LONDON METROPOLITAN UNIVERSITY

Islington college
(इस्लिङ्टन कलेज)

Module Code & Module Title

CS5002NI SOFTWARE ENGINEERING

Assessment Weightage & Type

35% Individual Coursework

Year and Semester

2022-23 Spring

Student Name:Aasutosh Kumar Verma

London Met ID:22085760

College ID:NP01AI4S230020

Assignment Due Date:5/7/2024

Assignment Submission Date: 5/6/2024

Title (Software Engineering):McGregor Institute of Botanical Training

Word Count (Where Required):6270

I confirm that I understand my coursework needs to be submitted online via Google Classroom under the relevant module page before the deadline in order for my assignment to be accepted and marked. I am fully aware that late submissions will be treated as non-submission and a marks of zero will be awarded.

Abstract

This initiative is an platform designed for individuals who have a keen interest, in botany. It provides an array of functions to nurture a community of plant lovers. Participants can register for courses ranging from short term certifications to in depth undergraduate and postgraduate programs. Additionally the platform offers the option to purchase a selection of plants enhancing their experience. A distinctive feature of this system is its capability for users to receive expert advice based on their requirements, such as selecting suitable plants, for their local soil conditions. Moreover the platform promotes community interaction through a forum where users can exchange ideas pose queries and gain knowledge from one another. The primary goal of this endeavor is to establish a informative and cooperative environment dedicated to all things related to botany.

Acknowledgements

I am profoundly grateful to Rubin Sir, Ujal Sir, and Ishan Sir for their unwavering support and expert guidance throughout my project. Their knowledge and insights have been a guiding light, enriching my work and helping me overcome challenges.

Ujal Sir’s practical approach to Agile, Scrum, and report preparation, backed by real-world examples, has significantly enhanced my understanding of these areas. His contributions have been pivotal in the successful completion of this project.

Rubin Sir’s detailed lessons on program cost calculation and the Constructive Cost Model (COCOMO) have been invaluable. His informative videos and thorough explanations have equipped me with essential skills and highlighted the practical implications of these concepts.

Ishan Sir’s comprehensive teachings on various testing methodologies have been instrumental in my project. His emphasis on conducting the majority of testing before launching the main function and allocating a portion for maintenance has provided a practical framework for efficient project management.

Additionally, Rubin Sir’s insightful teachings on use cases have significantly enhanced my understanding of system design and requirements gathering. His guidance has been instrumental in helping me grasp the practical applications of use cases.

I am deeply thankful for their guidance and support. Their contributions have made a significant difference, and for that, I am extremely grateful.

Table of Contents

1. Introduction

Welcome to the McGregor Institute of Botanical Training, a unique place where the green thumbs of Ireland and Nepal unite. Nestled in the heart of Godawari, Lalitpur, we've been nurturing minds and plants for nearly seven years. Our roots are in Ireland, but our branches extend to Nepal, offering a diverse range of undergraduate and postgraduate courses in agriculture and horticulture, all under the esteemed affiliation of Dublin City University.

Recently, we've noticed a blossoming interest in agriculture, and we're excited to cultivate this passion further. We're introducing a variety of short-term certification courses in horticulture, open to anyone with a keen interest in the field. But our vision extends beyond academics. We're also planning to offer a wide variety of plants for sale, some for a nominal fee and others completely free!

Our dream is to grow a community of plant enthusiasts, providing a platform where ideas can be shared, rare plants and forests can be protected, and questions can be answered by experts. This forum will serve as a fertile ground for discussions, collaborations, and growth.

Our proposed system includes features such as user registration, program enrollment, plant purchasing, payment processing, expert recommendations, report generation, certification exams, and a vibrant forum for discussions. We're excited to embark on this journey with you, nurturing both your knowledge and love for plants. Welcome to our community!

2. Methodology

2.1. Agile Methodology

The proper process study or analysis of all the steps, methods used to develop a
software us known as methodology. There are many methodologies to develop a
software such as waterfall methodology, Prototype methodology, Spiral methodology, Agile methodology, Kanban and many more. Among these I as a
project manage, I have chosen a Agile methodology for the development of a
software.

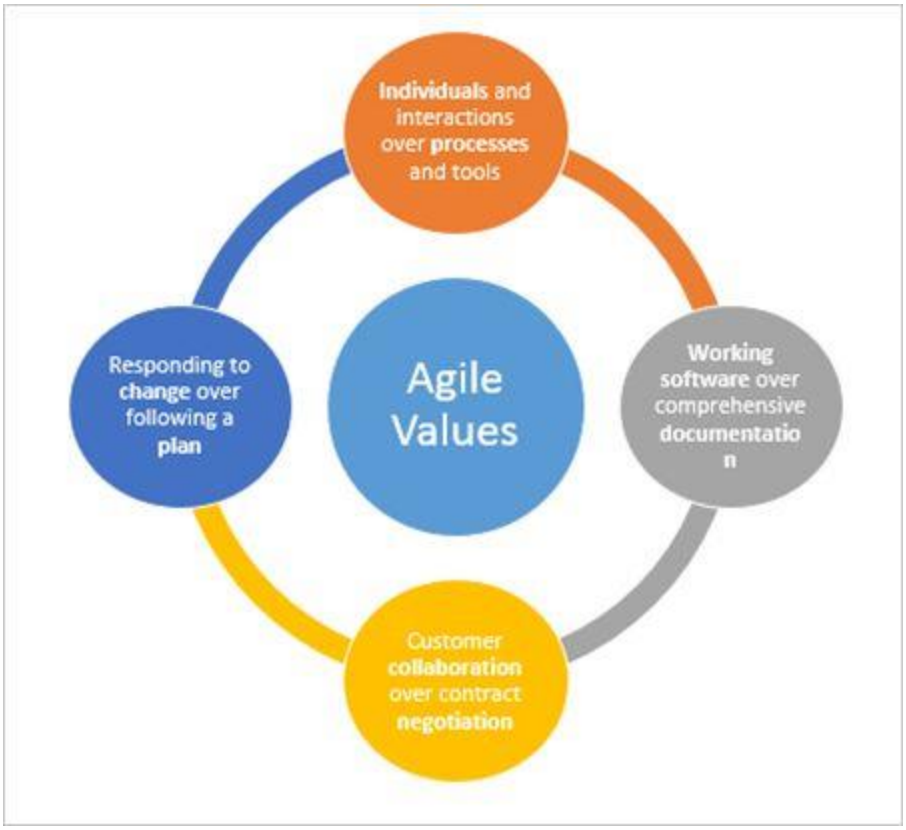


Figure 1Agile

A group of software engineers who felt that the conventional development

process was overly complex and burdened by documentation requirements came up
with the Agile methodology as a better alternative. The Agile Manifesto, the group's
founding statement, contained 12 principles and 4 guiding ideals for the Agile
concept. Software development may be organized around collaboration, iteration,
learning, and value delivery using agile techniques. Development teams tackle
complex projects in small chunks and time-boxed cycles. The goal is for your team to
have a distinct, coordinated approach to task definition and completion to increase
flexibility, shorten time to market, and produce software of higher caliber. (Aha!,
2020)! Inside Agile I feel for my project, scrum methodology is best for software
development. Scrum is an Agile framework that focuses on cross-functional
teamwork, accountability, and iteration in order to develop, deliver, and support
complex products. The Scrum framework is organized into key roles, events, and
artifacts.

