



Islington college
(इरिलिङ्टन कलेज)

Module Code & Module Title
CS5002NI SOFTWARE ENGINEERING

35% Individual Coursework

Year and Semester

2022-23 Spring

Student Name: Divya Shrestha

London Met ID: 22085527

College ID: NP01CPS230022

Group: C18

Assignment Due Date: May 7, 2024

Assignment Submission Date: May 6, 2024

Word Count: 5440

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.

Table of Contents

1	Introduction	1
2	Work Breakdown Structure	2
3	Grantt Chart	4
4	Use Case	6
4.1	Use Case Diagram.....	10
4.2	High level use case description.....	11
4.2.1	Register in the System.....	11
4.2.2	Join the Program	12
4.2.3	Make Payment.....	12
4.2.4	Ask for Recommendations.....	12
4.2.5	Report Preparation	13
4.2.6	Purchase Plant	13
4.2.7	Take Certificate Exams.....	13
4.2.8	Forum	14
4.3	Expanded use case description.	15
4.3.1	Expanded Use Case Description: Purchase Plant.....	16
4.3.2	Expanded Use Case Description: Ask for Recommendations	17
5	Collaboration Diagram	18
6	Sequence Diagram	21
7	Class Diagram.....	26
8	Further Development	31
8.1	Software Architecture model	31
8.2	Design Pattern	33
8.3	Programming Language.....	35
8.4	Testing.....	36
8.5	Maintenance.....	37
9	Conclusion	38
10	Prototype	40
10.1	Home page	40
10.2	Register member.....	41

10.3	Purchase Plant	43
10.4	Payment.....	46
10.5	Join the Program.....	48
10.6	Choose a Course	49
10.7	Take Certificate Examination	50
10.8	Form	51
10.9	FeedBack.....	52
10.10	Ask for recommendation.	53
10.11	Report Generation	54
11	References.....	55

Table of Figure

Figure 1: Use Case Diagram	10
Figure 2: Collaboration Diagram	20
Figure 3: Sequence Diagram	24
Figure 4: Class Diagram.....	29
Figure 5: Agile Software model.	32
Figure 6: MVC pattern figure.	33

Table of Tables

Table 1: Register in the System.....	11
Table 2: Join the program.....	12
Table 3: Make payment.	12
Table 4: Ask for recommendation.	12
Table 5: Report Preparation	13
Table 6: Purchase Plant	13
Table 7: Take Certificate Exams	13
Table 8: Participate in Forum.....	14
Table 9: Purchase Plant expanded use case.	16
Table 10: Ask for Recommendation expanded use case.....	17

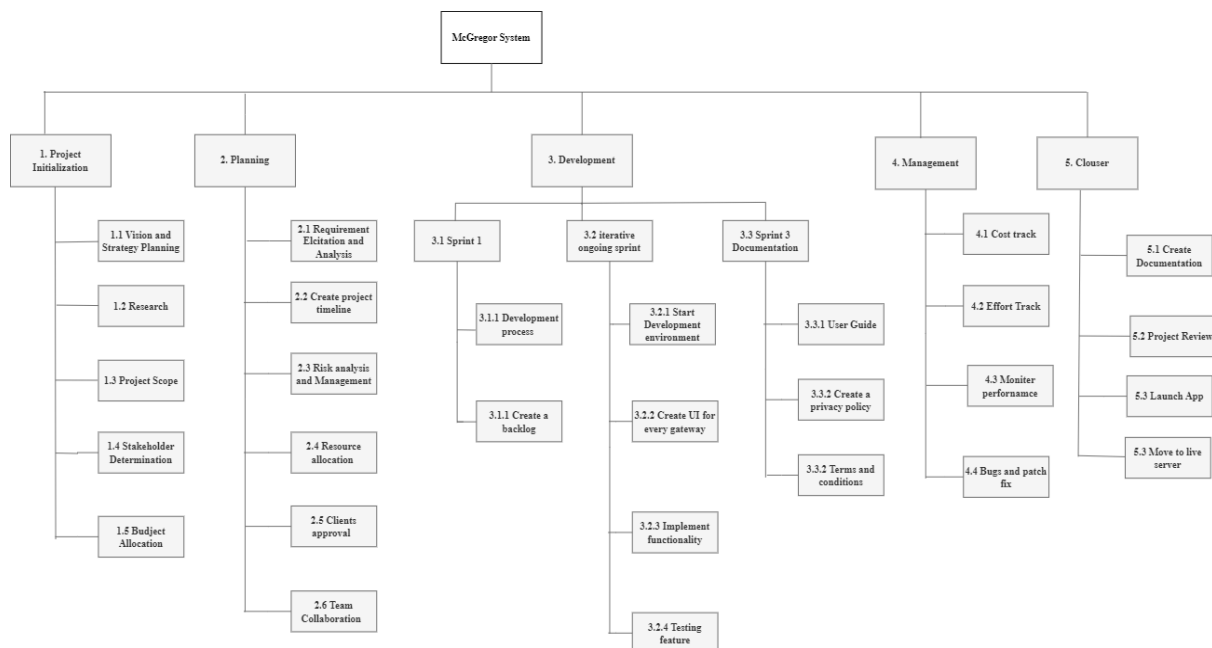
1 Introduction

In the first individual coursework that was part of the 'Software Engineering' course developing OOP models for McGregor Institute of Botanical Training, we became engaged as the first individual assignment to be completed during the 'Software Engineering' module, which includes the analysis and creation of software modeling of McGregor Institute of Botanical Training considering the Object - Oriented paradigm. Having the guiding burden of botanical knowledge, McGregor Institute, the beacon of light which appears in the constellation of the growing worlds of agriculture and horticulture, becomes the gathering point of people who have the soul of a scientist or even of a professional, and we want to enlighten their senses and guide them into this timeless topic.

McGregor Institute of Botanical Training, an Ireland-based institution located in Godavari, Lalitpur, has been operating in Nepal for nearly seven years. Affiliated with Dublin City University, it offers a range of undergraduate and postgraduate courses specializing in agriculture and horticulture. In response to a recent surge in interest in agriculture, McGregor Institute plans to introduce short-term certification courses in horticulture. Additionally, they aim to sell various plant varieties at minimal or no cost to foster a community of plant enthusiasts. Their vision includes creating a platform or forum where members can discuss ideas, organize programs for the protection of rare plants and forests, and seek expert advice on plant-related queries. This project focuses on designing a system for McGregor Institute of Botanical Training to facilitate its educational offerings and community-building initiatives in Godavari, Lalitpur, Nepal.

2 Work Breakdown Structure

A Work Breakdown Structure (WBS) is a visual and hierarchical method of project management that helps in segmenting the scope of a project into manageable pieces or sections (Raeburn, 2024). It is a task-oriented, particularly related with the goals of the projects, which helps them visualize all the tasks required to finish their projects. WBS identifies all the steps each work should take through detailed work and the possible outcomes required to complete the projects goals and objectives. The main role of this software is not only for implementation of Project Planning, Project Scheduling, Project Budgeting, Risk Management, Resource Management, Task Management, and Team Management. It is probably the major project management tool available to a project, by building the WBS and defining it according to the project phases, sub-projects, tasks, deliverables, and work packages, a clear view is provided which will help the project manager to avoid problems like extensions of the deadline, delays, and cost overruns.



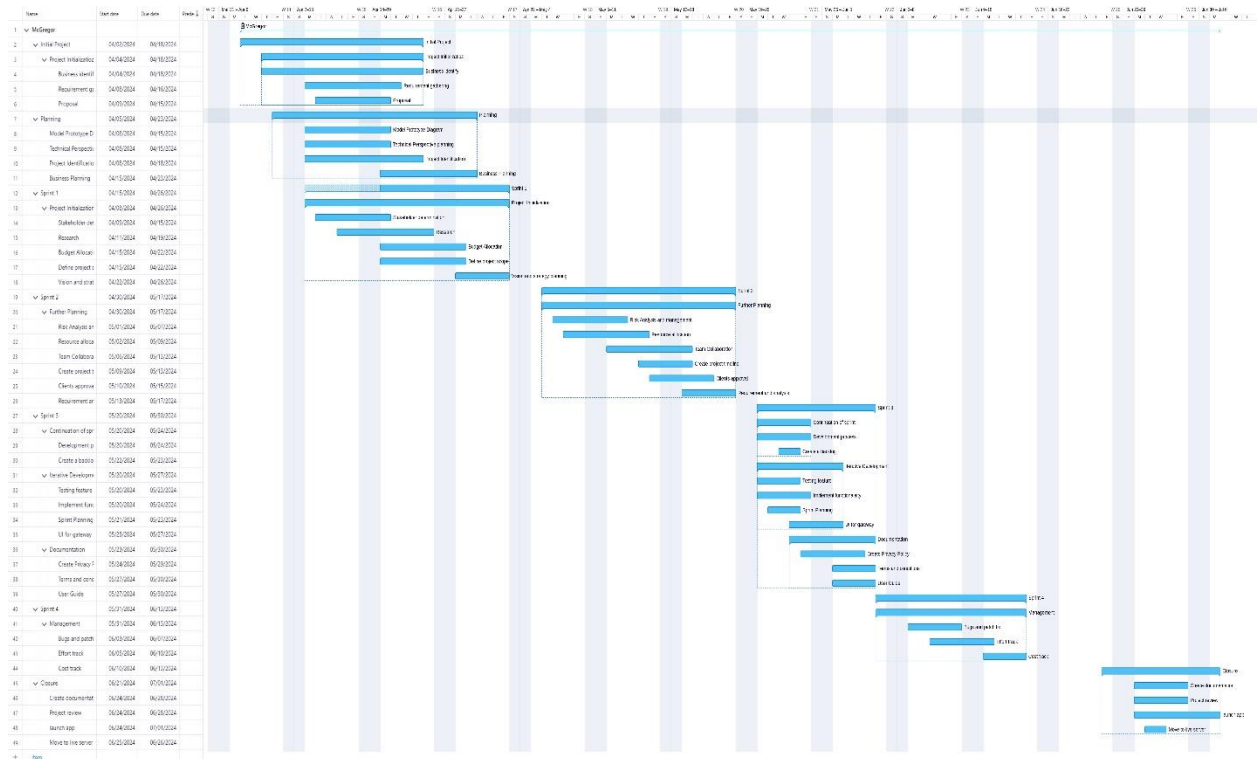
- The first step is project initialization phase, formulating strategies and planning process. This involves research, visualization, understanding the scope of the project. This is the initial phase of the project.
- The next phase is planning of the project. In this phase the risk management and project budget are calculated. The client or the company is told about these factors and if everything is approved resources and team are managed to work for the project.
- The next phase is development, which include mentioning the requirements for architecture, system designing, development team backlog management, building a staging environment, configuring a version control platform, deploying initial codebases, and structuring content plus the optimization of SEO and user experience.
- The next phase is management, where all the testing, reviewing, debugging and if any issues are there to be solved, are solved.
- The final phase is the closure of the project, where the final documentation of the project is done, and the app is launched in the live server.

Overall, The WBS is a road map from the beginning to the end of the McGregor System, the system that transforms the idea into a successful project. The WBS reviews the main elements of the successful implementation of the project step-by-step.

3 Gantt Chart

The Gantt chart is a horizontal bar chart, which is applied for the planning of projects to visualize a timeframe. It feels rare that this flowchart is not depicted because in most cases, a project timeline, sequence of activities, and responsible parties to work on them is present. based on splitting the complicated tasks into smaller and simple activities, the Gantt chart enables the project managers to prepare, schedule and track the project progress even though it is maintaining the attention on all the critical path and possible delays to guarantee that the project is delivered on time (Ramos, 2021).

