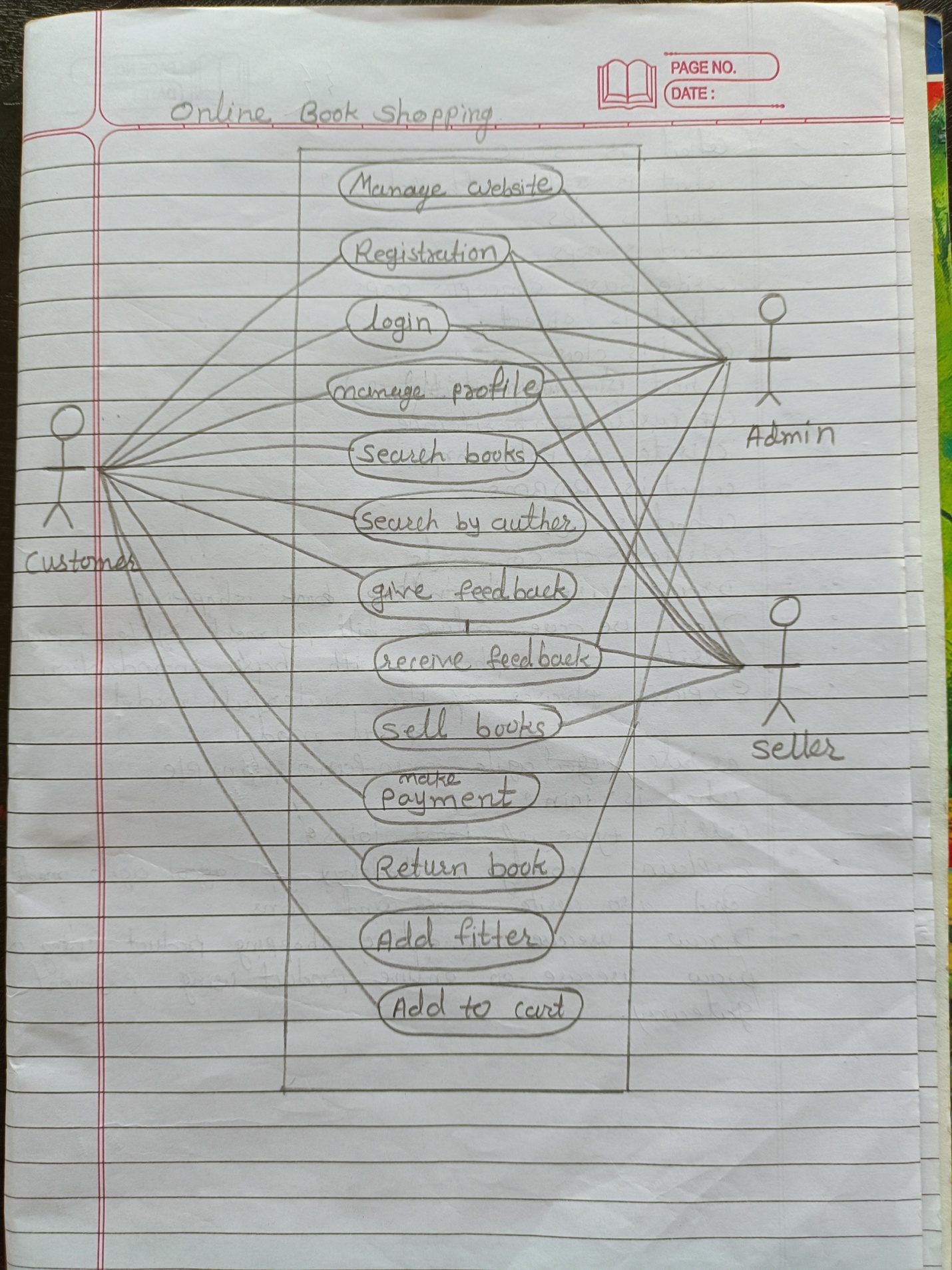