1. Scrum roles

a. Product Master

The scrum master is an authority on the scrum process, as his name indicates. He ensures that each member of the scrum team is aware of how the framework functions and aids in their adaptation to the agile

environment. He is the scrum master. (Westland, 2022)

b. Product Owner

The scrum product owner controls sprint planning keeps track of the product log, and actively participates in scrum meetings. Because they oversee backlog grooming and prioritize user stories to facilitate greater cooperation, they essentially take on the role of a project manager. (Westland, 2022)

2. Scrum events

a. Sprint Planning

Teams decide how to accomplish this goal by starting with the highest priority items in the product backlog. Do your research and only start with
items that are ready when preparing a sprint. Additionally, keep in mind that planning is a quick process and avoid getting caught up in the particulars. Just start working toward achieving the goals. Keep the strategy cooperative. The team should additionally query the stakeholder
and product owner. (Westland, 2022)

b. Daily Scrum Meeting

Every member of the scrum team gathers for these 15-minute sessions to
discuss the tasks they will be working on during the day and to share any
challenges they may be encountering. There is no need to extend the duration of the daily scrum meeting because additional in-depth discussions may be had during sprint reviews and retrospectives. (Westland, 2022)

c. Scrum Review

You should reflect on the sprint and identify what succeeded and failed. The knowledge may then be used to improve current sprints and duplicate successful ones while minimizing unsuccessful ones. Set the tone for the meeting by thanking everyone who participated, providing brief introductions, and starting the sprint review process. (Westland, 2022)

d. Scrum Retrospective

The scrum team has a chance to consider the previous sprint and assess what went right and wrong at the sprint retrospective meeting. In order to prioritize user stories and enhance product performance, stakeholders' and customers' opinions are also solicited. (Westland, 2022)

3. Scrum Artifacts

a. Product backlog

The product owner will create a list of all the work that needs to be done and rank it in priority order. This creates the backlog for your projects. They accomplish this by deciding which products are necessities, which are less important, and which can wait until a later time. That implies that each item's worth must be obvious. What is the item's impact, risk, and potential contribution to learning? (Westland, 2022)

b. Sprint Backlog

The user stories that the scrum team will be working on during a single sprint make up the sprint backlog, to put it simply. It's crucial to ensure that

the most essential user stories are continually being worked on and that none of them go overlooked. (Westland, 2022)

c. Product Increment

The phrase "product increment" refers to all of the product backlog items that have been finished during a sprint as well as the total of all of the user
stories and backlog items that have been finished. (Westland, 2022)

3. Gantt Chart/WBS

A chart which includes all the information about the timing, scheduling, and
monitoring the project which is in horizontal bar graph. It is the most useful chart
to plan and meet the deadline and present outstanding output. Gantt chart is
usually made at the beginning of the project as project managers and team
members can view the task schedules, dependencies, and progress. There ar3e
different methodologies which can used to develop one’s project such as
prototype methodology, Waterfall methodology, Spiral methodology, Agile

methodology. Recently, the most used methodology to develop the software is
Agile methodology. I have chosen to use Agile methodology to develop "Allgemen Book and rental" project. The benefits of agile methodology are:

1. Greater stakeholders' engagement
2. Predictable costs and scheduling
3. Flexibility amidst change
4. High quality products

There are different methodologies inside agile methodology. Among them I have
choose Scrum methodology. Similarly, I have created the Gantt chart according
to the scrum methodology. While making scrum methodology gantt chart we
need to make it include scrum roles, scrum events and scrum artifacts. As the
sprints plays the main role in scrum methodology as it provides freedom for all
the team members to respond for every kind of changes. No one is responsible
for specific things. Sprint is short period of time where the team works to
complete the tasks.
I have used Sprint-1, Sprint-2, Sprint -3 and Sprint -4 … where the time period is
separated for login, admin, purchase , joining and payment.etc. Scrum has one specific
benefit that after every finish of one sprint there is always evaluation step. At

every evaluation step we can recheck all the progress and rechange at right
time. This does not need to wait for the longer time to review. We can review and
make changes at every one sprint end.

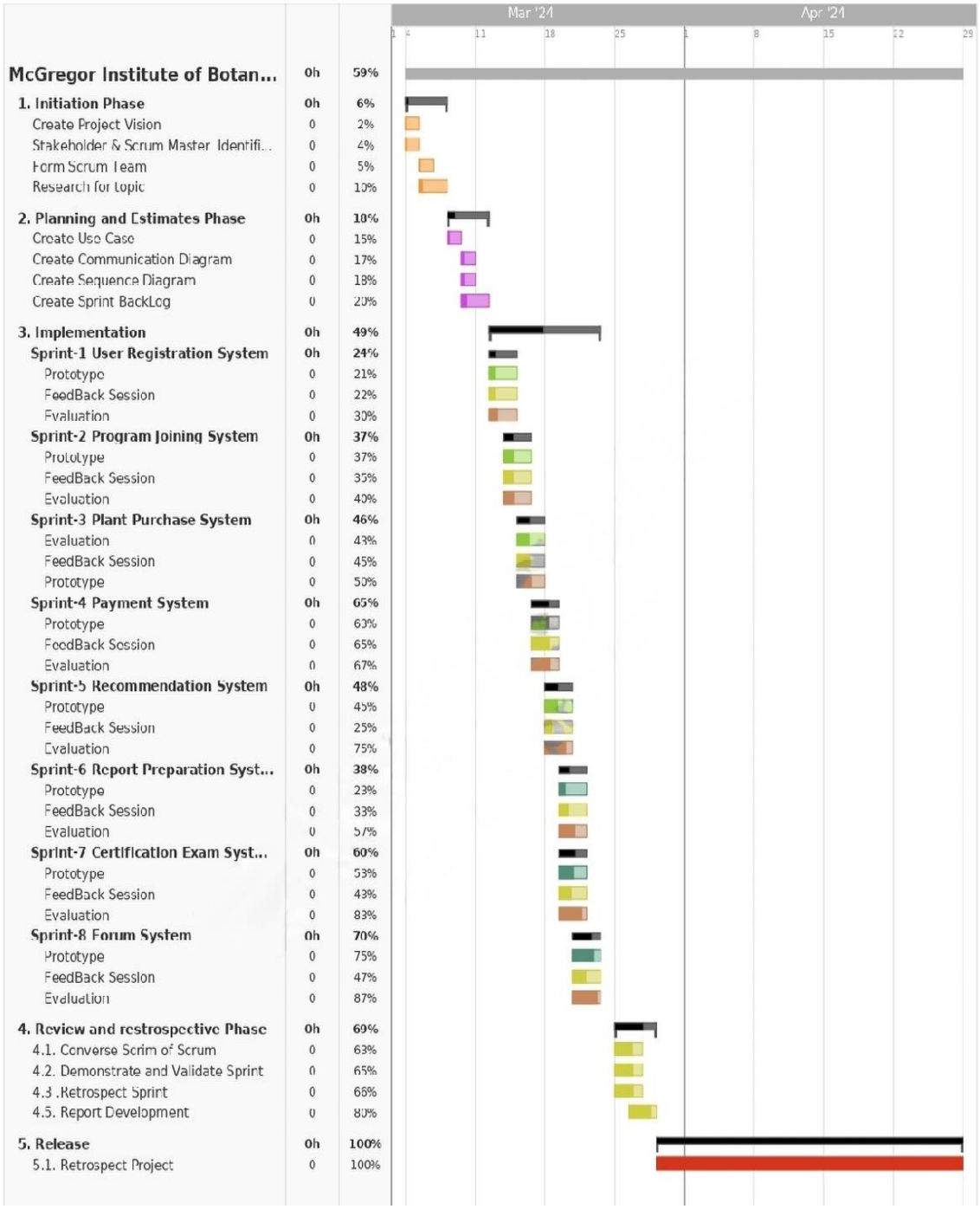


Figure 2 Gantt Chart

A Work Breakdown Structure (WBS) is like a roadmap for our project. Imagine we are planning a road trip. we wouldn't just hop in the car and start driving, right? we first decide where we want to go, then we figure out the best route to get there, and along the way, we identify the landmarks or stops we want to make.

In the context of software development, the final destination is the completion of the software project. The route consists of the main tasks or phases of the project, such as design, development, testing, and deployment. The landmarks or stops are the smaller tasks or deliverables within these phases.

