**Task 01**

**1 1 What is System Development Life Cycle (SDLC)**

**Introduction**

The Software Development Life Cycle (SDLC) is the process through which normal business practices are applied to the creation of software applications Planning, requirements gathering, design, development, testing, deployment, documentation and maintenance are the main processes that are typically followed

**Steps in SDLC**

* Planning
* Identifying Requirements
* Constructing Product Architecture
* Developing the Product
* Testing
* Deployment
* Operations and Maintenance

**1) Planning**

The research for the product under consideration is a part of this step of the software development life cycle In this step, a thorough market analysis is completed and product viability is estimated An in-depth technical study is also performed while taking into account the products' technological elements The benefits and drawbacks of the software development methodology are examined in this step We may then try to increase the benefits and decrease the drawbacks once this has been determined Developers should often solicit user feedback in order to make sure that the customer's expectations are understood

Meetings with consumers, distributing questionnaires for them to complete, conducting surveys, etc are all examples of ways to get feedback When a consumer is unsure of what they want, the team's role is to make sure they give them the right information, get the right feedback, comprehend their wants and then offer a solution The most crucial action in this phase is getting the customer's consent

**Steps to follow in the planning stage**

1 Assemble the project team

2 Justify Project

3 Choose a development methodology

4 Develop a project schedule

5 Produce a project development plan

**2) Defining Requirements**

The process of defining requirements involves deciding what the application will perform and what it will need For instance, a social media application would need the ability to connect with a buddy A search function may be required while using an inventory programme The criteria also include a list of the resources needed to finish the project For instance, a group could create software to manage specialized industrial machinery Without the machine, the process is impossible

After the client chooses a software solution, the stages of the software development life cycle move forward with a careful analysis of the requirements The team looks over documents linked to the project and evaluates the ecology of the client right now This is one of the steps in the software development life cycle that some would consider transitional when coupled with planning and requirements analysis

**Steps to follow in defining the requirements stage**

1. Study the current system
2. Determine system requirements
3. Write requirements report

**3) Designing Product Architecture**

The creation of the specification design document (SDD) is connected to this phase of the software development life cycle The product architects are tasked with writing this document All the characteristics of the software program that the team will create are detailed in the contents of this paper Additionally, all of the software application's budgetary expenses are listed in this document, along with the time needed to develop each feature Additionally, the customer and all stakeholders must approve the design document software As I mentioned earlier, the approval of the stakeholders and customers is crucial because, in some circumstances, revisions to this document are necessary due to the constraints of the stakeholders

**Steps to follow in the designing stage**

1. Identify potential solutions
2. Evaluate solutions and select the best
3. Select hardware and software
4. Develop application specifications
5. Obtain approval to implement a new system

These are some of the design elements examples

* **Architecture**  The programming language to be used , business standards , the overall architecture and the use of any boilerplate are all stated
* **Communications**  A primary server or other instances of the software may be among the external assets with which the programme communicates A variety of communication protocols are used and they are all covered in this section
* **Platforms** Specifies the operating systems, such as Apple android , Windows , Linux , or even gaming consoles , that the software will run on
* **Programming**  Not just the programming language , but also the methods used by the application to solve issues and carry out duties
* **Security**  Describes the program's security features, such as password security, SSL traffic encryption and safe user credential storage
* **User-Interface** Explains how customers interact with the software and how it reacts to their input

**4) Developing the Product**

The longest phase in the software development life cycle is this one The software application is developed during this stage The team's developers begin and finish developing the software during this phase, including any features that were outlined in the specification design document The developers make numerous significant decisions during this phase, which is the most critical element of the software development life cycle For software development, they must select the right programming language Each stage in this phase must be meticulously carried out and the team's communication, especially between the developers and the quality assurance team, must be excellent

The quality assurance team's responsibility is to guarantee that the generated product meets the quality standards set by the clients Therefore, having effective communication between the development team and the quality assurance team will lead to a product that meets the needs of the client and that they will actually like using

**Steps to follow in the development stage**

1. Purchase and install hardware and software
2. Develop applications
3. Application Testing
4. Complete Documentation
5. User Training
6. Data Conversion to New System