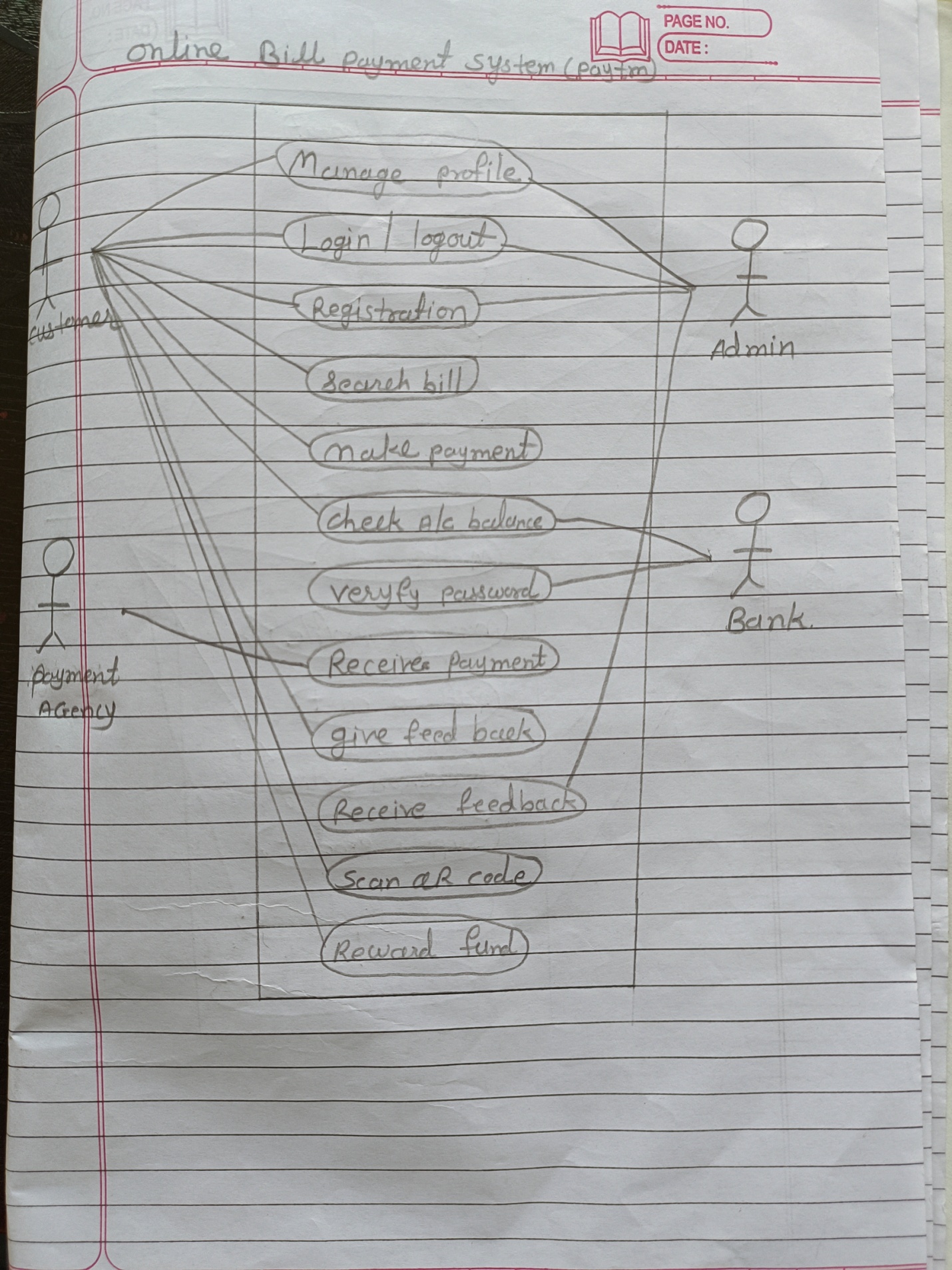
Assignment