So, a WBS is essentially a detailed map of your project, breaking down the journey from start to finish into manageable chunks. This makes the project easier to understand, plan, and manage. It's a crucial tool for project managers and teams, helping them stay organized and on track.

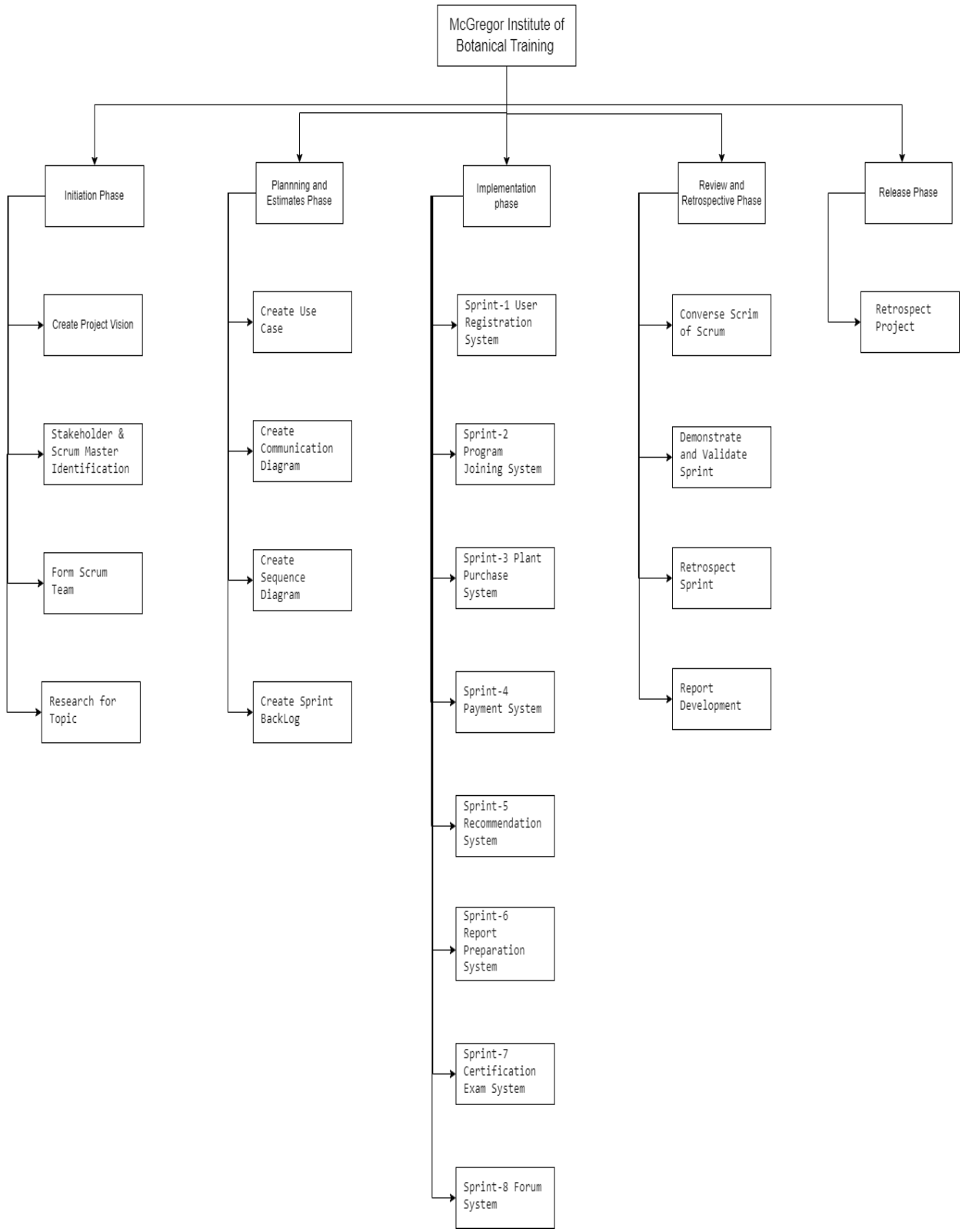
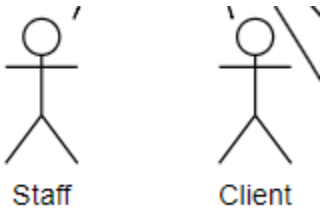

McGregor Institute of Botanical Training
Initiation Phase
Create Project Vision
Stakeholder & Scrum Master Identification
Form Scrum Team
Research for Topic
Plannning and Estimates Phase
Create Use Case
Create Communication Diagram
Create Sequence Diagram
Create Sprint BackLog
Implementation phase
Sprint-1 User Registration System
Sprint-2 Program Joining System
Sprint-3 Plant Purchase System
Sprint-4 Payment System
Sprint-5 Recommendation System
Sprint-6 Report Preparation System
Sprint-7 Certification Exam System
Sprint-8 Forum System
Review and Retrospective Phase
Converse Scrim of Scrum
Demonstrate and Validate Sprint
Retrospect Sprint
Report Development
Release Phase
Retrospect Project

Figure 3 WBS

4. Use case Diagram

4.1. Use Case Diagram

The information about a system and its users may be distilled into a use case
diagram. It is often displayed as a visual picture of interactions among different
parts in a system. Use case diagrams will identify the events in a system and
how those events flow but use case diagram does not indicate how those events
are implemented. Use case diagram usually describes and showcase the
objectives of user and system's aim. It gives a description of the sequence of
events that happens when users engage with a system. (techtarget, n.d.) thus
the use case diagram consist of actors, use case and
The components of a use case diagram are described below:



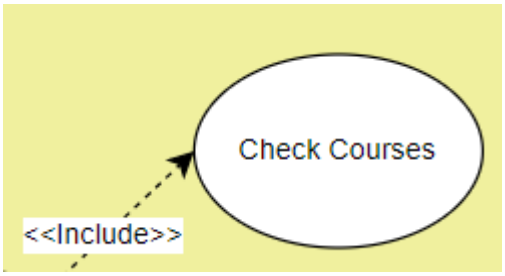
1. Actors

An actor portrays any entity (or entities) that perform certain roles in each system. The different roles the actor represents are the actual business
roles of users in each system. An actor in a use case diagram interacts with a

use case. For example, for modeling a organizations application, a customer entity
represents an actor in the application.

2. Use Case
An individual business functionality of a system is represented visually by a use case in a use case diagram. Here, "distinct business functionality" is
the important phrase. To choose a business process as a potential candidate
for modeling as a use case, you must ensure that the business process is
discrete in nature. (e-education.psu.edu, n.d.) it is the System's Function which is named by verb. Each actor must link to a use case while some use
cases may not be linked to actors.

3. Boundary of system –
It is the rectangle box to the system which is bounded to overall system.

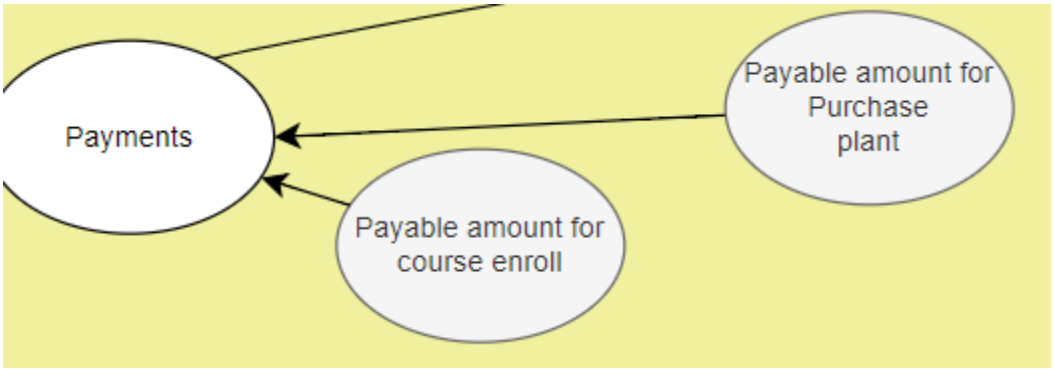


4. Include

Include shows the relationship between the use cases. It includes the use case which is must for the use case. For example, Login includes User Authentication.