**5) Testing the Product**

Prior to making the program accessible to users, testing is essential Security testing is one of many testing categories that can be automated Other testing strategies require a specialized environment, so for complicated deployments, think about creating a mimicked production setting

To ensure proper operation , It is important to carefully review each function Performance testing on the program's many parts is necessary to ensure that they all work seamlessly together and to eliminate any processing hangs or lags The testing process aids in lowering the number of defects and errors that users encounter User satisfaction and usage have therefore increased

**Steps to follow in the testing stage**

1. Test Planning
2. Test Case Designing and Development
3. Test Environment Setup
4. Test Execution
5. Test Closure

**6) Deployment**

While the application is being deployed, users can access it Many businesses want the deployment process to be automated This might be as straightforward as the company's website has a payment portal and download link Another possibility is the download of an app for a smartphone Deployment could be difficult as well Moving a company-wide database to a recently developed application is one example The upgrade may take longer than anticipated because the database is tied to so many other systems

**Steps to follow in the deployment stage**

1. processes and preparation for deployment
2. product introduction
3. transferring the product's ownership
4. the deployment phase's completion

**7) Operations and Maintenance**

The cycle of development is almost finished at this time The application is now being used in the field after completion Contrarily , the concept of operation and maintenance is still vital During this phase , users discover flaws that were overlooked during testing New development cycles could be necessary in order to overcome these challenges

Iterative development models provide updates that include patches as well as new features for the next releases A fresh development cycle may be started for each new release

**Steps to follow in the operational & maintenance stage**

1. Determining the issue
2. Examining potential solutions to the issue
3. Developing the required coding modifications
4. Verifying the issue has been resolved after the upgrades
5. Putting the modified coding into use in order to update the software with the

**Advantages of SDLC**

* Improved management oversight of the whole development project
* Clarity understanding about the requirements
* A shared vision of the final product among all parties
* Results like time, cost and output are predictable
* Decreased chances of getting errors

**Disadvantages of SDLC**

* The end-user doesn ' t learn the solution to what some people might view as an important issue until the system is practically complete
* Depending on how the developers perceive the user needs , they may or may not receive a system that fulfils them Loss of translation is a possibility
* It is difficult for users to analyze intermediate products and determine whether a certain product (such as a data flow diagram) satisfies their business needs
* It costs money and takes time to provide documentation and is hard to stay updated

**1 2 Types of SDLC Models**

Different software development life cycle models have been developed and are used to guide the software development process These models can be called " Software Development Process Models " To guarantee success in development process, each process model adheres to a set of stages particular to its type

**Waterfall Model**

The earliest and most basic of the structured SDLC methods is the Waterfall methodology , which calls for completing one phase before moving on to the next

Each phase has a separate project plan and is informed by the information from the phase preceding it The concept of waterfall is simple to grasp and manage The " Linear Sequential Model " is another name for this model

Figure 1 1 - Water Model

**Advantages of Waterfall Model**

* Each step has clear activities to follow, deliverables to meet and reviews to conduct, making management of each easy
* Simply to understand and very easy to use
* Recommended for small or mid sized projects
* Requirements are clear and fixed
* The stage remains simple and controllable thanks to sequential stages and no overlapping

**Disadvantages of Waterfall model**

* Higher Maintenance cost
* It's possible that the programming won't be competitive enough in the market to meet the standards
* Risk and uncertainty are very high
* Progress within stages is challenging to measure
* Cannot change requirements while the project is ongoing
* Not recommended for long term projects

**Suitable when,**

* The requirements are steady and well-documented
* There is no uncertainty because the requirements are explicit and comprehensive
* technology that is static
* short term project

### 

### **Iterative Model**

It doesn't take much planning to use this kind of model According to this idea, a method where the complete project is divided into smaller parts is required Iterations are each distinct small component Each iteration of the project, after being divided into smaller units called iterations, passes through all seven stages of the software development life cycle The best feature of this type of model is that after one iteration, a small chunk of the project is completed and may be released to the market Customers and other stakeholders gain a lot from this since they may test the product on the market and are involved in every step of its development

