S5002NI Software Engineering

McGregor Institute

20% Group Coursework

AY 2023-2024

Group Name:			
S.N.	**Student Name (Section name: AI3)**	**College ID**	**University ID**
1	**Sushant Hona**	**NP01AI4S230001**	**22085776**
2	**Asmita Basnet**	**NP01AI4S230019**	**22085764**
3	**Bineet Ratna Shakya**	**NP01AI4S230008**	**22085767**
4	**Dipawoli Malla**	**NP01AI4S230016**	**22085768**
5	**Tsewang Norbu Gurung**	**NP01AI4S230018**	**22085778**

Table of Contents

Table of Figures

1. Introduction

In the landscapes of Nepal, where diversity in flora and fauna thrive, McGregor Institute of Botanical Training has served as a beacon of knowledge. Having celebrated a staggering seven years of nurturing passion for agriculture and horticulture, the institute now finds itself at a point where innovation is non-negotiable. An increased rate of interest in these domains requires McGregor Institute to shift from conventional methodologies of handling it's services to a more dynamic and innovative approach.

With the increased demands for botanical education, the Institute envisions a goal of not just meeting the increasing demand but also building a community which is deeply connected and rooted in the love for plants. Going further than just the traditional confines of academia, the institute aims to create a multifaceted platform- one that integrates education along with plant transactions and community interactions. The platform will not be a mere transactional space but be a canvas for cultivating a community of enthusiasts in the heart of the Himalayan Country.

McGregor Institute embarks on a new transformative journey among this ever-evolving landscape- a journey to develop a software system that will serve the institute dynamically over a long period of time. This system shall streamline the enrolment, plant purchases, and create a forum for discussions, idea sharing as well as expert interactions. It is not a mere software project but rather a vision to lay the foundation for the future of this industry itself.

This project, led by the team consisting of Sushant Hona, Tsewang Norbu Gurung, Dipawoli Malla, Asmita Basnet and Bineet Ratna Shakya, unfolds itself with the goal of paving the way for a flourishing community in Nepal. This document holds the records detailing the intricacies of this project, from it's first inception, the envisioned milestones, capturing the sheer essence of the Institute's pursuit of knowledge, community, and sustainability.

2. Project Charter

2.1. Problem Statement

The McGregor Institute, renowned and respected botanical institute passes a 7-year milestone of operating in the country of Nepal and is currently experiencing a surge of interest in agriculture as well as horticulture. This has rendered the existing manual processes incapable of keeping pace with and meeting growing demands of offering undergraduate and postgraduate courses. The growing demand necessitates the introduction of short-term certification courses in horticulture especially.

2.2. Business Case

As a response to the growing interest in agriculture in the country, McGregor Institute plans to not only meet the demand for the aforementioned short-term certification but also create a flourishing community centered around the love for the plants. Beyond just academic and educational offerings, the institution envisions a possibility of selling various plant varieties grown by their own staff, students and even professionals at minimal fees or even free for certain special events. This initiative goes beyond just transactions; it’s about creating a community of plant enthusiasts in the country. In order to facilitate this goal, a multifaceted platform is mandatory – one that can seamlessly integrate course enrolment, plant sales as well as a community forum for discussions, sharing of ideas and expert interactions.

2.3. Goal Statement

Our goal is to develop a seamless software system capable of catering to the evolving needs of McGregor Institute. This system will streamline the process of enrolment while also providing user-friendly interfaces for plant purchases. The establishment of a community through an engagement forum will also be facilitated. The system when implemented is expected to not just simply address the immediate challenges faced by the institute but rather set the stage for future adaptation as well as growth.

2.4. Timeline

- **January 2, 2023: Project Kick-off and Requirements Gathering**
- **February 15, 2023: System Design and Specification**
- **March 10, 2023: Individual Function Design Assignment**
- **April 30, 2023: System Implementation Commencement**
- **June 15, 2023: Testing and Quality Assurance**
- **August 1, 2023: Finalization and Submission Preparation**

- **September 15, 2023: Finalization and Submission**
- **October 1, 2023: Performance Testing and Optimization**
- **November 1, 2023: Security Compliance Check**
- **December 1, 2023: Scalability and Usability Assessment**
- **January 3, 2024: Project Conclusion and Handover**

Milestones:

- **Requirements Gathering and Analysis**
    - Engage with the stakeholders of the McGregor Institute to understand all the unique requirements.
    - Existing systems will be analysed to identify all the opportunities for improvement.
- **System Design and Specification**
    - Develop a design which aligns with McGregor Institute’s specifications and needs.
    - Specify all the technical details including user interfaces and architecture of the system.
- **Individual Function Design**
    - Functions will be assigned to team members, ensuring it aligns with the overall design of the system.
- **System Implementation**
    - Development will be begun with a focus on creating a scalable and user-friendly system.
    - Regularly assess the progress and adaptation of implementation strategies will be done based on emerging insights from said assessment.
- **Testing and Quality Assurance**
    - Thorough testing shall be conducted to ensure that all functionalities meet the standards.
    - All bugs will promptly be addressed.
- **Finalization and Submission**
    - Compilation of components into a cohesive software system will be completed.
    - All the comprehensive documentation will be submitted for final approval.

2.5. Scope

- **Inclusions**
    1. **User Registration:** A simplified process for members of the community to join the institute.
    2. **Course Enrolment:** A user-friendly enrolment system for short-term certification courses.
    3. **Plant Purchasing:** A visually neat and appealing interface for purchasing plants of diverse varieties from the institution.
    4. **Payment Processing:** A secure payment gateway for course fees and plant purchases will be integrated.
    5. **Expert Recommendations:** A feature which enables community members to seek advice on plant selection based on their location and soil conditions from experts.
    6. **Forum Engagement:** A forum for members of the community to host discussions on plant-related issues and topics.
    7. **Notification System:** A system of personalized notifications to keep users informed and updated regarding relevant topics.
    8. **Report Preparation:** A feature for the admin to generate financial reports, employees record as well as reports related to any user.
    9. **Take Certification Exams:** A feature for users to take mock tests, check results and give certification exams after fulfilling all requirements.
- **Exclusions**
    1. **None**

2.6. Team Members

1. **Sushant Hona** – Project Lead
2. **Tsewang Norbu Gurung** – Design and User Experience
3. **Bineet Ratna Shakya** – Development Lead
4. **Dipawoli Malla** – Quality Assurance
5. **Asmita Basnet** – Community Engagement and Documentation

3. Software Requirements Specification

3.1. Functional Requirements

- **User Registration:**
  - A valid email address can be used by a user to create an account.
  - The registration process should include an account verification.
- **Course Enrolment:**
  - Users should be capable of browsing courses, viewing their details, and enrolling in a chosen course.
  - The system should be able to track their enrolments and provide application confirmation.
- **Plant Purchasing:**
  - A catalogue should display the availability of plants along with their details.
  - The chosen plants can be added to a shopping cart, checked out and then be paid for securely by the user.
- **Payment Processing:**
  - Integration with a secure payment gateway such as but not limited to “Esewa”, “Khalti” and mobile banking.
  - The payment records should be stored for administrative purposes.
- **Expert Recommendations:**
  - Users should be able to submit their queries along with details regarding their location and soil conditions.
  - Experts can respond to those queries with insights and plant recommendations.
- **Forum Engagement:**
  - Members of the community can post, comment and even upvote on posts in the forum.
  - A notification system can alert related users about activity on their posts.
- **Notification System:**
  - Users will receive personalized and targeted notifications based on their activity.
  - These notifications can include but are not limited to course updates, forum interactions, expert responses, and results of taken examinations.
- **Report Preparation:**

  - Administrative access is mandatory to generate financial, employee and user-related reports and they will be provided with visual data for analysis.
- **Take Certification Exams:**
  - Users are allowed to take mock tests, view those results, and participate in certification exams after applying and matching requirements.

3.2. Non-Functional Requirements

- **Performance:**
  - The system should be perfectly capable of handling concurrent user interactions without hampering the performance.
- **Security:**
  - User data, along with personal as well as payment details should be encrypted and stored securely.
- **Scalability:**
  - The architecture of the system should allow for expansion in the future in order to accommodate a healthy and growing user base.
- **Usability:**
  - The user interface must be accessible and intuitive to cater to a diverse user base.

3.3. Design and Implementation Constraints

- **Technology Stack:**
  - The system will be developed using the MERN [MongoDB, Express.js, React, Node.js] stack.
- **Compliance:**
  - The system will adhere to GDPR (General Data Protection Regulation) regulation and best practices.

3.4. External Interfaces Required

- **User Interfaces:**
  - A web-based interface for users for easy access.
  - Admin dashboard for the admin to facilitate internal management.
- **Hardware Interfaces:**
  - The system will be compatible with standard servers and cloud hosting.
- **Software Interfaces:**

    - The system will integrate Stripe, Esewa, Khatlti and MoBanking APIs.
- **Communication Interfaces:**
    - Secure Socket Layer (SSL) will be used for transfer of data.

3.5. Other Non-Functional Requirements

- **Reliability:**
    - The system should be online 99.9% of the time with pre notified schedules for maintenance.
- **Documentation:**
    - Comprehensive documentation for users, administrators and even developers will be available for access.

4. Group Tasks

4.1. Environment Model Specification

- **Context Level Diagram**

The content level diagram provides an overview of the system while identifying external entities and high-level processes.

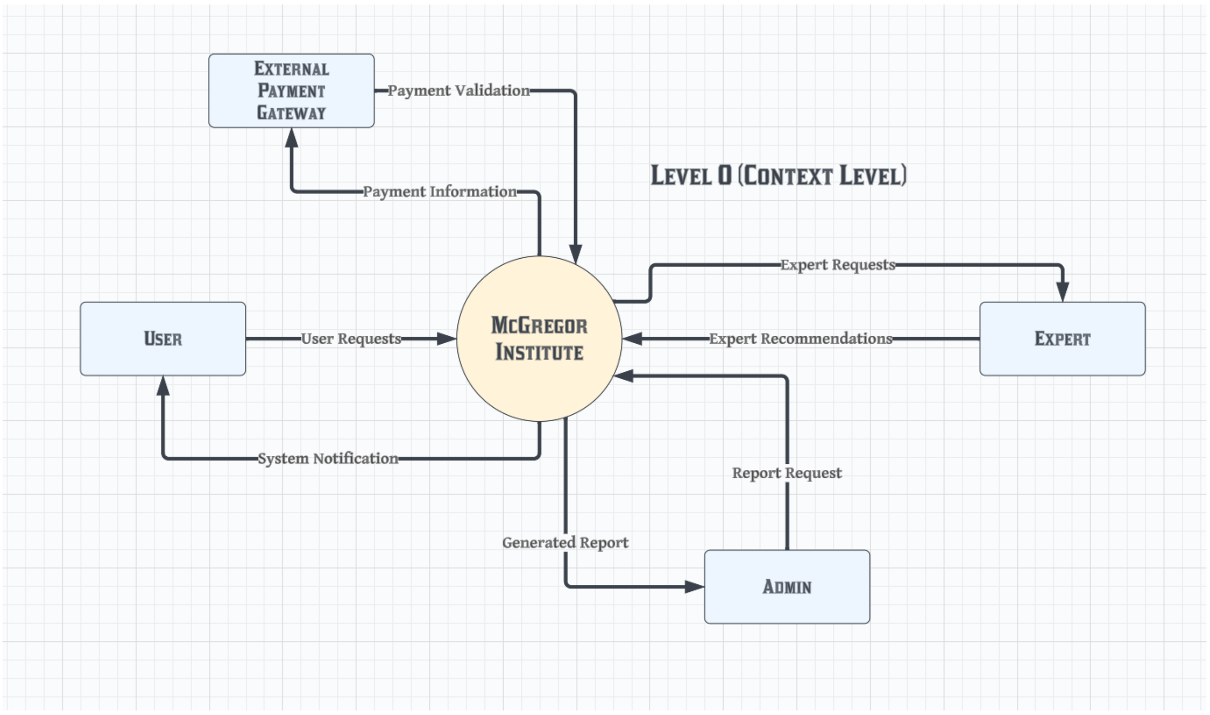

Figure 1: Context Level Diagram of the System

- **Data Flow Diagram**

The Data Flow Diagram (DFD) illustrates the flow of data within the system while showcasing processes, data stores as well as data stores.

  - **Level 1 DFD**

Level 1 DFD depicts the detailed data flow for each high-level process.

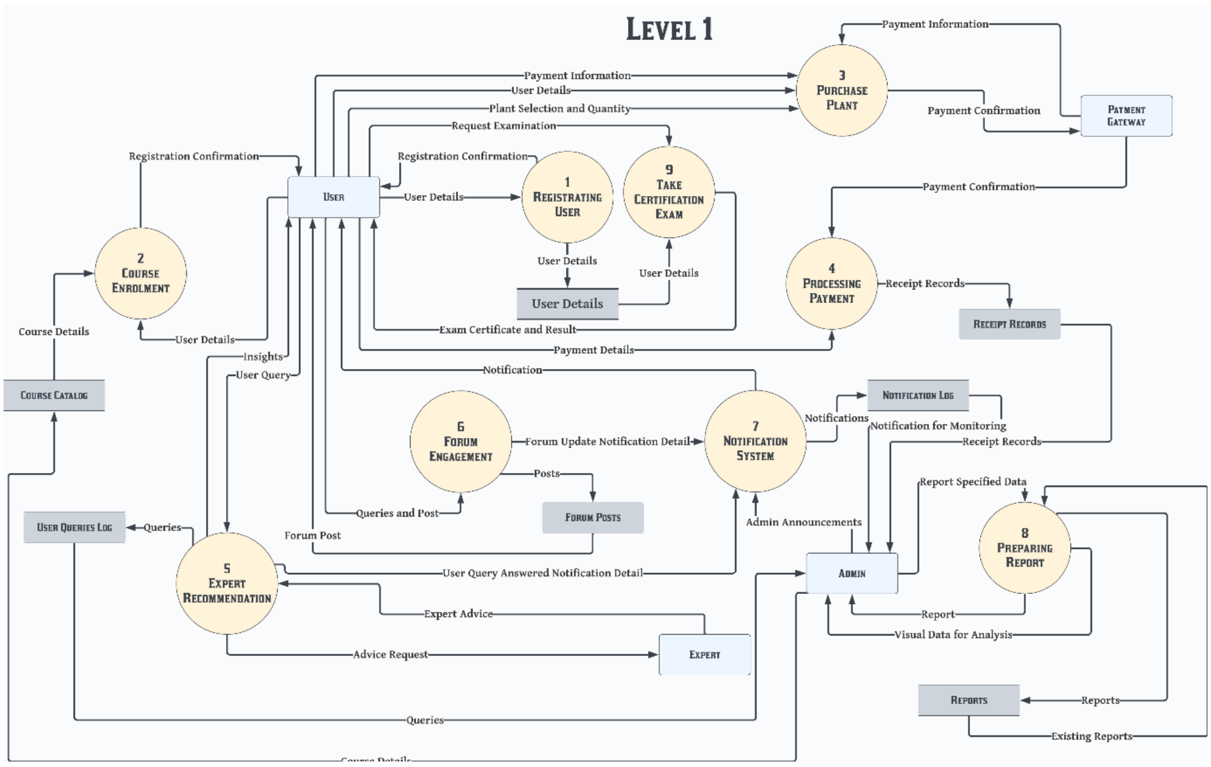

Figure 2: Level 1 DFD of the System

- **Level 2 DFD**

The Level 2 DFD breaks down the following processes into even more detail:

- Registering User
- Expert Recommendations
- Notification System

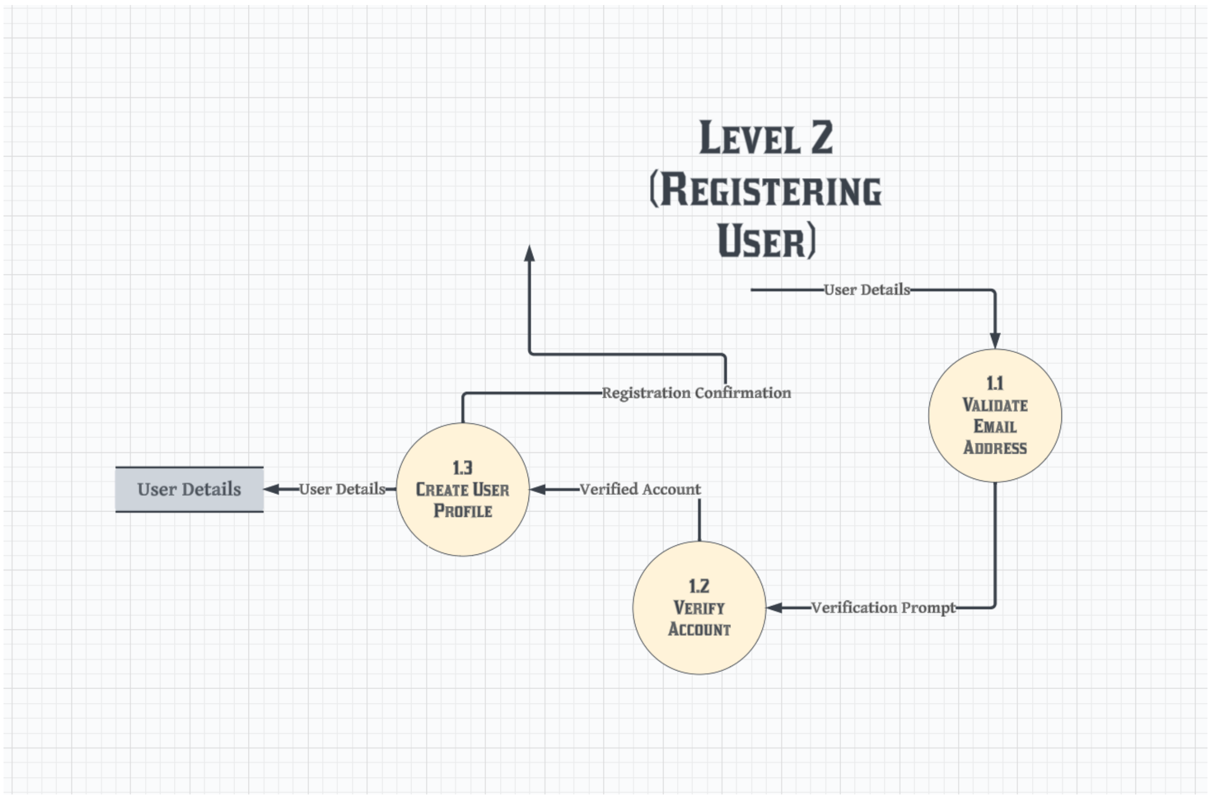

Figure 3: Level 2 Partial DFD of User Registration Feature

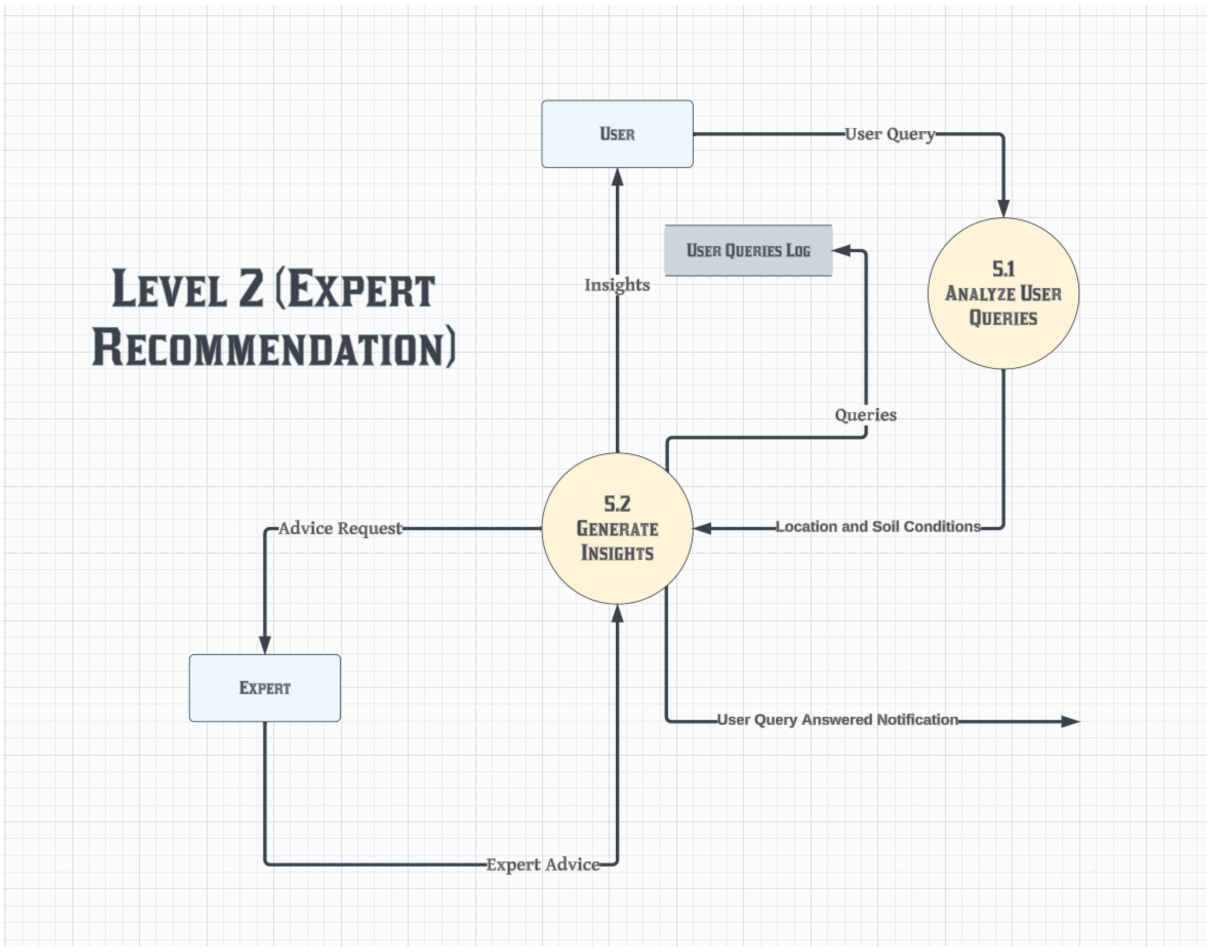

Figure 4: Level 2 Partial DFD of Expert Recommendation Feature

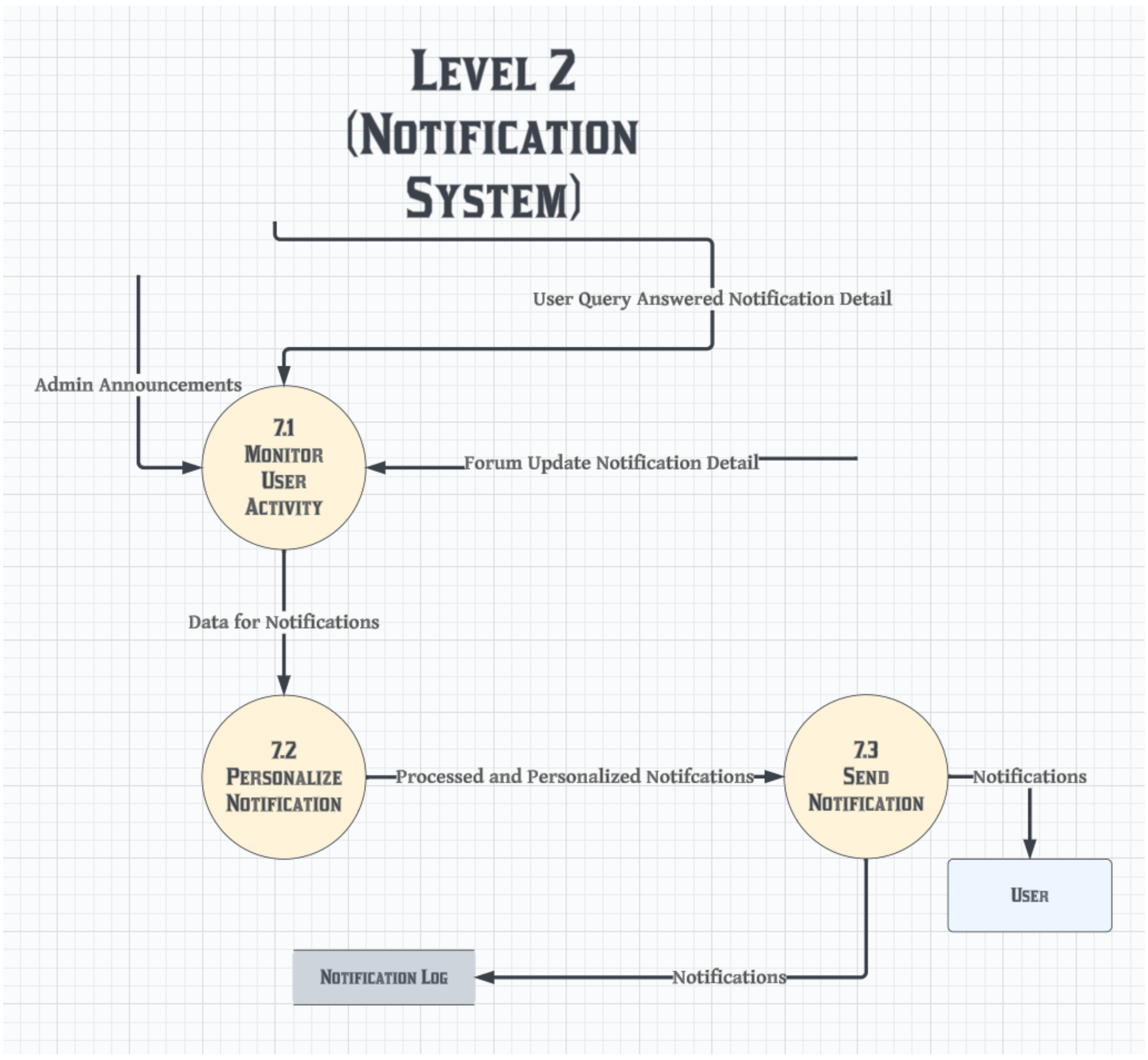
LEVEL 2 (NOTIFICATION SYSTEM)
User Query Answered Notification Detail
Admin Announcements
7.1 MONITOR USER ACTIVITY
Forum Update Notification Detail
Data for Notifications
7.2 PERSONALIZE NOTIFICATION
Processed and Personalized Notifcations
7.3 SEND NOTIFICATION
Notifications
USER
NOTIFICATION LOG
Notifications

Figure 5: Level 2 Partial DFD of Notification System Feature

4.2. Internal Model Specification for the System

- **Entity Relationship Diagram (ERD)**

The ERD illustrates the relationship among various entities in the McGregor Institute system.

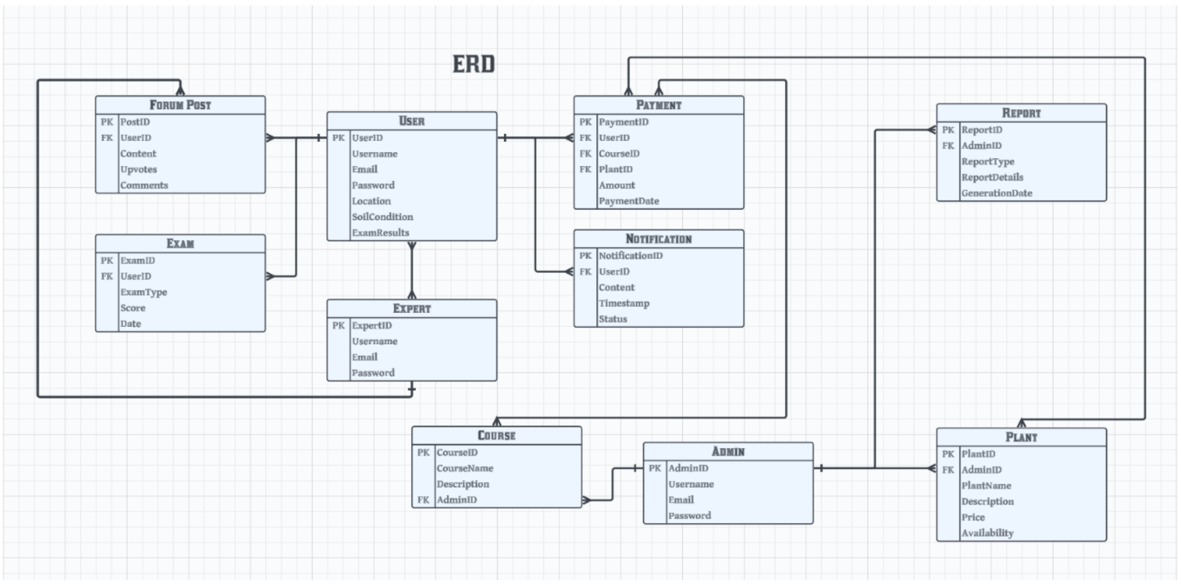

Figure 6: ERD of the System

- **Data Dictionary**

  - **Entities:**
    - **User:**
      - **Description:** Represents individuals interacting with the system.
      - **Attributes:**
        - **UserID** (integer, primary key)
        - **Username (string)**
        - **Email (string)**
        - **Phone Number (string)**
        - **Address (string)**
    - **Expert:**
      - **Description:** Represents experts providing advice to users.
      - **Attributes:**
        - **ExpertID (integer, primary key)**
        - **Username (string)**
        - **Email (string)**
        - **Password (integer)**

- **Admin:**
  - **Description:** Represents admins managing the system
  - **Attributes:**
    - **AdminID (integer, primary key)**
    - **Name (string)**
    - **Email (string)**
    - **Password (string)**
- **Payment:**
  - **Description:** Represents the external entity handling payment transactions.
  - **Attributes:**
    - **TransactionID (integer, primary key)**
    - **Amount (float)**
    - **UserID (integer, foreign key)**
    - **PlantID (integer, foreign key)**
    - **CourseID (integer, foreign key)**
    - **PaymentDate (datetime)**
- **Course:**
  - **Description:** Contains information about the courses offered.
  - **Attributes:**
    - **CourseID((integer, primary key)**
    - **CourseName (string)**
    - **Description (text)**
    - **AdminID (integer, foreign key)**
- **Notification:**
  - **Description:** Stores notifications sent to user.
  - **Attributes:**
    - **NotificationID (integer, primary key)**
    - **UserID (integer, foreign key)**
    - **Content (string)**
    - **Timestamp (datetime)**
    - **Status (string)**
- **Reports:**
  - **Description:** Stores reports requested by administrators.

