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: Ankit Karna

London Met ID: 22067501

College ID: NP01CP4A220017@islingtoncollege.edu.np

Assignment Due Date: Tuesday, May 7, 2024

Assignment Submission Date: Tuesday, May 7, 2024

Title (Where Required): McGregor Institute of Botanical Training

Word Count (Where Required): 5645

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.

ACKNOWLEDGEMENT

My sincere gratitude goes out to Mr. Rubin Thapa, our Software Engineering module leader and Mr. Dipesh Raj Adhikari who is out Tutor for the same module for their great advice and assistance during this coursework. Their advice and views were helpful in shaping this project. Their collaboration was crucial to this project's successful conclusion.

ABSTRACT

The objective of this report is to improve the educational and community-building initiatives for the McGreggor Institute of Botanical Training by presenting the design and analysis of a suggested online system. Numerous features, including user registration, course enrollment, plant purchases, payment processing, expert recommendations, and community forums, are supported by the system's design. A thorough model that shows the system's operations through use cases, class diagrams, and sequence diagrams was created using OOP analysis and design techniques. This report describes the development methodologies, design patterns and architectural decisions that were made to successfully meet the needs of the institute. In addition to streamlining administrative duties, the system is intended to promote a thriving community of plants enthusiasts, enabling information sharing and cooperative learning.

TABLE OF CONTENTS

TABLE OF TABLES

TABLE OF FIGURES

1. Introduction

The process of designing, creating, testing, and maintaining software is known as software engineering. It is an organized, methodical method of software development with the goal of producing software that is dependable, high-quality, and maintainable. A wide range of methods, instruments, and strategies are used in software engineering such as requirement evaluation, design, testing and maintenance. To enhance the software development process, new tools and technologies are continuously being developed in this quickly expanding sector (GeeksforGeeks, 2024).

In this coursework, the goal is to design a system for “McGregor Institute of Botanical Training”. Situated in the center of Nepal, the McGregor Institute of Botanical Training is a revolutionary establishment committed to developing a profound enthusiasm for horticulture and agriculture.

The McGregor Institute has been an inspiration for anyone interested in learning more about the complex world of plants for more than seven years thanks to its unwavering dedication to botanical education. As part of Dublin City University, we provide a wide range of undergraduate and graduate programs together with to an innovative new offering of short-term certification courses designed to meet the growing need for horticulture.

We at McGregor Institute do more than just teach, we foster a thriving community of plant enthusiasts who are bound together by an understanding of botanical delights. Through our dedicated forum, our platform not only makes it easier to enroll in courses and acquire plant varieties, but it also encourages vibrant discussions and idea exchanges. It is the ideal place to start your botanical adventure, regardless of your level of experience. Whether you are a beginner searching for advice on plant selection or an experience enthusiast wishing to impart your knowledge. Come grow with us as we plant curiosity seeds, foster development, and create a vibrant community of plant enthusiast who are committed to protecting our natural heritage.

2. Methodology

A process or set of procedures used in software development is known as software development methodology. Once more, very general, but that it includes stages like design and development. Usually, it takes the shape of distinct stages. Its purpose is to explain the how of a software product's life cycle (Alliance Software , 2024).

For this project, I am going to use Scrum Methodology.

2.1 Scrum

Scrum is an agile project management framework that encourages self-organization, reflection, and experience-based learning among team members. It was inspired by rugby. Its concept can be applied to a variety of collaborative scenarios, albeit they are primarily used in software development. Meetings, resources, and roles all work together in Scrum to efficiently manage and organize work while encouraging flexibility and ongoing development (Software Development, 2024).

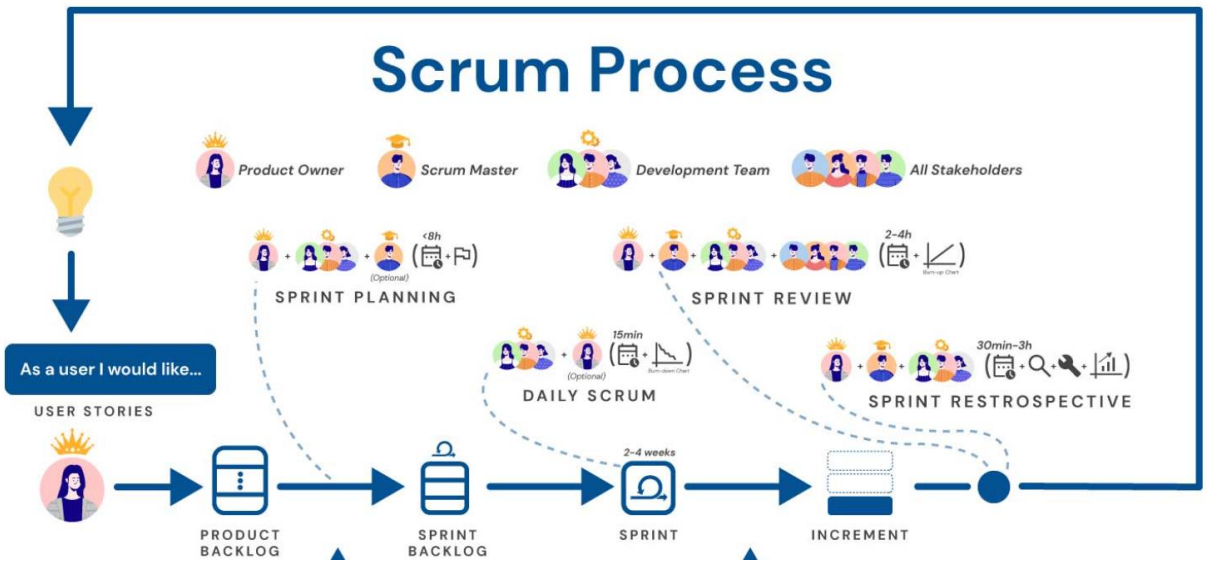

Figure 1: Scrum Process (donetonic, 2022).

2.2 Roles in Scrum

Inside a scrum team, there are quite a few fundamental jobs, and although team members collaborate, each roles have distinct tasks and obligations.

These are the main participants that make up a scrum team:

- **Scrum Master**: The scrum master is a person with qualification or experience in the Scrum framework. They mentor another team members on various procedures by using their knowledge.

- **Product Owner**: On a Scrum team, the product owner oversees creating high-value products. Their expertise lies in managing development teams, and they carefully consider project choices to make sure they complement team objectives.

- **The Development Team**: With Scrum, development team is made up of qualified experts who oversee producing a top-notch product at the conclusion of each sprint. They work well in teams and possess strong organizational, time-management, and problem-solving skills.

- **Stakeholders**: Stakeholders in Scrum represent different viewpoints from other departments or outside entities and offer feedback and influence over the project's result. Despite not being on the development team, their views are important (indeed, 2022).

2.3 Why use Scrum?

The scrum technique provides several unique benefits in relation to the McGregor Institute of Botanical Training's development project for a multipurpose educational and e-commerce platform:

- **Rapid and Progressive Distribution**: The project team may deliver certain system components early, like user registration and the fundamentals of course enrollment, thanks to Scrum's iterative nature. This implies that instead of waiting for the system to be finished, the institute may start getting input from users and adjusting faster.

- **Development Focused on Users**: The development team guarantees that the system satisfies user needs- a crucial component for features like interactive forum and plant suggestion system-by involving stakeholders, including users, in sprint review.

- **Improved Quality of Product**: In particular, for complex aspects like payment handling and integral regional data for plant selections, regular sprint evaluations and testing are essential for identifying problems early on.

- **Enhanced Project Management and Forecasting**: Project progress is kept on track by Scrum's regular stand-ups and sprint reviews, which make precise budget and timeframe projections possible.

- **Adaptability in Defining Priorities**: By prioritizing tasks according to client and stakeholder input, Scrum guarantees that the most useful features are built first.

For example, it can prioritize certification examinations if users show a high level of interest in them.

- **Morale and Teamwork**: Scrum encourages everyone to participate in conversations and decision-making which improves team morale and product quality.

- **Expense Reduction**: The institution may more effectively handle its budget by building the framework in sprints, distributing cash gradually, and modifying financial plans in response to the needs and results of the most recent projects. This gradual spending is essential to an educational institution's efficient resource management.

- **Mitigation of Risk**: System failures and user unhappiness are reduced by frequent reviews and feedback adjustments in Scrum, which addresses problems early in the development process rather than after they are deployed.

- **Participation of Stakeholders**: Scrum involves all stakeholders-faculty, students, and plant enthusiasts-and offers several avenues for their feedback. This involvement is essential to developing a system that appeals to users and advances community objectives.

All these benefits add up to a method of development that is better managed, effective, and in line with the strategic goals of the McGregor Institute of Botanical Training, which guarantees the system's successful implementation as an educational and community-focused resource.

3. WBS (Work Breakdown Structure)

A project's visual, ordered, and deliverable-oriented breakdown is represented by a work breakdown structure (WBS). Project managers can use this diagram to see all the tasks needed to finish their project and to break down the scope of their work (ProjectManager, 2024).

The below diagram shows the work breakdown struct of McGregor Institute of Botanical Training.

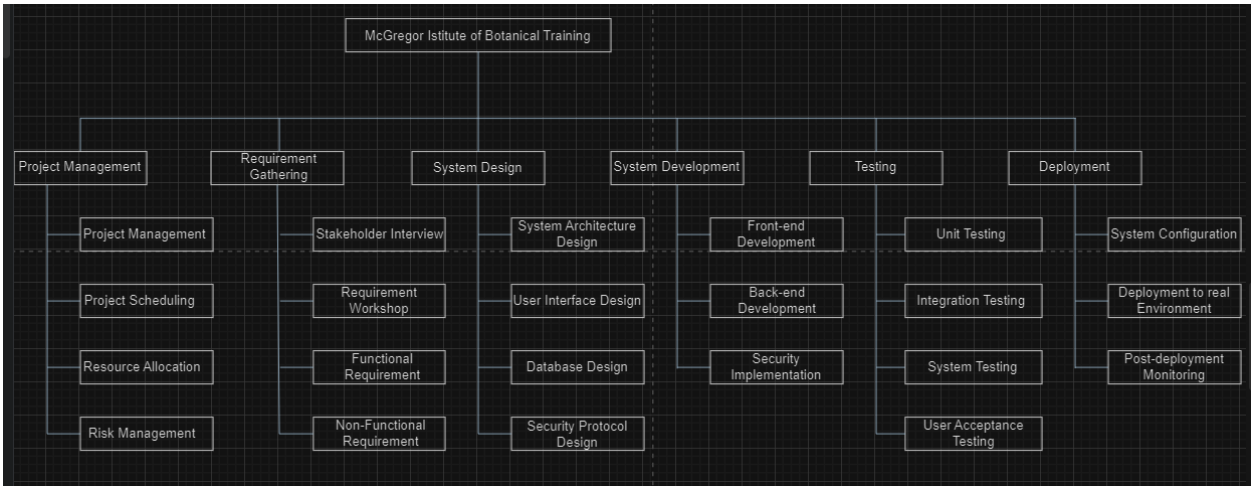

Figure 2: WBS structure.