A Gantt chart is commonly used in both waterfall and agile methodologies because they both address task progress according to linearity. This project is applied with agile methodology where the Gantt chart which is sliced into small periods of time which are called sprints: Each sprint will be at least two or maximum four weeks and each sprint will start with a sprint planning meeting where the sprint goal will be discussed and the activities to be performed and how the activities will be achieved should be planned. Firstly, the team comes up with the goals of the projects and decides deliverable, and then the team plans sprint schedule to fulfil sprint goals and timeline goals. The development sprints require the teams to deliver usable outcome with every increment and so build a prototype of the product. The product that is the finished work after each sprint undergoes a review stage. At this point, the testing, control of quality, and the engagement of stakeholders are involved. Sprint retrospective is what the team does at the end of the sprint to analyze the sprint and may pinpoint the areas where they can improve.



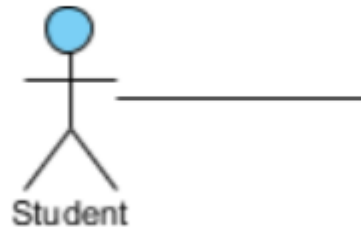
Each action of the work is shown by the blue bar which the size of it indicates how much time it lasts. The tasks are placed either horizontally or vertically, depending on what works best, and the timeline runs from right to left. Some tasks are linked to arrows showing that several tasks need to be done in specific sequence, AB in which A must be done before B can be started. A paler color would be the most suitable choice for the lines that extend from some bars and represent progress with tasks or predictions for future work. Text is placed beside each bar that give details about separate tasks, for example 'Task Name', 'Start', 'Finish', 'Duration', where WP1.1 stands for part of the larger task. The Gantt chart above makes it possible to visualize the entire timeline of a project as well as facilitating the monitoring of completion rates against the schedule plan. It is the ideal tool for project coordinators, team members, and major stakeholders who are dedicated to smooth planning and executing of complicated projects.

4 Use Case

A UML (Unified Modeling Language) use case diagram holds the central position in the initiation phase of a software project by being used for outlining the system or software requirements (figma, 2024). Use cases is not a store of predicted behaviors, instead, use cases depicts expected behaviors rather illuminating technical details of their performance. To achieve this purpose, various use cases can be conveyed on either line-digesting pieces or visual material. Use case diagram composition is made up of several different use case notation symbols that are the UML language icons. They are:

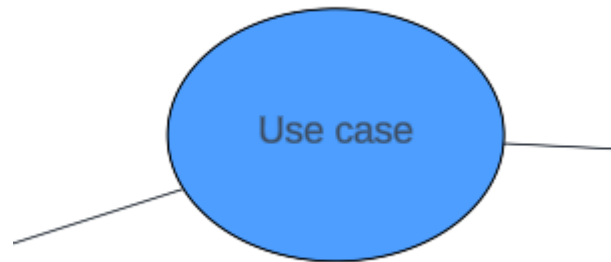
- **Actor:**

In the context of use case diagram, an Actor is an external entity involved in the system activity to accomplish some given goals and tasks. The actors listed in the system are usually called out using noun phrases which describe their role or function emerging in relation to the system. A Stick drawing of a human is used to symbolize it.



- **Use case:**

On the Use Cases diagram in a use case map automated and manual processes are incorporated as two key functions of the system. They work as if they are a complete record of tasks which the system is meant to run through, for the system to obtain its goal. For each one-Use Case functioning phase or property of the system will be described with concrete behavioral requirements. An oval shape entity is used to define it.



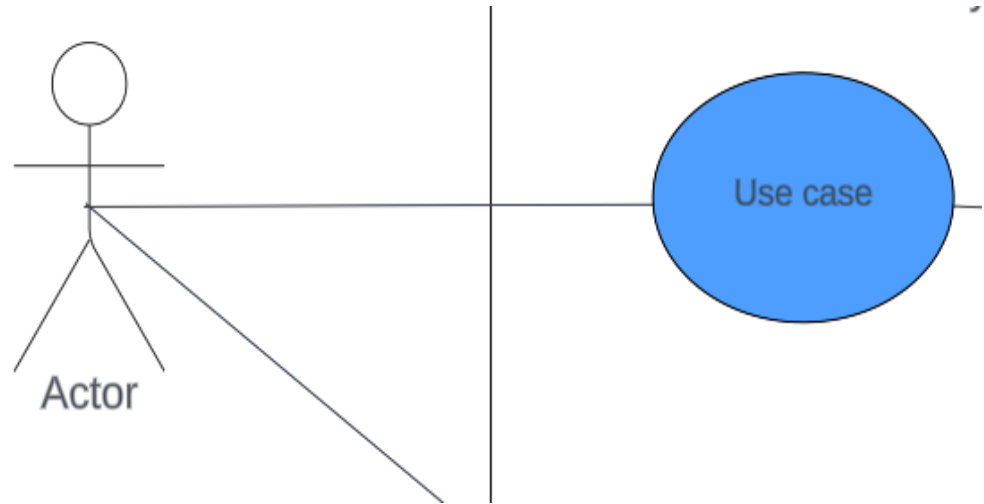
- **Communication Link:**

In a use case diagram, an actor's engagement with a particular use case is illustrated by a solid line connecting the actor to the respective use case. This linkage visually represents the interaction between the actor and the system function encapsulated by the use case.



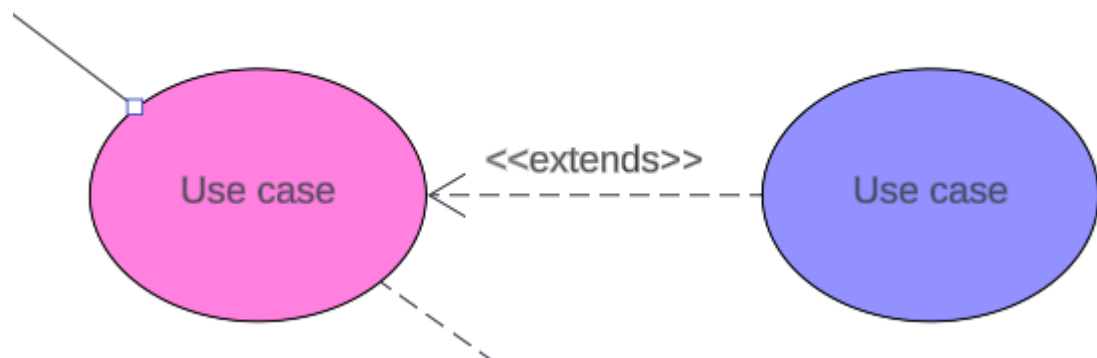
- **Relationships:**

The use cases within a use case diagram can have varying types of relationships. Determining the connection between two use cases is at the discretion of the software analysts who are working on the use case diagram.



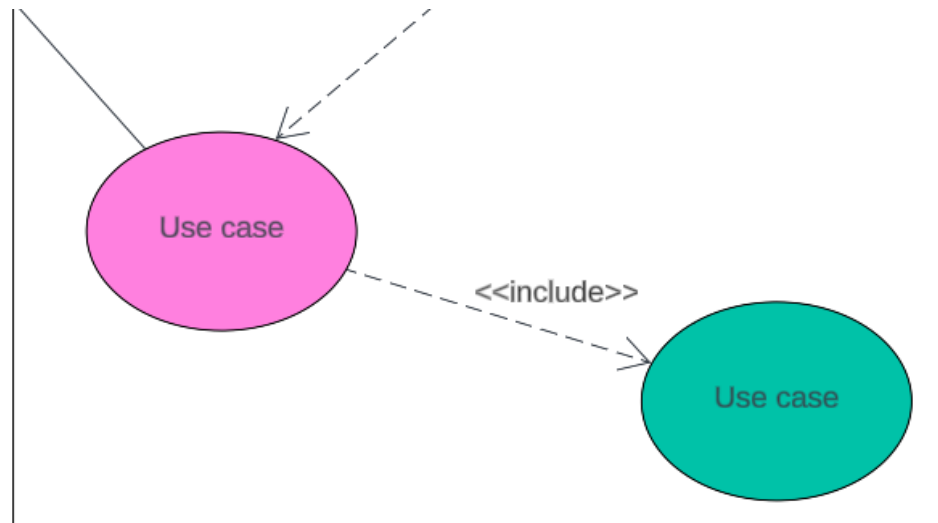
- **Extends**

Shows optional functionality or system behavior. Presented in a use case diagram through a dotted arrow with the label extends.



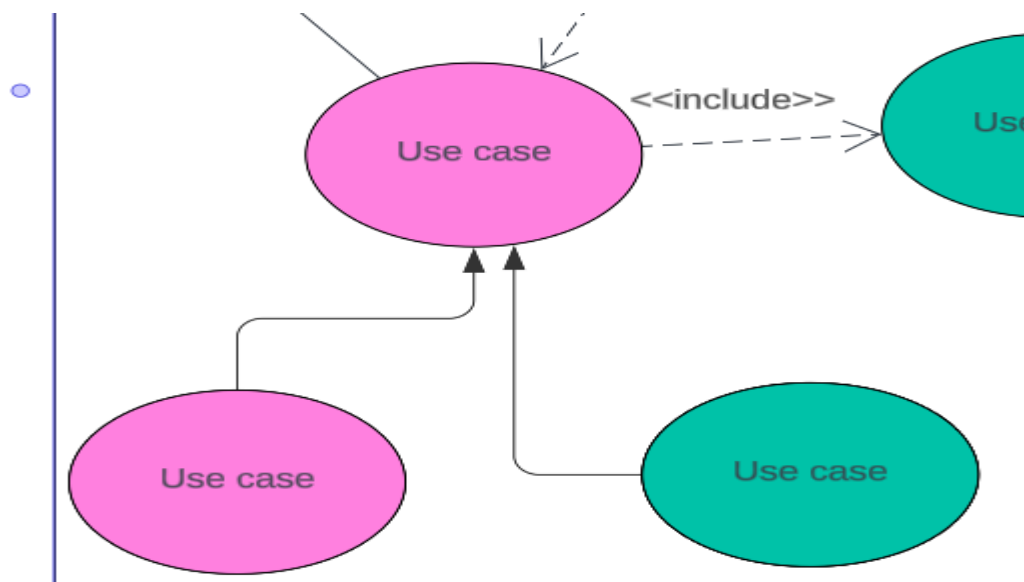
- **Include**

Adds additional functionality which is not specified in the base use case. Presented in a use case diagram through a dotted arrow with the label include.



- **Generalization**

A parent-child relationship where Child is an enhancement of parent. Identified as a directed arrow with a triangle arrowhead.