- **Attributes:**
  - **ReportID (integer, primary key)**
  - **AdminID (integer, foreign key)**
  - **ReportTyle (string)**
  - **ReportDetails (text)**
  - **GenerationDate (datetime)**

- o **Forum Post**
  - **Description:** Stores posts made by users and admins on the forum.
  - **Attributes:**
    - **PostID (integer, primary key)**
    - **UserID (integer, foreign key)**
    - **Comments (string)**
    - **Content (text and images)**
    - **Upvotes (integer)**
- o **Exam**
  - **Description:** Stores the details regarding the exams.
  - **Attributes:**
    - **ExamID (integer, primary key)**
    - **UserID (integer, primary key)**
    - **ExamType (string)**
    - **Score (float)**
    - **Date (datetime)**

- **Data Flow:**
  - o **User Details:**
    - **Description:** Carries user details from the User external entity to the Registering User process.
    - **Data Type: JSON**
  - o **Registration Confirmation:**
    - **Description:** Carries the registration confirmation from the Registering User process to the User external entity.
    - **Data Type: String (Confirmation message)**
  - o **Course Details:**

- **Description:** Carries course details from the Course Enrolment process to the User external entity.
- **Data Type: JSON**

o **Course Selection:**

- **Description:** Carries course selection from the User external entity to the Course Enrolment process.
- **Data Type: JSON**

o **Payment Information:**

- **Description:** Carries payment information from the Course Enrolment and Purchase Plant processes to the Processing Payment process.
- **Data Type: JSON**

o **Plant Selection and Quantity:**

- **Description:** Carries plant selection and quantity from the User external entity to the Purchase Plant process.
- **Data Type: JSON**

o **Payment Gateway:**

- **Description:** Carries payment information from the Processing Payment process to the Payment Gateway external entity.
- **Data Type: JSON**

o **Payment Confirmation:**

- **Description:** Carries payment confirmation from the Payment Gateway external entity to the Processing Payment process.
- **Data Type: String (Confirmation message)**

o **Advice Request:**

- **Description:** Carries advice request from the User external entity to the Expert Recommendation process.
- **Data Type: JSON**

o **Expert Advice:**

- **Description:** Carries expert advice from the Expert external entity to the Expert Recommendation process and back to the User external entity.
- **Data Type: JSON**

o **User Query:**

- **Description:** Carries user query from the User external entity to the Forum Engagement process**.**
- **Data Type: JSON**

- **Forum Post:**
  - **Description:** Carries forum posts from the Forum Engagement process to the User and Admin external entities and from the Forum Posts data store to the Forum Engagement process.
  - **Data Type: JSON**
- **Notification:**
  - **Description:** Carries notification from the Notification System process to the User external entity.
  - **Data Type: JSON**
- **Report:**
  - **Description:** Carries report from the Preparing Report process to the User external entity.
  - **Data Type: JSON**
- **Exam Certificate and Result:**
  - **Description:** Carries exam certificate and result from the Take Certification Exam process to the User external entity.
  - **Data Type: JSON**
- **User Details:**
  - **Description:** Carries user details from the Registering User process to the User Details data store.
  - **Data Type: JSON**
- **Course Details:**
  - **Description:** Carries course details from the Course Enrolment process to the Course Details data store.
  - **Data Type: JSON**
- **Plant Selection and Quantity:**
  - **Description:** Carries plant selection and quantity from the Purchase Plant process to the Plant Selection and Quantity data store.
  - **Data Type: JSON**
- **Notification Log:**

- **Description:** Carries the notification log from the Notification System process to the Notification Log data store.
- **Data Type: JSON**

- **Reports:**
  - **Description:** Carries reports from the Reports data store to the Preparing Report process.
  - **Data Type: JSON**
- **Receipt Records:**
  - **Description:** Carries receipt records from the Course Enrolment and Purchase Plant processes to the Receipt Records data store.
  - **Data Type: JSON**
- **Course Catalogue:**
  - **Description:** Carries the course catalogue from the Course Catalogue data store to the Course Enrolment process.
  - **Data Type: JSON**
- **User Queries Log:**
  - **Description:** Carries the user queries log from the User Queries Log data store to the Expert Recommendation process.
  - **Data Type: JSON**
- **Manage Details:**
  - **Description:** Carries the command "Manage Details" to update information.
  - **Data Type: String (Command)**

- **Process Specification (Pspecs)**

Pspecs outline the steps and interactions for each process in the McGregor Institute’s system.

1. **Registering User (Process 1):**
   a. Description: This process handles the registration of new users into the system.
   b. Steps:
      i. Users initiate registration by providing required details.
      ii. System validates user information and sends a verification email.
      iii. Upon email confirmation, users can log in.

2. **Course Enrolment (Process 2):**
   a. Description: Facilitates users to browse, view details, and enrol in chosen courses.
   b. Steps:
      i. Users log in and browse available courses.
      ii. Users select a course, and the system updates Course Details.
      iii. Users receive enrolment confirmation.

3. **Purchase Plant (Process 3):**
   a. Description: Allows users to browse, select, and securely purchase plants.
   b. Steps:
      i. Users view the plant catalogue with details.
      ii. Selected plants are added to the shopping cart.
      iii. Users proceed to checkout, providing necessary details.
      iv. Payment processing through integrated gateways (Esewa, Khalti, MoBanking).
      v. Confirmation and update of Receipt Records.
4. **Processing Payment (Process 4):**
   a. Description: Manages the processing of payment transactions.
   b. Steps:
      i. Payment information from Course Enrolment and Purchase Plant processes is received.
      ii. Integration with payment gateways (Esewa, Khalti, MoBanking).
      iii. Payment confirmation received and stored.
5. **Expert Recommendation (Process 5):**
   a. Description: Users can seek expert advice on plant selection based on their location and soil conditions.
   b. Steps:
      i. Users submit queries with location and soil details.
      ii. Experts receive queries and provide recommendations.
      iii. Expert Advice is stored, and users receive notifications.
      iv. Users can interact further for clarifications.

6. **Forum Engagement (Process 6):**
   a. Description: Community members can engage in discussions, post queries, and upvote content in the forum.
   b. Steps:
      i. Users post queries or comments in the forum.
      ii. Forum Engagement process updates Queries and Posts data store.
      iii. Admins can post responses and upvote content.
      iv. Notifications sent for forum updates.
7. **Notification System (Process 7):**
   a. Description: The system sends personalized notifications to keep users informed about relevant activities.
   b. Steps:
      i. Notification triggers based on user activity (course updates, forum interactions, expert responses).
      ii. Notifications sent to respective users.
      iii. Notification Log is updated for future reference.
8. **Preparing Report (Process 8):**
   a. Description: Admins can generate financial reports, employee records, and user-related reports for analysis.
   b. Steps:
      i. Admin accesses the Report Preparation feature.
      ii. Selects the type of report to generate (financial, employee, user).
      iii. System compiles and generates the report.
      iv. Report is available for download and analysis.
9. **Take Certification Exam (Process 9):**
   a. Description: Users can take mock tests, view results, and participate in certification exams after fulfilling requirements.
   b. Steps:
      i. Users request to take a certification exam.
      ii. Admin validates the request.
      iii. Users take the exam.
      iv. Exam result and certificate are sent to the users.

Notes:

Ensure data flows align with processes, updating relevant data stores.

Include error-handling mechanisms for each process.

Specify data types for inputs and outputs.

4.3. Design Specification

- Structure Chart (Upper Level)

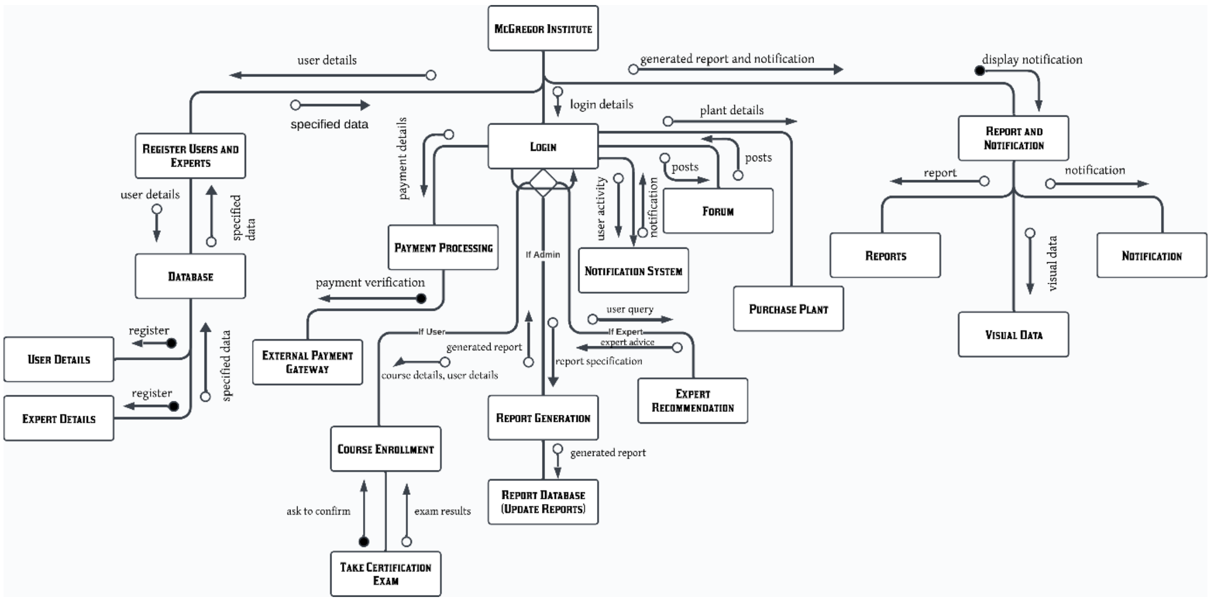

Figure 7: Structure Chart of the Overall System

4.4. Assignment Diary: Project "GreenHarbor"

In this journey, we meticulously document our assumptions, responsibilities, and collaborative strategies using this comprehensive assignment diary.

- **Document Assumptions**
  - Assumption 1: Hardware and Software Accessibility
    - Details: We assume that each member of the team possesses the necessary hardware as well as the software required for seamless development. Regular checks will still be conducted to address any dependencies.
    - Mitigation: A contingency plan has been developed which includes a shared pool and tech support channel which will help resolve any issues quickly.
  - Assumption 1: Stakeholder Feedback Timelines
    - Details: We anticipate active engagement for constructive insights.
    - Mitigation: Regular feedback sessions have been scheduled at key project milestones.
  - Assumption 3: GDPR Compliance
    - Details: Adherence to GDPR regulations is assumed to be respected and followed.
    - Mitigation: Regular compliance checks will be carried out.
- Omissions
  - Omission 1: Hardware Configurations
    - Details: The diary does not delve into the specific hardware configurations required.
    - Mitigation: A detailed hardware guideline document will be created and shared with the appropriate team members.
  - Omission 2: External Dependencies
    - Details: Potential external dependencies impacting the project are not explicitly covered.
    - Mitigation: A risk and security assessment session shall identify external dependencies.
- Group Member Responsibilities
  - Sushant Hona (Project Lead):

    - Details: Overseeing project coordination and ensuring alignment with McGregor Institute’s vision.
    - Additional: Individual feature design for “Report Generation”
  - Tsewang Norbu Gurung (Design and User Experience):
    - Details: Crafting the user interfaces for an immersive user experience.
    - Additional: Individual feature design for “Purchase Plant”
  - Bineet Ratna Shakya (Development Lead):
    - Details: Leading the development using the MERN stack.
    - Additional: Individual feature design for “Take Certification Exam”
  - Dipawoli Malla (Quality Assurance):
    - Details: Conducting tests to ensure the system functionality meets and exceeds the set standard.
    - Additional: Individual feature design for “Join the Program” feature.
  - Asmita Basnet (Community Engagement and Documentation):
    - Details: Engage with the community for feedback and managing a comprehensive document.
    - Additional: Individual feature design for “Make Payment”
- **Group Meetings**
  - Frequency: Weekly meetings, every Thursday at 3:00 PM.
  - Agenda: Progress updates, challenges discussion, and planning for the upcoming week.
  - Communication Channel: Video conferencing via Zoom for meetings supplemented by meetings held in the university, supplemented by Sack for ongoing collaboration.

As project “GreenHarbor” unfolds, this assignment diary will serve as our live document, evolving to meet the dynamic needs of the project. Regular updates ad reflections will ensure its relevance and effectiveness in steering the project towards success.

5. Individual Task

5.1. Purchase Plant

- **Context Level:**

The feature "Purchase Plant" operates within the broader system where users can interact with the McGregor Institute platform to purchase plants.

Major Inputs: User requests for plant names and quantities

Major Outputs: The purchase status which indicates the success of the transaction.