5. Extend

Extend shows the relationship between the use cases. It includes the use case which is optional. For example: if the calling extends adding a call.



6. Generalization

The child use case inherits the behavior and meaning of the parent use case. • The child may add to or override the behavior of its parent.

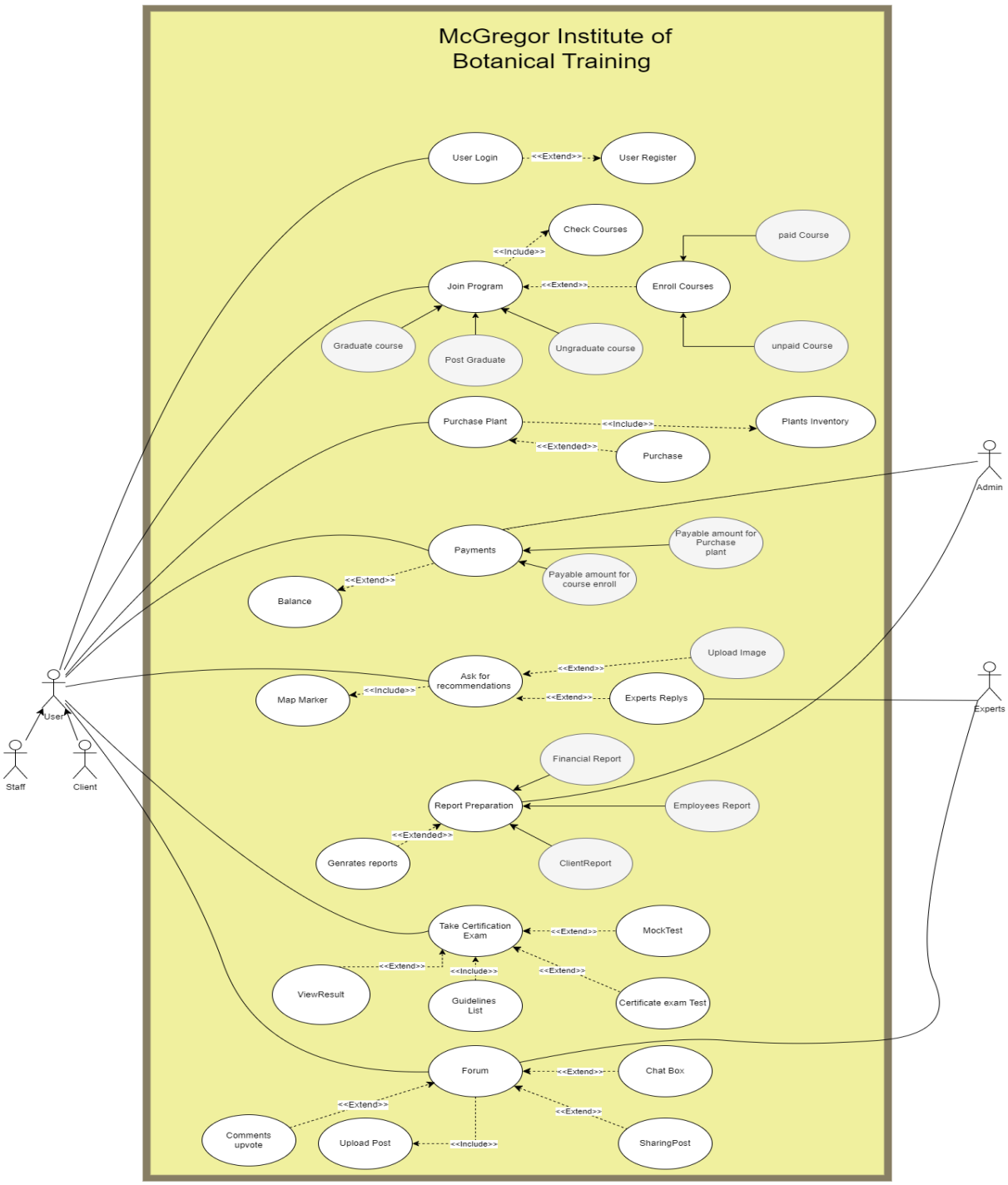


Figure 4 USECASE

4.2. High Level Use Case Diagram

4.2.1. Register user

Use case: Login

Actor: Users

Description: The user attempts to log in to the system using their credentials.

If the user is not registered or their login attempt fails, they are request for registration process where they can create a new account.

4.2.2. Join the program

Use case: Join the program

Actor: User

Description: Users can enroll in the program based on their academic qualifications. Once enrolled, they have the option to select from a variety of both paid and unpaid courses, depending on the specific program they have joined.

4.2.3. Purchase plant

Use case: Purchase plant

Actor: User

Description: Users have the ability to search and select plants to know their rates and information. If they find a plant they're interested in, they also have the option to purchase it.

4.2.4. Payments

Use case: Payments plant

Actor: User

Description: Once users have selected plants to purchase or a paid course to enroll in, they are required to complete the payment process

before they can finalize their purchase or enrollment. All payments are recorded for future analysis.

4.2.5. Ask for recommendations

Use case: Ask for recommendations

Actor: User,Expert

Description: Users are required to provide the location of their site when seeking recommendations. If they wish to obtain more detailed information, they have the option to upload images. Based on the user's data and any uploaded images, experts can provide the information the user is seeking.

4.2.6. Report Preparation

Use case: Report preparation

Actor: Admin

Description: The admin has the ability to generate reports to gain insights into the company's financial details, employee information, and monthly client details. This data is crucial for business analysis and the analytical process."

4.2.7. Take certification exam

Use case: Take certification exam

Actor: User

Description: Users have the opportunity to take certification exams or mock tests. However, they must familiarize themselves with the exam guidelines before attempting any exam.

4.2.8. Forum

Use case: Forum

Actor: User,Expert

Description: There is a community feature available for users where they can post comments. If users find certain comments helpful, they have the option to express their appreciation through ‘likes’. Experts can also engage in these discussions by replying to user comments and providing suggestions. However, users need to create a post before they can start commenting. If users find a post particularly interesting or useful, they have the option to share it.

4.3. Expanded Use Case Diagram

4.3.1. Join the program

Use case: Join the program

Actor: User

Description: Users can enroll in the program based on their academic qualifications. Once enrolled, they have the option to select from a variety of both paid and unpaid courses, depending on the specific program they have joined.

Typical course of Events:

Table 1 JOIN THE PROGRAM

Actor Action	System Response
1. User open system.Select join the program form system category.	2.The system display list of Program .
3. User select program based on their Qulification or degree.	4. The system display a list of paid and unpaid courses based on user's selection..
5. User selects a course..	6. The system checks if the selected course is paid. If it is, the system calculates the payment amount and sends it to the user.
7. User received payments amounts for final process before joining program.	
8. User pays the amount as per mention to enroll.	9. The system receive amounts notifying admin and save it for future record and creates a new users inside progam.
	10. System send user a notification that he/she has joined the program.

	11. User receive notification for program enrolled.	

Alternative Courses:

Line 3:- The new user didn’t see a suitable program based on qualifications.Use case ends,

Line 5:- The new user didn’t like any courses inside program.Use case ends.

Line 8:- The new users didn’t want to pay for program.Use case ends.

4.3.2. Purchase Plant

Use case: Purchase plant

Actor: User

Description: Users have the ability to search and select plants to know their rates and information. If they find a plant they're interested in, they also have the option to purchase it.