4.1 Use Case Diagram

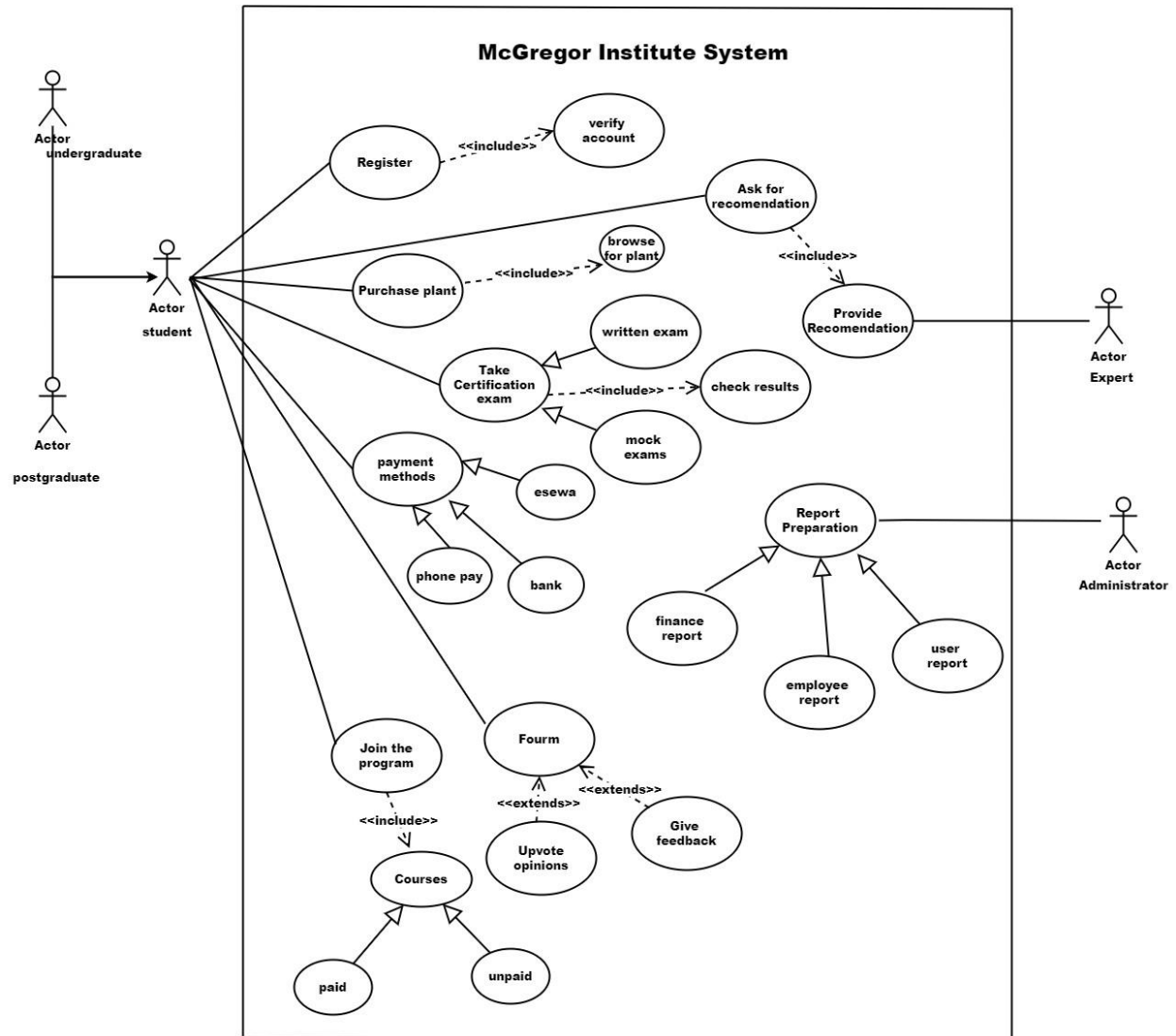


Figure 1: Use Case Diagram

4.2 High level use case description

High-level Use Case (UC) provides a comprehensive brief of his task, written in unstructured sentences make up 2-3 sentences. The main goal of this activity is to provide the level of each instance detail so that the team can arrange the related use cases as a part of the elaboration phase. In the present of the use case diagram, there are fifteen descriptions, each with three parts, the name of the use case, the involved actors, and short brief description of task at hand. Such structured process facilitates in cleaning up system functionalities and interaction with external entities and, thereby, is foundation for coming stages.

4.2.1 Register in the System

Use Case	Register in the System
Actor	User
Description	<p>Any user who are interested to join, can register themselves into the system by filling out personal details, contact number and password.</p> <p>The system validates the information and creates an account such that user can log into the system</p>

Table 1: Register in the System

4.2.2 Join the Program

Use Case	Join the Program
Actor	User
Description	<p>Anyone interested who have registered into the system can join the program and enroll themselves in the Courses.</p> <p>The system records the enrollment, and they can participate in paid or unpaid Courses.</p>

Table 2: Join the program.

4.2.3 Make Payment

Use Case	Make Payment
Actor	User
Description	User who has enrolled in the system should make a payment if they want to purchase any kind of plats or courses, they have enrolled in.

Table 3: Make payment.

4.2.4 Ask for Recommendations

Use Case	Ask For Recommendation
Actor	User, Expert
Description	Any user who has query with specific details (location, soil condition) can send the request to the system. The system sends the request to the expert who provides recommendation.

Table 4: Ask for recommendation.

4.2.5 Report Preparation

Use Case	Report Preparation
Actor	Admin
Description	Admin selects the type of report to generate. Its either finance, user, or employee report. System processes the request and generates the report.

Table 5: Report Preparation

4.2.6 Purchase Plant

Use Case	Purchase Plant
Actor	User
Description	Users access the plant catalog; user selects the plant to purchase and add them to their card and processed to checkout after the order is confirmed

Table 6: Purchase Plant

4.2.7 Take Certificate Exams

Use Case	Take Certificate Exam
Actor	User
Description	User selects the desired certification exam, i.e. either mock test or quiz. After the completion of the exam, the system evaluates the answer, then sends the result. The user must check their results

Table 7: Take Certificate Exams

4.2.8 Forum

Use Case	Forum
Actor	User
Description	Users access the form to read existing posts and comments. Users can also create new posts or comments on existing ones. Users can also interact with others post and comments.

Table 8: Participate in Forum

4.3 Expanded use case description.

The expanded use case description is situated between the concise explanation of the specific use case within the software application. It is a precise form of way in which the system deals with the users towards making an aggregation of objectives. This exhaustive use case description presents the list of preconditions, postconditions, and the subsequent sequence of events to be followed, together with alternative flows and paths that could lead to unique results. Through the painstaking demonstration of the interactions between the system and its users along with other scenarios, and the delineation of various scenarios and contingencies, the system will be seen with more clarity hence the overall effectiveness of the system will be fully understood.

4.3.1 Expanded Use Case Description: Purchase Plant

Use Case: Purchase Plant

Action: User

Description: This use case describes the process by which a user can browse and purchase plants from the institute's online catalog

Course of Event

Actor Action	System Response
1. The user logs into the system using their credentials.	2. The system checks into the system and verifies the user, and sends them to Catalog
3. User selects the plant they want to purchase and adds them to their list	4. The system shows all the items in the items users have selected
5. The user reviews the selected in its cart and makes necessary adjustment like quantity	
6. The user proceeds to the checkout page	7. The system receives a payment request and sends them the payment form
8. The user fills all the payment details and confirms the order	9. The system receives the payment details and ask for the payment conformation
10. The user confirms the payment	11. The system process the payment and send email of conformation
	12. The details are sent to the admin for report preparation

Table 9: Purchase Plant expanded use case.

Alternate Flow:

- **Line 4:** If the selected plant is out of stock, the user is notified during the checkout process, and they can choose to remove the item or replace it.
- **Line 9:** If there is an error during the checkout process, the user is notified the error and are asked to try again.

4.3.2 Expanded Use Case Description: Ask for Recommendations

Use Case: Ask for Recommendation

Action: User, Expert

Description: This use case describes the process by which a user can browse and purchase plants from the institute's online catalog

Course of Event

Actor Action	System Response
1. The user logs into the system using their credentials.	2. The system checks into the system and verifies the user
3. User selects the "Ask for recommendation" section and request the form	4. The system gives the form to the user
5. The user fills the query they have in the form such as location, soil conditions, etc. and submits it!	6. The system sends the form to the Expert
7. Experts receive the form and analyzes the form	
8. The Expert provides all the recommendations and queries about the user and sends it to the system	9. The system receives the report from the expert, and send it to the user
10. The user receives the report from the system	

Table 10:Ask for Recommendation expanded use case.

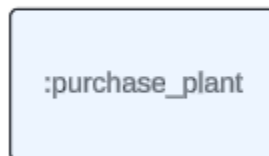
Alternate Flow:

- **Line 6:** If the user provides insufficient information, the expert requests for additional information details via system.
- **Line 3:** if there is an error while submitting the form, it will notify the user and ask them to try again.

5 Collaboration Diagram

A collaboration diagram is a type of communication diagram under the Unified Modeling Language (UML) which shows interacting entities by drawing their interrelationships and connections. The diagram itself is formed by the exchange of messages and revealing the connections between objects to make up the cooperation within the process. Rather like a flowchart, a co-operation chart shows the functions, the role, and the behaviors of each element at the instant of control which are then put into action in the real time mode of the whole system (geekforgeeks, 2024). The following are the cooperation diagram's four main components:

- Object: Rectangular shapes that include name labels. The naming label adheres to the object name: class name convention. If an item possesses a characteristic or a condition that directly affects the cooperation, it should also be mentioned.



- Actor: Stick figure of a person. The use case is initiated by one actor, with each actor having a name and a role.



- Link: A solid line is used to represent the connection between the two elements and links actors to objects. Messages can be sent over each connection.



- Message: These are shown by a labelled arrow next to a link. These messages, which also contain the sequence number, are sent between objects to transmit information about the activity.



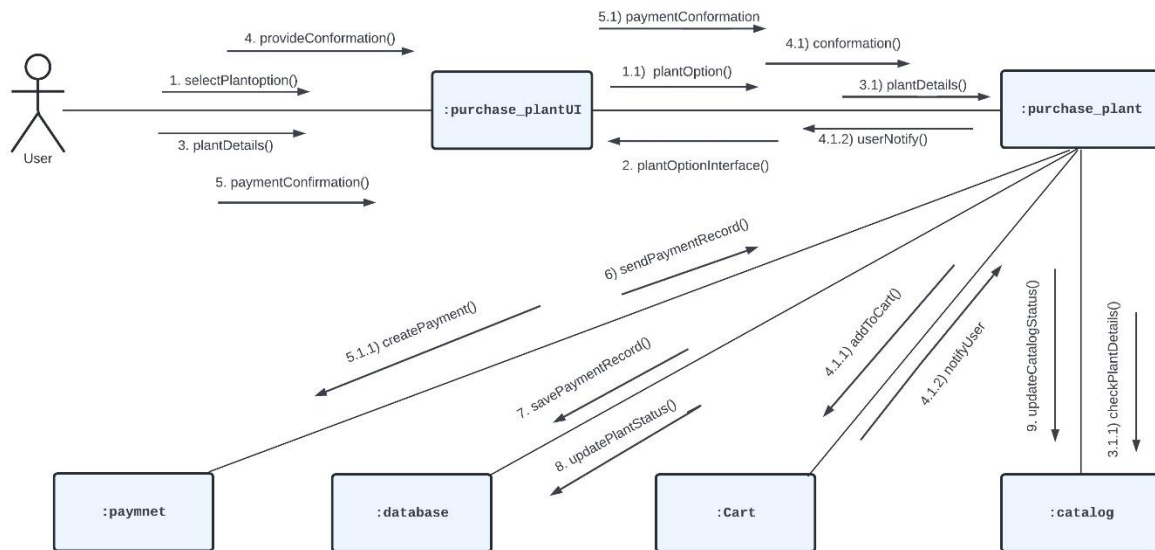


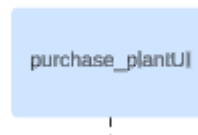
Figure 2: Collaboration Diagram

This collaboration diagram highlights the important procedural steps of that plant purchasing process. The process initiates with the consumer giving the command by requesting a plant of his choice. The self-schematic first approaches its catalog section by means of the interface and then proceeds to get the required field detail. The selected plant is then automatically added to the cart and proceeding through the controller. Next the payment object is called where the customers payment details are safely recorded and fade into the database controller containing the status of the plant and that is how it is dealt with. Then the controller signals the catalog's object about the plant's status to keep the systems consistent.