4. Gantt Chart

One tool for project management which assists with scheduling and planning projects of any kind is the Gantt Chart. They are especially helpful for visualizing projects. An activity's graphical representation versus time is called a Gantt Chart, and project managers can use it to track their progress (Association for Project Management, 2024).

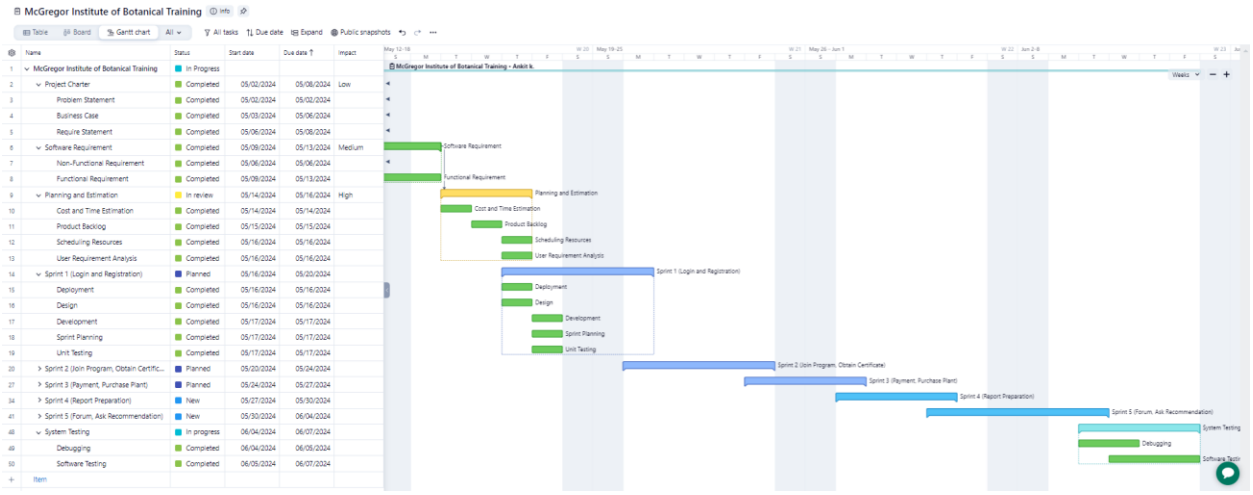

Figure 3: Gantt Chart for System Development.

The Gantt Chart for the development of McGregor Institute of Botanical Training was done using an Agile Methodology known as Scrum. It reflects all the works done along with their time duration. It also shows all the schedules related to deliver the requirements of McGregor Institute of Botanical Training.

The first phase in the gantt chart above is planning and estimation where there the team members plans about the projects, schedules resources, analyses the user requirements and product backlog with the client. In the first sprint, the team works on the development of login and registration which is tested and then deployed. In the second sprint, the team worked on join program and obtain certificate page. Similarly, in the third sprint, the team

worked on payment and purchase plant page. Then, on the fourth sprint, the team worked on report preparation page and in the last sprint, the team worked on forum and ask recommendation page. At the last phase of gantt chart, system testing was done.

5. Use Case Diagram

The main source of software and system specifications for an unfinished project is a UML use case diagram. Use cases define the desired behavior (what) rather than the precise process for achieving it (how). Once they are specified, use cases can indicate both written and visual representation (Visual Paradigm, 2024).

The notations used in use case diagrams are as follows:

- **System**

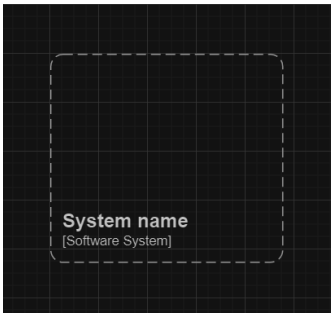

Figure 4: System boundary structure of a use case.

Use a rectangle that has use cases inside of it to draw the boundaries of the system. Put actors outside the bounds of the system.

- **Use Case**

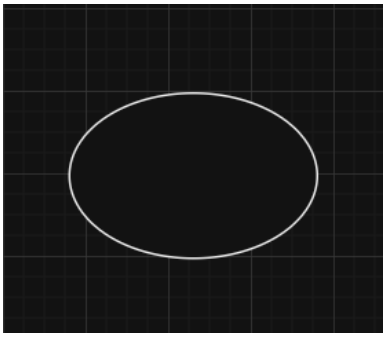

Figure 5: Use case structure of a use case.

Illustrate use case scenarios with ovals. Put verbs that stand in for the various functions of the system on the ovals.

- **Actor**

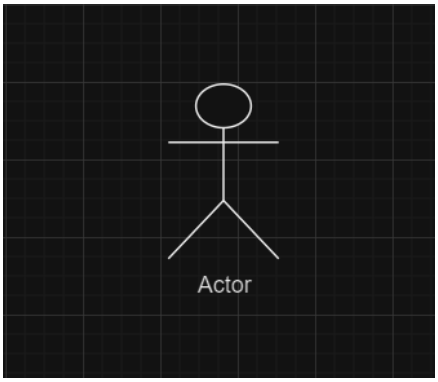

Figure 6: Actor structure of a use case.

Customers of a system are Actors. Put the actor system's model on it when one system acts as the actor of another (Edrawsoft, 2024).

- **Communication Link**

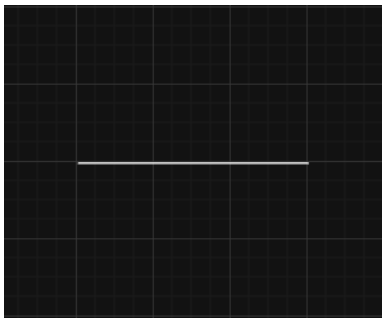

Figure 7: Communication link structure of a use case.

Actors and use cases are connected using this notation, which signifies that messages are exchanged between the two.

- **Association**

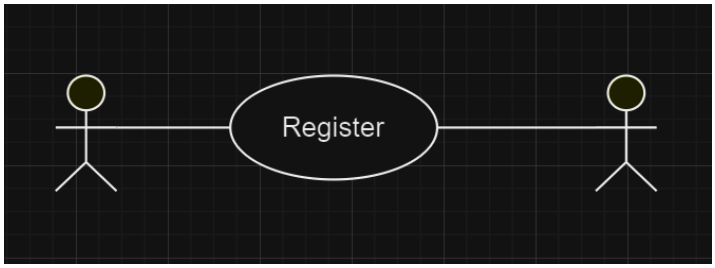

Figure 8: Association structure of a use case.

Actors indicates users or other systems engaging with the system under design in any use case diagram. They can take part in more than one use case, but they must be connected to at least one. Similarly, several actors may be involved in a single use case. The foundation of a well-defined user-system interaction paradigm is this mutual relationship (Cinergix, 2022).

- **Generalization of an actor**

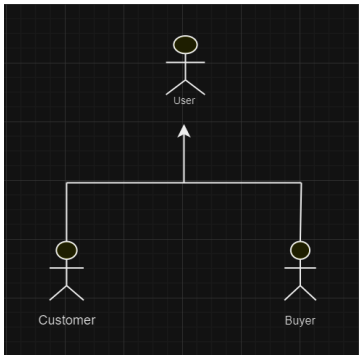

Figure 9: Generalization structure of a use case.

When an actor generalizes, they can take on the part of another actor. All the ancestor's use cases are passed on to the descendant. There are several use cases unique to that job in the descendent. Here, the actor "User" is generalized into "Customer" and "Buyer" (Cinergix, 2022).

- **Extends**

Figure 10: Extend structure of a use case.

An extend connection can be used in UML modeling to indicate that the behavior of one-use case is extended by another use case. Relationships of this kind make system specifics visible that are typically concealed in use cases (Visual Paradigm, 2022).

- **Include**

Figure 11: Include structure of a use case.

An include relationship in UML modeling is one where the feature of one-use case is contained in another use case. The use case model's repetition of functionality is supported via a confinement relationship (Visual Paradigm, 2022).

