LONDON METROPOLITAN UNIVERSITY

islington college
(इस्लिङ्टन कलेज)

CS5002NI Software Engineering

McGregor Institute of Botanical Training

20% Group Coursework

2023-2024

Assignment Due Date: January 5, 2024

Assignment Submission Date: January 5, 2024

Group 1		
SN	**Student Name**	**University ID**
1	Apil Thapa (Leader)	22067753
2	Arbit Bhandari	22068111
3	Jenish Katuwal	22068751
4	Miraj Deep Bhandari	22067814
5	Rijan Paudel	22067807

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 mark of zero will be awarded.

Table of Contents

Table of Figures

Table of Tables

1. Introduction

The McGregor Institute of Botanical Training, an Ireland-based institution located in Godavari, Lalitpur, has been providing diverse undergraduate and postgraduate courses in agriculture and horticulture for nearly seven years. With an affiliation to Dublin City University, the institute is now preparing to address the evolving needs of plant enthusiasts by introducing short-term certification courses in horticulture. Beyond academic offerings, McGregor Institute plans to create a dynamic community and a platform for plant enthusiasts to engage, share ideas, and contribute to the protection of rare plants and forests.

This report outlines the structured software engineering approach employed to design and implement various functions within McGregor Institute's system. The project charter establishes the problem statement, business case, goal statement, timeline, scope, and team members. The software requirement specification (SRS) details functional and non-functional requirements, design and implementation constraints, and external interfaces required for the system. To stick to structured software engineering principles, significant parts of the system are designed using the Yourdon method. Additionally, the report provides a detailed specification of both group and individual tasks.

The group task involves environmental model specifications, internal model specifications, and design specifications for the entire system. The environmental model includes context-level and data flow diagrams (DFDs), while the internal model comprises an entity-relationship diagram (ERD), data dictionary, and process specifications. The design specification encompasses a structure chart for the whole system. For individual tasks, each group member selects a specific function (Make Payment, Purchase Plant, Report Preparation, Join the Program, or Take Certification Exam) and provides an in-depth analysis. This includes environmental model specifications, internal model specifications, and design specifications specific to the chosen function. An assignment diary documents assumptions, inconsistencies, responsibilities, and group meetings.

This systematic approach aims to demonstrate practical knowledge of structured software engineering principles, meeting the project's objectives, and encouraging successful collaboration within the group.

2. Project Charter

2.1 Problem Statement

- McGregor Institute aims for a big 10 crores turnover next year, but lacking a central platform for courses, plant sales, and community engagement might slow down reaching this financial goal.

- Relying on traditional communication methods creates challenges for McGregor Institute in effectively engaging with plant enthusiasts. This limitation affects the formation of a passionate user base crucial for reaching the desired financial milestone.

- The lack of online assistance for plant advice and exams introduces a risk of reduced student engagement, presenting an obstacle for McGregor Institute in reaching its financial goal.

2.2 Business Case

Project Name	**Project Manager**
Course and Plant management system	Apil Thapa Arbit Bhandari Jenish Katuwal Miraj Deep Bhandari Rijan Paudel
Date of Project Approval	**Duration**
5th Jan 2024	30 days
Contribution To Business Strategy	1. Short-term horticulture certification courses. 2. Dedicated platform/forum for plant enthusiasts. 3. Online platform for plant sales, secure payments, and bulk purchases. 4. Secure payment system for course enrollment and plant transactions. 5. Certification exams, mock tests, and user-friendly knowledge-sharing forum.

Costs	1. Plant Seeds: Cost per seed: Rs. 100 Estimated quantity: 5,000 seeds Total Cost: Rs. 500,000 2. Course Teachers: Number of teachers: 5 Monthly payment per teacher: Rs. 50,000

	Duration: 1.5 months Total Cost: Rs. 375,000 3. Learning Zone Rent: Monthly rent: Rs. 200,000 Duration: 1.5 months Total Cost: Rs. 300,000 4. Software Development: Development cost estimate: Rs. 3,000,000 5. Design and Testing: Design and testing costs: Rs. 500,000 6. Infrastructure: Server and database setup: Rs. 800,000 7. Training and Implementation: Training program and implementation: Rs. 200,000 8. Support and Maintenance: Ongoing support and maintenance: Rs. 400,000 Total Estimated Project Cost: Rs. 6,075,000

Expected Return on Investment	Total Estimated Revenue=Rs. 7,000,000 $ROI = \frac{\text{Total Estimated Revenue} - \text{Total Project Cost}}{\text{Total Project Cost}} * 100$ $x = \frac{7,000,000 - 6,075,000}{6,075,000} * 100 \approx 15.26\%$ Updated Expected ROI: 15.26% In this scenario, with the increased total estimated revenue, the project is expected to be profitable, and the ROI is positive at approximately 15.26%
Risk Assessment	1. Unpredictable changes in demand for courses and plants. 2. Low user adoption of the online platform and forum. 3. Development and implementation difficulties in creating the online platform.