Table 2 PURCHASE PLANT

Actor Action	System Response
1. User open system.Select Purchase plant form system category.	2.The system display search area for plant .
3. User search and select plants as per their choice.	4. The system display plants rates and information.
5. User make purchases.	6. The system checks if the selected plants is in stock. If it is, the system calculates the payment amount and sends it to the user.
7. User recived payments amounts for final process before purchase of plants.	
8. User pays the amount as per mention to purchase plant.	9. The system receive amounts and save it for future record and notify admin about plant sold.
	10. System send user a notification that the amount has been received.

Alternative Courses:

Line 3:- The new user didn't see a suitable plants to purchase.Use case ends,

Line 5:- The new user selected plants is not in stock.Use case ends.

Line 8:- The new users didn’t want to pay for plants.Use case ends.

5. Communication Diagram/Collaboration Diagram

Communication Diagram also known as collaboration diagram is one of the
architectural designing steps of the software development which elaborate the
relationship and interaction between objects to show the behavior of a particular
use case. It elaborates the part of a use case. It depicts a pattern of interaction
between objects and identifies the participants by their connections to one
another and the messages they exchange. It models the message outgoing and
incoming by different applications.

The components of communication diagram are explained below:

1. Object

It is represented as a rectangle.

2. Actor

It is represented as a stick figure. It starts the communication and has unique
role and name.

3. Links

It is a connecting line between the actors or objects which shows the relationship between the objects.

4. Messages

It is the arrow pointing from client to the supplier object. Messages are represented order wise which gives the flow of information step wise.

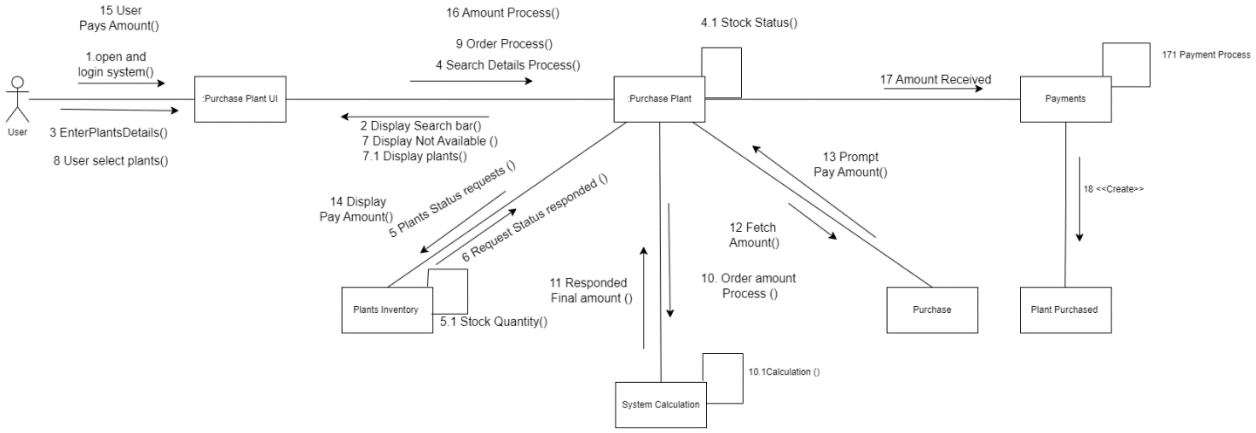


Figure 5 COMMUNICATION

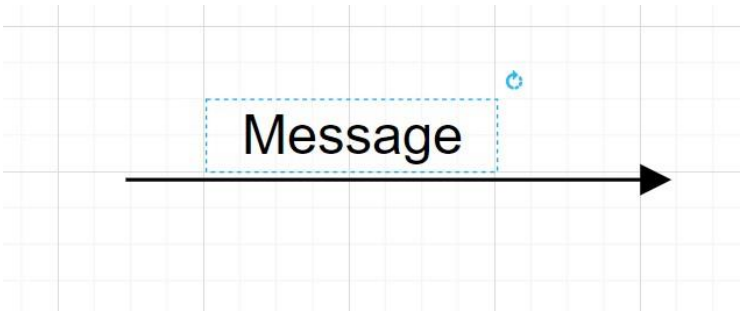
6. Sequence Diagram

Sequence Diagram is only also one of the architectural designing steps to
develop software. There is a connection between communication diagram
and sequence diagram. It represents the timeline or the flow of message between objects which is usually top to bottom. It is also elaboration of the
part of a Use Case diagram. Here I have chosen to make sequence diagram
of Purchase Plant.

The components of Sequence Diagram are explained below:

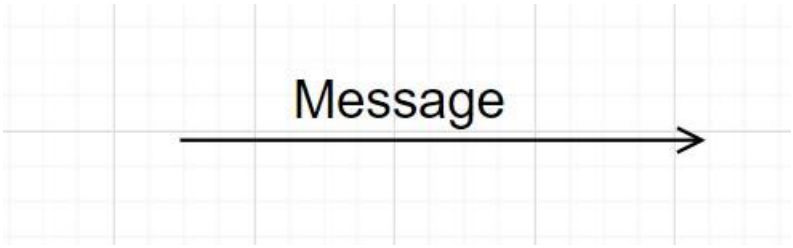
1. Activation Bar
It is the rectangle placed on the lifeline. The length represents the duration.
2. Object Lifeline
It is vertical dashed line. It represents the time of presence.
3. Messages
It is the arrow from caller to receiver which is in any direction whether it may be from caller or from receiver. There are different types of messages in sequence diagram which are explained below:

a. Synchronous Message
It requires response and waits for it.



b. Asynchronous message

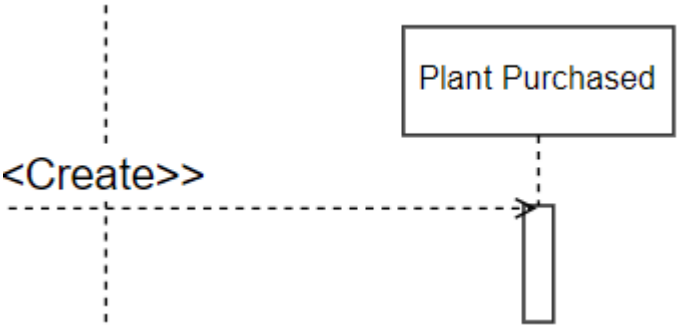
It is a forward giving message where caller does not need to wait for the response



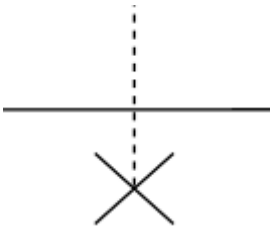
c. Return Message

It is the return response of synchronous message.

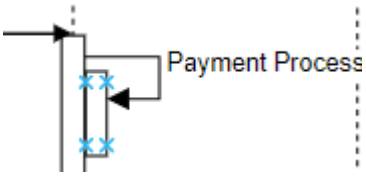
d. Creation Message



e. Destructive message



f. Reflexive message



The main difference between communication diagram and sequence diagram
is their focus, Timing, and level of details. Both the diagrams serve the same
purpose, which is used to model the communication between objects.
The sequence diagram that I have created is shown below:
Sq-diagram