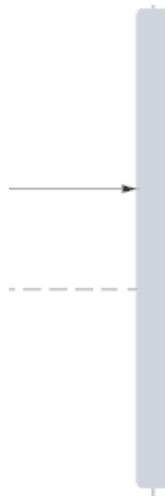
6 Sequence Diagram

A sequence diagram in Unified Modelling Language (UML) shows the flow of messages passing between objects during an interaction. Using lifelines to represent the objects, a sequence diagram shows the messages that the objects exchange throughout the course of the interaction (lucidchart, 2024). Notations in a sequence diagram are:

- **Object:** The object symbol shows how a particular object will act inside a framework of a system. Class characteristics shouldn't be given in this format.



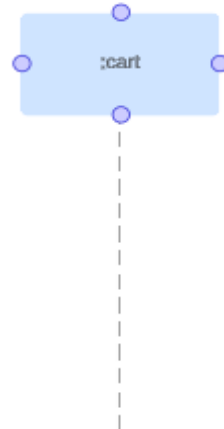
- **Activation box:** The Rectangle attached to the lifeline, the length of which represents the duration of the task.



- **Actor:** The entities which have interactions to the system.



- **Lifeline:** A lifeline denotes the time of an object, or an individual involved in the sequence diagram



- **Message:** The Arrow pointing from caller to the recipient, which shows flow of direction and is accompanied by a message signature description. The different lines and arrowhead represent different kinds of message being sent.
- **Return Message:** These messages are responses to calls, and they are denoted by a dashed line with a lined arrowhead.



- **Synchronous Message:** These messages are call for a function to start. It is represented by a straight line with pointed solid arrowhead.



- **Asynchronous message:** An arrowhead-lined solid line is used to represent this. Response is not necessary for asynchronous message before the sender can proceed.



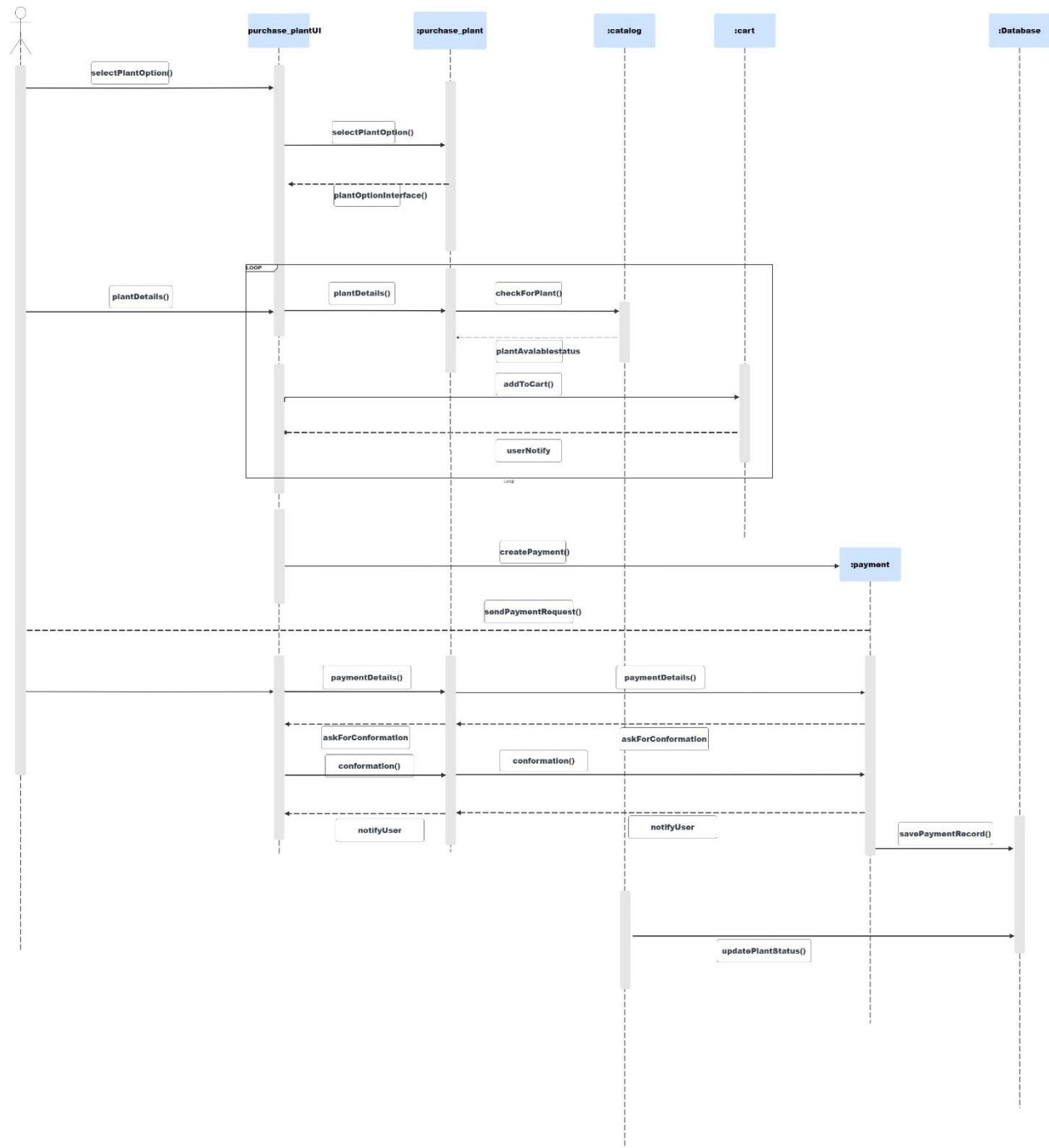


Figure 3: Sequence Diagram

Above is the sequence diagram of a Purchase Plant entity of the use case for McGregor Institute System, there is a single actor, User. One boundary Object called "Purchase PlantUI" and controller object "Purchase Plant". There are three pre-existing object catalog, cart and database, and another object named payment, which is called later after some functions are called out.

The sequence diagram begins with the actor "User" invitation, then the controller object starts the interface. Afterwards necessary methods are called and returned, modelling the use case's logic and interactions between components that make up the process.

7 Class Diagram

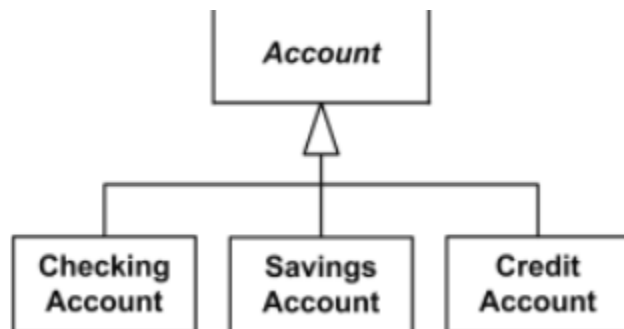
A class diagram provides a static perspective of an application, describing its attributes, operations, and constraints. It serves as a visual aid for documenting and constructing executable code for software applications (lucidchart, 2024). Due to their direct mapping with object-oriented languages, class diagrams are a popular tool for modelling object-oriented system.

Normally class diagram is divided into three parts:

- Upper Section: Contains name of the class
- Middle Section: Contains attributes of the class
- Lower Section: Contains methods of the class

The class diagram is connected to each other through different types of relationships. They are:

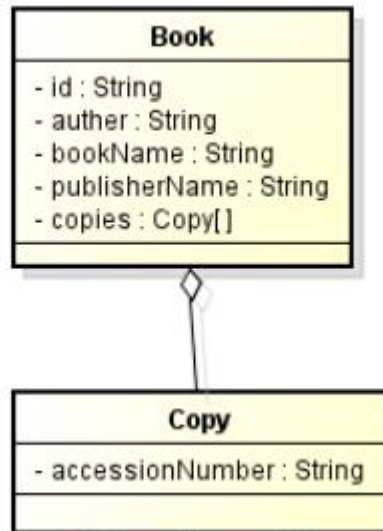
- Generalization: Generalization is the type of relationship between a parent and a child class where the child class is an improvement on the parent class.



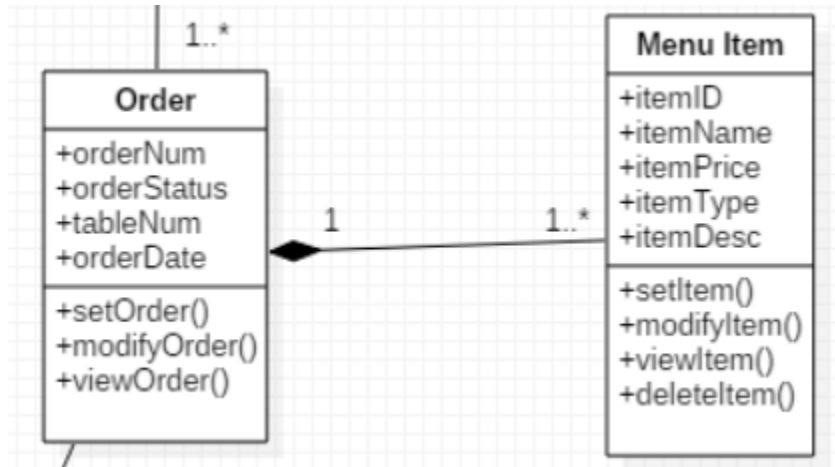
- Association: Association is when there is a tangible relationship between two or more objects.



- Aggregation: Aggregation is a type of association that signifies a “has-a” relationship and is a more specific form of association.



- Composition: Composition also signifies a “has-a” relationship and is a stronger form of aggregation. It illustrates how the parent and child are interdependent, therefore if one component is removed, the other portion must also be removed.