* **What is SDLC**
* SDLC is a Structure impose on the development of software product that defines the process for planning,implimantation,testing documentation,deployment and on going maintance and support.
* **What is agile methodology?**
* Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product. Agile Methods break the product into small incremental builds.
* These builds are provided in iterations.
* Each iteration typically lasts from about one to three weeks.
* Every iteration involves cross functional teams working simultaneously on various areas like planning, requirements analysis, design, coding, unit testing, and acceptance testing.
* At the end of the iteration a working product is displayed to the customer and importantstakeholders.
* 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.
* **What is SRS**
* A software requirements specification is a complete description of the behavior of the system to be developed.
* It includes a set of use cases that describe all of the interactions that the users will have with the software.
* Use cases are also known as functional and nofunctional requirements.
* **What is OOPS**
  + Identifying objects and assigning responsibilities to these objects. Objects communicate to other objects by sending messages.
  + Messages are received by the methods of an object
  + An object is like a black box.
  + Objects of a program interact by sending messages to each other.
* **Write basic concepts OOPS**
* Object
* Class
* Encapsulation
* Inheritance
* Polymorphism
* Overriding
* Overloading
* Abstraction
* **What is object**
* An object represents an individual, identifiable item, unit, or entity, either real or abstract, with a well-defined role in the problem domain.
* An "object" is anything to which a concept applies.
* This is the basic unit of object oriented programming.
* That is both data and function that operate on data are bundled as a unit called as object
* **What is class**
  + When you define a class, you define a blueprint for an object.
  + This doesn't actually define any data, but it does define what the class name means, that is, what an object of the class will consist of and what operations can be performed on such an object.
  + A class represents an abstraction of the object and abstracts the properties and behavior of that object.
  + Class can be considered as the blueprint or definition or a template for an object and describes the properties and behavior of that object, but without any actual existence.
  + An object is a particular instance of a class which has actual existence and there can be many objects for a class.
  + In the case of a car or laptop, there will be a blueprint or design created first and then the actual car or laptop will be built based on that. We do not actually buy these blueprints but the actual objects.
* **What is Encapsulation**
* Encapsulation is the practice of including in an object everything it needs hidden from other objects. The internal state is usually not accessible by other objects.
* Encapsulation is placing the data and the functions that work on that data in the same place. While working with procedural languages, it is not always clear which functions work on which variables but object-oriented programming provides you framework to place the data and the relevant functions together in the same object.
* **Write inheritance**
* Inheritance means that one class inherits the characteristics of another class. This is also called a “is a” relationship
* One of the most useful aspects of object-oriented programming is code reusability. As the name suggests Inheritance is the process of forming a new class from an existing class that is from the existing class called as base class, new class is formed called as derived class.
* This is a very important concept of object-oriented programming since this feature helps to reduce the code size.
* Inheritance describes the relationship between two classes. A class can get some of its characteristics from a parent class and then add unique features of its own.
* In general, Java supports single-parent, multiple-children inheritance and multilevel inheritance (Grandparent-> Parent -> Child) for classes and interfaces. Java supports multiple inheritances (multiple parents, single child) only through interfaces.
* In a class context, inheritance is referred to as implementation inheritance, and in an interface context, it is also referred to as interface inheritance.
* **Write polymorphism**
* Polymorphism means “having many forms”.
* It allows different objects to respond to the same message in different ways, the response specific to the type of the object.
* The most important aspect of an object is its behaviour (the things it can do). A behavior is initiated by sending a message to the object (usually by calling a method).
* The ability to use an operator or function in different ways in other words giving different meaning or functions to the operators or functions is called polymorphism.
* Poly refers too many. That is a single function or an operator functioning in many ways different upon the usage is called polymorphism.
* E.g. the message displayDetails() of the Person class should give different results when send to a Student object (e.g. the enrolment number).
* The ability to change form is known as polymorphism.
* There is two types of polymorphism in Java
  + Compile time polymorphism(Overloading)
  + Runtime polymorphism(Overriding)
* **Draw use-case online book shopping**