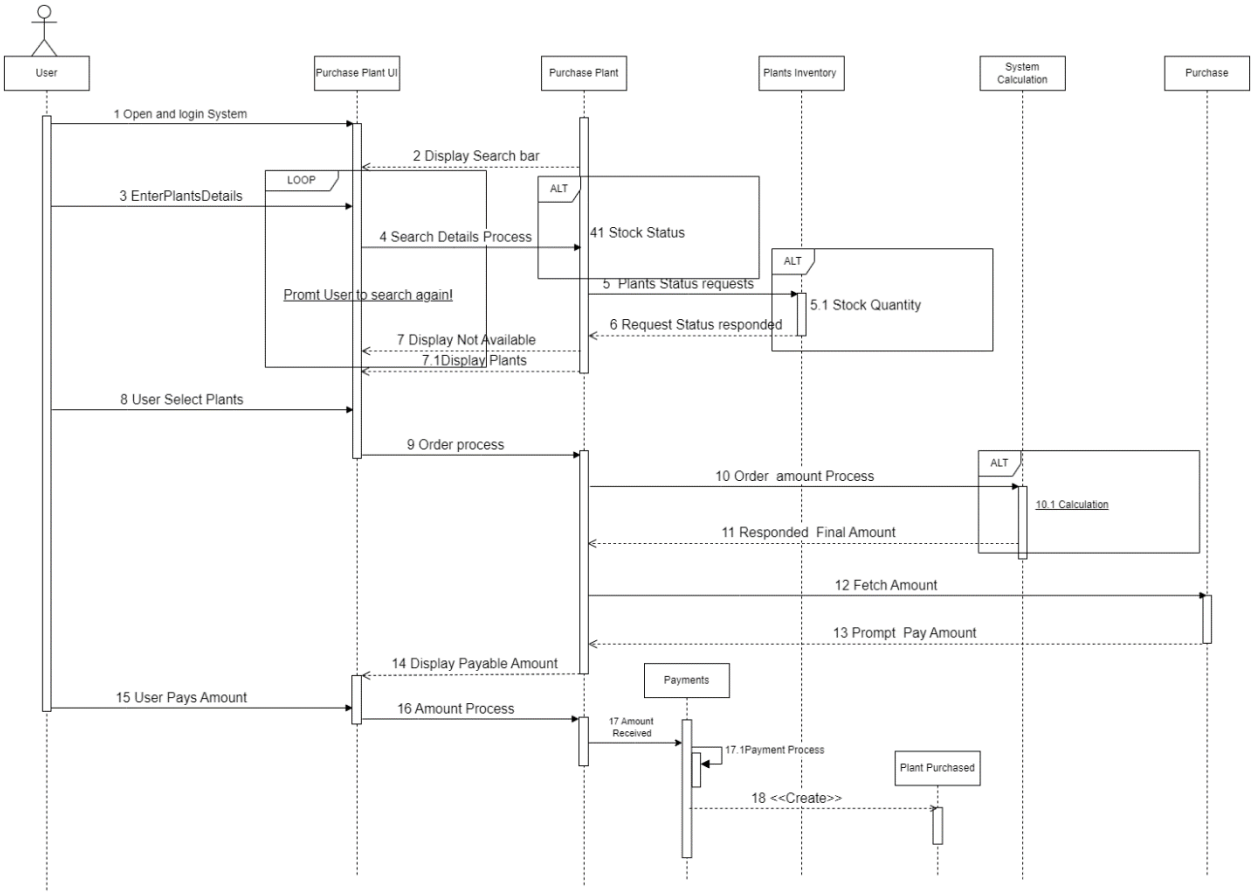


Figure 6 SEQUENCE

7. Class Diagram

A class diagram is a static diagram. It reflects an application's static view.

A class

diagram is used not only for visualizing, defining, and recording various facets of a framework, but also for creating executable code for a software program. A class diagram depicts a class's properties and operations, as well as the system' constraints. Since class diagrams are the only UML diagrams that can be specifically mapped for object-oriented languages, they are commonly used in the modeling of object-oriented structures.

The aim of a class diagram is to represent an application's static view. Class diagrams are the only diagrams that can be precisely traced for object-oriented languages, making them common throughout the construction process. UML diagrams such as activity diagrams and sequence diagrams will only show the application's process flow; however, class diagrams are a little different. It is the most widely used UML diagram by programmers.

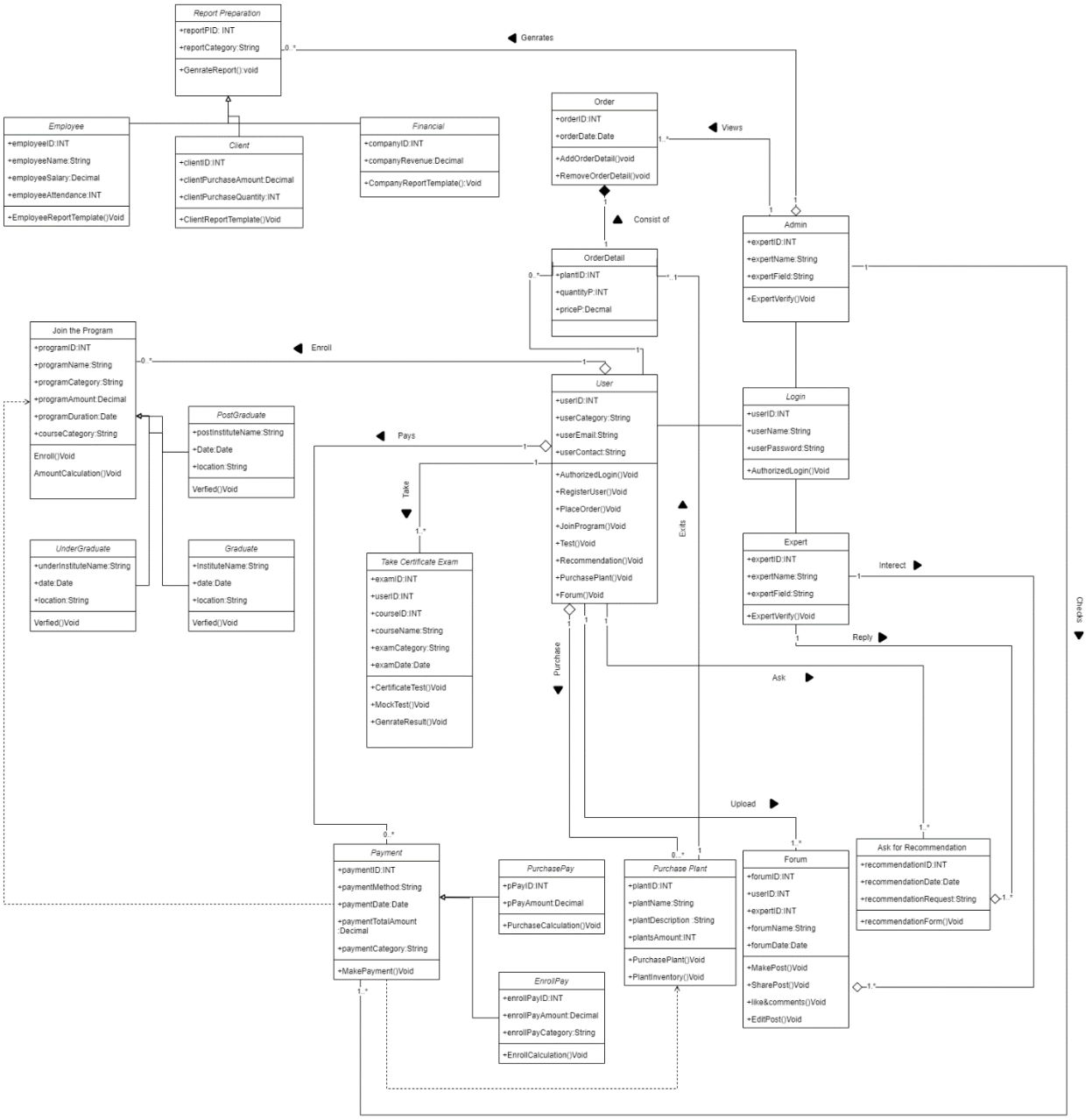


Figure 7 CLASS DIAGRAM

8. Further Development

8.1. Layered Architecture

Layered Architecture, also known as the n-tier architecture, is a design pattern that organizes a software system into a set of horizontal layers. Each layer has a specific responsibility and communicates only with adjacent layers. The separation of concerns is a key principle, making it easier to understand, maintain, and extend the system.