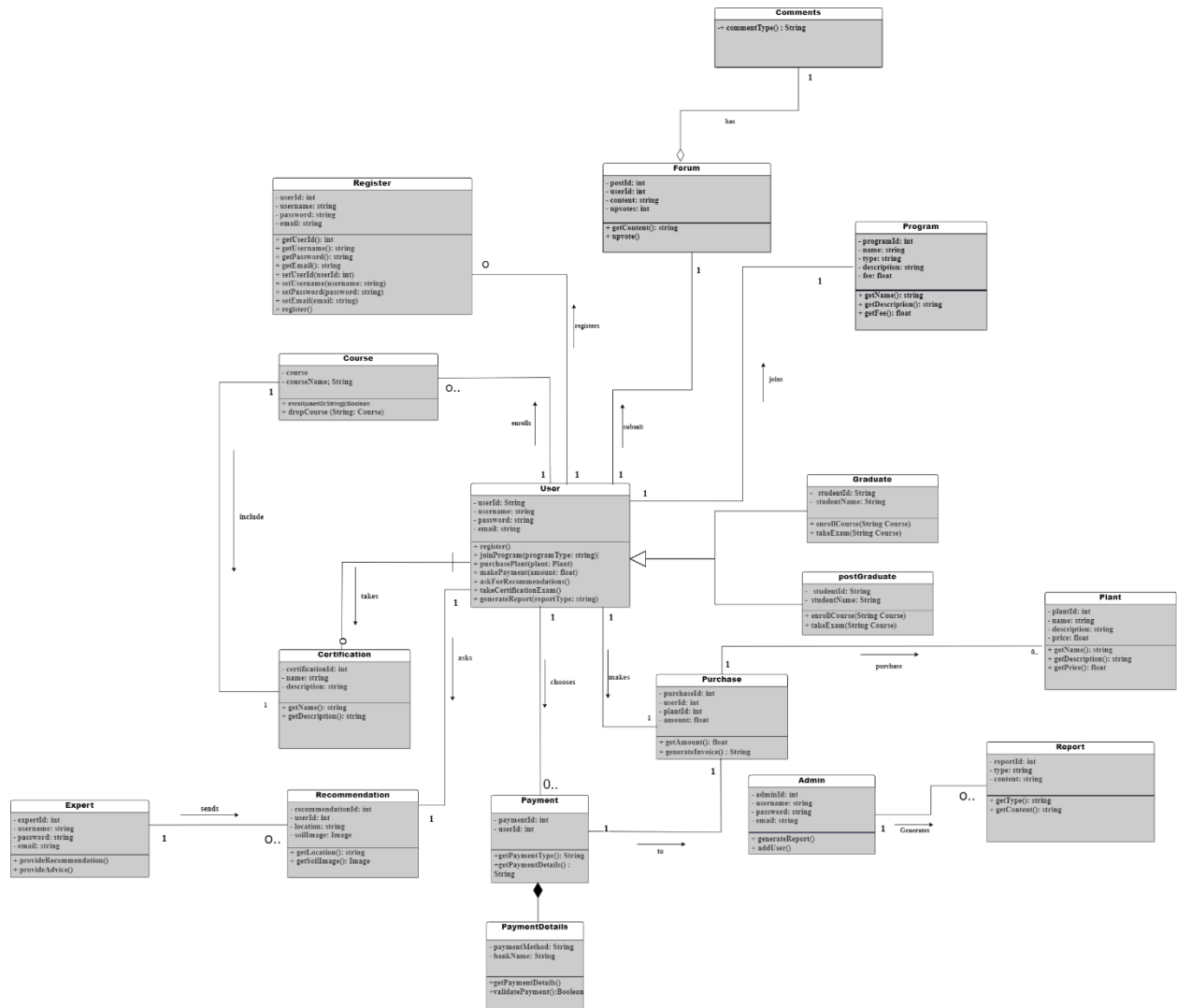


Figure 4: Class Diagram.

The class diagram indicates a system with many roles of users, and multiple interactions between them in an educational and e-commerce site. The “User” class in the “core” defines function related to registration, enrollment in courses, and interacting with forums, among others. Students can move into “graduate” or “postgraduate” levels to enroll into any courses and do examinations correspondingly. The experts provide advice tailored to each need, and for the “Administrators” to modulate user reports and supervise the system. Courses start, attend, or pay for “Payment” as the users themselves handle the process at their own discretion, and the methods for doing transactions. Moreover this system has purchase module with detailed purchase information as

“PlantPurchase” and “PurchaseDetails” managed under the same. Forums and conversations augment human interaction and, in this way, make the platform more community oriented.

8 Further Development

8.1 Software Architecture model

Software architecture of the system can act as a business plan about the interaction between the leading components of the system. It includes all the specifications implementation for good running system with extra precaution consideration the security issues. Lastly, the whole process of organizing work distribution, and identifying which technological solutions is the best for this task, always has a crucial direct influence on whether the product is well-structured, and with good performance, and it brings the desirable result.

The software's main agenda is based on three factors:

- **Requirement Iterative Development:** The Project integrates many steps and functionalities, i.e., user registration, program enrollment, plants and related accessories purchase, payments and even internet forum is made available during this iterative development. This progressive method allows for successful addition of requested functionalities which makes it a good fit for implementation of functionalities.
- **Flexibility to Accommodate Changing Requirements:** The project refers to the online system to be created for the training institute which in accordance with the educational requirements of the institute may undergo modifications due to evolving needs or incoming of new technologies. The option that enables adjustment in addressing sudden changes would be an asset.
- **Emphasis on User Involvement and Feedback:** The system aims to serve a community of plant enthusiasts and learners, incorporating user feedback throughout the development process is crucial. The model should support frequent interactions with stakeholders to ensure that the system meets their needs effectively.

Considering these factors, Agile Software development model, such as Kanban, or Scrum would be suitable for this project. Agile method focuses more on collaboration with the stakeholders, iterative development, and flexibility to adjust the changing needs. By delivering working software in incremental stages, Agile enables early feedback and adjustments, which suits the project's dynamic nature and the importance of user input for ongoing improvement.

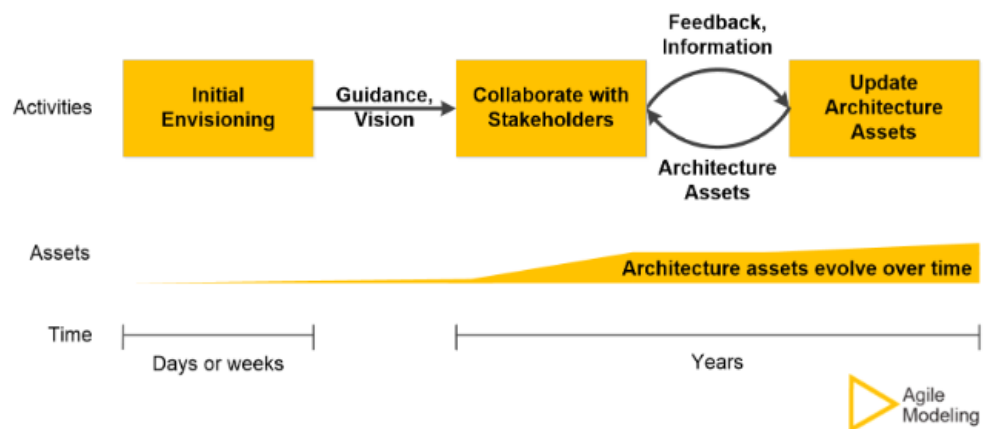


Figure 5: Agile Software model.

8.2 Design Pattern

Design patterns are responses to common challenges encountered by software engineers during the development processes. Design patterns offer developers a shared platform, standard terminology, and are scenario specific.

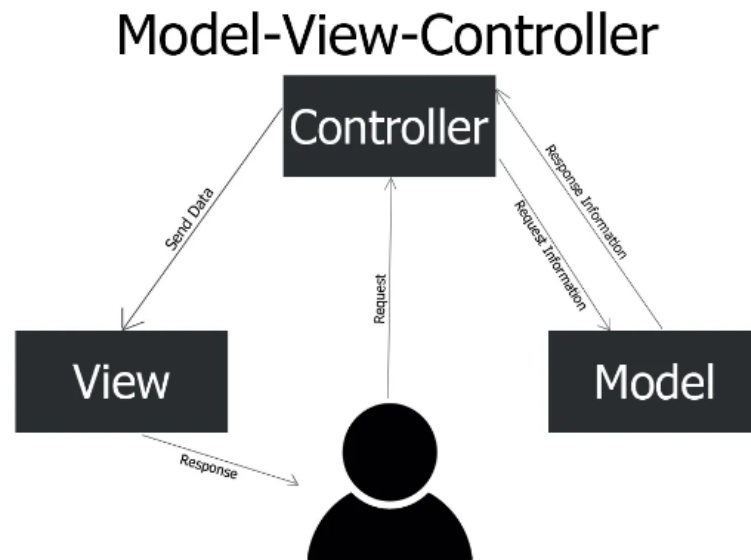


Figure 6: MVC pattern figure.

- **Model:** The backend containing all the data logic. The model oversees managing data from any source whether it is from a database, API, or JSON object.
- **View:** The frontend or graphical user interface. View component is accountable for displaying the information obtained from the Model in a user-friendly and easily comprehensible manner for the user.
- **Controller:** The application's control center, determining how data is displayed. The role of the controller is to collect, modify, and deliver data to the user. The controller essentially links the view and the model. The controller retrieves data from the model and initializes the views using getter and setter functions.

The MVC architecture pattern simplifies the development of complex applications to a more feasible process. It allows you to segregate the frontend and backend code. This way, each side may be managed and changed without interfering with the other. Given the benefits of Separation of Concerns, Simultaneous Development, Testability, Scalability, and Code Reusability that the MVC design pattern provides, it is recommended that this pattern be used for the system's future development.

8.3 Programming Language

The system can be designed, and the development phase can be done in Java as the main programming language, alongside the Object-Oriented Programming Paradigm and the Model View Controller design pattern.

Java is a multi-operational, object-oriented, network-dominant programming language that perhaps will be. also used a standalone platform. It is an easily, safely, and dependably programming language that saves time and efforts. are applied in creating different from mobile apps and business software to big data applications and server-side technologies.

Java has a significant and engaged developer community. There is an extensive range of resources and libraries accessible for developers to leverage. Java is also known for its high portability, which enables code written in Java to be conveniently deployed and executed on various hardware and operating systems without modification, making it a versatile and flexible language.

8.4 Testing

Testing software is a crucial aspect of developing high-quality software that functions correctly. In this phase the software functionality is evaluated for errors and deficiencies. It also enhances efficiency and effectiveness. Testing constitutes a vital component of the Software Development Life Cycle (SDLC). There are two general methods in which software can be evaluated: Functional Testing and Non-functional Testing. Functional testing methodologies can be employed while carrying out tests for the system. There are many types of functional testing, each with their own goals and criteria. They are as follows:

- **Unit Testing:** Unit testing is a category of software testing that validates the functionality of individual software units and determines whether the corresponding code performs as intended. A unit refers to the most basic testable component of an application.
- **Integration Testing:** Integration testing is the second stage of the software testing process, following unit testing. During integration testing, units or separate software components are tested in conjunction with one another. The purpose of this testing method is to detect flaws in the interactions of integrated components and units.
- **System Testing:** System testing is the process of testing integrated software. The goal is to assess the system's adherence to the specified requirements. The quality assurance team assesses how each component of the application or software works together in a comprehensive, integrated environment during system testing.
- **Regression Testing:** Regression testing makes sure that a component continues to function properly after new components are added to the application. When anything changes, such as adding a new module to the software, regression testing is performed.

8.5 Maintenance

The Software Development Life Cycle includes software maintenance. Its primary purpose is to alter and update software applications after they have been delivered to correct faults and improve performance. Software is a simulation of the real world. When the real-world changes, the software must be updated where possible.

Software maintenance is a broad activity that encompasses error correction and optimization. It is critical that the McGregor software system meets the needs of end users and functions efficiently and effectively. To ensure that we carry out multiple maintenance actions. These actions can be categorized as follows:

- **Corrective Maintenance:** Corrective maintenance tries to correct any residual flaws in specs, design, coding, testing, and documentation, among other places.
- **Adaptive Maintenance:** This entails altering the software system to respond to variations in the environment, such as hardware or software modifications.
- **Preventive Maintenance:** It is a measure taken to guard against the system becoming outdated where an existing system utilizing outdated technology is reengineered using modern technology. This upkeep stops the system from failing.
- **Perfective Maintenance:** It describes enhancing software performance or efficiency or limiting software to increase changeability. This might include enhancing the functionality of the current system or increasing computational effectiveness and so on

9 Conclusion

To summarize our object-oriented development class, we were dealing with the creation of complex modules for the McGregor Institute of botanical learning. This was a very new experience for us since it was something that we couldn't have even imagined. The first time that I was attempting to develop class diagrams, use cases diagrams, and sequence diagrams I faced the challenge. It was these things most importantly, therefore, showing both who interacts with who and how they interact, which to me, proved to be the tough tasks.