Table 1: Table of Business Case.

2.3 Goal Statement

The central objective of our institute's project is to establish an educational and botanical excellence platform. Our primary focus is on delivering exclusive postgraduate and undergraduate courses, ensuring a quality educational experience in the field of agriculture and horticulture. We aspire to empower individuals through effective learning opportunities that not only meet but exceed industry standards.

Simultaneously, we're dedicated to offering high-quality plants for purchase, enhancing the learning environment, and supporting a sustainable business model. also, providing educational programs and botanical products known for their effectiveness, quality, and positive impact on individuals and the environment. This integrated approach is designed not only to enhance education.

2.4 Timeline

Phases (stages)	Main Tasks
Phase 1: Planning (4 days)	1. Kick-off meeting to discuss project goals, roles, and initial planning. 2. Outline project objectives and define key deliverables.
Phase 2: Requirement Gathering, System Design, and Development (1.5 weeks)	1. Implement backend and frontend components based on the finalized design. 2. Review and finalize the design with the development team. 3.Conduct regular team meetings to ensure alignment with project goals.
Phase 3: Testing and Bug Fixing (6 days)	1. Begin system testing to identify major issues. 2. Initiate bug fixing and improvements.
Phase 4: Deployment and Maintenance (1.5 weeks)	1. Monitor system performance and address immediate issues. 2. Conduct training sessions for end-users if necessary.

Table 2: Table of Timeline

Total Project Time: 4.5 weeks (32 days)

This indicates the overall duration of the project, highlighting the 4.5-week timeframe, equivalent to 32 days.

2.5 Scope

2.5.1 User Registration

- New users will be able to register in the system to access its features.

- User profiles will include essential information for identification and communication purposes.

2.5.2 Program Enrollment

- Users can join various programs, including undergraduate and postgraduate courses, as well as short-term certification courses.

- The system will support both paid and unpaid courses.

2.5.3 Plant Purchase

- Users will get access to view different plant varieties available for purchase.

- A shopping cart functionality will be implemented for bulk plant purchases.

2.5.4 Payment Processing

- A secure payment system will be integrated to facilitate transactions for both course enrollment and plant purchases.

- Payment data will be securely stored for record-keeping.

2.5.5 Expert Recommendations

- Users can request plant recommendations from experts.

- Experts will respond based on user-provided site locations, soil condition images, and other relevant information.

2.5.6 Report Preparation

- Admins will have the access to generate detailed reports, including financial reports, employee reports, and user-related reports.

2.5.7 Certification Exams

- Users can take mock tests at their convenience and check the results.

- Certification exams can be undertaken after fulfilling specified prerequisites.

2.5.8 Forum

- A discussion forum will be implemented for users to engage in conversations about plants.

- Users can post queries, share opinions, comment on posts, and upvote valuable contributions.

2.5.9 Notification System

- A notification system will be in place to alert users about relevant activities, updates, and responses to their posts.

2.6 Team Members

Team member's name	Task
1. Apil Thapa (Project Manager)	1. Led the project planning and coordination. 2. Gathered and prioritized requirements from stakeholders.
2. Miraj Deep Bhandari (Data Analyst)	1. Conducted data research and gathered insights. 2. Ensured data quality and integrity.
3.Jenish Katuwal (Backend Developer)	1. Implemented server-side functionalities. 2. Developed backend infrastructure and logic.
4. Rijan Paudel (Front-End Developer)	1. Implemented user interfaces and clientside functionalities. 2. Conducted testing and debugging for frontend components.
5. Arbit Bhandari (UI/UX Designer)	1. Collaborated with developers for effective design implementation. 2. Designed user interfaces and overall user experience.

Table 3: Table of Team Members.

3. Software Requirements Specification (SRS)

3.1 Introduction:

McGregor Institute of Botanical Training (MIBT) is an established institute in Nepal, offering undergraduate and postgraduate courses in agriculture and horticulture affiliated with Dublin City University. With growing interest in horticulture, MIBT plans to launch a variety of short-term certification courses and an online platform for plant enthusiasts. This SRS document outlines the requirements for this online portal.