Here are the typical layers in a Layered Architecture:

- **Presentation Layer**: This is the topmost layer responsible for handling user interactions with the software system. In your case, this would be the user interface where users can register, join programs, purchase plants, ask for recommendationetc.

- **Business Logic Layer (Application Layer)**: This layer handles aspects related to accomplishing functional requirements. It determines how the system processes requests, applies rules, and carries out business operations. In this project, this would involve the logic for course registration, plant purchasing, payment processing, etc.

- **Data Access Layer**: This layer manages data storage, retrieval, and communication with databases. It abstracts the underlying database system and provides methods to interact with data. In this project, this would handle all data interactions for user information, course details, plant details, etc.

The Layered Architecture aligns well with the Agile methodology as it promotes separation of concerns and modular development. Each layer can be developed and

tested independently, allowing for iterative development and continuous integration, which are key principles of Agile.

8.2. MVC Design Pattern.

The Model-View-Controller (MVC) is a well-known design pattern in software engineering. It's like a blueprint for building user interfaces and it's particularly useful because it separates your project into three interconnected parts. This separation is great because it allows each part to be developed and managed independently of the others.

Here's how it works:

- **Model**: Think of the Model as the brain of our application. It's where all our data and the rules for manipulating that data live. For this project, the Model would contain all the data and business rules for the McGregor Institute of Botanical Training.

- **View**: The View is all about presentation - it's what our users see and interact with. In this project, the View would be the user interface where users can register, join programs, purchase plants, and so on.

- **Controller**: The Controller is the go-between for the Model and the View. It processes all the business logic and incoming requests, manipulates the Model data, and updates the View accordingly. In this project, the Controller would handle all the communication between the Model and View, processing user requests and updating the Model or View as necessary.

So, in the context of our project, using the MVC pattern means you'd have one part of your codebase (the Model) dedicated to all our data and business rules, another part (the View) dedicated to everything the user sees and interacts with, and a third part (the Controller) that ties the two together. This makes our code easier to manage and

understand, and it aligns well with Agile principles of iterative development and continuous feedback.

8.3. Development Plan.

Programming Platforms: We'll be using Eclipse as our Integrated Development Environment (IDE). It's like our toolbox for coding - it has all the tools we need like code completion, debugging, and integrated testing. It's a popular choice for Java development.

Server: Our server of choice is Apache Tomcat. It's reliable, open-source, and supports servlets and JSPs. Plus, it integrates well with Eclipse, making our development process smoother.

Database: For our database needs, we'll be using XAMPP. It's a free and open-source solution that gives us the Apache HTTP Server, MariaDB database, and interpreters for scripts written in the PHP and Perl programming languages. It's like a one-stop-shop for our database needs.

Tools/Resources:

- **Java Development Kit (JDK**): This is like our Java dictionary. It's what we need to develop Java applications.

- **Spring Framework:** This is for creating enterprise applications and provides an inversion of control container for the Java platform. It's like the backbone of our Java application.

- **Hibernate:** This helps us map an object-oriented domain model to a relational database. It's like a translator between our code and the database.

- **Maven:** This is for managing our project's build, reporting, and documentation. It's like our project manager, keeping everything organized.

- **Git:** This is for version control to track changes in source code during software development. It's like our project history book, keeping track of all the changes we make.

- **Priority Order of Artifacts/Features:** We'll start with core features that deliver basic functionality and then incrementally add more complex features. This is in line with the Agile principle of iterative development. For example, we might start with user registration and login, then move on to course registration and plant purchasing, and finally add the forum and recommendation features. It's like building a house - we start with the foundation, then build the walls, and finally add the roof.

8.4. Testing Plan.

8.4.1. White Box Testing:-

- **Unit Testing:** This is the first level of testing where individual components of the software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. In Java, you can use JUnit for unit testing.

- **Integration Testing:** After unit testing, integration testing is performed. This is a level of software testing where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing.

8.4.2. Black Box Testing:-

- **User Acceptance Testing (UAT):** This is the final level of testing performed after System Testing and before moving the software application to the Market or Production environment. The aim of UAT is to validate the end to end business flow.

- **Smoke Testing:** It is performed on the latest build to check whether the build is stable enough to proceed with further levels of testing. Smoke testing is also known as Build Verification Testing.

- **Sanity Testing:** It is performed after receiving a software build with minor changes in code, or functionality, to ascertain that the bugs have been fixed and no further issues are introduced due to these changes. The goal is to determine that the proposed functionality works roughly as expected.

8.4.3. Black Box/White Box Testing:-

- **Regression Testing:** It is performed after the enhancements or the bug fixes in the software application. The aim of regression testing is to ensure that the changes made in the software application have not affected the existing functionalities.
- **System Testing:** This is the third level of testing performed after Integration Testing and before Acceptance Testing. The purpose of this level of testing is to evaluate the system's compliance with the specified requirements.

8.5. Maintenance Plan.

Corrective Maintenance: This involves fixing bugs and defects that were not discovered during the initial testing phase. This is a reactive modification of the software performed after delivery to correct discovered problems.

Adaptive Maintenance: This involves updating the software to cope with changes in the environment, such as new operating systems or new hardware. It's about keeping the software up-to-date with the evolving world of technology.

Perfective Maintenance: This involves updating the software to improve performance or maintainability, or to provide a better basis for future enhancements. Often, this is done by refactoring the code to make it cleaner and easier to understand.

Preventive Maintenance: This involves performing activities to prevent the occurrence of errors. It includes tasks like code optimization and error detection.

Regular Updates: Regularly update the application to handle bugs and add new features. This could be in response to user feedback or as new requirements come up.

Performance Monitoring: Monitor the application's performance to ensure it is running smoothly and efficiently. Use monitoring tools to keep track of various metrics like response time, server load, error rates, etc.

Security Updates: Keep the software up-to-date with the latest security patches to protect against vulnerabilities. This is crucial to protect your users' data and maintain their trust in your application.

Data Backup: Regularly backup data to prevent loss. This is especially important for applications dealing with critical data.

9. Prototype Development

Figure 8 PROTOTYPE 1

1. Login UI

User enter ID and Password if wrong register acc0unt options appears..

Figure 9 PROTOTYPE 2

2.Register UI

When a user wants to join our community, they start by filling out a registration form with their details. They'll also need to upload two images. The first image is for their profile picture, a friendly face we can associate with their account. The second image is a way for us to verify their date of birth and name, and for this, they'll need to upload an image of their citizenship card. Once they've filled out all the necessary information and uploaded the images, they simply click 'Submit' to complete their registration.

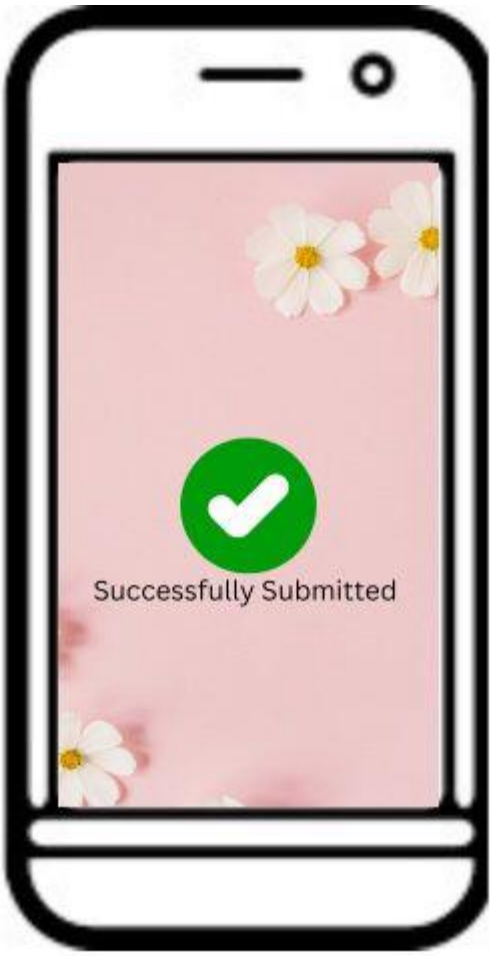


Figure 10 PROTOTYPE 3

3.Registered UI

Once the registration form is submitted, the user can now log in to their account. They'll use their username, which they provided during registration, as their ID. For the password, they'll use the one they set up during registration. After entering these details,

Figure 11 PROTOTYPE 4

4.Login UI

User enter details for login and click login button.