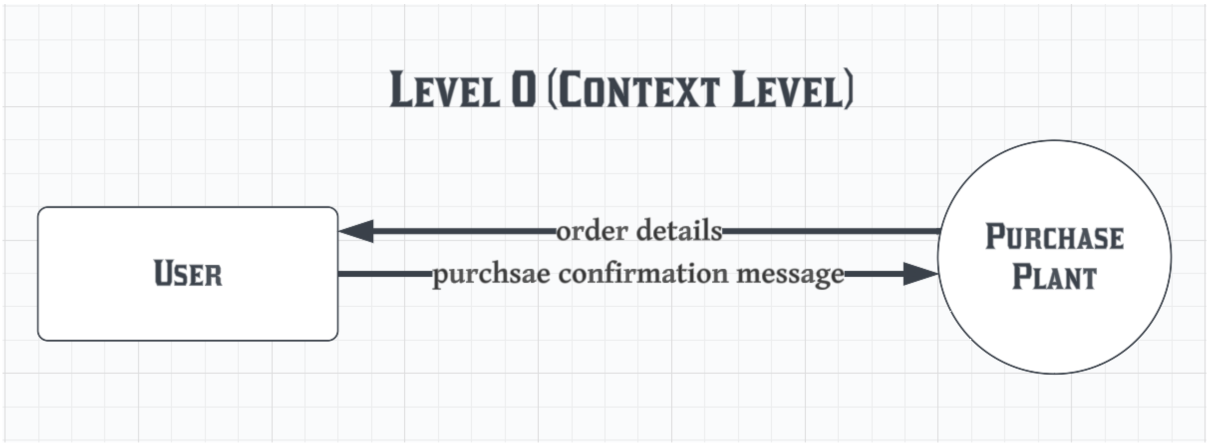

Figure 8: Level 0 DFD of Purchase Plant Feature

- **Internal Model Specification for the Feature**
  - **Level 1 DFD Fragments:**
    The level 1 DFD for this feature illustrates the high-level processes involved in the feature such as user input validation, cart management, and order confirmation.

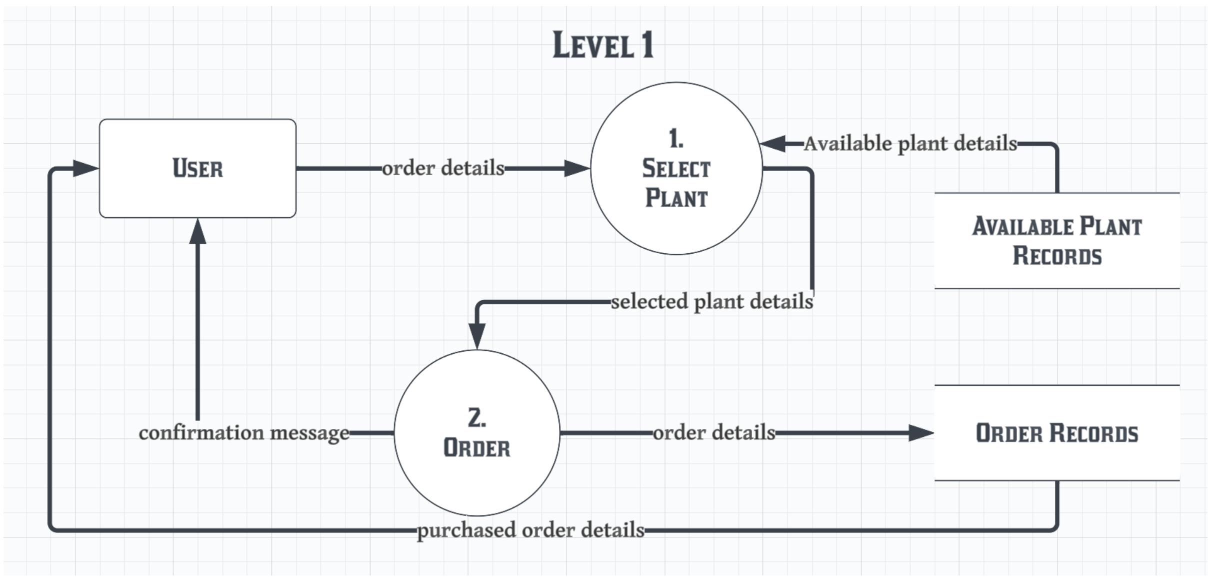

Figure 9: Level 1 DFD of Purchase Plant Feature

- **Level 2 DFD Fragments:**
  The level 2 DFD for this feature breaks down the "Order" process further, depicting the interaction between modules such as AddToCart, OrderConfirmation and OrderPlaced.

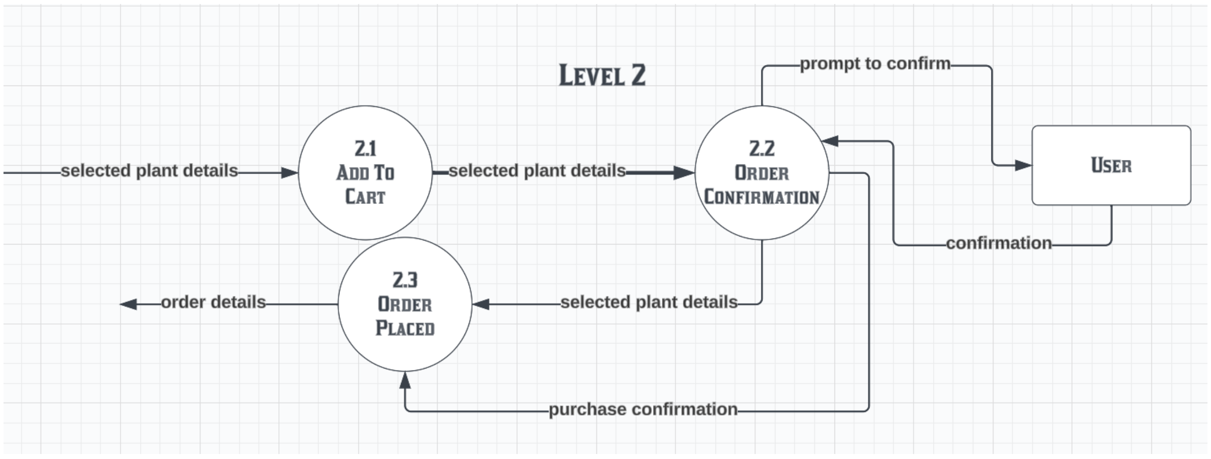

Figure 10: Level 2 DFD of Purchase Plant Feature

- **Design Specification**
  - **Structure Chart:**

    The structure chart for this function outlines the hierarchy of modules which are involved in this feature. Modules such as Purchase plant, Add to cart, and Generate receipt.

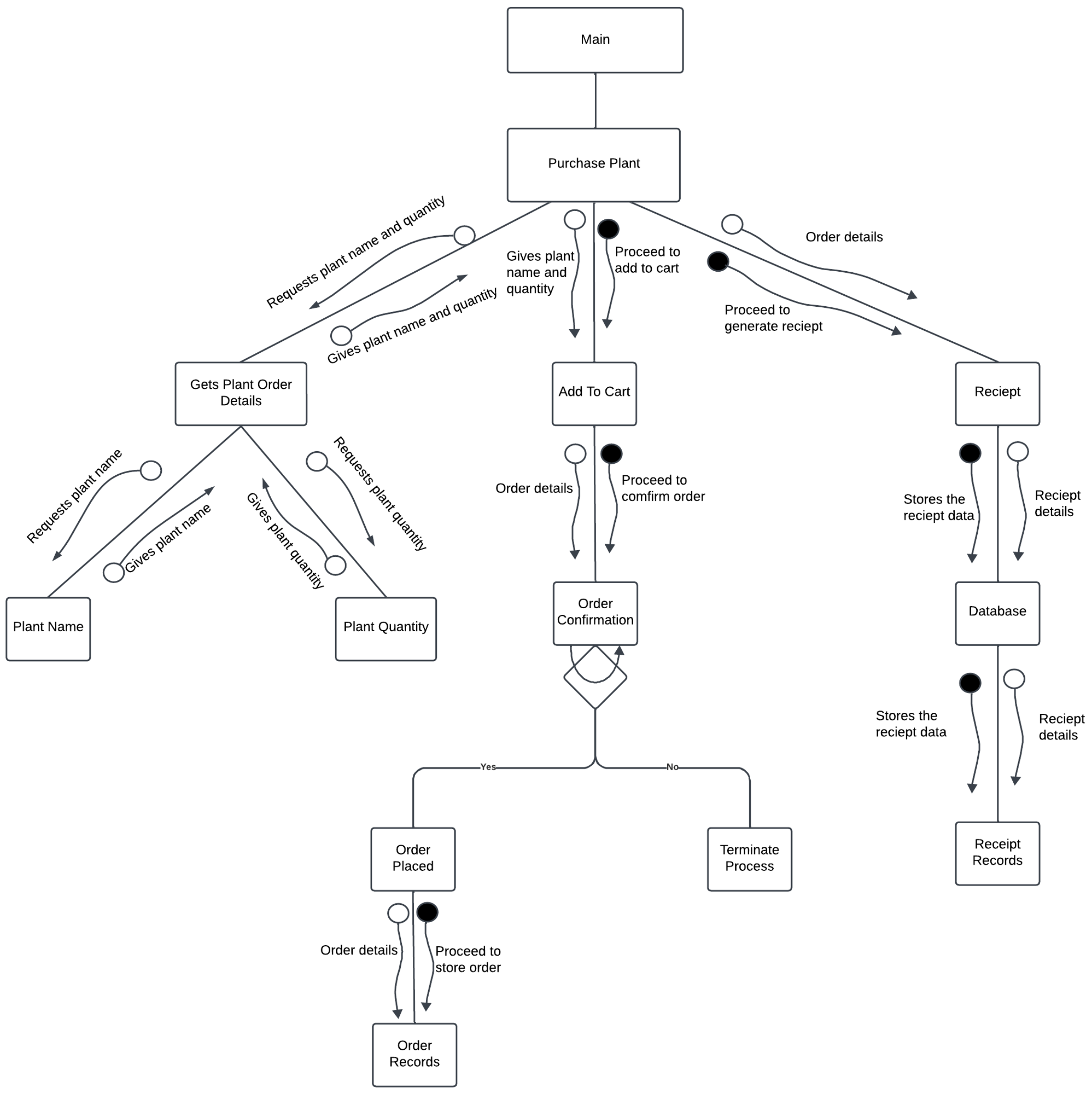

Figure 11: Structure Chart of Purchase Plant Feature

  - **Module Specifications (MSpecs):**

- **Module Name:** Purchase Plant Module
  - **Purpose:** This module facilitates the purchase of plants by the user based on the plant name and quantity.

- **Pseudocode:**

```
# Request plant name and quantity from the user
PlantName = RequestPlantName()
PlantQuantity = RequestPlantQuantity()

# Validate plant name and quantity
IF PlantName IS VALID AND PlantQuantity IS VALID:
    # Add plant to cart and display order details
    PurchaseStatus = AddToCart(PlantName, PlantQuantity)
    IF PurchaseStatus == SUCCESS:
            DisplayOrderDetails(PlantName, PlantQuantity)
            # Confirm order and generate receipt
            IF ConfirmOrder() == YES:
                PlaceOrder(PlantName, PlantQuantity)
                GenerateReceipt(PlantName, PlantQuantity)
            ELSE:
                Terminate()
            END IF
        END IF
    ELSE:
        PurchaseStatus = INVALID_INPUT

    RETURN PurchaseStatus
```

- **Input Parameter:**
    - PlantName: The name of the plant chosen.
    - PlantQuantity: The quantity of the plant chosen.
- **Output Parameter:**
    - PurchaseStatus: Indicates the status of the purchase (SUCCESS, INVALID_INPUT, etc)
- **Global Variable:**
    - OrderRecords: The database that stores the order information and history.
- **Local Variable:**
    - PlantName: Temporary variable to store the chosen plant's name.
    - PlantQuantity: Temporary variable to store the chosen plant quantity.
- **Calls:**
    - RequestPlantName(): Subroutine to request the plant name from the user.
    - RequestPlantQuantity(): Subroutine to request the plant quantity from the user.
    - AddToCard(PlantName, PlantQuantity): Subroutine to add the plant to the cart and update inventory.
    - DisplayReceipt(PlantName, PlantQuantity): Subroutine to display the order details to the user.

    - ConfirmOrder(): Subroutine to ask the user to confirm the order.
  - **Called By**: The main system control flow.
- **Feature Description:**

"Purchase Plant" enables users to interact with the McGregor Institute platform to buy plants. The feature includes processes such as user input validation, cart management, and order confirmation. Users request specific plant names and quantities, and the system validates and processes the order, providing feedback on the transaction's success.

5.2. Report Generation

- **Context Level**

The feature "Report Preparation" operates within the broader system, it allows only the administrators to interact with the McGregor Institute platform to generate and display reports.

Major Inputs:

- **SpecifiedData:** The data specified by the admin for the generation of the report.
- **ReportType:** The type of report chosen by the admin (e.g., user, financial, etc.)

Major Outputs:

- **ReportStatus:** Indicates the status of the report generated (SUCCESS, INVALID_INPUT, etc.).

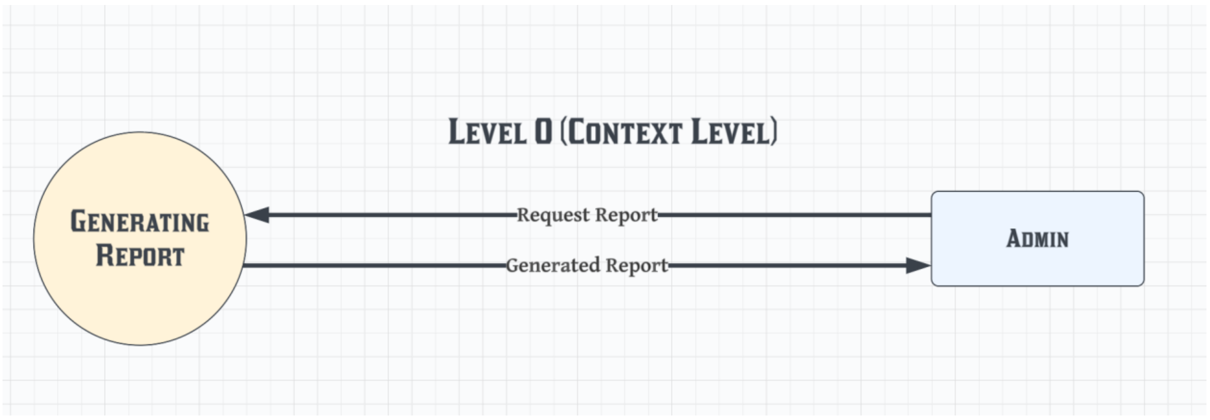

Figure 12: Level 0 DFD of Report Generation Feature

- **Internal Model Specification for the Feature**
    - **Level 1 DFD Fragments:**

      The level 1 DFD for the "Report Preparation" feature illustrates the high-level processes involved, including collecting data, generating report as well as storing and displaying said report.