3.2 Purpose:

This SRS document outlines the software requirements for the development of a comprehensive system to manage the institute's operations, including course enrollment, plant sales, community forums, and administrative tasks.

3.3 Scope

The objective of this MIBT online portal is to automate crucial operations for:

- **Streamlining course registrations:** By integrating user data and course catalogs, the portal automatically manages enrollments and payments, eliminating manual processing.

- **Optimizing plant sales:** User-friendly browsing, cart systems, and secure payment gateways automate transaction processes and track inventory details.

- **Fostering a dynamic plant community:** Discussion forums, plant recommendations based on user data and location, and expert consultations through the platform automate knowledge sharing and engagement.

- **Empowering administrative decisions:** Comprehensive reports on finances, employee activities, and user behavior inform resource allocation, marketing strategies, and course content optimization.

- **Enhancing user experience:** Personalized notifications, readily available course materials, and convenient features like mock tests automate information access and personalized support.

3.4 Functional Requirements

Functional requirements (FRs) define the specific features, behaviors, and capabilities that a software system must possess to meet the needs and expectations of its users. These requirements outline the functionalities that the system should perform and describe how users will interact with the system to achieve their goals. Functional requirements are crucial for guiding the design, development, and testing phases of a software project (Benslimane, Cysneiros and Bahli, 2007).

These are guidelines for the system's operation, including how it should react to particular inputs, what it should do, and how to behave in various situations. It describes the required functionality that product developers must incorporate in order to enable users to carry out their duties.

3.4.1 Register in the System

- Browse plants and view certification courses without registering.
- Register by providing basic information for transactions and course enrollments.
- Log in using email and password for secure access.
- Manage profiles, update information, and perform account-related actions.
- Admins can activate, suspend, or delete user accounts as needed.

3.4.2 Join the program.

- Explore graduate, postgraduate, and certificate courses without restrictions.
- Log in to enroll in courses, with both free and paid options.
- Make payments for paid courses through wallets or bank accounts.
- Gain access to certificates, materials, assignments, and mock tests within enrolled courses.
- Admins can manage course catalogs, add, modify, and price courses.

3.4.3 Purchase plants:

- Explore plant varieties, view detailed information, and add to the shopping cart.
- Log in to complete the purchase process.
- Make payments through wallets, bank accounts, or cash on delivery.
- Provide additional details for online delivery.
- Admins can manage plant inventory, set prices, and update stock levels.

3.4.4 Payment

- Ensuring secure transactions for plant purchases and course enrollments.
- Prompting users for OTP codes for added security.
- Maintaining a secure database with comprehensive payment records.

3.4.5 Forum

- Log in to create, reply to, and engage in discussions on plant-related topics.
- Moderate content, manage spam, and prevent abuse for a positive environment.
- Express opinions through upvotes and downvotes on posts and comments.

3.4.6 Ask for Recommendations

- Undergo profile verification by submitting essential details and documents.
- Request plant recommendations from experts by providing detailed information.
- Upload pictures and videos for more accurate suggestions.
- Engage in live video calls for personalized assistance.

3.4.7 Report Preparation

- Access and generate financial reports detailing revenue and expenses.
- Generate user reports covering activity, course enrollments, forum participation, and plant purchases.
- Generate employee reports for performance metrics and insights.

3.4.8 Certification Exams

- Access timed mock tests for enrolled courses with feedback.
- Take online certification exams with secure access and anti-cheating measures.
- Receive immediate results and detailed score reports.

3.4.9 Get Notification

- Sends personalized notifications based on user activity and preferences.
- Provides course registration confirmation, reminders, and updates.
- Notifies users of new forum posts, replies, plant purchase confirmations, and delivery updates.
- Delivers expert recommendations and exam reminders.

3.5 Non-Functional Requirements

Non-functional requirements (NFRs) are aspects of a software system that define its overall characteristics and constraints rather than specifying specific behaviors or features. These requirements focus on the qualities and properties that contribute to the system's performance, security, usability, maintainability, and other non-functional aspects. Non-functional requirements are essential for ensuring that the system operates effectively, reliably, and meets user expectations in terms of performance and user experience (Benslimane, Cysneiros and Bahli, 2007).

3.5.1 Design and Implementation Constraints:

Design and implementation constraints refer to limitations and conditions that may impact the development, deployment, and operation of a software system (Van Hentenryck, Saraswat and Deville, 1998). Considering the context of McGregor Institute of Botanical Training and its goals. Design and implementation constraints outline specific requirements and preferences set by the client for the development of the McGregor Institute of Botanical Training's online portal. These constraints shape the technical aspects of the system to align with the client's vision.