Figure 12 PROTOTYPE 5

5.System UI

Once the user logs in, they're greeted with a menu displaying six different options. Alongside these options, they'll see their profile icon and a speaker icon. This speaker icon represents our AI Assistant, a helpful tool designed for those who may have difficulty seeing. By default, the AI Assistant is always on, ready to guide users through our system using audio prompts. Users can respond using their mobile microphone.

We've also considered users who may be deaf, mute, or both. For them, we've integrated a feature that uses the mobile camera to track eye

movements. A blink will select an option, and moving the eye to the side will navigate back. In this scenario, the user selects ‘Purchase Plants’.

Figure 13 PROTOTYPE 6

6.Selects option in System UI

purchase plant options selected.

Figure 14 PROTOTYPE 7

7.Purchase Plant UI

Upon entering the ‘Purchase Plants’ section, users will find a search bar. This search bar is designed to make the plant shopping experience as smooth as possible. Users can search for plants by their name or the category they belong to.

To make the platform more user-friendly, especially for our visually impaired friends, we’ve featured two plants by default. This allows users to quickly select a plant without having to navigate through the entire catalog. It’s just one of the ways we’re striving to make our platform accessible and enjoyable for everyone.

Upon entering the ‘Purchase Plants’ section, users will find a search bar. This search bar is designed to make the plant shopping experience as smooth as possible. Users can search for plants by their name or the category they belong to.

To make the platform more user-friendly, especially for our visually impaired friends, we’ve featured two plants by default. This allows users to quickly select a plant without having to navigate through the entire catalog. It’s just one of the ways we’re striving to make our platform accessible and enjoyable for everyone.

Figure 15 PROTOTYPE 8

8.Purchase Plant UI

User can type in search bar to find plants as per their interest for purchase.

Figure 16 PROTOTYPE 9

9.Purchase Plant UI

Based on user search plants will be displayed with left and right sliding features.

Figure 17 PROTOTYPE 10

10.Purchase Plant UI

User select plants for purchase.

Figure 18 PROTOTYPE 11

11.Purchase Plant UI

Once a user selects a plant, they're presented with detailed information about it. This includes the plant's name, its category, and the stock available for purchase. There's also a 'Purchase' button, which users can click when they decide to buy the plant.

To make the buying process even more user-friendly, we've added a feature that allows users to select the quantity of the plant they want to buy. This feature includes an option to increase or decrease the quantity as per their needs. Once they've selected the desired quantity,

Figure 19 PROTOTYPE 12

12.Purchase Plant

User click on purchase icon and a hovor effects is dispalyed .

Figure 20 PROTOTYPE 13

13. Purchase Plant UI

Order Details Displayed.

And prompt user for pay.

Figure 21 PROTOTYPE 14

14. Purchase Plant UI

User click on pay icon and a hovor effects shows.

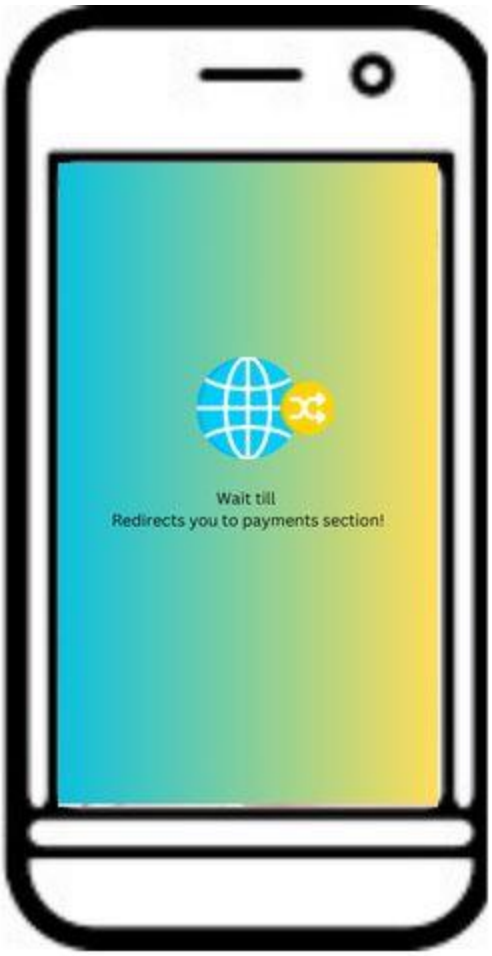


Figure 22 PROTOTYPE 15

15. Process UI

The system calculate and procees amounts. Redirecting user in payments section for final process.

Figure 23 PROTOTYPE 16

16 Payments UI

This is payments section where user purchase amount is being loaded.

Figure 24 PROTOTYPE 17

17.Payments UI

User see the total amount and payment icon.

Figure 25 PROTOTYPE 18

18. Payments UI

Once the user has selected their plants and is ready to make a purchase, they simply click on the ‘Pay’ button. This will lead them to the payment processing stage. Here, they’ll encounter an ePayments User Interface, which is separate from our software. This interface allows users to choose their preferred method of payment, such as eWallet or mobile banking.

Please note that the payment process is handled by a different software for security reasons. Once the payment is successfully processed, the user will be notified, and their purchase will be confirmed.

Figure 26 PROTOTYPE 19

19.Payments UI

A colorful pay icon will appear showing user payments received and decuted from user bank or wallet account.

Figure 27 PROTOTYPE 20

20.Payments UI

After Payments completes . A Confirmation message will appear for letting user know there were no problem whlile making payments.

Figure 28 PROTOTYPE 21

21.System UI

After payments the user get redirect to system menu UI.

Figure 29 PROTOTYPE 22

22.System UI

Now, User can choose other options if insterested.

10. Conclusion

Finally, I acknowledged how much I knew about Agile methodology and how to act as a software engineer after completing the coursework. I assume I have acquired a significant amount of expertise on different topics such as Use Case, Gantt Chart, Sequence Diagram, Collaboration Diagram, Class Diagram, and several others. Working on this coursework,

on the other hand, was very challenging. I have come to know about many software development process. I must acknowledge about many methodologies to develop software. Among them I have choose Agile methodology to complete the further development. The whole process of completing every phase and moving from one to another is whole new level of development that helps to enhance the project one by one. In addition, I conducted extensive analysis, reading a variety of journals, blogs, and posts among other sources. Finally, I ‘d like to express my gratitude to all my teachers, tutors for their unwavering support. To summarize I think this coursework is giving me a full guidance for our oncoming final year project and as well as to develop real life software.

11. References

Aha!, 2020. Aha!. [Online]
Available at: https://www.aha.io/roadmapping/guide/agile/developmentmethodologies#:~:text=Scrum%20is%20the%20most%20popular,scrum%20master%20or%20product%20owner.
[Accessed 04 04 2023].

e-education.psu.edu, n.d. e-education.psu.edu. [Online]
Available at: e-education.psu.edu/geog468/l8_p4.html
techtarget, n.d.

techtarget. [Online]
Available at: https://www.techtarget.com/whatis/definition/use-case-diagram

Westland, J., 2022. PM. [Online]
Available at: https://www.projectmanager.com/blog/scrum-methodology [Accessed 04 03 2022].

ttps://www.glassdoor.com/Interview/Aha-Software-Engineer-Interview-Questions-EI_IE1102346.0,3_KO4,21.html