Figure 1 2 - Iterative Model

**Advantages of Iterative Model**

* Supports parallel development
* Progress can be measured
* Easy to test and debug
* The design is mainly concentrated and documentation effort is reduced
* Can change requirements while the project is ongoing
* It is less expensive when changing the requirements

**Disadvantages of Iterative Model**

* Not Suitable for small Sized projects
* Requires more planning and resources
* Highly skilled risk Analysis required
* For the later stages, risks cannot be predicted
* Higher management required

**Suitable when,**

* The demands for the finished software are clearly stated
* The project is large
* Clear-cut fundamental tasks may later be improved upon
* Market limitations are well-known and when the programme is delivered, they will still be relevant
* There is potential to incorporate new technology

**Spiral Model**

The waterfall model ' s regulated, methodical elements are combined with the spiral model ' s iterative development concept This spiral model , which places a strong emphasis on risk analysis , combines the waterfall model with an iterative development process model to create the spiral model Each spiral iteration enables either incremental new introductions or gradual enhancements to the item

Figure 1 3 - Spiral Model

**Advantages of Spiral Model**

* Highly Scalable Model
* New features can be added later on
* High-level risk management
* Helps users become involved in the development process early (Beta Program)
* A well-defined method can be used to adapt when changing requirements

**Disadvantage of Spiral model**

* Costly and time-consuming to develop the final software
* Risk management requires highly qualified professionals
* The rate of project failure increases with poorer risk management
* Unsuitable for quick, easy projects
* It takes more time and effort to document

**Suitable when,**

* Financial limitations
* Risk assessment is a top priority responsibility
* Requirements and long-term initiatives evolve with time
* The consumer himself does not completely understand the requirements
* Evidence of Concept is required

**Agile Model**

The agile method allows for the client to evaluate each development iteration ' s results and express his level of satisfaction The agile software development life cycle paradigm has this as one of its advantages One of its shortcomings is that in the absence of clearly defined needs , it is difficult to anticipate the resources and development costs Extreme programming is a real world example of how the agile paradigm is applied The fundamental unit of this idea is a sprint, which is a part of the Scrum process

Figure 1 4 - Agile Model

**Advantages of Agile Model**

* Focus is placed mainly on developing skill sets in a people-oriented manner
* High customer engagement consistently improves the quality
* Really adaptable and practical approach
* Early software that is only partially functional
* Tasks are dispersed fairly and with less resources
* Minimized risks

**Disadvantages of Agile Model**

* Cannot resolve pricing issues
* Significant customer interaction is needed
* Required more dedication and time
* Projects can drag on forever because there is no obvious conclusion

**Suitable when,**

* Rapid changes in customer requirements
* Early investment rewards
* Clients don't have an exact financial limit or schedule
* Prior to any rivals entering the market, you must seize control

**V Model**

The V approach is a variation on the waterfall approach where development and testing are done concurrently The V Model takes the stages after implementation in an upward direction, making a "V" shape, as opposed to having the stages in a linear manner

Early test planning distinguishes the V Model from the Waterfall Model

All of the validation activities (unit testing, integration testing, system and acceptance testing) are on the right side of the V Model, whereas all of the verification activities (requirement analysis, architecture, design) are on the left The Development Stage ties these two sides together

Figure 1 5 - V Model

**Advantages of V Model**

* Can identify issues when testing earlier
* Gaps in requirements might be found quite early
* The milestone can be attained sooner
* Small, quick projects do not profit from the V Model
* Because the stages are so well-structured, they are simple to control and manage

**Disadvantages of V Model**

* The end of the life cycle is when working software becomes available
* Lack of flexibility and rigidity
* V Model is insufficient for complex and object-oriented programmes
* Current projects receive no benefits
* It is not possible to support changing requirements to a greater extent

**Suitable when,**

* The project team is aware of the technology and it is static
* Clearly stated requirements
* Short-term initiatives
* Experts with high levels of ability are required
* Testing in advance is necessary