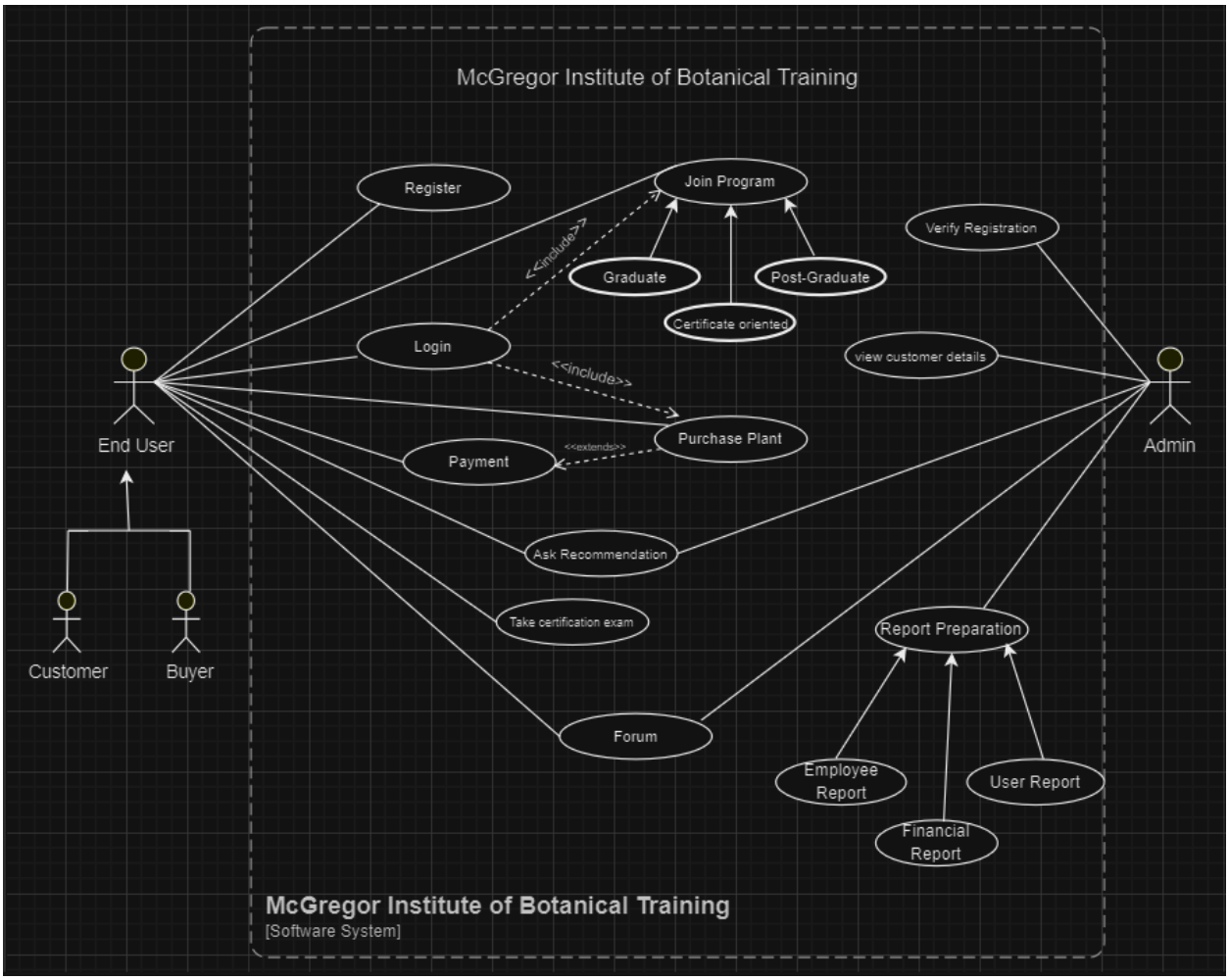

Figure 12: Use Case diagram of McGregor Institute of Botanical Training.

6. High Level Use Case

Multiple use cases may be called by a high-level use case. A functional explanation of a procedure from the perspective of the end user. It provides a non-technical description of how the system's users interact with one another (empirica, 2024).

I. Register

Use Case	Register
Actor	End User
Description	It allows a new user to create an account by providing their personal details in the system.

Table 1: High level use case of Register.

II. Login

Use Case	Login
Actor	End User
Description	Users can log in to the system after they are registered and access its features.

Table 2: High level use case of Login.

III. Join Program

Use Case	Join Program
Actor	End User
Description	It allows users to enroll in available educational programs or short courses.

Table 3: High level use case of Join Program.

IV. Payment

Use Case	Payment
Actor	End User
Description	After enrollment in a course or buying a plant, users make payment.

Table 4: High level use case of Payment.

V. Purchase Plant

Use Case	Purchase Plant
Actor	End User
Description	Users can browse, select, and purchase various plants from the platform.

Table 5: High level use case of Purchase Plant.

VI. Ask Recommendation

Use Case	Ask Recommendation
Actor	End User
Description	Users can submit their questions for which plants or crops would be best suited for their specific location and conditions.

Table 6: High level use case of Ask Recommendation.

VII. Take Certification Exam

Use Case	Take Certification Exam
Actor	End User
Description	Users can take certification exam related to their courses after meeting certain criteria.

Table 7: High level use case of Take Certification Exam.

VIII. Report Preparation

Use Case	Report Preparation
Actor	Admin
Description	Admin can generate and view reports related to finance, users, and employees.

Table 8: High level use case of Report Preparation.

IX. Forum

Use Case	Forum
Actor	End User, Admin
Description	Users can post, comment and interact in discussion with the plant enthusiast related to plant and horticulture.

Table 9: High level use case of Forum.

X. Verify Registration

Use Case	Verify Registration
Actor	Admin
Description	Admin verifies the registration of the users.

Table 10: High level use case of Verify Registration.

XI. View Customer Details

Use Case	View Customer Details
Actor	Admin
Description	Admin can see all the details of the users registered in the system.

Table 11: High level use case of View Customer.

7. Expanded Use Case

Adding features to a preexisting use case without changing the original use case could be done using an extended use case (Visual Paradigm, 2024).

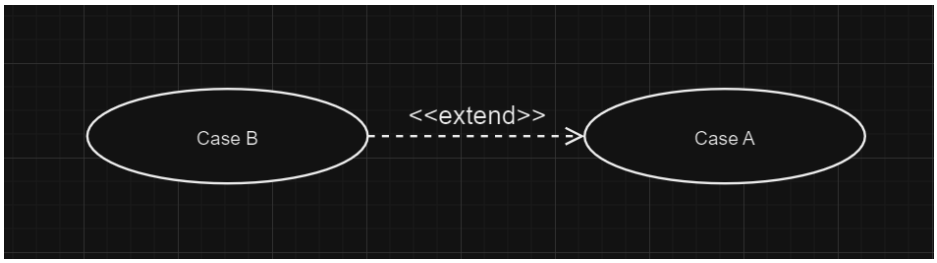

Figure 13: Example of Extended use case.

Case A and Case B are the two use cases in the above example. Case B is an extension of Cas A, meaning that it modifies Case A's fundamental behavior while adding new capabilities. Case B's function is carried out when a specific circumstance within Case A is satisfied.

i) Login

Use Case Name: Login

Actor: End User

Purpose: Log in to the system "McGregor Institute of Botanical Training"

Overview: Before entering the system, the system displays a login page where users should enter their valid login credentials to access different functions such as buying plants, enroll in courses, taking certification exam, participate in forum, etc. If the user enters invalid details, then the system asks the user to re-enter their user ID and password by showing suitable message.

Pre-conditions: User needs to login using valid login credentials.

Post-conditions: The system logs in the user and show the appropriate homepage.

Main Flow:

Actor Action	System Response
1. The user enters their user ID and password.	
2. The user clicks on the login button	
	3. The system checks if the user ID or password is missing or not.
	4. The system validates the user login credentials.
	5. The use case ends.

Table 12: Main flow of Login.

Alternative Flow:

Actor Action	System Response
3i. The user ID or password is not there.	
	1. The system asks the user to enter the missing credential.
	2. The use case goes back to step 1 of the main flow.
4i. Invalid login credentials.	

	1. The system shows a proper error message as “Please enter correct user ID or password”
	2. The system asks the customer to re-enter user ID and password.
	3. The use case then goes to step 1 of the main flow.

Table 13: Alternate flow of Login.

ii) Join Program

Use Case Name: Join Program

Actor: End User

Purpose: Join different graduate, post graduate or certification courses in the system “McGregor Institute of Botanical Training”

Overview: The procedures a user must follow to become a member of a program and enroll in various kinds of paid and unpaid courses are described in this enhanced use case. Courses from graduate-level to short-term certification programs are offered by the program.

Pre-conditions: There must be courses in the program that are open for enrollment.

To register for courses, the user needs to have the required liberties.

Post-conditions: The user is enrolled in the chosen course.

Payment processing is done, if applicable.

The user is granted access to the course materials and enrollment confirmation.

Main Flow:

Actor Action	System Response
1. The user logs in to the system.	
2. The user goes to the join program section.	
	3. The system displays different list of courses, filtering by category.
4. User selects an option in which they wish to enroll in.	
	5. The system asks for payment if the chosen course is paid, otherwise asks for the confirmation of course.
6. The user pays for the course, if necessary, otherwise confirms the enrollment.	
	7. System records the user's enrollment and provide access to course materials.
8. Gets the confirmation message along with access to course materials.	9.

Table 14: Main flow of Join Program.

Alternative Flow:

Actor Action	System Response
	5i. If in the paid course, if entered payment detail is incorrect or the user has insufficient balance in their e-wallet, then payment won't happen.
	6. The use case goes back to step 1 of the main flow.

Table 15: Alternate flow of Join Program.

8. Collaboration Diagram

Communication diagrams are another name for collaboration diagrams. They can show how objects interact to carry out a certain use case's operation or a particular use case's component (Wondershare EdrawMax, 2024).

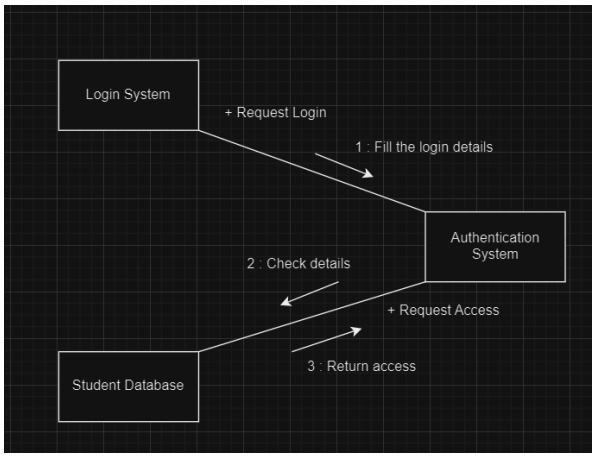

Figure 14: Example of Collaboration Diagram.

The communication flow is briefly described in the above diagram for the student database system. A login request made by a student is sent to the login system. In this case, the software's authentication system examines the request. Access is allowed if the system locates a matching student record in the database. On the other hand, an error message indicating the unsuccessful authentication request is returned if no such record is found.

Collaboration diagram of "Join Program" Use Case.

The four major components of a collaboration diagram include:

i. **Objects:**

They represent an instance of a class participating in interactions, showcasing its attributes and behavior.

ii. **Actors:**

They represent an external entity interacting with the system, typically a user or another system component.

iii. Links:

They depict the communication or interaction between objects or actors, illustrating the flow of messages or data between them.

iv. Message between objects:

Represented by arrow, it signifies a communication or interaction between instances of classes, indicating the exchange of information or triggering of behaviors.

Process:

The domain class objects, control objects and boundary object were first identified which are represented in rectangular box and are underlined in the diagram below. The name of domain classes is Course, Enrollment and User. Similarly, the name of the control object is JoinProgram, and the name of boundary object is JoinProgramUI. The actor is drawn to interact with the system's user interface after identifying the objects. The "End User" is the actor in this case.

The End User sends their personal details through enterPersonalDetails method to the boundary object and the boundary object then forwards the personal details to the controller object as recordPersonalDetails method. The controller then sends the details to the domain object User and Enrollment through constructors User and Enrollment respectively. The number 2 in the diagram has "*" sign because it is a repeating process, so the controller object asks the domain object Course to provide it with course details through getCourseDetails method and then the controller forwards the message to the boundary object through displayCourseDetails method. Then the boundary object shows the course details to the actor.

After seeing the course details, the actor then chooses a course and sends it to the boundary object through enterChosenCourse method. The boundary object then

forwards it to the controller through recordChosenCourse method and it forwards it to the domain object Enrollment through recordCourses method.