****

* **Draw use-case online bill payment system(Paytm)**

****

* **Write SDLCphases with basic introduction**

SDLC Phases

Requirements Collection/Gathering Establish Customer Needs

Analysis Model And Specify the requirements- “What”

Design Model And Specify a Solution – “Why”

Implementation Construct a Solution In Software Testing Validate the solution against the requirements

Maintenance Repair defects and adapt the solution to the new requirements

**Requirement Gathering**

* Features
* Usage scenarios
* Although requirements may be documented in written form, they may be incomplete, unambiguous, or even incorrect.
* Requirements will Change!
* Inadequately captured or expressed in the first place
* User and business needs change during the project
* Validation is needed throughout the software lifecycle, not only when the “final system” is delivered.
* Build constant feedback into the project plan
* Plan for change
* Early prototyping can help clarify the requirements
* Requirements definitions usually consist of natural language, supplemented by diagrams and tables.
* **Three types of problems can arise**:
* • Lack of clarity: It is hard to write documents that are both precise and easy-toread.
* • Requirements confusion: Functional and Non-functional requirements tend to be intertwined.
* Requirements Amalgamation: Several different requirements may be expressed together.
* Types of Requirements:
* • Functional Requirements: describe system services or functions.
  + • Compute sales tax on a purchase
  + • Update the database on the server
* • Non-Functional Requirements: are constraints on the system or the development process.
* Non-functional requirements may be more critical than functional requirements.
* If these are not met, the system is useless!
* **Analysis Phase**
* 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.
* This phase starts with the requirement document delivered by the requirement phase and maps the requirements into architecture.
* The architecture defines the components, their interfaces and behaviors.
* The deliverable design document is the architecture.• This phase represents the “how” phase.
* Details on computer programming languages and environments, machines, packages, application architecture, distributed architecture layering, memory size, platform, algorithms, data structures, global type definitions, interfaces, and many other engineering details are established.
* The design may include the usage of existing components.

**Design Phase**

* 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.

**Implementation Phase**

* 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.
* 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.

**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

**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
* Maintenance is the process of changing a system after it has been deployed. **Corrective maintenance**: identifying and repairing defects

**Adaptive maintenance**: adapting the existing solution to the new platforms. **Perfective Maintenance**: implementing the new requirements

* **Explain phases of the waterfall model.**

The waterfall is unrealistic for many reasons, especially:

• Requirements must be “frozen” to early in the life cycle

• Requirements are validated too late

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.

Advantages

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.

Clearly defined stages.

Well understood milestones.

Easy to arrange tasks.

Process and results are well documented.

Disadvantages

* 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.
* Not suitable for the projects where requirements are at a moderate to high risk of changing. So risk and uncertainty is high with this process model.
* It is difficult to measure progress within stages.
* Cannot accommodate changing requirements.
* No working software is produced until late in the life cycle.
* Adjusting scope during the life cycle can end a project.
* Integration is done as a "big-bang. at the very end, which doesn't allow identifying any technological or business bottleneck or challenges early.
* **Write phases of spiral model.**

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. Requirements are complex and need evaluation to get clarity.
* New product line which should be released in phases to get enough customer feedback.
* Significant changes are expected in the product during the development cycle.

Advantage

Changing requirements can be accommodated.

* Allows for extensive use of prototypes
* Requirements can be captured more accurately.
* Users see the system early.
* Development can be divided into smaller parts and more risky parts can be developed earlier which helps better risk management.

**Disadvantages**

* Management is more complex.
* End of project may not be known early.
* Not suitable for small or low risk projects and could be expensive for small projects.
* Process is complex.
* Spiral may go indefinitely.
* Large number of intermediate stages requires excessive documentation.
* **Write agile manifesto principal.**

**Individuals and interactions** - in agile development, self-organization and motivation are important, as are interactions like co-location and pair programming.

**Working software** - Demo working software is considered the best means of communication with the customer to understand their requirement, instead of just depending on documentation.

**Customer collaboration** - As the requirements cannot be gathered completely in the beginning of the project due to various factors, continuous customer interaction is very important to get proper product requirements..

**Responding to change** - agile development is focused on quick responses to change and• continuous development.

* **Explain working methodology of agile model and also write pros and cons.**
* Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product.
* 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.
* Iterative approach is taken and working software build is delivered after each iteration. Each build is incremental in terms of features; the final build holds all the features required by the customer.
* Agile thought process had started early in the software development and started becoming popular with time due to its flexibility and adaptability.