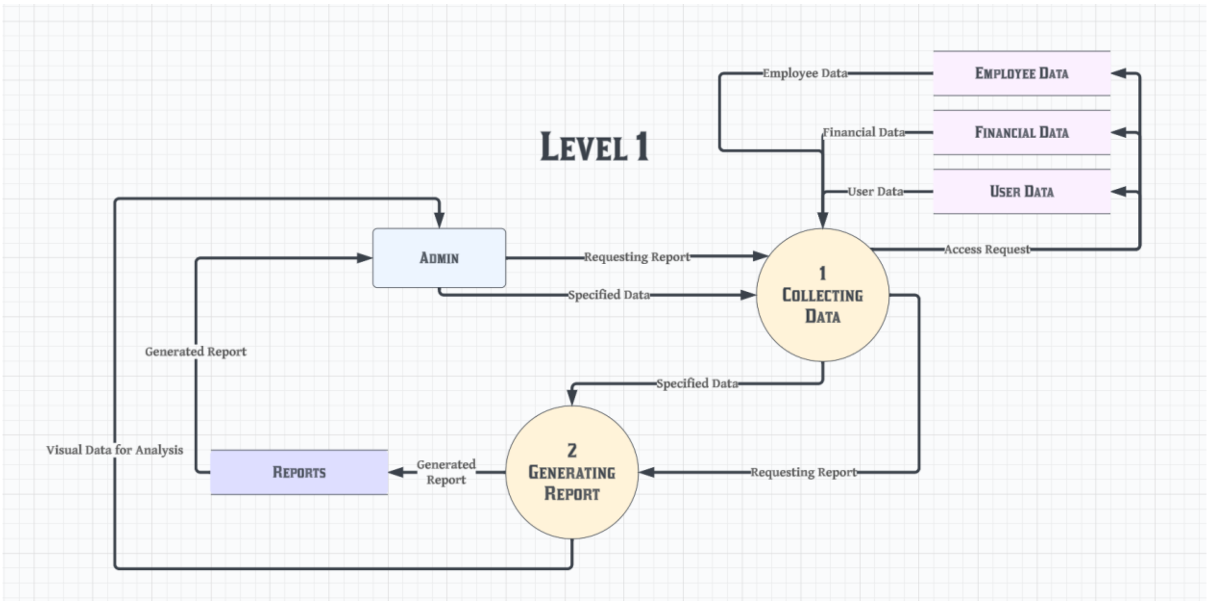

Figure 13: Level 1 DFD of Report Generation Feature

- **Level 2 DFD Fragments:**

  The level 2 DFD for this feature breaks down the “Generating Report” process, depicting the interactions between modules such as FormattingData, DisplayingData and GeneratingVisualData.

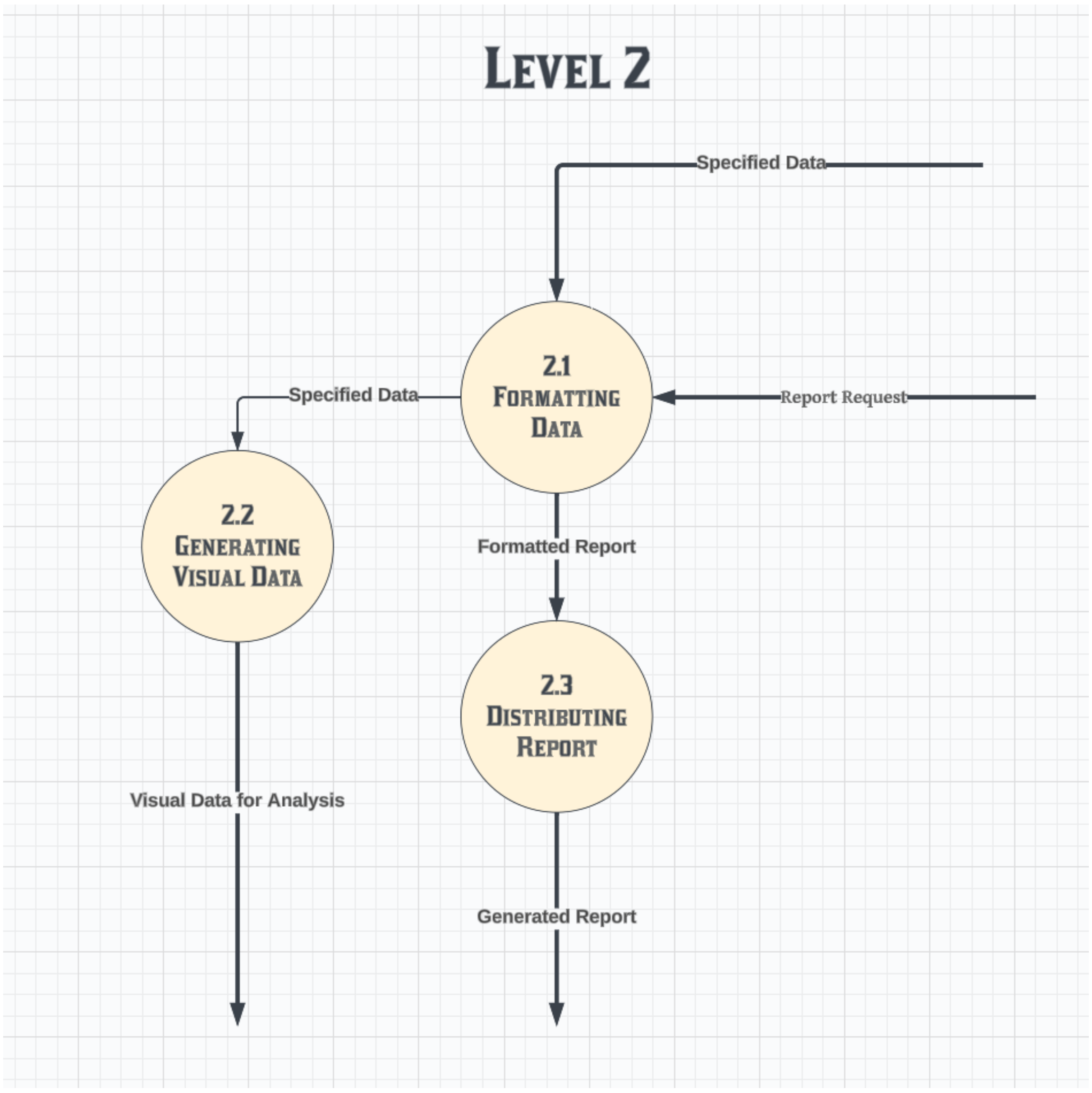

Figure 14: Level 2 DFD of Report Generation Feature

- **Design Specification**
    - **Structure Chart:**
      The structure chart for the “Report Preparation” function outlines the hierarchy of modules involved, these include ReportEngine, FormattingData and GeneratingVisualData etc.

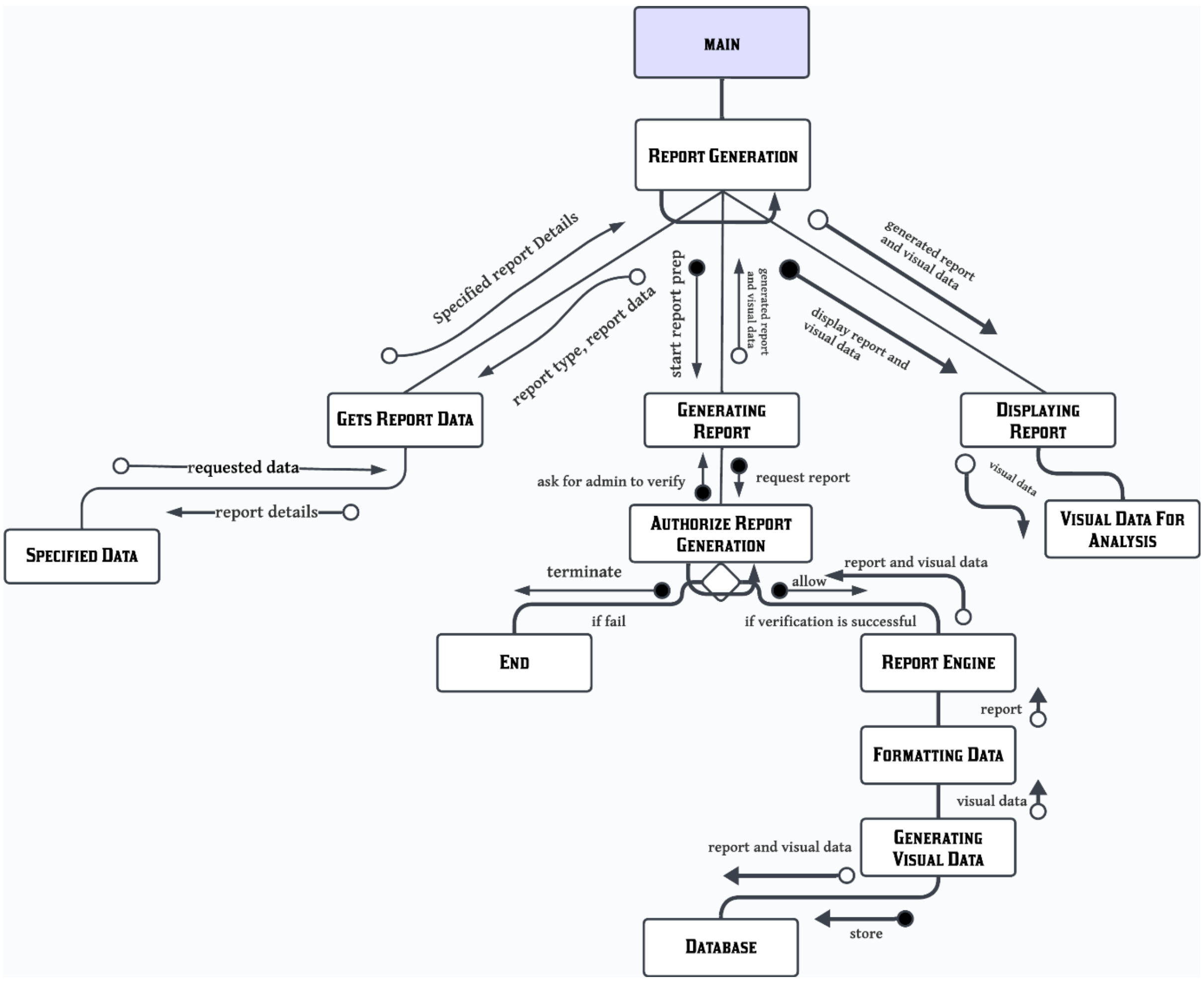

Figure 15: Structure Chart of Report Generation Feature

- **Module Specification (MSpecs)**
    - **Module Name:** Report Preparation Module
    - **Purpose:** This module facilitates the generation and display of reports based on the admin's specification.
    - **Pseudocode:**

```
PROCEDURE PrepareReport():
    INPUT: SpecifiedData, ReportType
    OUTPUT: ReportStatus

    # Validate specified data and report type
    IF SpecifiedData IS VALID AND ReportType IS VALID:
        # Generate report and visual data
        ReportStatus = GenerateReport(SpecifiedData, ReportType)
        IF ReportStatus == SUCCESS:
            DisplayReport(SpecifiedData, ReportType)
        END IF
    ELSE:
        ReportStatus = INVALID_INPUT

    RETURN ReportStatus
END PROCEDURE
```

- **Input Parameters:**
    i. **SpecifiedData:** The data specified by the admin for the report generation.
    ii. **ReportType:** The type of report chosen by the administrator.
- **Output Parameters:**
    i. **ReportStatus:** Indicates the status of the report generation (SUCCESS, INVALID_INPUT, etc.)
- **Local Variables:**
    i. **SpecifiedData:** Temporary variable to store the admin's specified data.
    ii. **ReportType:** Temporary variable to store the admin's chosen report type.
    iii. **ReportStatus:** Variable to store the status of the report generation.
- **Calls:**
    i. **GenerateReport(SpecifiedData, ReportType):** Subroutine to handle report generation and visual data creation.
    ii. **DisplayReport(SpecifiedData, ReportType):** Subroutine to display the report and visual data to the user.

- **Called By:**
  - **i.** The main system control flow.
- **Feature Description:**

The "Report Preparation" feature is designed for administrators to generate and display reports within the McGregor Institute platform. Administrators input specified data and select report types. The system validates the input, generates reports, and displays them based on the administrator's preferences. The feature provides insights into various aspects such as user activity, financial data, etc.

5.3. Join the program feature with Enrollment System

The user can join the program through the enrollment system.

- **Context Level for this feature**

Context Level is the highest level in Data Flow Diagram. It is also referred to as Level 0 Data Flow Diagram. (context diagram, 2024)This high-level interaction encapsulates the entire process of a user joining a program without detailing the internal processes or data stores that support this feature. The diagram shows that the user provides details to the system (input), and in return, the system provides enrollment details to the user (output).

Major Input:

retrieve user_details: This represents the information that the user submits to the Enrollment System. This data includes personal information, educational background, course preferences, and any other details required for enrollment.

Major Output:

view enrollment_details: This is the information that the enrollment system provides back to the user to view. The details about the course details, course confirmation, application details and payment requirement for the course are provided to the users to view.

Data Flow Diagram Level 0

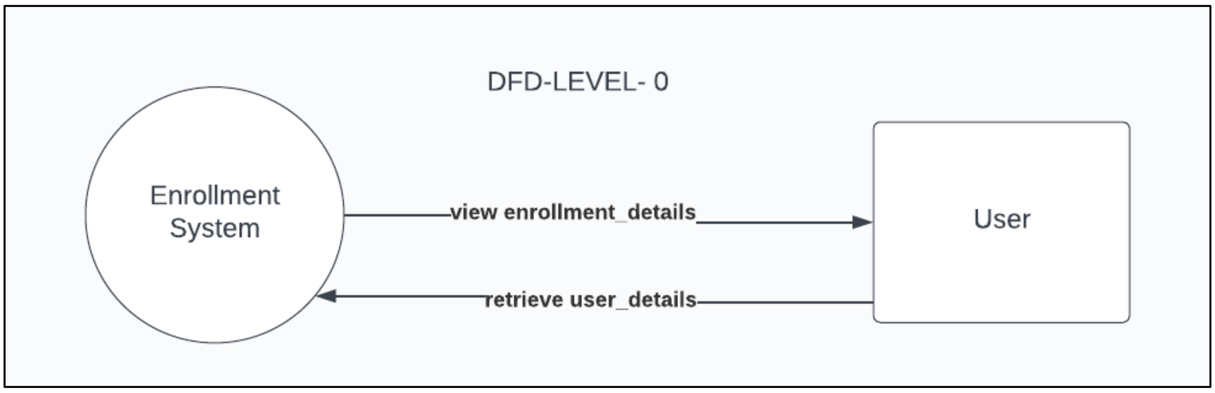

Figure 16: Level 0 DFD of Join the Program Feature

- **Internal Model Specification for the Feature**:

The Internal Model Specification details how data moves through the system from the point of course selection, through application, and ending with payment processing, including interactions with various data stores that hold the necessary details for each step.