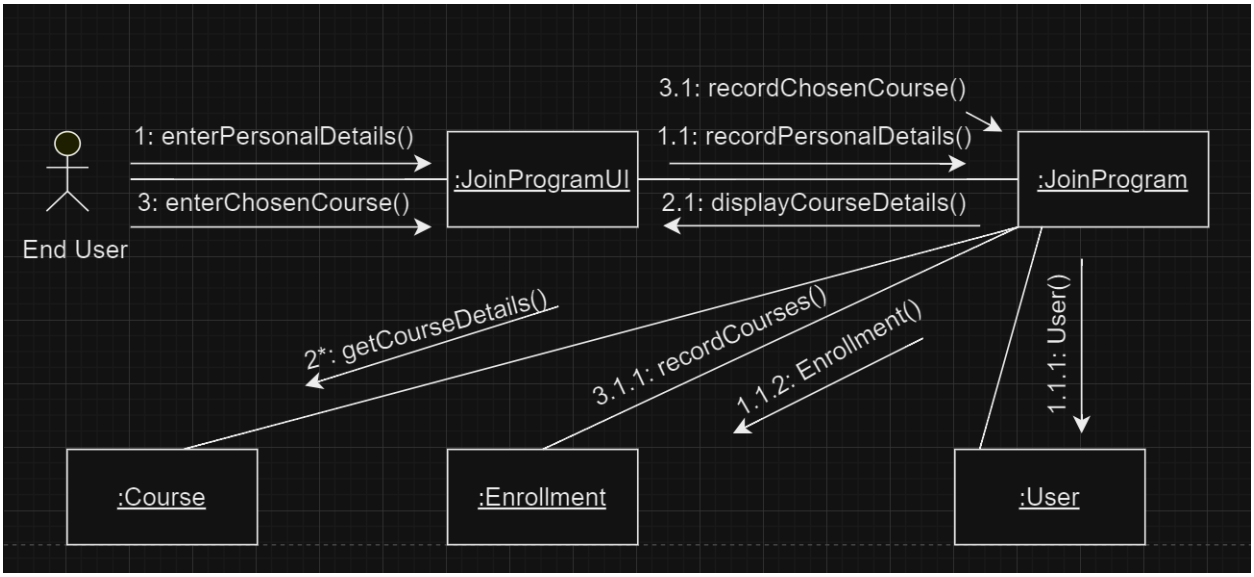

Figure 15: Collaboration Diagram of JoinProgram use case.

9. Sequence Diagram

A sequence diagram shows how things interact with one another in a sequential way. It is used to show the order in which messages travel through the system (Guru99, 2024).

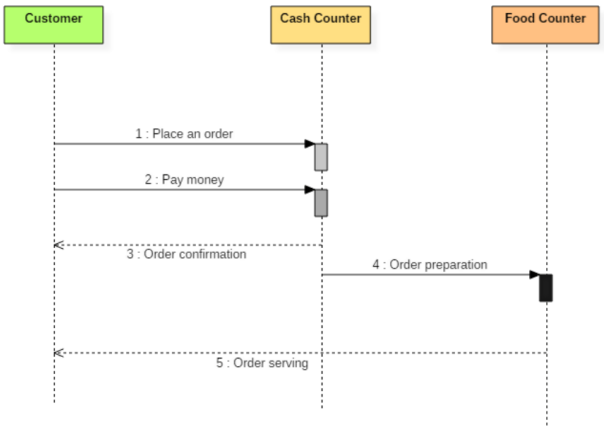

Figure 16: Example of Sequence Diagram (Guru99, 2024).

In the above example, for the sake of clarity and accuracy, each step in the organized flow from placing an order-to-order serving is indicated by a distinct message type in the sequence diagram. Changing this recommended order puts the stability of the software at risk and may result in crashes or incorrect results. Using a variety of communication types across the diagram reduces system complexity and promotes more dependable outcomes.

Sequence diagram of "Join Program" Use Case.

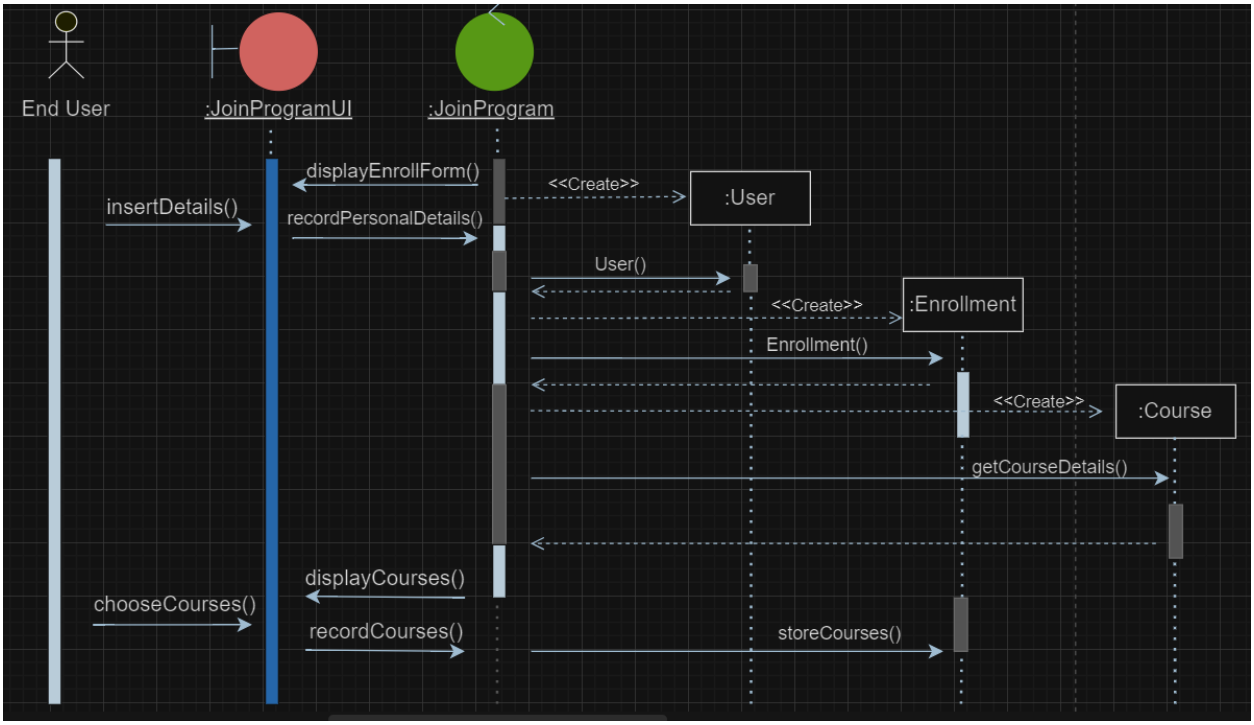

Figure 17: Sequence Diagram of JoinProgram use case.

The process of a user registering in a program through an interface, including multiple system components, is represented by the sequence diagram above.

The flow of the diagram is:

I. **End User to JoinProgramUI**: The user interacts with the JoinProgramUI which could be a webpage or application form.

- **insertDetails():** The user inserts their personal details into the form.

II. **JoinProgrsmUI to Joinprogram**: The JoinProgramUI sends the user details to the controller object JoinProgram, which handles the backend logic.

- **displayEnrollForm()**: This method shows the enrollment form or page to the user.

- **RecordPersonalDetails()**: This method holds the information provided by the user.

III. **JoinProgram**: The creation of objects and data storage operations are managed here.

- **User Creation**: **User()**: The “Usrer” object stores the personal details recorded above.
- **Enrollment Creation**: **Enrollment()**: It links the user to the program they are enrolling in.
- **Course Handling**: **Course()**: This object retrieve about the courses available for the user to enroll in from the getCourseDetails() method.

IV. Back to End User:

- **chooseCourse()**: The user chooses the course in which they want to enroll in from the UI.
- **displayCourses()**: The JoinProgramUI displays all the courses that are available to the user.
- **recordCourses()**: This method stores the details of the course chosen by the user.
- **StoreCourses()**: At last, the chosen course are stored in relation to the “Enrollment” object.

In this way, the flow of the sequence diagram completed each time a new user enrolls into the system.

10. Class Diagram

One of the main features of UML is the class diagram. They are essential in guiding in creating classes that will function as the base of your application (McNeish, 2022).

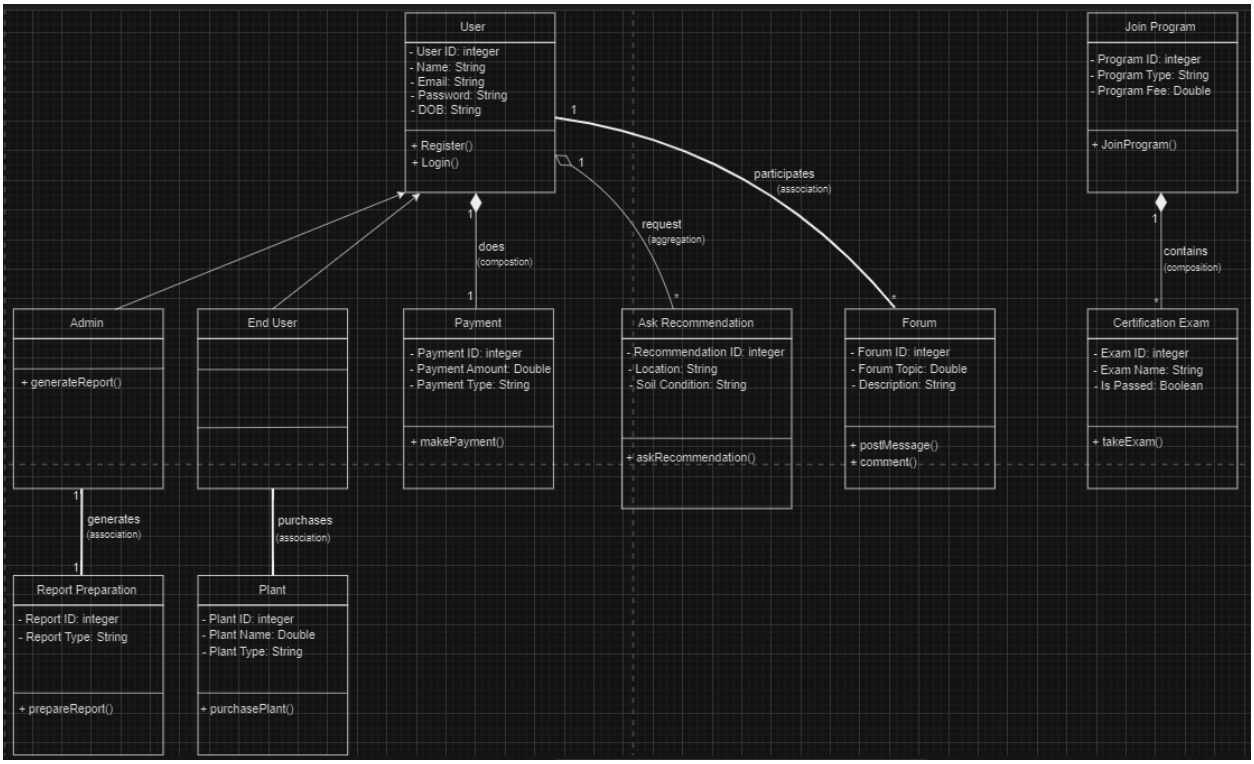

Figure 18: Class Diagram with Relation.