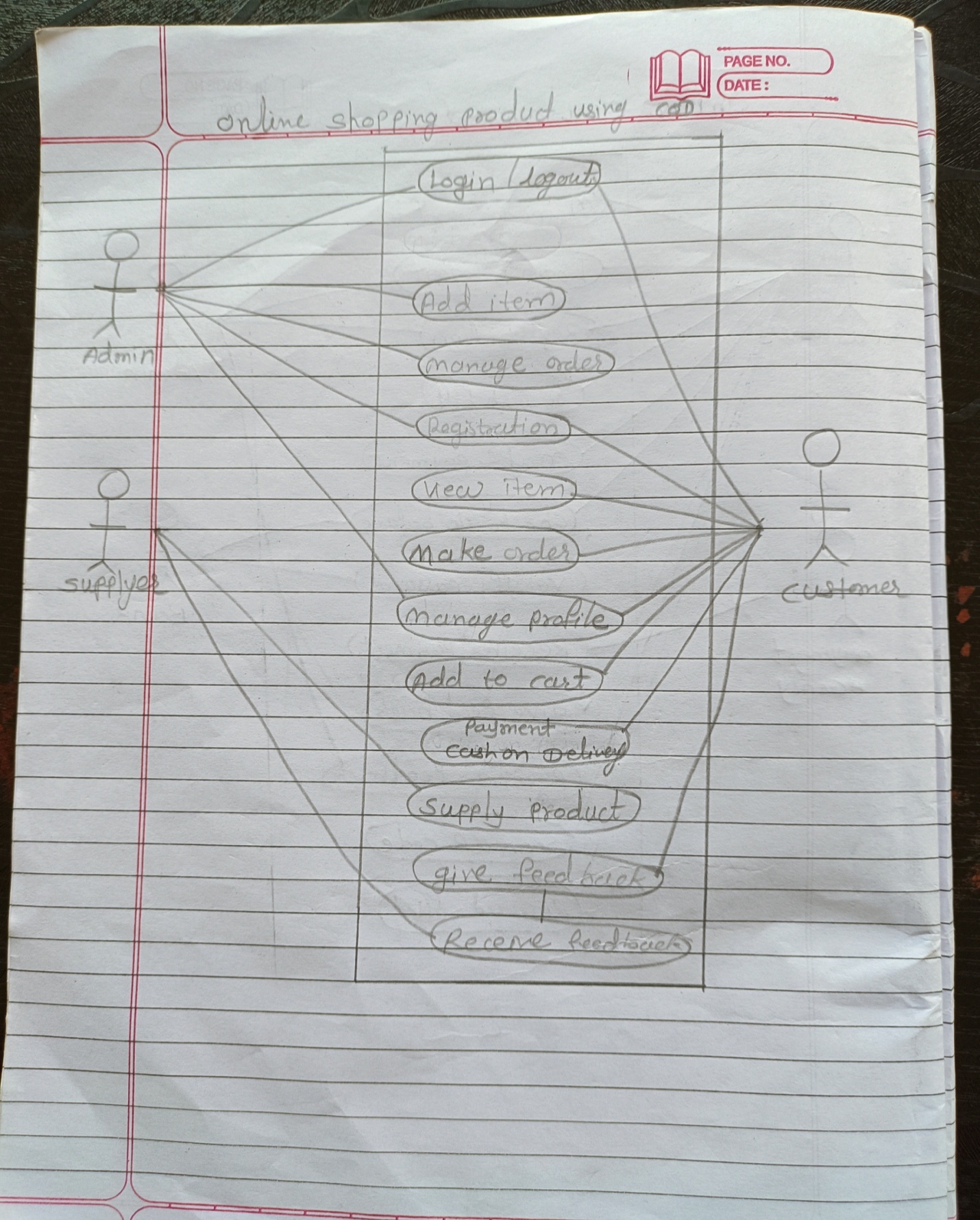
**Agile Model Work Flow**

**Pros**

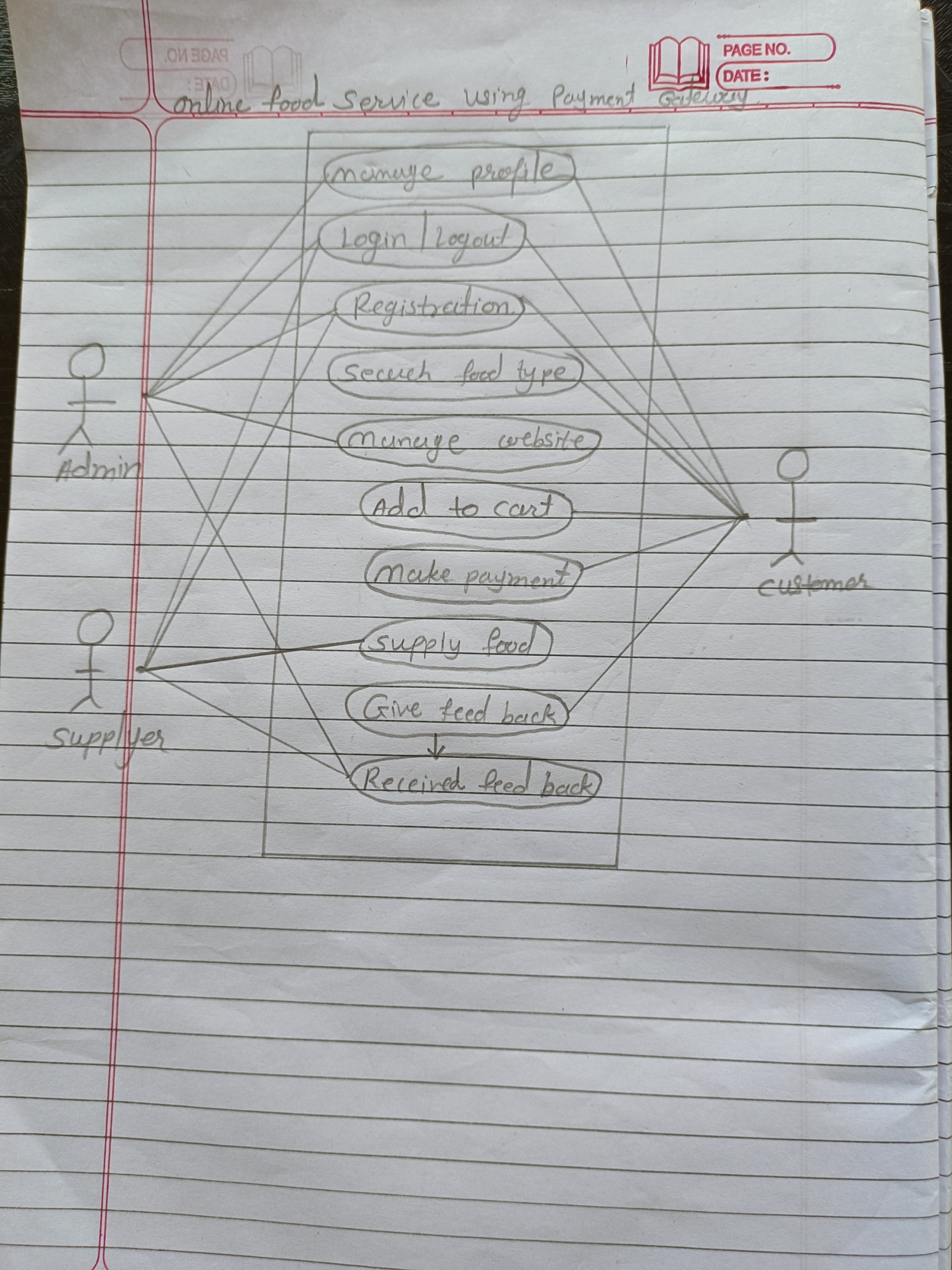
* Is a very realistic approach to software development
* Promotes teamwork and cross training.
* Functionality can be developed rapidly and demonstrated.
* Resource requirements are minimum.
* Suitable for fixed or changing requirements
* Delivers early partial working solutions
* Good model for environments 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
* **Draw use-case online shopping product using COD**

**D**

* **Draw Use-case on online product using payment gateway**

****