Here are some of these Constraints provided below:

3.5.1.1 Programming Language

The client insists on using Python with the Django framework for the development of the system. The choice for development is because of it’s, robust framework features, and compatibility with McGregor Institute's technical expertise.

3.5.1.2 Database Management System (DBMS)

The system must utilize PostgreSQL as the primary database management system. This is due to its open-source nature, strong support for complex queries, and the client's familiarity with its usage.

3.5.1.3 Cloud Service Integration

The client requires seamless integration with a specific cloud service provider, such as Amazon Web Services (AWS). This choice is influenced by the scalability, reliability, and cost-effectiveness offered by the selected cloud platform.

3.5.1.4 Development Methodology

The Agile development methodology is mandated by the client for iterative and collaborative development. This approach allows for flexibility in responding to changing requirements and ensures continuous client feedback throughout the development process.

3.5.1.5 User Authentication Framework

The client specifies the integration of OAuth 2.0 for user authentication, enabling secure and standardized access to the system using existing user credentials from recognized identity providers.

3.5.1.6 Mobile Responsiveness

The system must be designed to be fully responsive across various devices, with a particular emphasis on mobile devices. This aligns with the increasing trend of users accessing online platforms through smartphones and tablets.

3.5.1.7 Third-Party API Compatibility

The client emphasizes compatibility with specific third-party APIs for functionalities such as geo-location services, ensuring seamless integration with external systems that enhance the user experience.

3.5.1.8 Continuous Integration/Continuous Deployment (CI/CD)

The client advocates for the implementation of CI/CD pipelines to automate the testing, integration, and deployment processes. This accelerates development cycles and ensures the rapid release of updates and features.

3.5.1.9 Blockchain Integration

The client expresses interest in exploring blockchain technology for securing certain aspects of the system, such as transaction records or certification authenticity. Integration possibilities and potential benefits need to be thoroughly investigated.

3.5.1.10 AI/ML Integration

The client imagines future integration of artificial intelligence (AI) and machine learning (ML) capabilities to enhance user recommendations, automate repetitive tasks, and provide intelligent insights based on user behavior.

3.6 External Interfaces Required

3.6.1 User Interfaces

The user interface serves as the primary point of interaction between users and the McGregor Institute of Botanical Training's online portal. It is the face of the system, playing a crucial role in making the platform visually appealing and user-friendly.

The significance of the user interface cannot be overstated. It serves as the gateway through which users access services, explore courses, make plant purchases, engage in forums, and manage their profiles. A well-designed user interface contributes to the overall attractiveness of the system, ensuring that users can easily comprehend and utilize the diverse features available to them.

The user interface of our system encompasses the following features:

1. **Navigation Menu**

Clear and intuitive menu for easy access to different sections like Courses, Plant Store, Forums, Profile, etc.

2. **Homepage**

Engaging landing page displaying featured courses, new plant arrivals, and important announcements.

3. **Accessibility Considerations**

Prioritizing user experience for color blindness with a thoughtfully chosen color palette.

4. Plant Catalog

Browse plants with images, descriptions, and filtering options. Add to cart or Wishlist functionality.

5. Forum Section

Easy-to-navigate forum with categories, recent posts, and options to create new topics or participate in discussions.

6. User Profile

Personalized user profiles with options to manage settings, view enrolled courses, and track plant purchases.

7. Shopping Cart

Convenient and visible cart with details of selected plants and courses for easy checkout.

8. Search Functionality

Robust search bar allowing users to find courses, plants, or forum topics quickly.

9. Notifications

Alerts for new forum posts, course updates, and plant delivery status.

10. Payment Integration

Seamless integration for secure online payments with various options like wallets, cards, etc.

11. Plant Details Page

Detailed pages for each plant with specifications, price, and an option to add to the cart.

12. Course Details Page

Information-rich pages for each course, including syllabus, instructor details, and enrollment options.

13. Interactive Elements

Engaging buttons, sliders, and interactive elements to enhance user experience.

14. Consistent Branding

Maintain a consistent visual theme and branding throughout the platform for a cohesive user experience.

15. Help and Support

Accessible help sections, FAQs, and customer support options for user assistance.

16. Security Measures

Visual indicators of a secure connection, especially on pages involving transactions or personal information.

