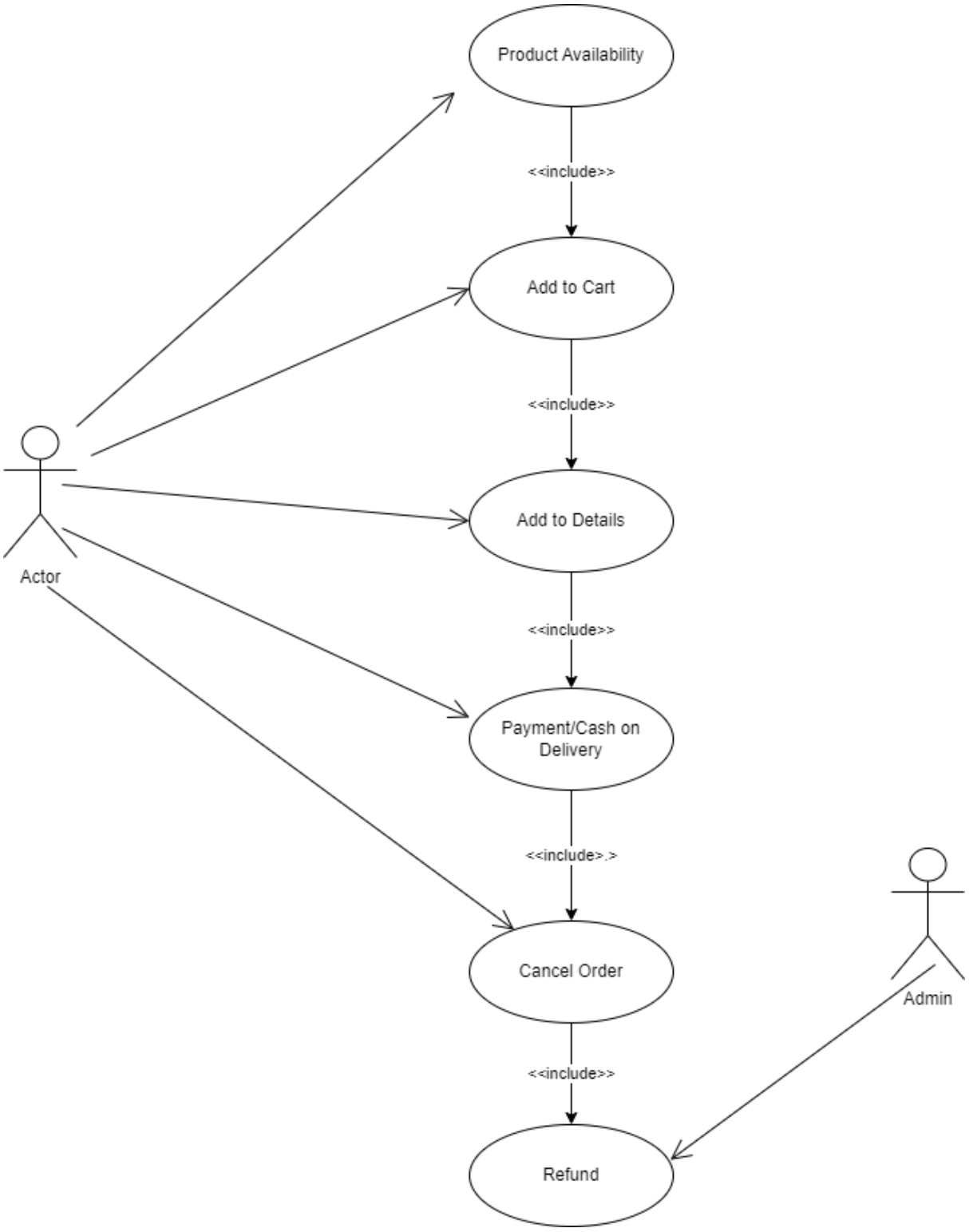
Pros:

- Is a very realistic approach to software development
- Promotes teamwork and cross training.
- Functionality can be developed rapidly and demonstrated.
- Resource requirements are minimum.
- Suitable for fixed or changing requirements
- Delivers early partial working solutions.
- Good model for environments that change steadily.
- Minimal rules, documentation easily employed.
- Enables concurrent development and delivery within an overall planned context.
- Little or no planning required
- Easy to manage
- Gives flexibility to developers

Cons:

- Not suitable for handling complex dependencies.
- More risk of sustainability, maintainability and extensibility.
- An overall plan, an agile leader and agile PM practice is a must without which it will not work.
- Strict delivery management dictates the scope, functionality to be delivered, and adjustments to meet the deadlines.
- Depends heavily on customer interaction, so if customer is not clear, team can be driven in the wrong direction.
- There is very high individual dependency, since there is minimum documentation generated.
- Transfer of technology to new team members may be quite challenging due to lack of documentation.

19) Draw use case on online shopping product using COD.



20) Draw use case on online shopping product using payment gateway.