**Incremental Model**

The incremental model , sometimes referred to as the successive version model , is a popular method for developing software in which the requirements for the product are divided or broken down into a number of independent modules or increments throughout the SDLC Every increment follows the SDLC incremental paradigm and is regarded as a sub project throughout This has an iterative model-like ring to it The incremental model is also known as the iterative enhancement model because this model is an improvement over the iterative model Instead of taking a giant stride forward , we accomplish our goals using the gradual strategy

**Types of Incremental Model**

* Staged Delivery Model
* Parallel Development Model

Figure 1 6 - incremental model

**Advantages**

* During the software life cycle, the software will be produced swiftly
* Requirements and limitation changes are flexible and less expensive
* Changes may be made at any stage of development
* In comparison to other models, this one costs less
* Errors can easily be noticed

**Disadvantages**

* It requires careful planning and design
* Problems may arise because not all requirements are gathered up front for the full programme lifecycle owing to system architecture
* The iteration phases are distinct and do not overlap
* Correcting an issue in one unit necessitates fixing the problem in all the units, which takes a lot of time

**Suitable when,**

* The system's requirements are grasped in full
* when a need for a product's early release materializes
* when the software engineering crew is undertrained or poorly skilled
* when high-risk characteristics and objectives are present
* Companies that focus on web applications and products are more likely to employ this model

**1 3 Suggested SDLC model for the proposed solution**

Eco Farm Pvt Ltd is a modern and innovative ecologically responsible agricultural business committed to ethical and sustainable farming methods The company places a significant emphasis on ethical and sustainable farming For all these years, Eco Farm has needed a proper farm management system to provide excellent client service and maintain customer records So in my opinion, the Agile model is the best suit for this system

**1 4 Mapping out the stages of the selected model**

1. **Requirements Gathering**

Pick out the most essential requirements, such as crop management, equipment tracking, weather forecasting, animal management and reporting and prioritize them in order

1. **Planning**

Since the beginning, Eco farm Pvt Ltd had to face various issues like, lack of sustainability, poor productivity due to higher costs, unnecessary decisions made by the farmers etc so to these problems they’ve decided to implement a new farm management system implementing this can enhance productivity, reduce costs, improve sustainability and empower farmers to make data driven decisions

* Registering new farmers
* Updating payments and methods (credit card or cash)
* Tracking the equipment by maintaining a record book (which includes the date that the equipment was taken and the handover date )
* Option to track livestock and Inventory statues of the farmers
* Weather forecasting news feed to know about climate changes

1. **Development**

When it comes to development, make sure that you’ve covered all the requirements mentioned And also organizing daily standup meetings to discuss the challenges, solutions and to track progress Usually the development team has a fixed duration called sprints (like 2 or 3 weeks) to implement the system (user stories)

1. **Testing**

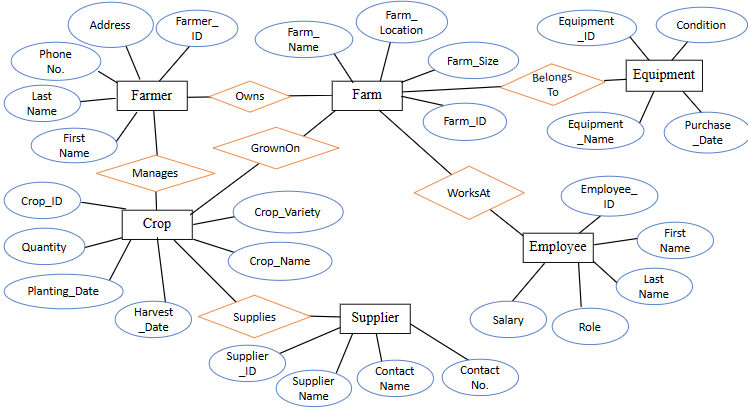
Requires continuous testing procedure throughout the development process in order to get successful output as expected Also can use modern testing tools to make sure the system is running properly without any other issues After testing, can introduce an experimental version of the system to showcase the completed features to stakeholders and gather feedback

1. **Deployment**

After ensuring that the system is stable and meets the necessary quality standards, can produce new system deployments This phase involves the process of making the system accessible and operational for its users