The course of project work under the Rational Unified Process (RUP) methodology becomes a useful and relevant instrument for addressing issues and planning of tasks. The Gantt chart is unquestionably one of the most visual and effective tools that we utilized to keep track of each progress phase so that is it always completed on schedule. I had an opportunity to draught the use case diagrams, communication diagrams, class diagrams as well as I expanded both my intellect and design competency. The above tools become indispensable elements of any software engineering project as the plan and becomes objective in goal attainment. Moreover, the making the McGregor systems prototype demystified the core features and the mechanics about how and this is the way the user and the system interacted which was basically the way any succession is defined.

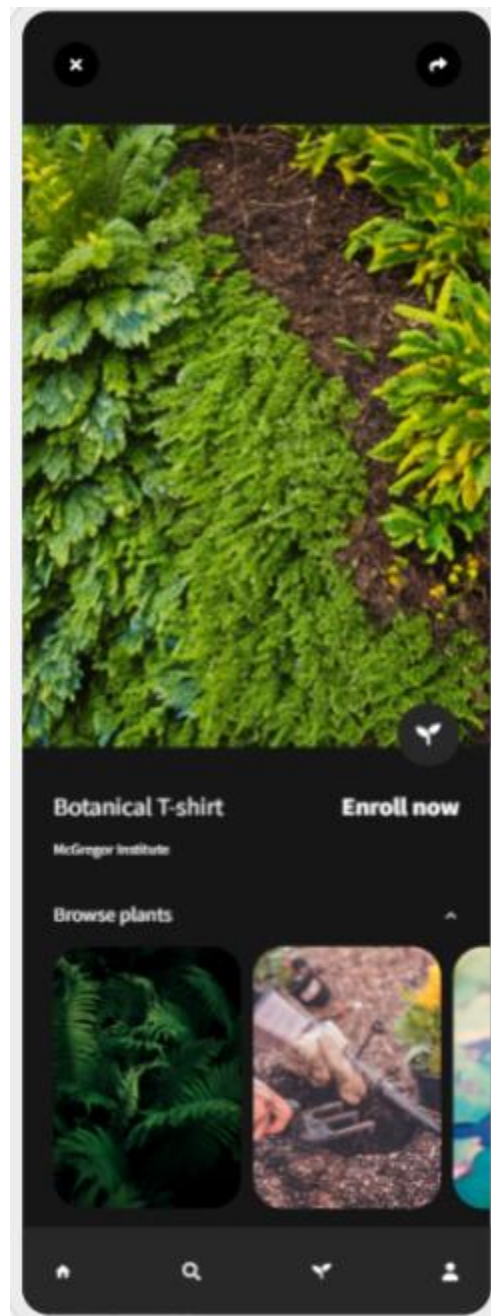
This course makes me develop technical skills, and it also provides me a groundwork in case I am assigned to software projects in future. Additionally, handsight of what I have established professionally is going to be a huge help to my future career because I'm going to be more confident in dealing with complex situations. Just that project has enhanced my self-confidence and career competition in the field software engineering, because of the strong base which it has provided for my future professional and education advancement.