- **Data Flow Diagram Level 1**

The DFD Level-1 diagram provided for the "Join the Program" feature of the Enrollment System gives a more detailed look into the main processes and data stores involved, compared to the context level diagram.

i. Select Course

Major Input: course_inquiry — This input comes from the user and represents a request for information about courses.
Data Store: course_details — This is where information about all the courses is stored. It includes details such as course names, descriptions, schedules, fees, availability, and any other relevant data about the courses offered.
Major Output: confirmed_course_details — The output of this process is the detailed information about the course that the user has confirmed. This will typically include confirmation that the course is available and confirmed for enrollment.

2. Apply for Course

Major Input: confirmed_course_details — This input signifies that the user has selected a course and is providing details required to apply for it.

Data Store: application_details— This is where the details of the user's application are stored. This includes the user's personal information, the selected course, application status, and any additional documentation.

Major Output: application_status — The output is the status of the user's application, which could be pending, accepted, waitlisted, or rejected, and is communicated back to the user.

3. Process Payment

Major Input: application_status— This input include user details for payment processing if the application is accepted.

Data Store: payment_details — This contains records of the payment transactions, including payment amounts, dates, and statuses (completed, pending, failed).

Major Output: payment_status — If the course requires payment, it provides payment details to proceed with payment. If the course does not require payment, the course is confirmed.

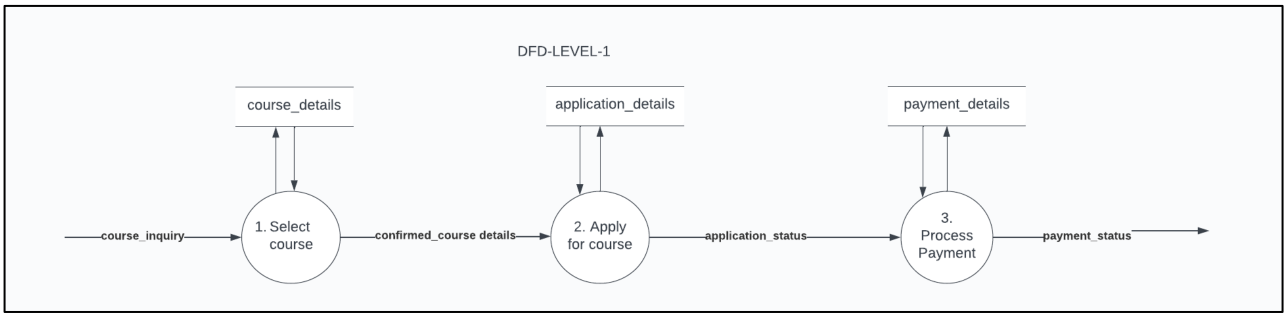

Figure 17: Level 1 DFD of Join the Program Feature

- **Data Flow Diagram Level 2**

The DFD Level-2 diagram provides a more detailed view of the "Join the Program" feature within the Enrollment System, breaking down the process of Select Course into subprocesses.

Process Specification:

Number: 1.1

Name: View Course List

Description: Retrieves a complete list of courses available for enrollment from the course database.

Input data flow: "course_inquiry" - The user's request to view available courses, including search criteria or filters.

Output data flow: "course_information" - Detailed information on all available courses that match the inquiry.

Number: 1.2

Name: Select Available Course

Description: Allows the user to select a course from the list of available courses.

Input data flow: "available_course_details" - Details of courses that are currently available for enrollment.

Output data flow: "requested_course_information" - Information about the course selected by the user.

Number: 1.3

Name: Check Course Slot Availability

Description: Verifies the availability of slots for the selected course.

Input data flow: "requested_course_information" - Information about the course chosen by the user.

Output data flow: "course_available_status" - The availability status of the selected course.

Number: 1.4

Name: Confirm Course

Description: Confirms the user's course selection and provides the user with confirmation details.

Input data flow: "course_available_status" - The status indicating whether the selected course has available slots.

Output data flow: "confirmed_course_details" - Confirmation details of the course, including slot availability and next steps for enrollment.

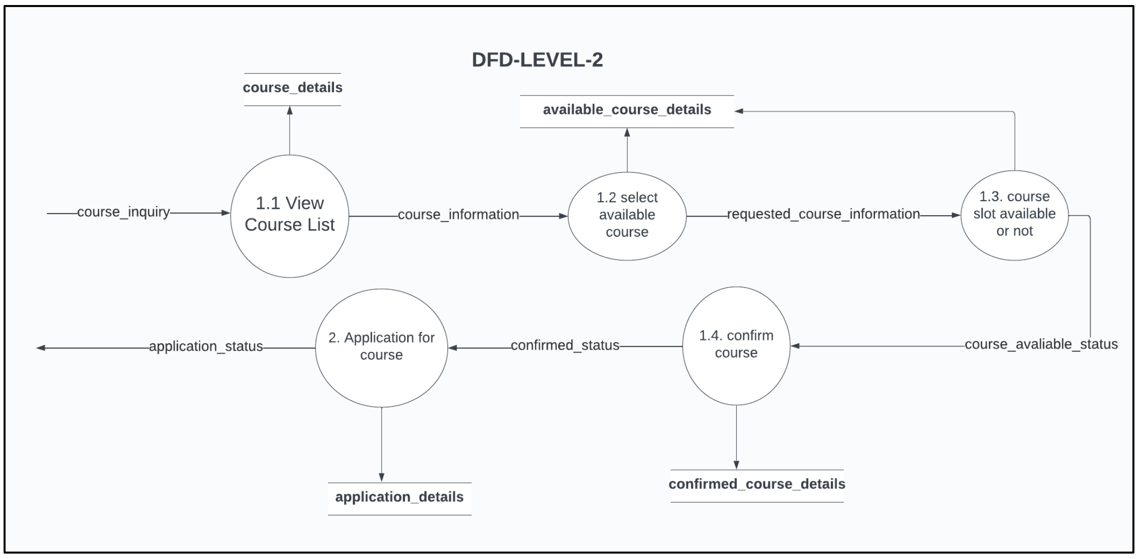

Figure 18: Level 2 DFD of Join the Program Feature

- **Design Specification**

i. Structure Chart

The structure chart represents the breakdown of the enrollment system into lowest functional modules.

Control Module:

Main: This is the top-level control module that orchestrates the overall process flow in the system.

Enrollment System: This module acts as a control module for the submodules related to course selection, application, and payment processing.

Sub Modules:

1. Select Course: A submodule under "Enrollment System" that handles course selection.

1.1 View Course List: A submodule of "Select Course" that retrieves the available courses.

1.2 Available Course: A submodule of “Select Course" that allows a user to pick any course.

1.3 Check Course Slot Availability: A submodule of "Select Course" that checks the availability of slots for a selected course.

1.4 Confirm Course: A submodule of "Select Course" that confirms the course selection with the user.

2. Apply for Course: A submodule under "Enrollment System" for processing course applications.

3. Process Payment: A submodule under "Enrollment System" responsible for handling payment transactions.

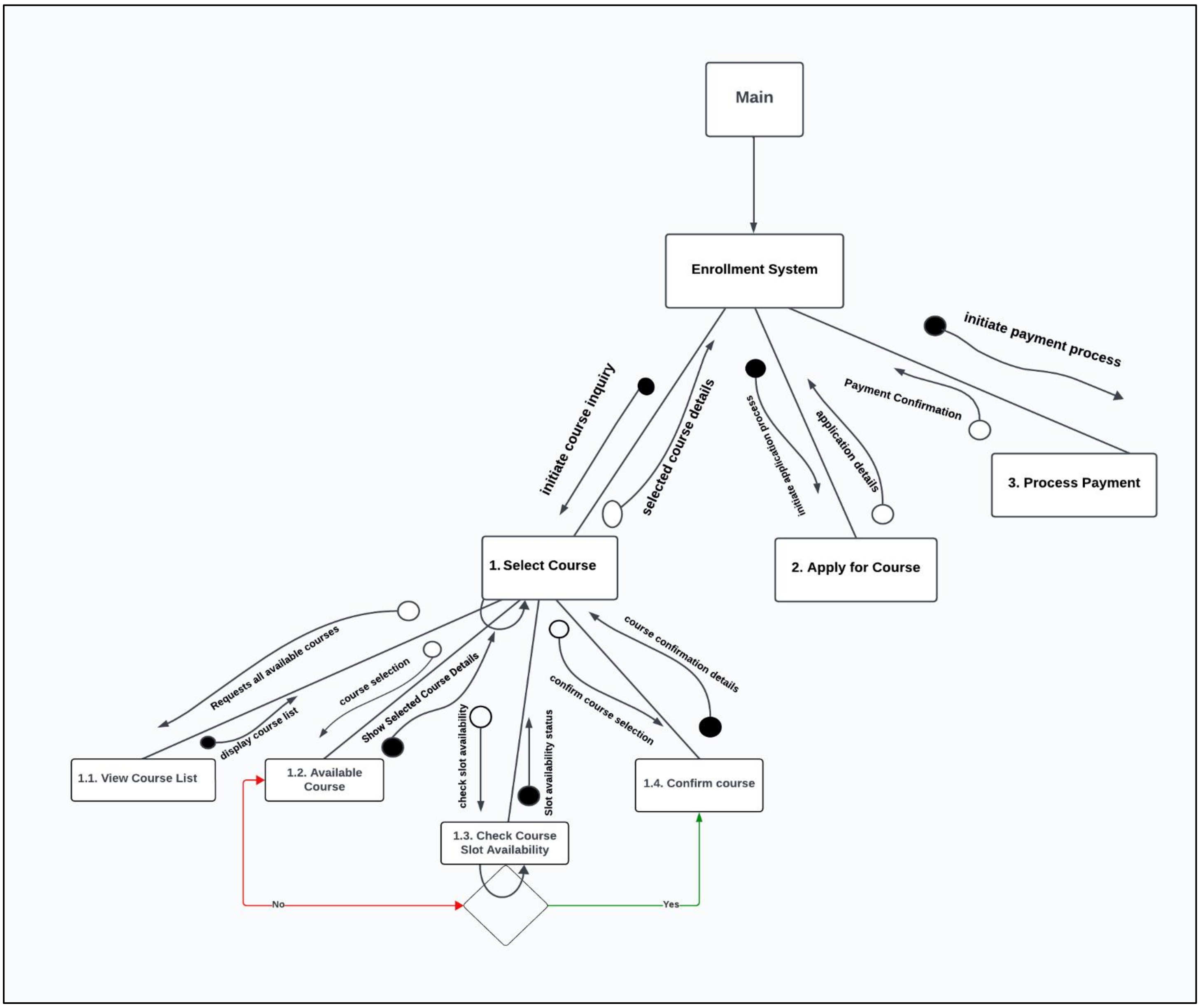

Figure 19: Structure Chart of Join the Program Feature

ii. Module Specification

Module Name: Select Course Module

Purpose: This module handles the user's course selection, including validating the course ID, retrieving course details, and checking for availability. It also informs the user of the selection status.

Pseudocode:

```
PROCEDURE SelectCourse():
INPUT: CourseID
    OUTPUT: SelectionStatus, CourseDetails

    # Validate the course ID input
    IF ValidateCourseID(CourseID):
        # Retrieve detailed course information
        CourseDetails = RetrieveCourseDetails(CourseID)
        # Check course availability
        SelectionStatus = CheckCourseAvailability(CourseDetails)
        IF SelectionStatus == AVAILABLE:
            DisplayCourseDetails(CourseDetails)
        END IF
    ELSE:
        SelectionStatus = INVALID_COURSE_ID
    END IF

    RETURN SelectionStatus, CourseDetails
END PROCEDURE
```

Input Parameters:

CourseID: The identifier for the user-selected course.

Output Parameters:

SelectionStatus: Indicates the status of the course selection (AVAILABLE, FULL, INVALID_COURSE_ID)

CourseDetails: Contains detailed information about the course if the selection is successful.

Local Variables:

CourseID: A temporary variable to store the course identifier input by the user.

CourseDetails: A variable to store the retrieved course information.

SelectionStatus: A variable to store the status of the course selection process.

Calls:

ValidateCourseID(CourseID): A subroutine to validate the course ID provided by the user.
RetrieveCourseDetails(CourseID): A subroutine to retrieve details about the course associated with the given ID.
CheckCourseAvailability(CourseDetails): A subroutine to check the availability of slots in the selected course.
DisplayCourseDetails(CourseDetails): A subroutine to display the course details to the user if available.

Called By:

The main system control flow with course enquiry request.

Feature Description:

"Join the Program" allows users to enroll in programs offered by McGregor Institute through the enrollment system. Users provide personal information, educational background, and course preferences. The feature involves processes like course selection, application submission, and payment processing. The system validates user details and provides enrollment information, including course confirmation and payment requirements.

5.4. Take Exam Certification

- **Context Level:**
- Major Inputs:
  1. Information provided by the user, including personal details and relevant identification data.
  2. Information related to the specific exam, such as exam type, duration, and any other relevant parameters.
- Major Outputs:
  1. Certification Exam Results: The outcome of the certification exam, indicating whether the user passed or failed.
  2. Certification Status: Indicates the overall status of the certification process (e.g., Certified, Not Certified).