**Task 02**

**2 1 Entity relationship Diagram (ER)**



**2 2 Data Dictionary for the ER Diagram**

**Farmer**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Data Name** | **Description** | **Data Type** | **No of Characters** | **Primary Key** | **Foreign Key** |
| **Farmer ID​** | A specific number to identify each farmer ​ | Int​ | 06​ | Yes​ | No​ |
| **First Name​** | First name of the farmer ​ | Var​ | 30​ | No​ | No​ |
| **Last Name​** | Last name of the farmer ​ | Var​ | 30​ | No​ | No​ |
| **Address​** | Farmer's residential address ​ | Var​ | 30​ | No​ | No​ |
| **Phone No​** | Farmer's contact No ​ | Int​ | 15​ | No​ | No​ |

**Farm**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Data Name** | **Description** | **Data Type** | **No of Characters** | **Primary Key** | **Foreign Key** |
| **Farm ID** | A specific number to identify each farm ​ | Int​ | 10 | Yes​ | No​ |
| **Farm Name** | Name of the farm ​ | Var​ | 30​ | No​ | No​ |
| **Farm Location** | The location where the farm is located ​ | Var​ | 30​ | No​ | No​ |
| **Farm Size** | The total size of the farm in acres ​ | Int | 10 | No​ | No​ |

**Crop**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Data Name** | **Description** | **Data Type** | **No of Characters** | **Primary Key** | **Foreign Key** |
| **Crop ID** | A specific number to identify each crop | Int | 08 | Yes​ | No​ |
| **Crop Name** | Name of the crop | Var | 30 | No​ | No​ |
| **Crop Variety** | The specific variety or strain of the crop | Var | 20 | No​ | No​ |
| **Planting Date** | The date when the crop was planted | Int | 10 | No​ | No​ |
| **HarvestDate** | The expected or actual date of crop harvest | Int | 10 | No​ | No​ |
| **Quantity** | Quantity of the crop | Int | 10 | No | No​ |

**Equipment**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Data Name** | **Description** | **Data Type** | **No of Characters** | **Primary Key** | **Foreign Key** |
| **Equipment ID** | A specific number to identify each equipment | Int | 06 | Yes​ | No​ |
| **Equipment Name** | Name of the equipment | Var | 30 | No​ | No​ |
| **Purchase Date** | The date when the equipment was purchased | Int | 10 | No​ | No​ |
| **Condition** | The condition or maintenance status of the equipment | Var | 20 | No​ | No​ |

**Employee**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Data Name** | **Description** | **Data Type** | **No of Characters** | **Primary Key** | **Foreign Key** |
| **Employee ID** | A specific number to identify each employee | Int | 10 | Yes | No |
| **First Name** | First name of the employee | Var | 30 | No​ | No​ |
| **Last Name** | Last name of the employee | Var | 30 | No​ | No​ |
| **Role** | Role of the employee | Var | 30 | No​ | No​ |
| **Salary** | The salary of the employee | Int | 10 | No​ | No​ |

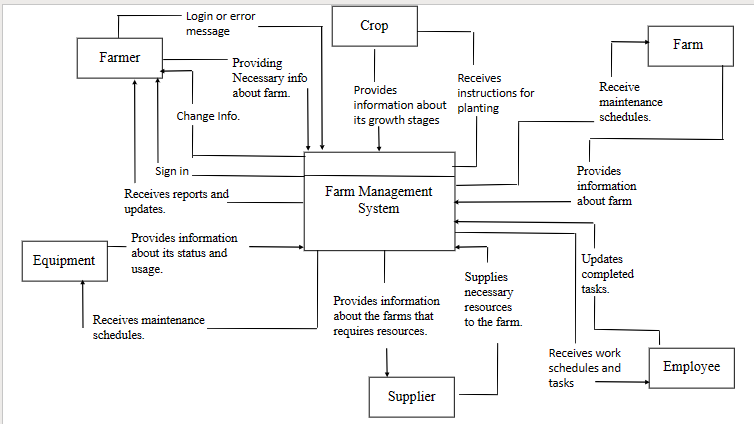
**Supplier**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Data Name** | **Description** | **Data Type** | **No of Characters** | **Primary Key** | **Foreign Key** |
| **Supplier ID** | A specific number to identify each supplier | Int | 10 | Yes | No |
| **Supplier Name** | Name of the supplier | Var | 30 | No | No |
| **Contact Name** | Name of the contact person at the supplier | Var | 30 | No | No |
| **Contact No** | The contact number of the supplier | Int | 20 | No | No |