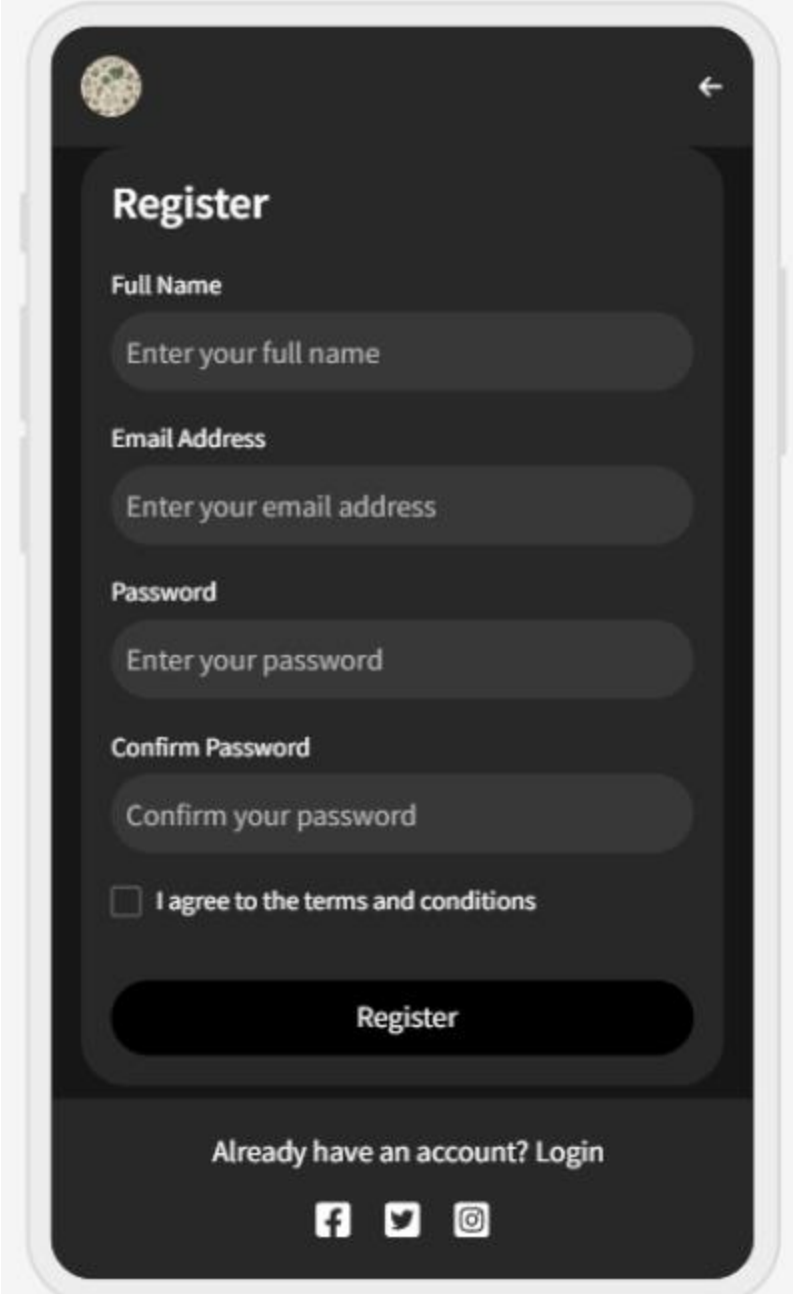
10 Prototype



10.1 Home page



10.2 Register member.





Register

Full Name
Enter your full name

Email Address
Enter your email address




Password
Enter your password

Confirm Password
Confirm your password

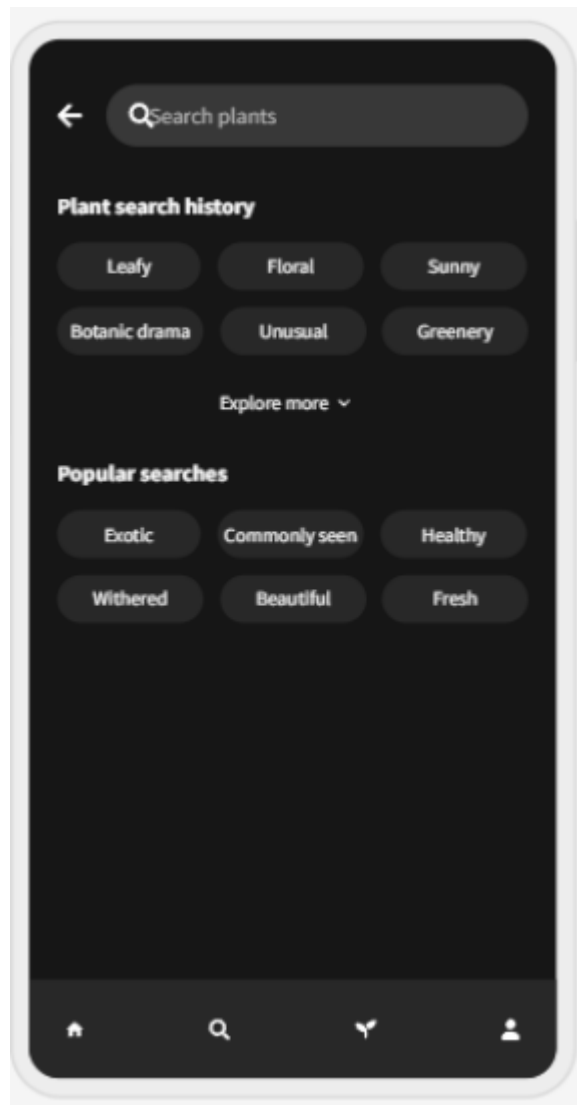
☐ I agree to the terms and conditions

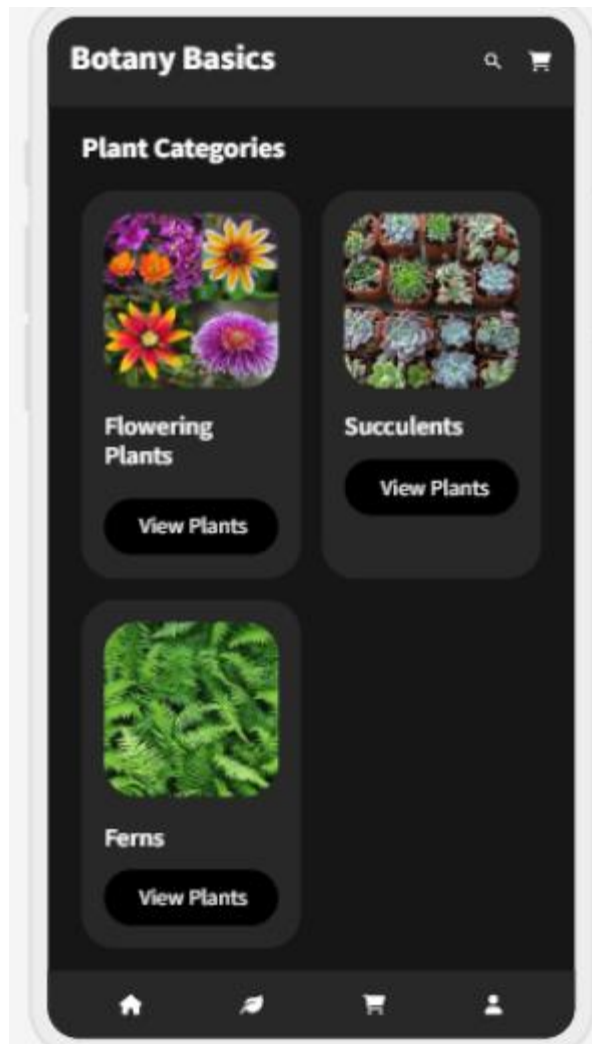
Register

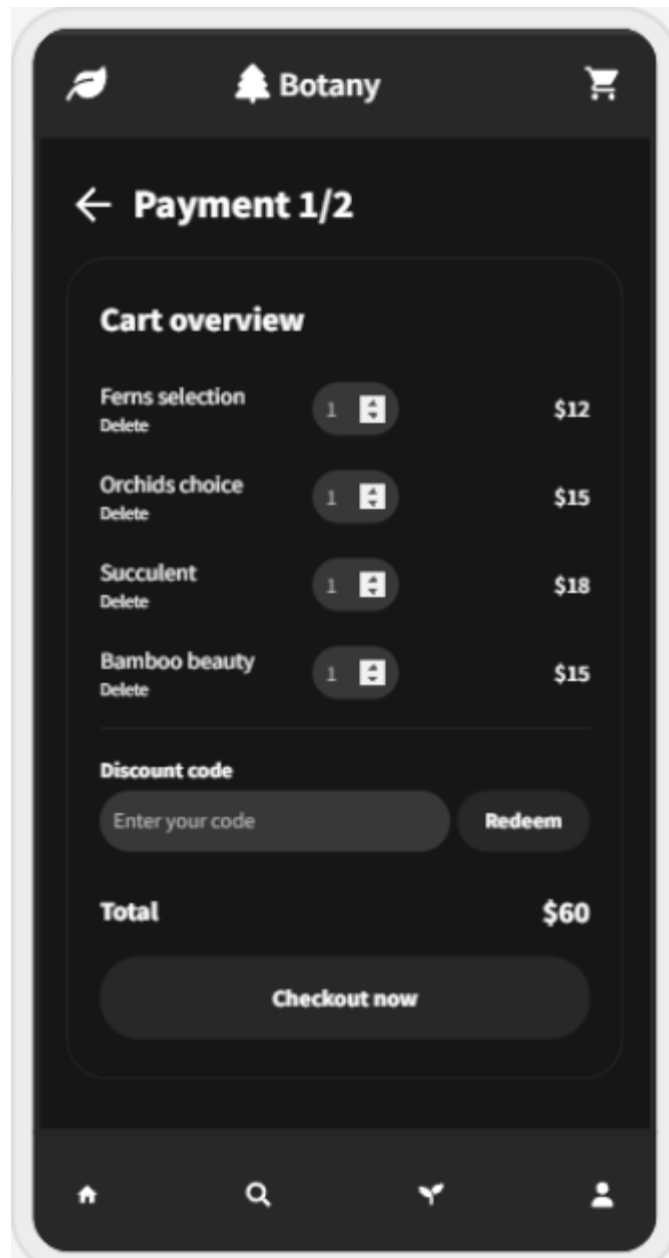
Already have an account? Login

10.3 Purchase Plant







10.4 Payment

basket.

Infor > Delivery > Payment > Order

Payment

Transactions are securely encrypted.

Cardholder's Name

Jane Doe

Card Number

9876 5432 1010 1213

Expiry Date

12/23

CVC Code

☒ Remember my card details

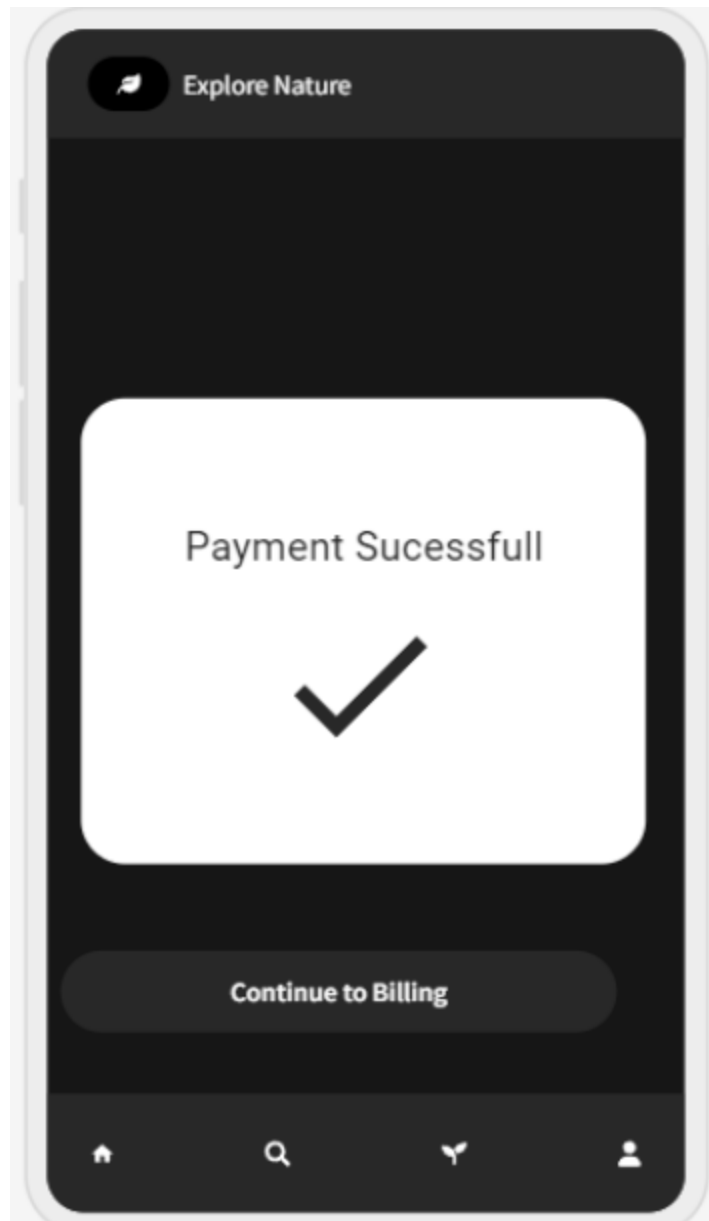
Billing Address

Select the billing address for payment.

☒ Same as Shipping Address

☐ Different Billing Address

Proceed to Summary



10.5 Join the Program

The image shows a mobile application interface for joining a botanical program. The app has a dark theme with green accents. At the top, there's a header bar with a green circular profile picture placeholder, the title "Join Program", and a back arrow. Below the header, a subtitle reads "Connect with nature through botanical studies". The main content is divided into three sections: "Program Details", "Course Options", and "Registration Form".

Program Details

Botany Exploration Program
Explore the diverse world of plants through this interactive program.

Eligibility: Graduate
Courses: Paid

Course Options

Botanical Illustration
Learn the art of botanical illustration with our expert instructors.
Paid Course

Plant Taxonomy
Discover the classification and naming of plants in this insightful course.
Unpaid Course