17. Logout and Account Management

Clearly visible options for logging out and managing account settings.

18. User-Friendly Forms

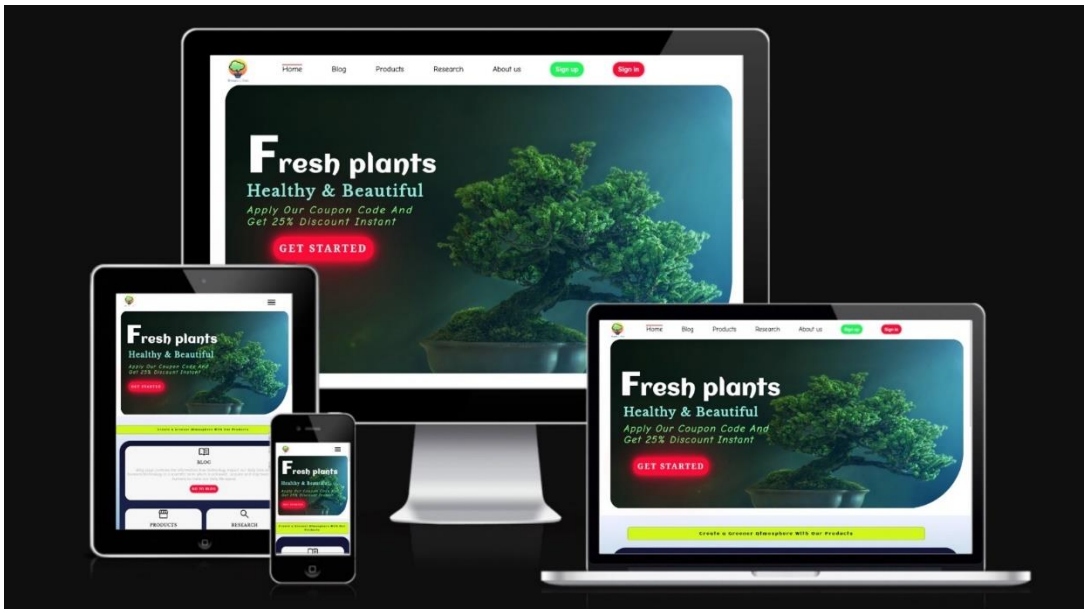
Easy-to-use forms for user registration, course enrollment, and plant purchases.

19. Social Sharing

Integration with social media for sharing achievements, courses, or plant discoveries.

20. Responsive Design

Ensuring the GUI adapts well to different screen sizes, especially for mobile devices.



3.6.2 Hardware Interfaces

The McGregor Institute of Botanical Training's online portal requires integration with specific hardware components to ensure seamless functionality. These hardware interfaces include:

Servers	Client/User Side (Devices with at least 4GB RAM)
❖ Interact with servers for hosting, data storage, and operations. ❖ Specifications aligned with expected user load and data storage needs.	❖ Users can smoothly access the platform on devices with at least 4GB RAM. ❖ Website optimized for performance on laptops and desktops.

High-Resolution Media Storage Devices	Client/user side (Devices with at least High-resolution media storage devices)
❖ Interface with storage devices for efficient storage and retrieval of media files. ❖ Users can view high-resolution images and videos on devices with standard specs.	❖ Users can view high-resolution images and videos on devices with standard specs. ❖ Platform optimized for efficient media streaming on devices with smaller screens.

Payment Processing Hardware	Client/User Side (Various devices including smartphones and tablets)
❖ Interface with payment processing hardware for secure transactions. ❖ Ensure encryption and protection of sensitive financial information.	❖ Users can easily make payments through smartphones and tablets. ❖ Responsive and user-friendly payment interface across different devices.

3.6.3 Software Interfaces

The McGregor Institute of Botanical Training's online portal relies on various software components to deliver its features. These software interfaces include:

1. **Operating Systems:** The system is designed to be compatible with a variety of operating systems, including Windows, Linux, Android, and iOS. This ensures broad accessibility across desktop and mobile devices.

2. **Web Browsers:** The online portal supports various internet browsers, including Mozilla Firefox, Internet Explorer, Safari, Opera, and Chrome. Users are encouraged to utilize these browsers for optimal performance and compatibility.

3. **Payment Gateway:** Integration with a secure payment gateway is essential for processing online payments during plant purchases and course enrollments. The interface must comply with industry standards for financial transactions.

4. **Database Management System (DBMS):** The system interacts with a DBMS to store and retrieve user data, course information, plant inventory details, and other relevant data. The DBMS should be scalable, efficient, and ensure data integrity.