The above class diagram shows the relationship between each use cases along with their relationship with each other.

Here is the description of each object of the class diagram:

i. User

Attributes: UserID, Name, Email, Password DOB with data types of integer, String, String, String and String respectively.

Methods: Register() and Login() for Registration and Login purposes respectively.

ii. Admin

Attributes: Same as User above because of inheritance.

Methods: generateReport to genertare various financial, user and employee reports.

iii. End User

Attributes: Same as User above because of inheritance.

Methods: No methods

iv. Payment

Attributes: PaymentID, PaymentAmount, PaymentType with data type of integer, double and String respectively.

Methods: makePayment() for the payment purposes.

v. Ask Recommendation

Attributes: RecommendationID, Location, SoilCondition with data type of integer, String and String respectively.

Methods: askRecommendation() for the recommendation purposes.

vi. Forum

Attributes: ForumID, ForumType, Description with data type of integer, String and String respectively.

Methods: postMessage() and comment() for posting the messages and comments respectively.

vii. Join Program

Attributes: ProgramID, ProgramType, ProgramFee with data type of integer, String and double respectively.

Methods: joinProgram() for joining the course purposes.

viii. Certification Exam

Attributes: ExamID, ExamName, IsPassed with data type of integer, String and Boolean respectively.

Methods: takeExam() for exam purposes.

ix. Report Preparation

Attributes: ReportID, ReportType with data type of integer and String respectively.

Methods: prepareReport() for making report purposes.

x. Plant

Attributes: PlantID, PlantName and PlantType with data type of integer, String and String respectively.

Methods: purchasePlant() for buying plant purposes.

11. Further Development

The project's next stages might be organized as follows, taking into consideration the Scrum methodology that was used for the creation of the online system for the McGregor Institute of Botanical Training:

11.1 Software Architecture

A system's software architecture is a representation of the design choices made for the general behavior and structure of the system. A system's architecture aids in the understanding and analysis of key features like security, availability, and modifiability by stakeholders (Software Engineering Institute, 2024).

For the software architecture, it is a good idea to use Microservices Architecture. The scalability needed to manage the many online system tasks, like user registration, user enrollment, plant purchases, and forum interactions, will be supported by this architecture. Because each microservice would manage a distinct aspect of the system's functionality, scalability and maintenance will be made simpler.

11.2 Design Pattern

In software design, design patterns are standard responses to often occurring issues. They function similarly to already prepared blueprints that you can modify to address persistent design issues in the code (Refactoring Guru, 2024).

11.2.1 Model-View-Controller (MVC)

The design pattern known as Model-View-Controller (MVC) will be applied to divide the user interface, business logic, and data access to manage the system's complexity. Each component can be developed, tested, and maintained independently thanks to this division, which is consistent with Scrum's incremental and iterative design.

11.2.2 Repository Pattern

A more structured approach to manage data access layers will be implemented using this pattern. It offers a layer of substitution between the data mapping and business logic levels by accessing domain objects using an interface akin to a collection. This is how it functions within the system:

Isolation: By isolating the data access logic, it facilitates system testing and maintenance.

Centralization of Data: Consistent data access rules are made possible by centralizing the logic behind data or web service access inside the application.

11.2.3 Singleton Pattern

With the help of this pattern, a class is guaranteed to have a single instance and a worldwide point of access. The Singleton pattern could be applied to the McGregor Institute's system in the following ways:

Configuration Files: Making sure the program has a single instance of configuration settings that are accessible from anywhere in the world.

Database Connection: Using a single instance to manage database connections saves overhead from repeatedly opening and closing connections.

11.2.4 Strategy Pattern

By using this design, a set of algorithms can be defined, each encapsulated, and made interchangeable. Strategy allows the algorithm to change without depending on the users using it. Within the framework of the McGregor Institute:

Payment Processing: Depending on the kind of payment method the user chooses, several payment procedures may be used.

Recommendation Algorithm: Diverse approaches that base recommendations for plants or classes on user preferences or historical usage.

11.3 Programming Paradigm

A programming paradigm is an essential method or approach to programming that offers a collection of ideas, concepts, and methods for creating and executing computer programs. It outlines the methods for expressing computations and solving problems, as well as the structure, organization, and flow of the code (Liyan, 2023).

It is a good idea to use Object-Oriented-Programming (OOP) as the programming paradigm. Because OOP can encapsulate behaviors and data into objects, it is especially well-suited for this kind of work. This fits in nicely with the microservices architecture and the Model-View-Controller (MVC) design pattern.

11.4 Development Plan

In this phase, the development will make use of Spring Boot for the Backend to effectively manage microservices and Angular for the frontend to take advantage of its MVC

capabilities for flexible interfaces. These services will be containerized using Docker, which will provide consistency between development, testing, and operational environments.

For the programming platform, Typescript with Angular for the frontend and Java for the backend services will be used in the development of the system.

The priority arrangement of features from high to low are:

i. Feature for User Interaction.
ii. Payment Processing System.
iii. Browsing of Plant.
iv. Enrollment or Purchase.
v. User Registration and Authentication.
vi. Admin Tools for Managing.

11.5 Testing

The testing method used in the development plan is White Box Testing. This was chosen to make sure the internal operations of the program comply with functional requirements and design criteria. This approach, which is included early in the Scrum development cycles, uses techniques such as statement coverage, branch coverage, condition coverage, and path coverage. These techniques aid in the early detection and correction of problems, code optimization, and security enhancement. Automated testing is made easier by tools like Junit or NUnit. These tools can be linked into the CI/CD pipeline to ensure ongoing software health assessments. This meticulous testing methodology, together with frequent code reviews and developer and tester interactions, guarantees that the system is secure, effective, and functionally sound, offering a dependable platform for the McGregor Institute educational and community-building activities.

11.5.1 **Test 1**: **User Registration**

Objective:	To test whether the user registration process works correctly or not.
Action:	i. Go to the user registration page. ii. Valid user credentials were provided by the user. iii. "Submit" button was clicked.
Expected Result:	The system should have checked the data and then create a new user. Then, the user would have got redirected to welcome page.
Actual Result:	The system checked the data and then created a new user. Then, the user got redirected to welcome page.
Conclusion:	The test was successful.

Table 16: Test 1 of User Registration.

11.5.2 **Test 2**: **Payment Processing**

Objective:	To test whether the payment processing system accurately handles the transactions or not.
Action:	i. A payment method was selected.

	ii. Valid payment credentials were provided by the user. iii. "Submit" button was clicked.
Expected Result:	The system should have checked the payment details and then the transaction proceeds without error. Then, the user would have received confirmation message regarding the transaction.
Actual Result:	The system checked the payment details and then the transaction proceeded without error. Then, the user received confirmation message regarding the transaction.
Conclusion:	The test was successful.

Table 17: Test 2 of Payment Processing.

11.6 Deployment

An automated testing and deployment pipeline (CI/CD) is part of the deployment strategy for the online system at the McGregor Institute, which guarantees consistent, dependable updates. The solution will be fully deployed on a flexible cloud platform such as AWS or Azure after first launching in a staged beat to gather user feedback. Mechanism for rollbacks and ongoing monitoring will be implemented to promptly resolve any problems and preserve system stability. This strategy guarantees a deployment that is scalable, dependable, and suited to the requirements of the institute.

12. Prototype

An early model, sample, or release of a product made to test, and idea or procedure is called a prototype. Prototypes are typically used to assess new designs to analyst and system user accuracy (Ramirez, 2018).

For User

I. The following UI appears when the system is first opened.

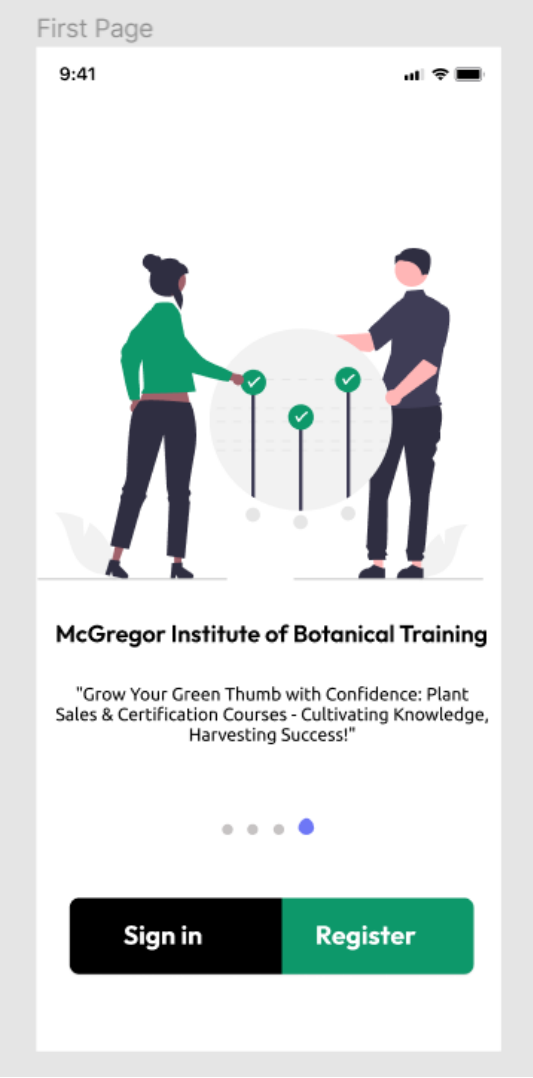

Figure 19: Prototype 1 showing the UI when the system is first opened.

It shows a welcome page with Sign in and Register button for the user to select.

II. The below prototype shows the login page for the user to enter their login credentials to enter the system if the user is already registered.