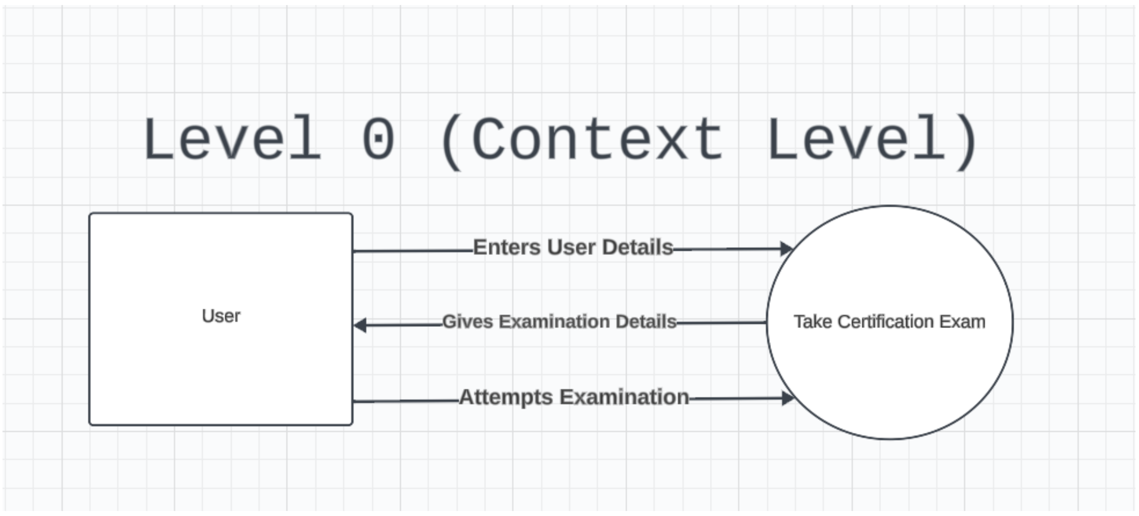

Figure 20: Level 0 DFD of the Take Certification Exam feature

- **Internal Model Specification for the Feature**:

- **Level 1 DFD Fragments:**
  Illustrates high-level processes - user details validation, exam attendance, exam data management, evaluation, certification, and outcome database updates in the Certification System.

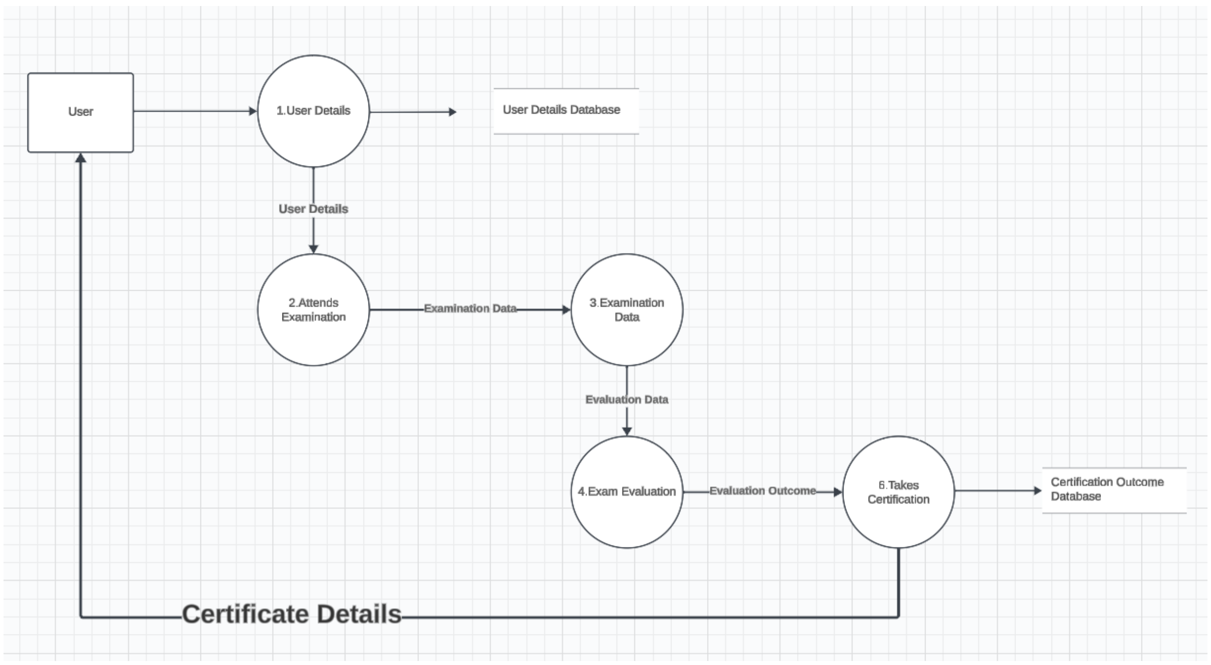

Figure 21: Level 1 DFD of the Take Certification Exam Feature

- **Level 2 DFD Fragments:**

  Illustrates detailed processes - validating and managing user details, check-in, exam execution, exam data management, response evaluation, certification opportunities, and outcome database updates in the Certification System.

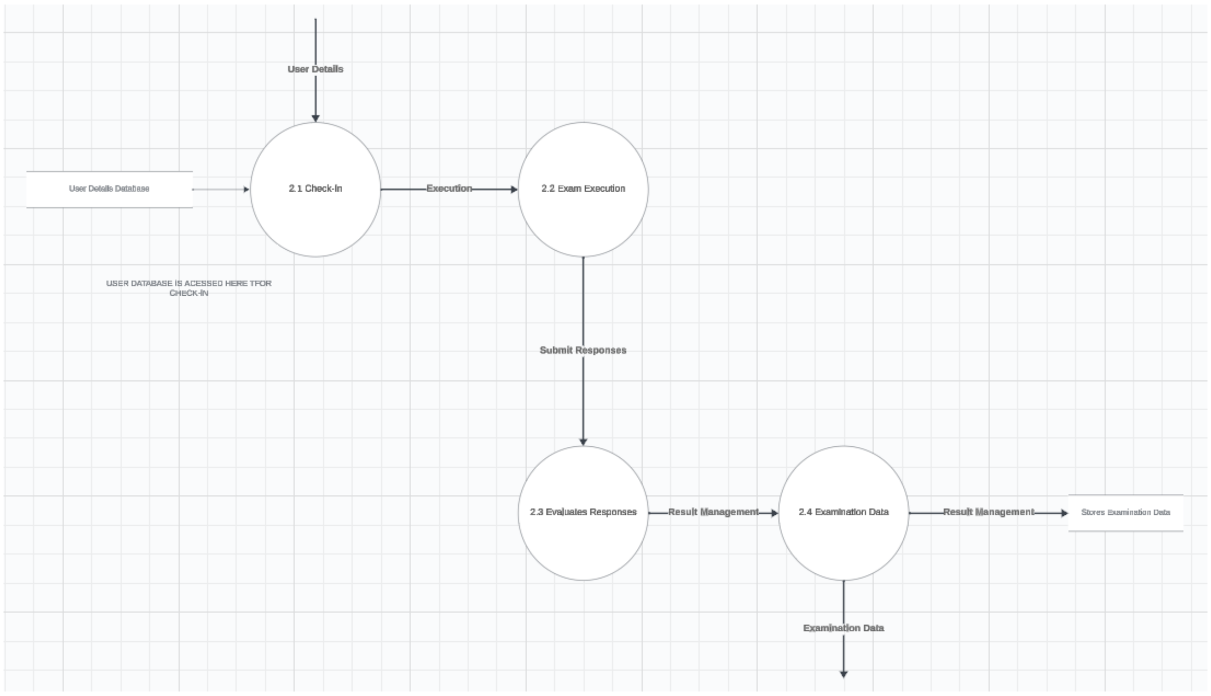

Figure 22: Level 2 DFD of the Take Certification Examination Feature

- **Design Specifications**
- **Structure Chart:**

This structure chart provides a hierarchical view of the "Take Certification Exam" Module, including their relationships and dependencies.

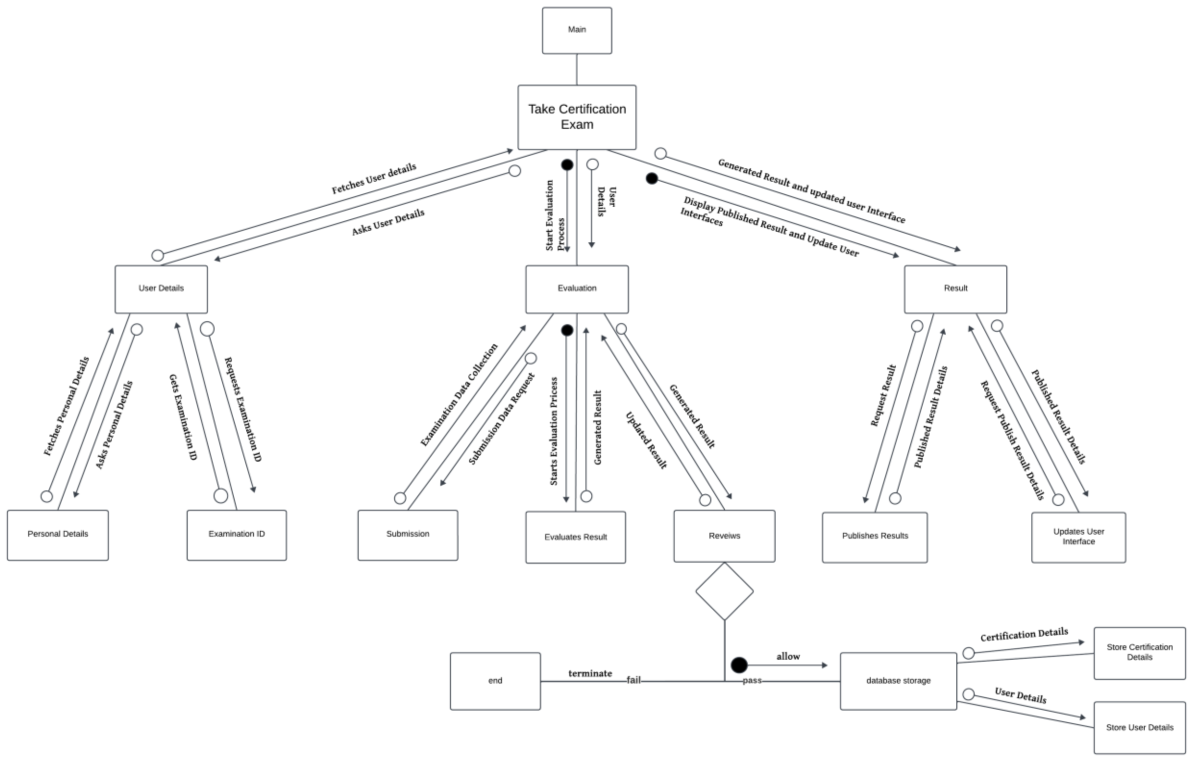

Figure 23: Structure Chart of the Take Certification Exam Feature

- **Module Specifications (MSpecs):**
- **Module Name**: Taking Certification Exam Module
- Purpose: This module facilitates the generation and display of examination data and certificate distribution.
- **Pseudocode:**

```
# Request user details
UserDetails = RequestUserDetails()

# Attends Examination
ExamAnswers = AttendExam()

# Validate user details and exam answers
IF AreUserDetailsValid(UserDetails) AND AreExamAnswersValid(ExamAnswers):
    # Evaluate exam and calculate score
    ExamScore = EvaluateExam(ExamAnswers)

    # Takes Certification
    CertificationStatus = TakeCertification(ExamScore)

    IF CertificationStatus == SUCCESS:
        DisplayCertificationDetails(UserDetails, ExamScore)
    ELSE:
        Terminate()
    END IF
ELSE:
    CertificationStatus = INVALID_INPUT

RETURN CertificationStatus
```

Input Parameters:

UserDetails: Data containing user details obtained through user input.

ExamAnswers: Data containing the answers submitted by the user during the examination.

Output Parameters:

CertificationStatus: Indicates the status of the certification process (SUCCESS, INVALID_INPUT, etc.).

Local Variables:

UserDetails: Temporary variable to store user details.

ExamAnswers: Temporary variable to store exam answers.

CertificationStatus: Variable to store the status of the certification process.

Calls:

RequestUserDetails(): Subroutine to obtain user details.

DisplayCertificationDetails(UserDetails, ExamScore): Subroutine to display

Called By:

The main system control flow.

Description:

The "Take Certification Exam" feature manages the process of users taking certification exams within the McGregor Institute platform. Users provide personal details and engage in the examination. The system validates user details, evaluates exam responses, and, based on the outcome, certifies or denies certification. Successful certifications result in the display of certification details.

5.5. Make Payments

5.5.1 Context Level

- **Data Flow Diagram Level 0**

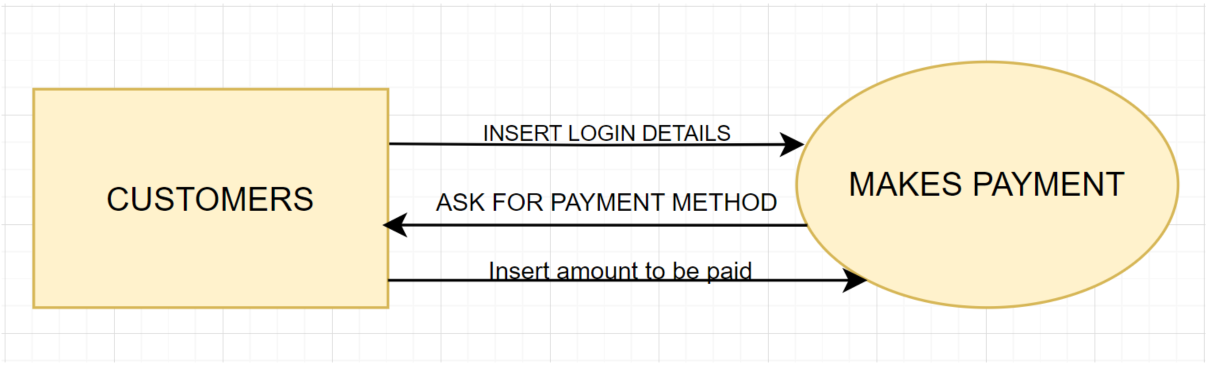

Figure 24: Level 0 DFD of the Make Payments Feature

5.5.2

Internal Model Specification for the Feature:

- **Data Flow Diagram Level 1**

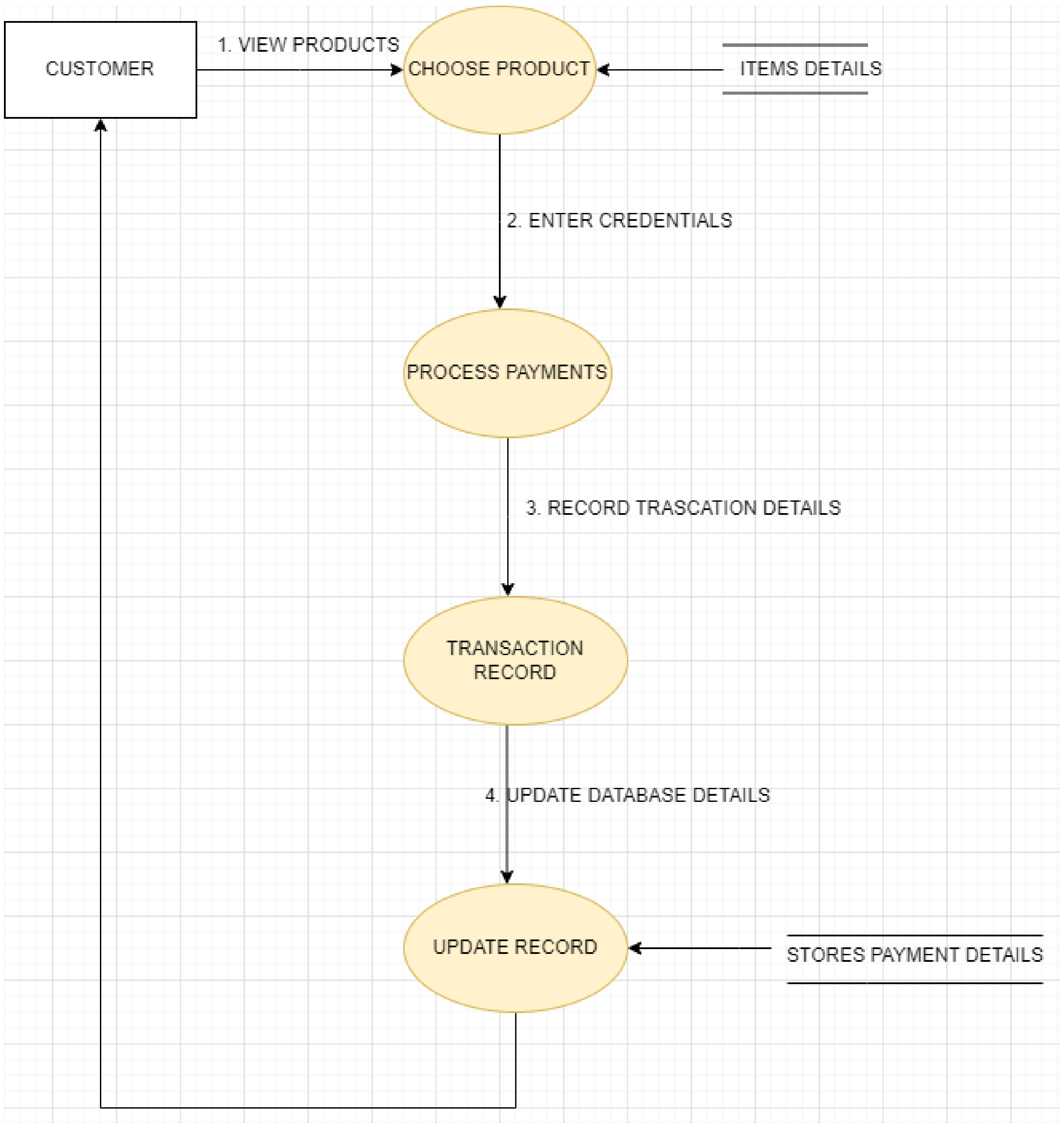

Figure 25: Level 1 DFD of the Make Payments Feature

- **Data Flow Diagram Level 2**

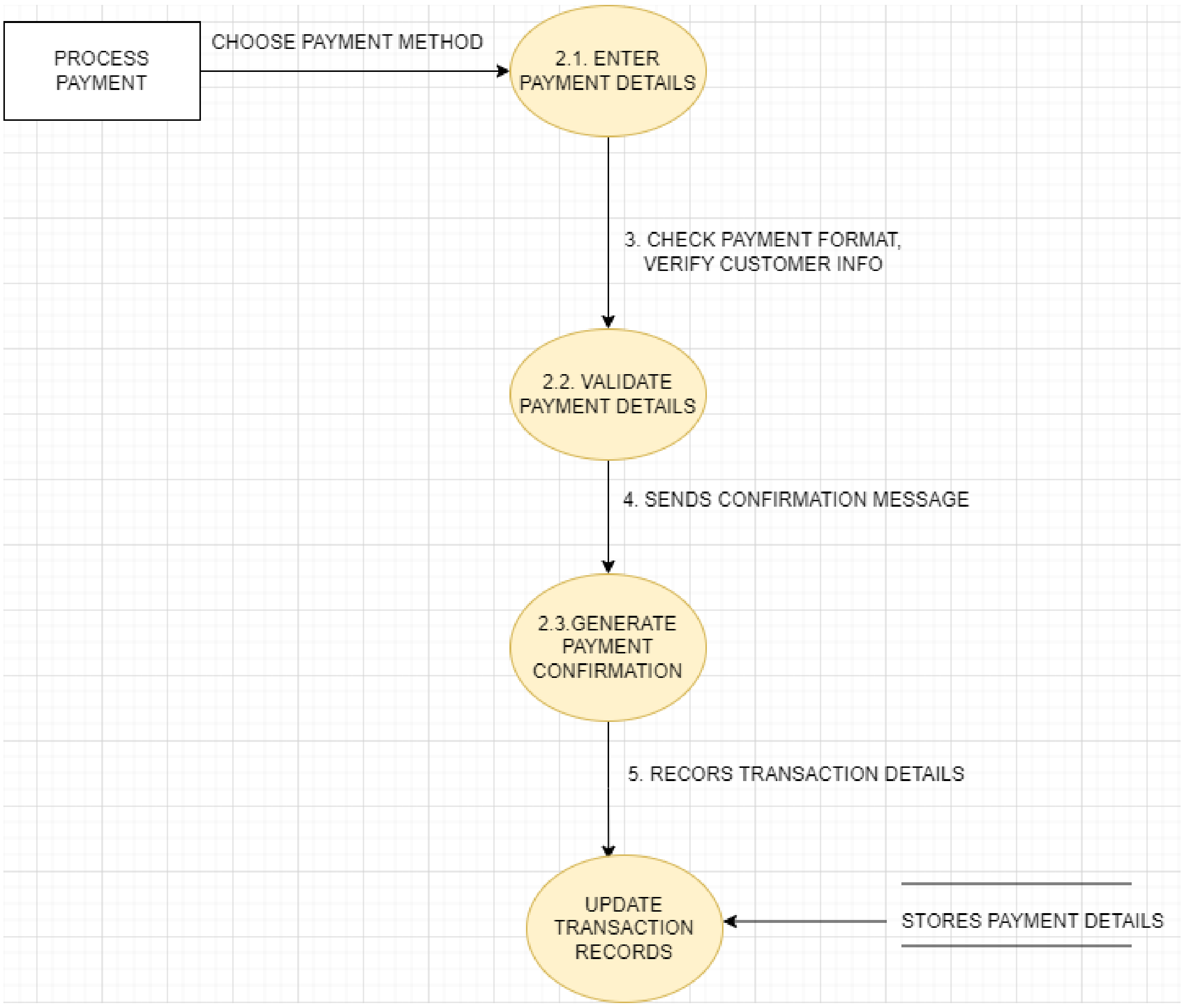
PROCESS PAYMENT
CHOOSE PAYMENT METHOD
2.1. ENTER PAYMENT DETAILS
3. CHECK PAYMENT FORMAT, VERIFY CUSTOMER INFO
2.2. VALIDATE PAYMENT DETAILS
4. SENDS CONFIRMATION MESSAGE
2.3.GENERATE PAYMENT CONFIRMATION
5. RECORS TRANSACTION DETAILS
UPDATE TRANSACTION RECORDS
STORES PAYMENT DETAILS

Figure 26: Level 2 DFD of the Make Payments Feature

5.5.3. Design Specification Diagram.

- **Structure Chart**

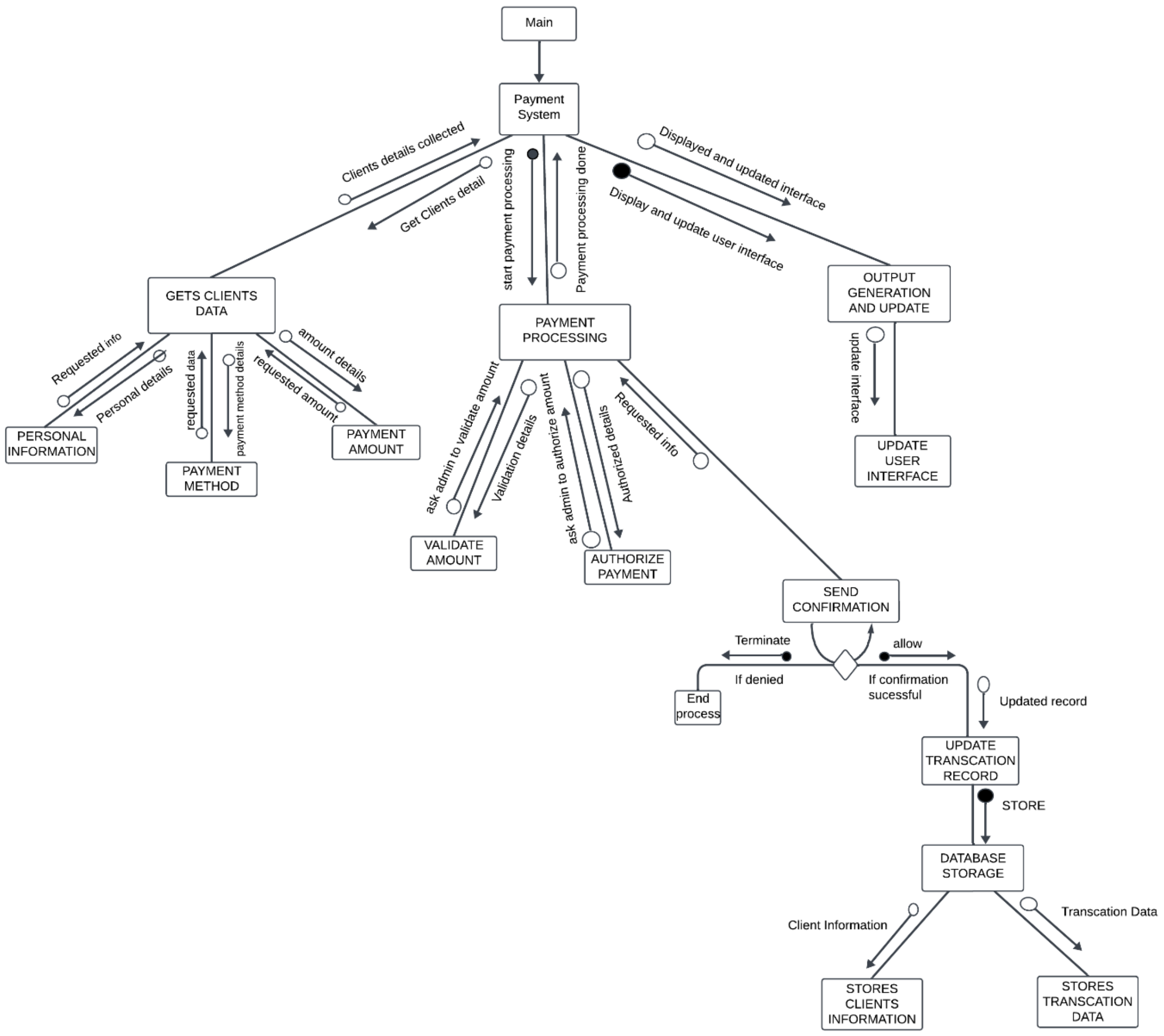

Figure 27: Structure Chart of the Make Payments Feature

- **Module Specifications (MSpecs):**
  - **Psuedocode:**

```
# Step 1: Request user details
UserDetails = RequestUserDetails()

# Step 2: Browse and Select Product
SelectedProduct = BrowseAndSelectProduct()

# Step 3: Validate user details and selected product
IF AreUserDetailsValid(UserDetails) AND IsProductValid(SelectedProduct):
    # Step 4: Display Order Summary
    DisplayOrderSummary(UserDetails, SelectedProduct)

    # Step 5: Choose Payment Method
    PaymentMethod = ChoosePaymentMethod()

    # Step 6: Enter Payment Details
    PaymentDetails = EnterPaymentDetails(PaymentMethod)

    # Step 7: Validate Payment
    IF ValidatePayment(PaymentDetails):
        # Step 8: Process Order
        OrderStatus = ProcessOrder(UserDetails, SelectedProduct, PaymentDetails)

        IF OrderStatus == SUCCESS:
            # Step 9: Display Order Confirmation
            DisplayOrderConfirmation(UserDetails, SelectedProduct, PaymentDetails)
        ELSE:
            Terminate()
        END IF
    ELSE:
        DisplayErrorMessage("Payment validation failed. Please check your details.")
        Terminate()
    END IF
ELSE:
    DisplayErrorMessage("Invalid user details or selected product.")
    Terminate()
END IF

# Step 10: Return Order Status
RETURN OrderStatus
```

- **Input Parameters:**
    1. UserDetails: Data containing user details obtained through user input.
    2. SelectedProduct: Data containing the product selected by the user.
    3. PaymentMethod: Data representing the chosen payment method.

- **Output Parameters:**
    1. **OrderStatus:** Indicates the status of the order processing (SUCCESS, INVALID_INPUT, etc.).
- **Local Variables:**
    1. UserDetails: Temporary variable to store user details.
    2. SelectedProduct: Temporary variable to store the selected product.
    3. PaymentMethod: Temporary variable to store the chosen payment method.
    4. OrderStatus: Variable to store the status of the order processing.
- **Calls:**
    1. RequestUserDetails(): Subroutine to obtain user details.
    2. DisplayOrderSummary(UserDetails, SelectedProduct): Subroutine to display the order summary.
    3. DisplayOrderConfirmation(UserDetails, SelectedProduct, PaymentDetails): Subroutine to display the order confirmation.
- **Called By:**
    1. The main system control flow.
- **Description:**

The "Make Payment" feature facilitates the seamless processing of user payments within the McGregor Institute platform. Users go through a step-by-step process, starting from providing personal details to selecting products and choosing payment methods. The system ensures the validity of user information, product selection, and payment details before processing the order. Successful transactions lead to the display of order confirmations.

6. Summary

This document serves to outline the functional as well as nonfunctional specifications along with the design details for the various features within the McGregor Institute platform. These features include but are not limited to “Purchase Plant”, “Report Generation”, “Join the Program”, “Take Exam Certification” and “Make Payment”. Each feature is presented with context level diagrams, internal model specifications (level 1, level 2 DFD fragments, design specifications, structure charts and even module specifications).

7. References

context diagram. (2024, January 05). Retrieved from edrawmax: https://www.edrawmax.com/context-diagram/