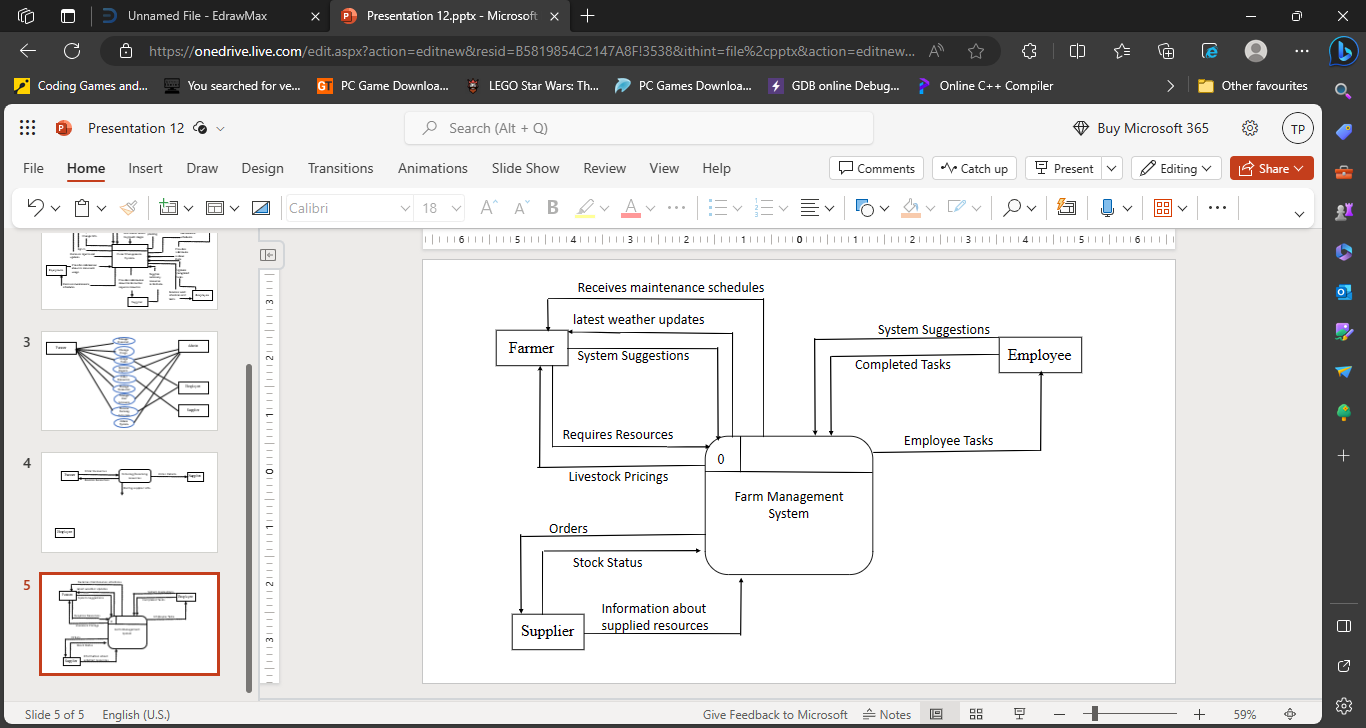
**Task 03**

**3 1 Structure System Analysis and Design**

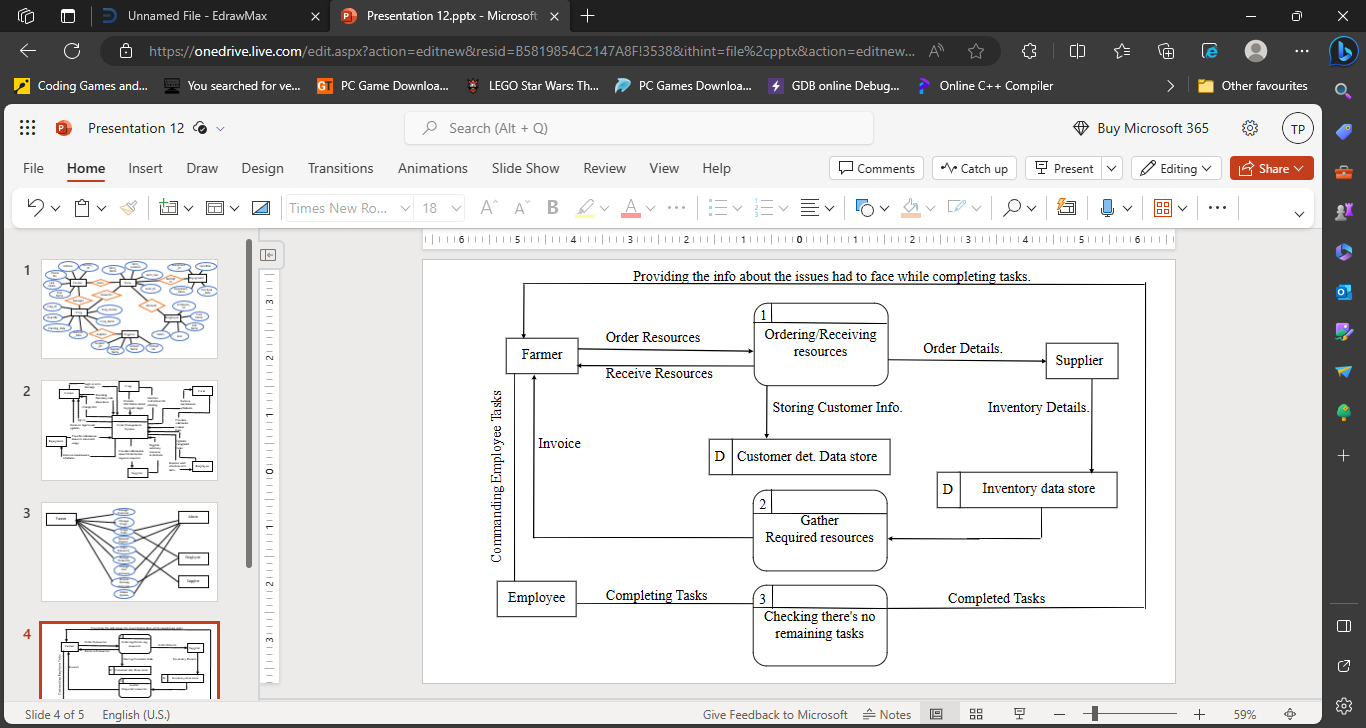
**3 1 1 DFD Context Level Diagram**



**3 1 2 DFD Level 0 Diagram**

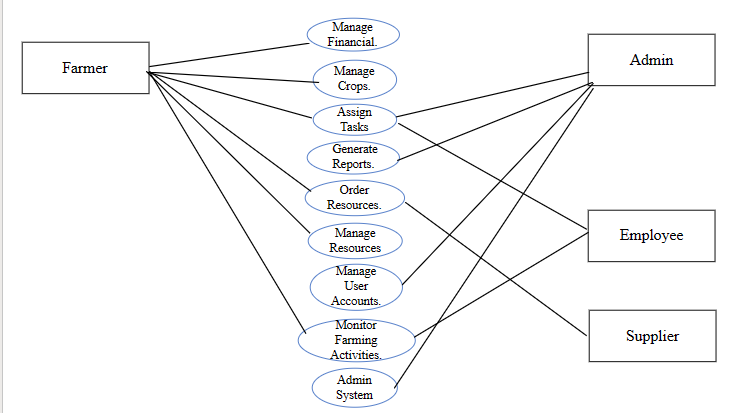


**3 1 3 DFD Level 1 Diagram**



**3 2 Object Oriented Analysis and Design Methodology**

**3 2 1 Use Case Diagram**



**3 2 2 Class Diagram**