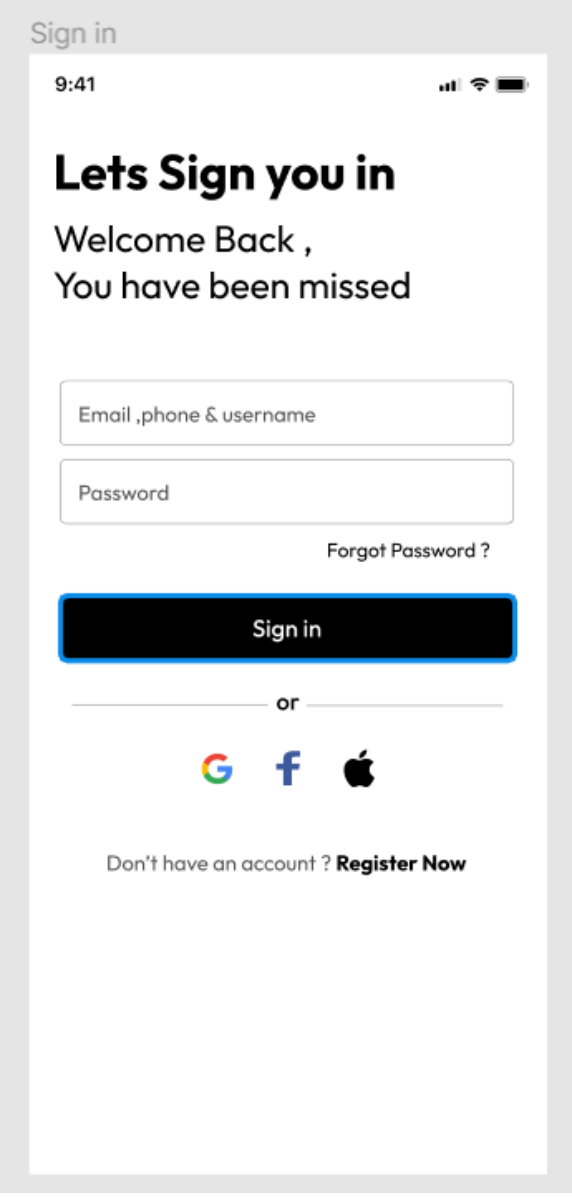

Figure 20: Prototype 2 showing the Login Page.

The user enters the information and gets access to the system.

III. The below prototype shows the register page for the user to enter their personal information to get registered into the system.

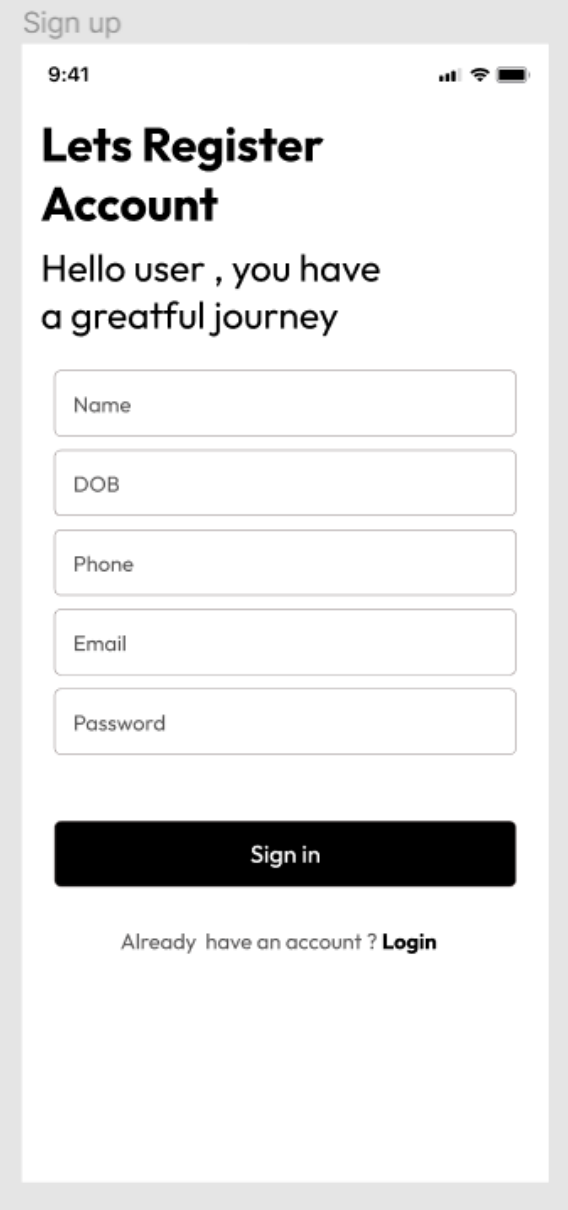

Figure 21: Prototype 3 showing the Register Page.

The user enters their personal information to register.

IV. The below prototype shows the homepage of the system.

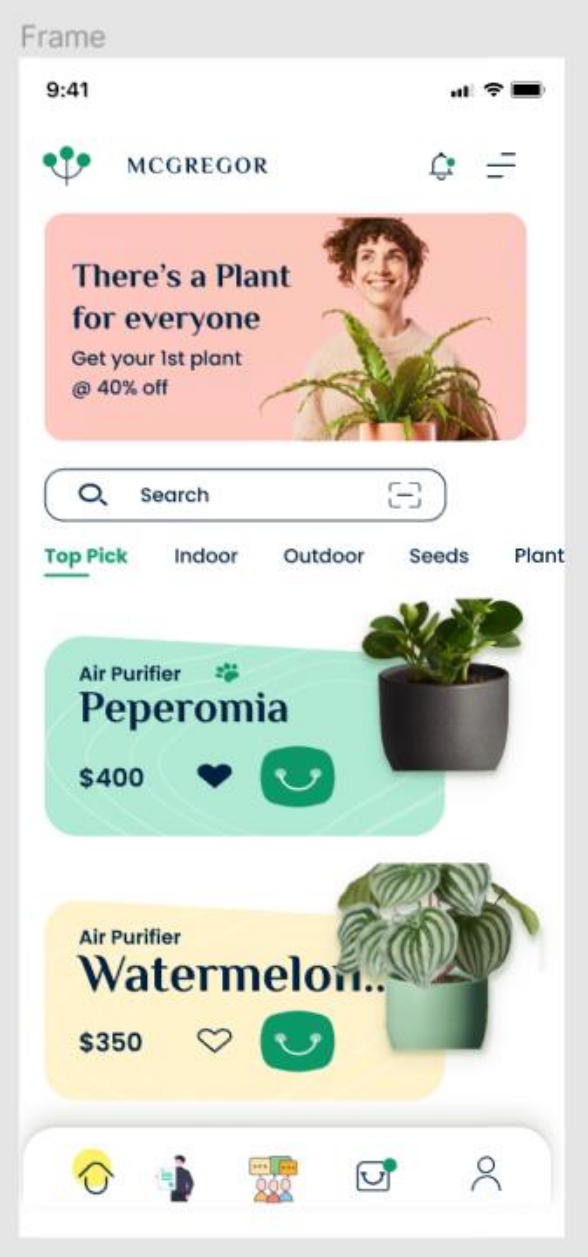

Figure 22: Prototype 4 showing the Home Page.

The user gets this homepage after they have successfully logged in to the system.

V. The below prototype shows the product page of the syste.

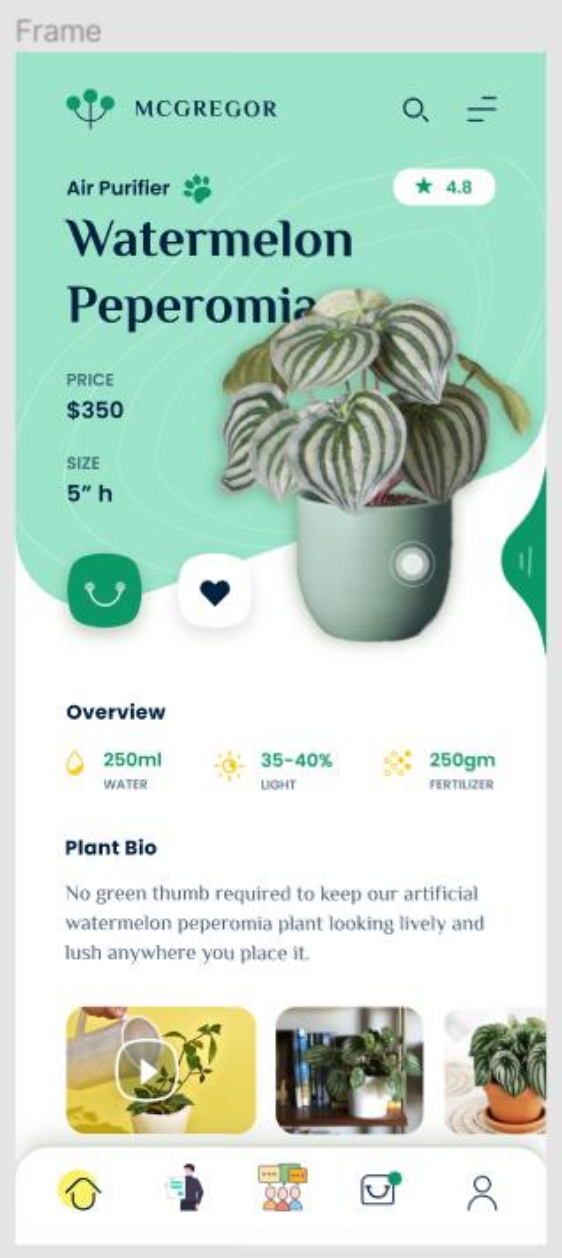

Figure 23: Prototype 5 showing the Product Details Page.

After the user have selected a plant to buy, this product detail page opens.

VI. The below prototype shows the cart page.

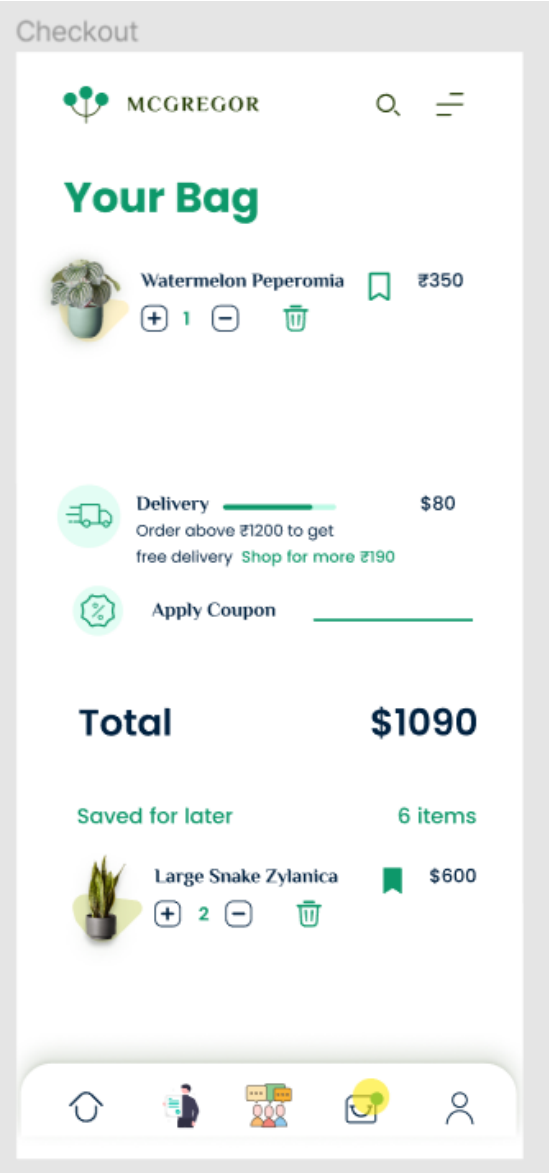

Figure 24: Prototype 6 showing the Cart Page.