Registration Form

Full Name

Email Address

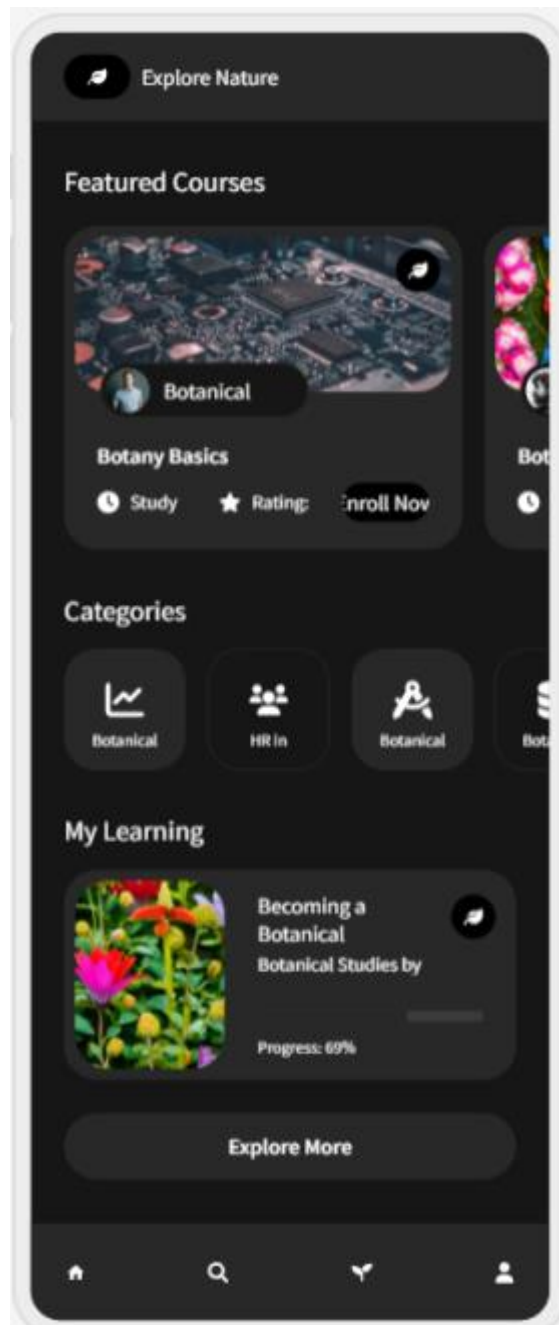
Educational Background

☐ Interested in Botanical Illustration Course

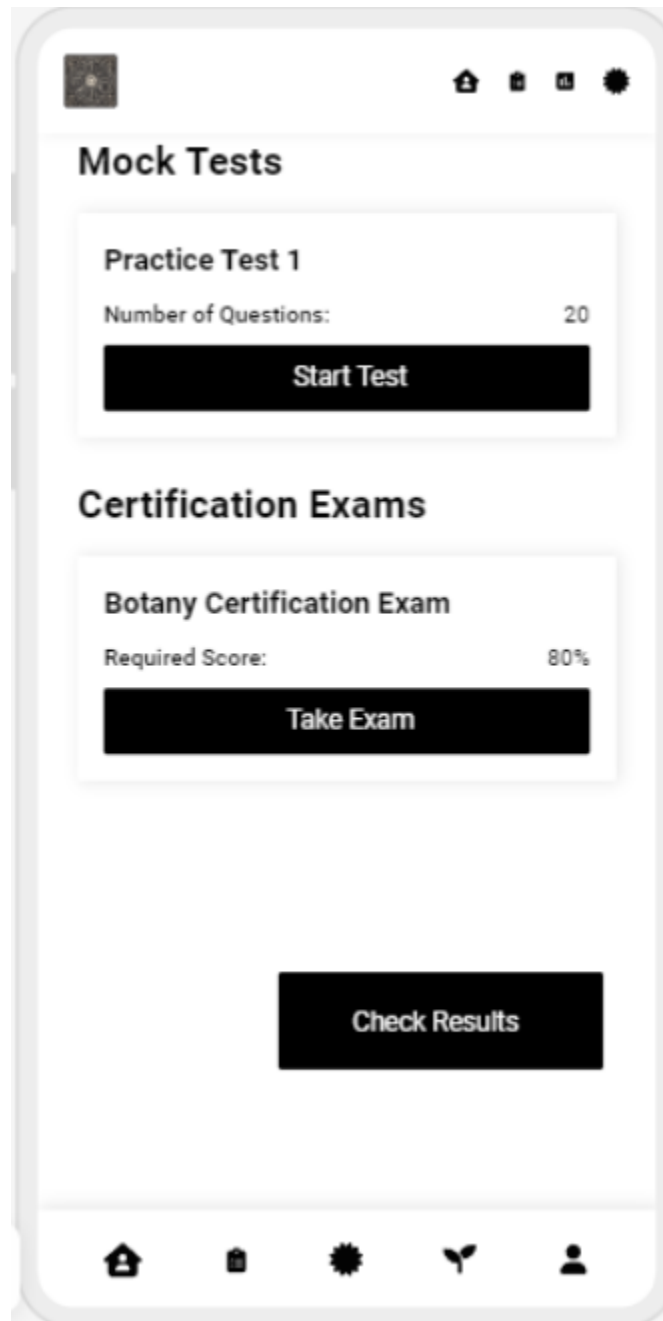
Submit

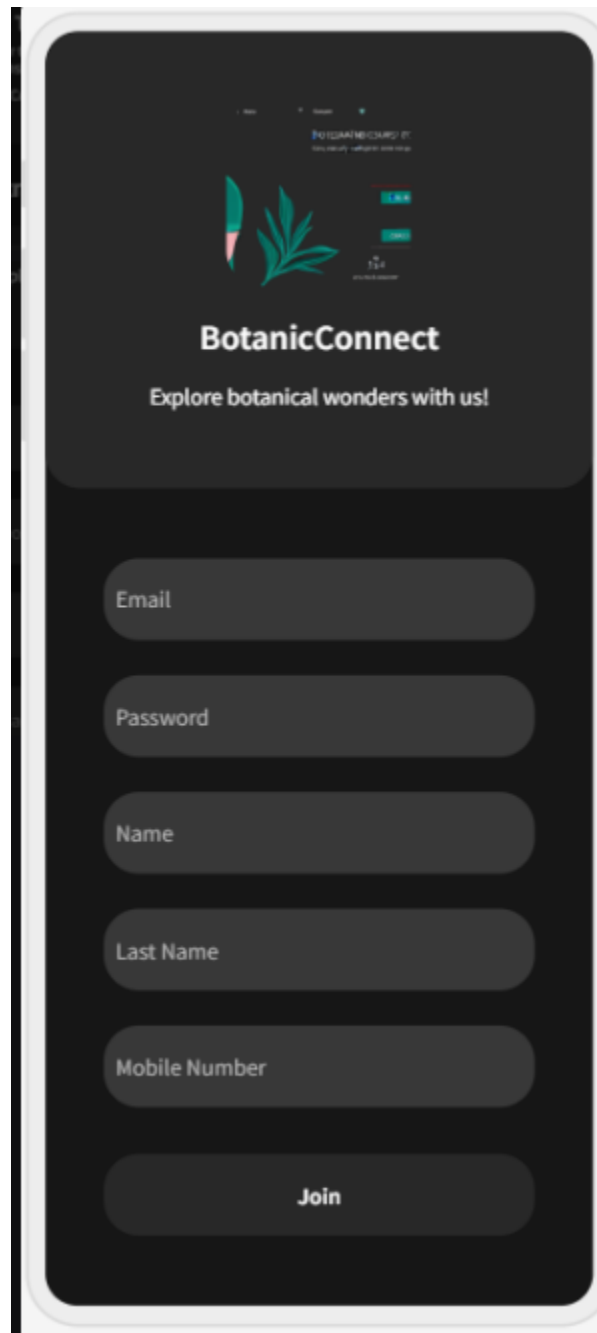
At the bottom, there is a navigation bar with four icons: a home icon, a calendar icon, a group of people icon, and a user profile icon.

10.6 Choose a Course



10.7 Take Certificate Examination



10.8 Form

The image shows a mobile app interface for BotanicConnect. At the top, there is a header with a logo featuring a green leaf and a pink flower, and the text "BotanicConnect" and "Explore botanical wonders with us!". Below the header, there is a registration form with the following fields: Email, Password, Name, Last Name, and Mobile Number. Each field is represented by a dark gray rounded rectangle with the field name inside. At the bottom of the form, there is a "Join" button, also represented by a dark gray rounded rectangle.

BotanicConnect

Explore botanical wonders with us!

Email

Password

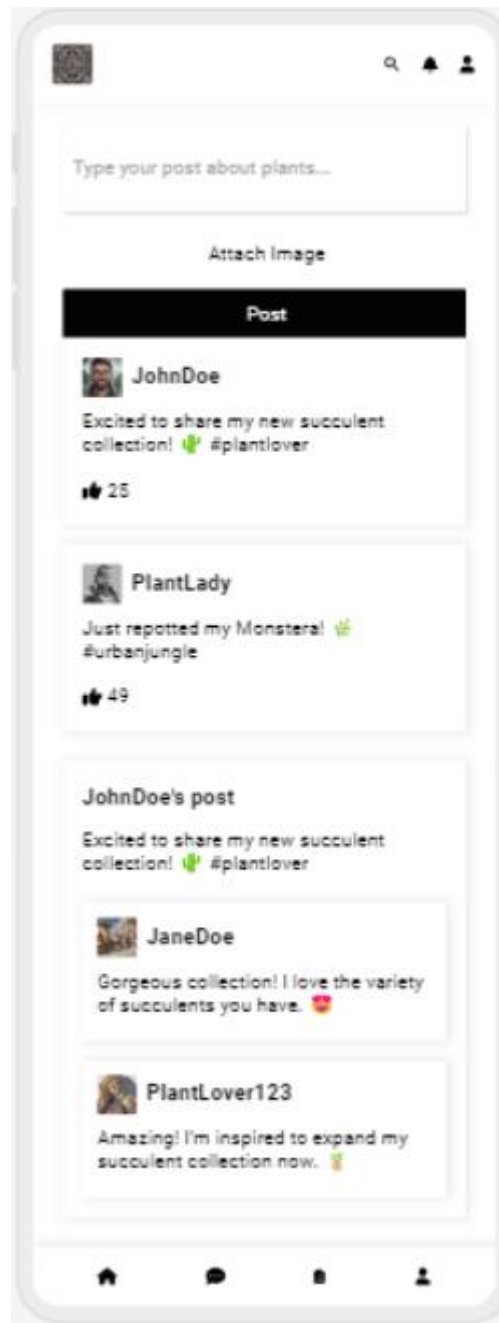
Name

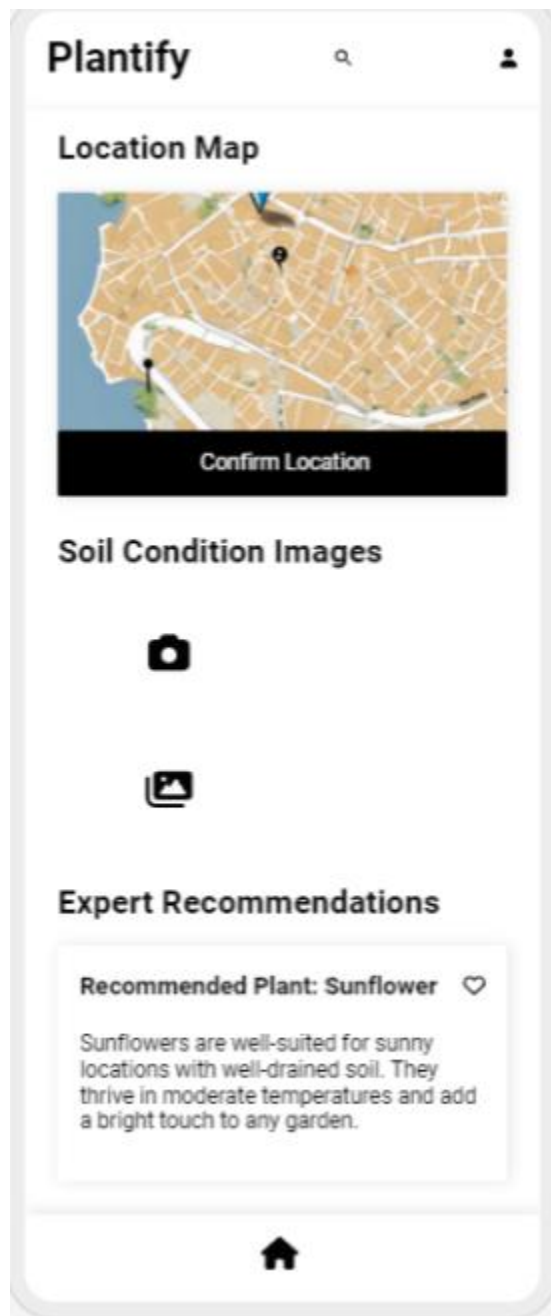
Last Name

Mobile Number

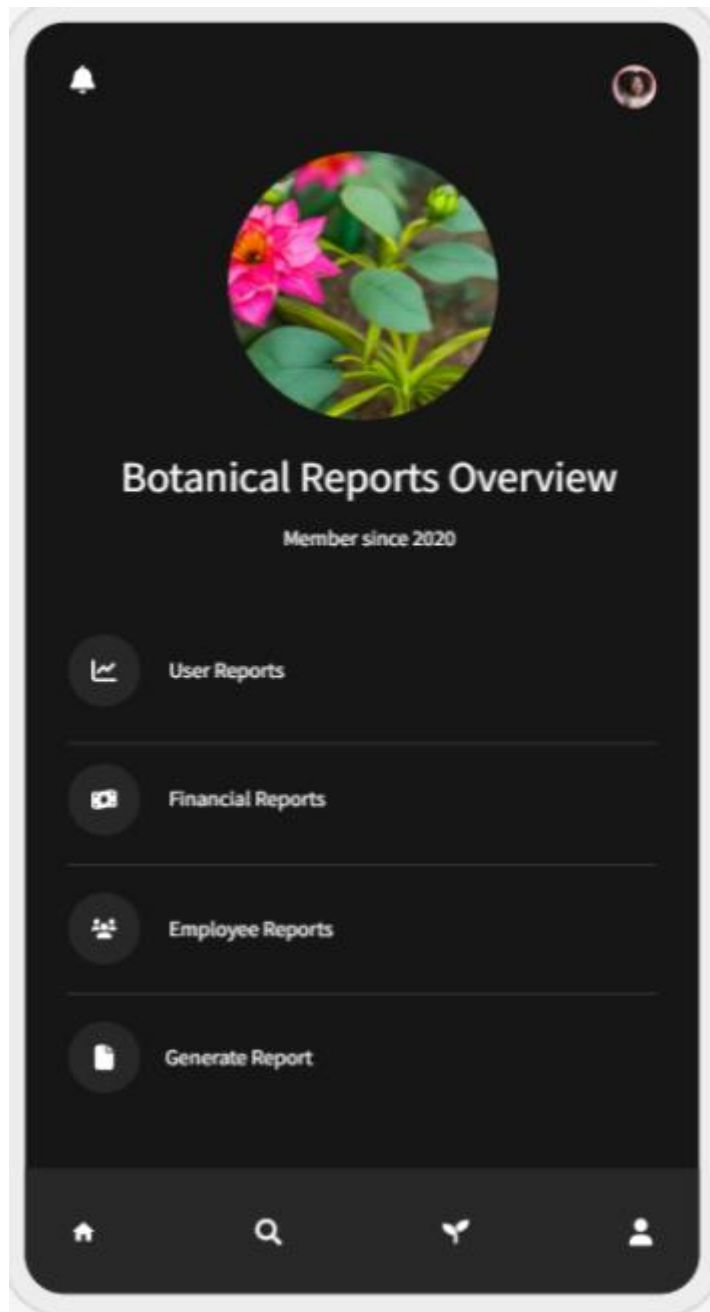
Join

10.9 FeedBack



10.10 Ask for recommendation.

10.11 Report Generation



11 References

figma, 2024. *figma.com*. [Online]

Available at: <https://www.figma.com/resource-library/what-is-a-use-case/>

[Accessed 01 05 2024].

geekforgeeks, 2024. *geekforgeeks.com*. [Online]

Available at: <https://www.geeksforgeeks.org/collaboration-diagrams-unified-modeling-languageuml/>

[Accessed 02 05 2024].

lucidchart, 2024. *lucidchart.com*. [Online]

Available at: <https://www.lucidchart.com/pages/uml-sequence-diagram>

[Accessed 02 05 2024].

Raeburn, A., 2024. *asana*. [Online]

Available at: <https://www.projectmanager.com/>

[Accessed 16 04 2024].

Ramos, D., 2021. *smartsheet*. [Online]

Available at: <https://www.smartsheet.com/content/gantt-chart-pros-cons>

[Accessed 01 05 2024].