After the user selects a plant to buy, the plant gets stored in the cart page for further process.

VII. The below prototype shows the payment process of the system.

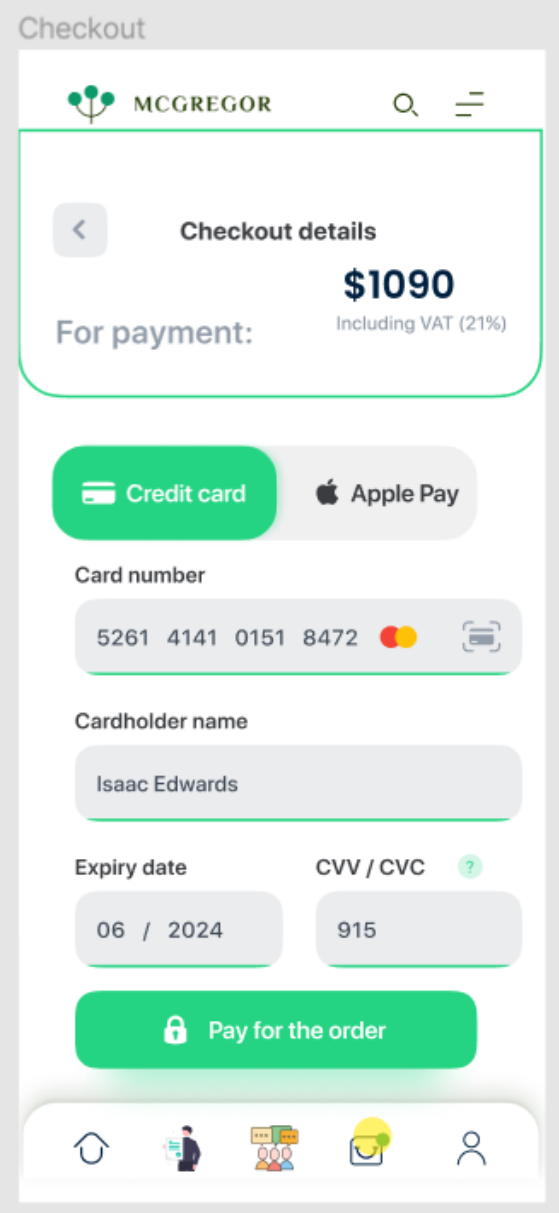

Figure 25: Prototype 7 showing the Payment Process Page.

After the user goes for further process, this payment option appears for them to enter their payment credential to buy the plant.

VIII. The below prototype shows the confirmation message.

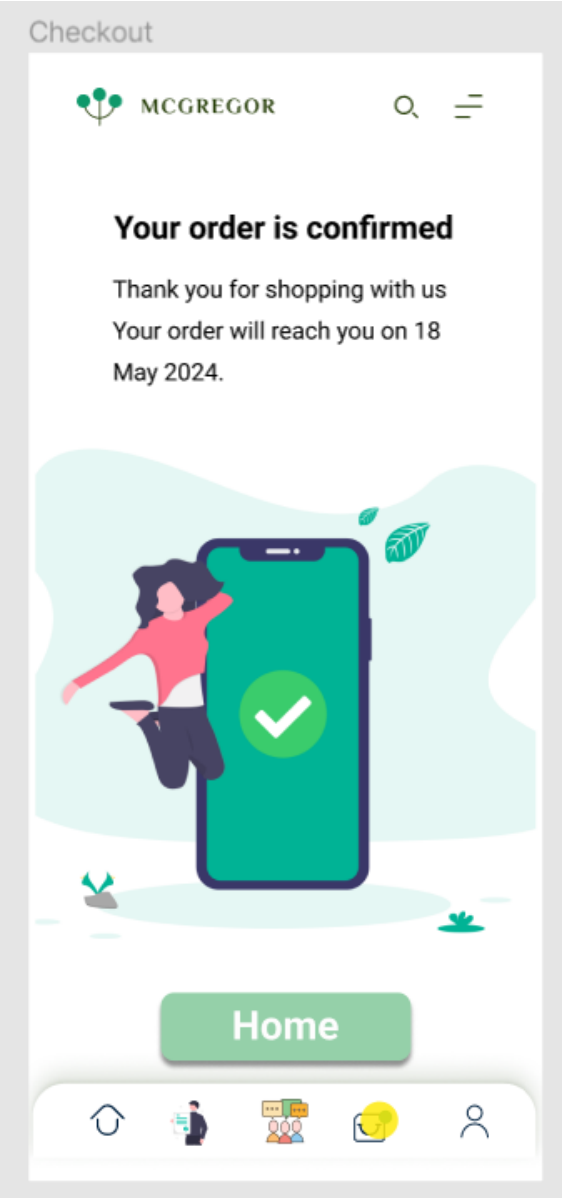

Figure 26: Prototype 8 showing the Product Purchased Confirmation Page.

After the payment gets accepted, the system shows the confirmation message to the user.

IX. The below prototype shows the forum for the user and expert.

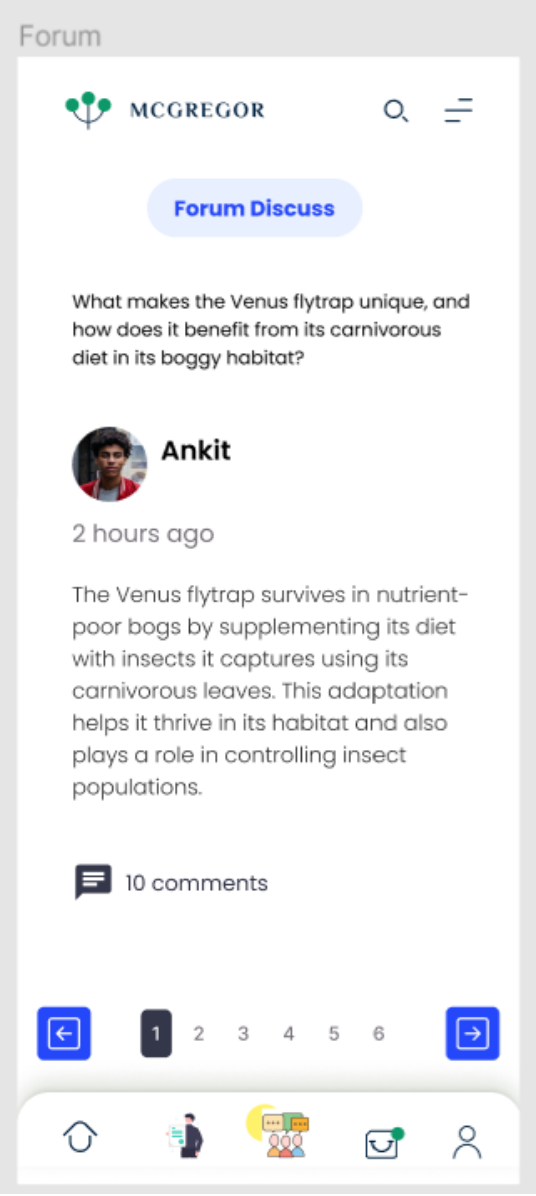

Figure 27: Prototype 9 showing the Forum Page.

If the user wants to ask something about the plant to the plant expert, they can post about them in the forum to get their answers from the expert.

X. The below prototype shows the different courses available for the user to enroll.

Figure 28: Prototype 10 showing the Courses Page.

User can select the course they want to enroll in.

XI. The below prototype shows the user profile page.

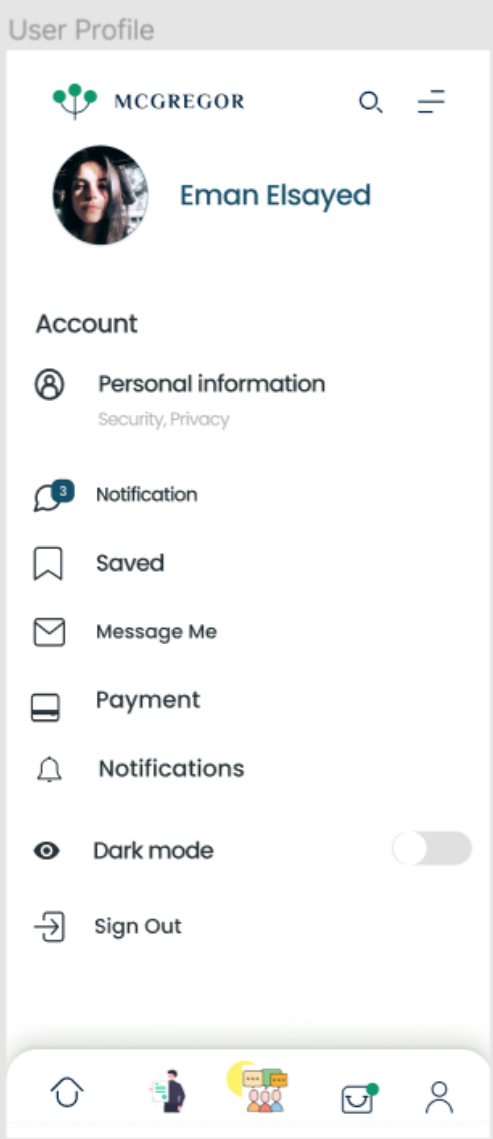

Figure 29: Prototype 11 showing the User Profile Page.

If the user wants to update their personal information or security information or if they want to log out from the system, they can do that from the user profile page.

XII. The below prototype shows the certification award for the user.

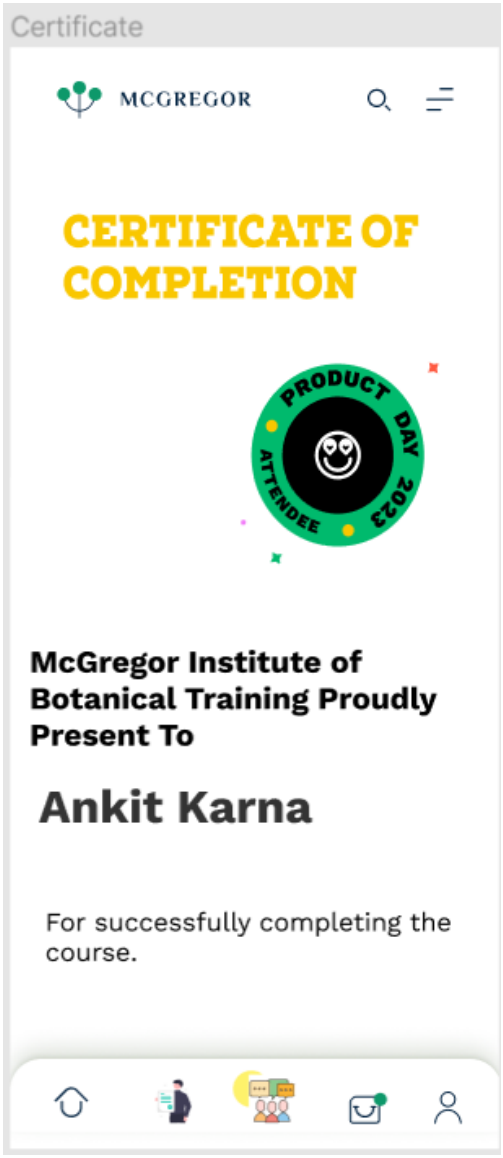

Figure 30: Prototype 12 showing the Certificate Obtaining Page.

After the user completes their certification courses, they are provided with the certificate.

For Admin

XIII. The below prototype shows the financial report of the system.

Figure 31: Prototype 13 showing the Financial Report Page.

If the admin wants to see the financial report of the system, they can see it in this page.

XIV. The below prototype shows the user report of the system.

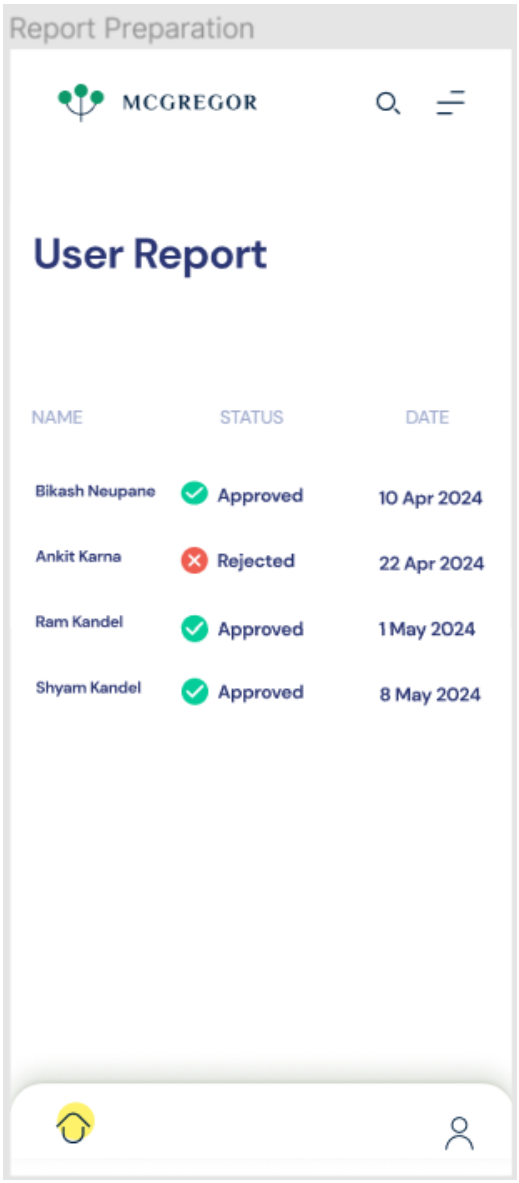

Figure 32: Prototype 14 showing the User Report Page.

The admin can see the report of all the users in this page.

XV. The below prototype shows the employee report of the system.

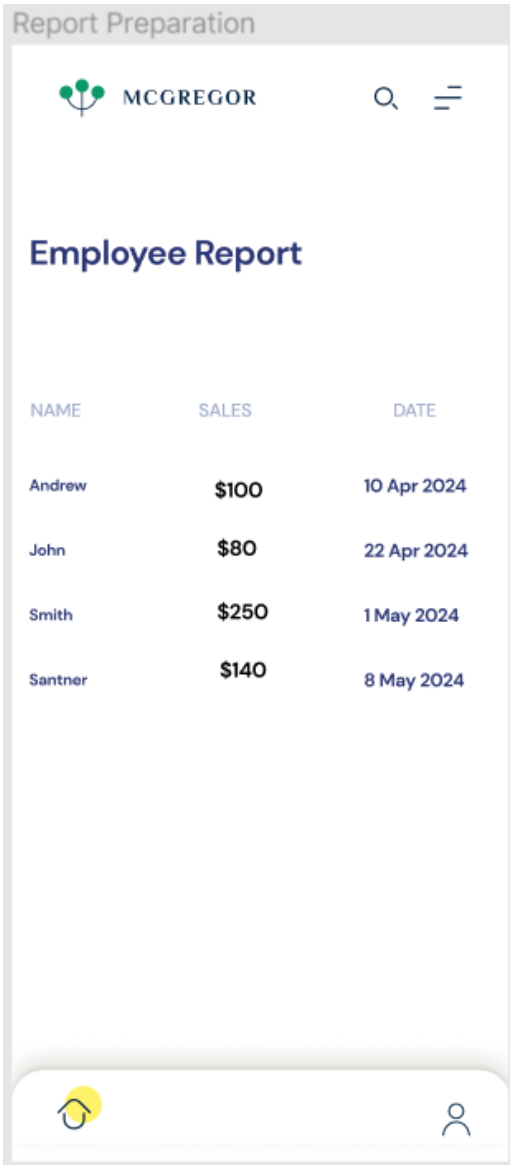

Figure 33: Prototype 15 showing the Employee Report Page.

The admin can see the report of all the employees in this page.

13. Conclusion

In conclusion, Object-Oriented Analysis and Design was used to carefully plan and carry out the construction of the online system for McGregor Institute of Botanical Training. With the help of an intuitive online platform, this method has successfully transformed complicated criteria into a thorough design that improves the institute's educational options and community involvement. With a strong development methodology and thorough planning stages guaranteeing a disciplined approach to execution and possible improvements, the project is well-positioned for future expansion. The McGregor Institute's operational effectiveness and educational experience should both be greatly enhanced by this method.

14. References

Alliance Software , 2024. *What is a software development methodology?.* [Online] Available at: https://www.alliancesoftware.com.au/introduction-software-development-methodologies/#:~:text=Software%20development%20methodology%20is%20a,the%20form%20of%20defined%20phases.
[Accessed 25 April 2024].

Association for Project Management, 2024. *What is a Gantt chart?.* [Online] Available at: https://www.apm.org.uk/resources/find-a-resource/gantt-chart/#:~:text=A%20Gantt%20chart%20is%20defined,to%20form%20a%20bar%20chart.
[Accessed 6 April 2024].

Cinergix, 2022. *Association Between Actor and Use Case.* [Online] Available at: https://creately.com/blog/diagrams/use-case-diagram-relationships/
[Accessed 24 April 2024].

donetonic, 2022. *What is scrum?.* [Online] Available at: https://donetonic.com/what-is-scrum/
[Accessed 25 April 2024].

Edrawsoft, 2024. *UML Use Case Diagram Symbols.* [Online] Available at: https://www.edrawsoft.com/uml-use-case-symbols.html
[Accessed 8 4 2024].

empirica, 2024. *High-level use case.* [Online] Available at: https://digitalhealtheurope.eu/glossary/high-level-use-case/#:~:text=A%20high%20level%20use%20case,a%20non%2Dtechnical%20way.%20(
[Accessed 28 April 2024].

GeeksforGeeks, 2024. *What is Software Engineering?.* [Online] Available at: https://www.geeksforgeeks.org/software-engineering-introduction-to-

software-engineering/
[Accessed 6 April 2024].

Guru99, 2024. *Sequence diagram example.* [Online] Available at: https://www.guru99.com/interaction-collaboration-sequence-diagrams-examples.html
[Accessed 28 April 2024].

Guru99, 2024. *What is a Sequence Diagram?.* [Online] Available at: https://www.guru99.com/interaction-collaboration-sequence-diagrams-examples.html
[Accessed 28 April 2024].

indeed, 2022. *Roles in a scrum team.* [Online] Available at: https://www.indeed.com/career-advice/career-development/scrum-team
[Accessed 26 April 2024].

Liyan, A., 2023. *What is a programming paradigm?.* [Online] Available at: https://medium.com/@Ariobarxan/what-is-a-programming-paradigm-ec6c5879952b
[Accessed 7 May 2024].

McNeish, K., 2022. UML Class Diagrams. *CODE Magazine,* Issue 11.

ProjectManager, 2024. *What Is a Work Breakdown Structure (WBS)?.* [Online] Available at: https://www.projectmanager.com/guides/work-breakdown-structure
[Accessed 7 May 2024].

Ramirez, V., 2018. *What is a Prototype?.* [Online] Available at: https://medium.com/nyc-design/what-is-a-prototype-924ff9400cfd
[Accessed 6 May 2024].

Refactoring Guru, 2024. *What's a design pattern?.* [Online] Available at: https://refactoring.guru/design-patterns/what-is-pattern
[Accessed 7 May 2024].

Software Development, 2024. *What is scrum?.* [Online] Available at: https://www.atlassian.com/agile/scrum [Accessed 25 April 2024].

Software Engineering Institute, 2024. *Software Architecture.* [Online] Available at: https://www.sei.cmu.edu/our-work/software-architecture/ [Accessed 7 May 2024].

Visual Paradigm, 2022. *The Four Types of Relationship in Use Case Diagram.* [Online] Available at: https://blog.visual-paradigm.com/the-four-types-of-relationship-in-use-case-diagram/
[Accessed 25 April 2024].

Visual Paradigm, 2024. *Extend Use Case Example.* [Online] Available at: https://online.visual-paradigm.com/diagrams/templates/use-case-diagram/extend-use-case-example/
[Accessed 28 April 2024].

Visual Paradigm, 2024. *What is Use Case Diagram?.* [Online] Available at: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/
[Accessed 8 4 2024].

Wondershare EdrawMax, 2024. *What Is A Collaboration Diagram in UML?.* [Online] Available at: https://www.edrawmax.com/article/collaboration-diagram-uml.html [Accessed 28 April 2024].