5. **Content Delivery Network (CDN):** For optimal performance in delivering high-resolution media, the system should interface with a CDN to cache and distribute content efficiently, reducing load times for users.

6. **Communication APIs:** Interfaces with communication APIs are required for functionalities such as sending notifications, emails, and managing user communications. This includes interfaces for email services and push notification services.

7. **Security Software:** Integration with security software and antivirus programs is crucial to ensure the protection of user data, financial information, and the overall system integrity.

3.6.4 Communication Interfaces:

In addition to the previously mentioned communication interfaces, the McGregor Institute of Botanical Training's online portal incorporates specific communication interfaces to ensure compatibility and effective data exchange. These interfaces include:

1) Communication Protocols

- **TCP/IP Network Protocol:** The system utilizes the TCP/IP network protocol to establish communication between different components, ensuring reliable and secure data transmission. This protocol is foundational for seamless communication across the online portal's architecture.

- **HTTP Protocol**: HTTP is employed for communication with web browsers and web servers. This protocol is crucial for seamless interactions between users and the online portal. It facilitates the transfer of hypertext, enabling the retrieval and display of web content.

2) Database Communication

- The system communicates with a database to retrieve and store information. This includes communication with a SQL server, where data related to course registrations, plant sales, and user profiles is managed. The communication with the database is fundamental for data integrity and efficient management of crucial information.

- Ensure that database communication encompasses essential functionalities such as data retrieval, storage, and modification. It's also beneficial to mention how the system ensures the security of database communication, such as through encryption or secure connections.

3.7 Other Non-Functional requirements

1. **Usability:** The goal is to make sure users can effectively use the system. Each interface includes a help and information section for users to remotely manage the system by reading its documentation.

2. **Security:** The system requires users to log in with a password containing more than eight characters to prevent unauthorized access. Only authorized corporate users with active usernames and passwords can access the secured pages. The system ensures high security and data integrity with multiple security keys, refreshing the system for 100% security and no data loss.

3. **Performance:** The system must respond to user instructions within 10 seconds. It should operate quickly, responding in half a second for complex tasks and a few seconds for simpler ones, aiming for expected performance in milliseconds.

4. **Availability:** The system must maintain continuous availability, 24 hours a day, 7 days a week. In the event of a major malfunction, any necessary repairs or fixes must be promptly executed during standard working hours. This is to ensure that the impact on business operations is minimized, and the system remains consistently recognizable and accessible.

5. **Error Handling:** The system must employ measures to minimize occurrences of errors. Additionally, users encountering errors should receive confirmation of error corrections to aid in a seamless recovery process. The system should autonomously handle errors, reducing the typical time needed for recovery to milliseconds.

6. **Ease of Use:** The system must feature a user-friendly interface designed to accommodate users with diverse levels of understanding. The interface should be characterized by simplicity, high quality, and minimal training requirements. This ensures that users can interact with the system effortlessly, regardless of their familiarity with the system, promoting a positive user experience.

7. **Maintainability:** Users should easily reset all optional and stored variables to default settings. The system should be easy to handle, with auto-maintenance using stored user data for information.

8. **Portability Requirements:** The software can be easily adapted to different networks or operating systems. This characteristic simplifies software distribution in any scenario, possibly involving the use of software development tools during product development.

9. **Data Backup and Recovery:** The system must regularly perform backups of its data to mitigate the risk of data loss. In the event of a system failure, there must be a well-defined and efficient data recovery process that ensures the restoration of data to its latest state within an acceptable timeframe.

10. **User Support:** The system must provide accessible and effective user support channels to address user queries and issues promptly. These support channels should include online help documentation, live chat, email support, or a dedicated customer service hotline, ensuring users can easily seek assistance and receive timely and helpful solutions.

11. **Multilingual Support:** If the system caters to a diverse user base, it must support multiple languages. This entails ensuring that the user interface, system documentation, and any communication from the system are available in more than one language. The goal is to promote inclusivity, facilitate user understanding, and enhance the user experience for individuals with different language preferences.

4. Group Tasks

4.1 Environmental Model Specification

4.1.1 Context Diagram

A context diagram in software engineering is a succinct visual overview of a system, illustrating its boundaries and interactions with external entities. It uses a central box for the system, surrounded by external entities connected by arrows denoting data flow. This high-level representation aids communication among stakeholders, providing a snapshot of the system's scope without delving into technical specifics(Van Hentenryck, Saraswat and Deville, 1998).

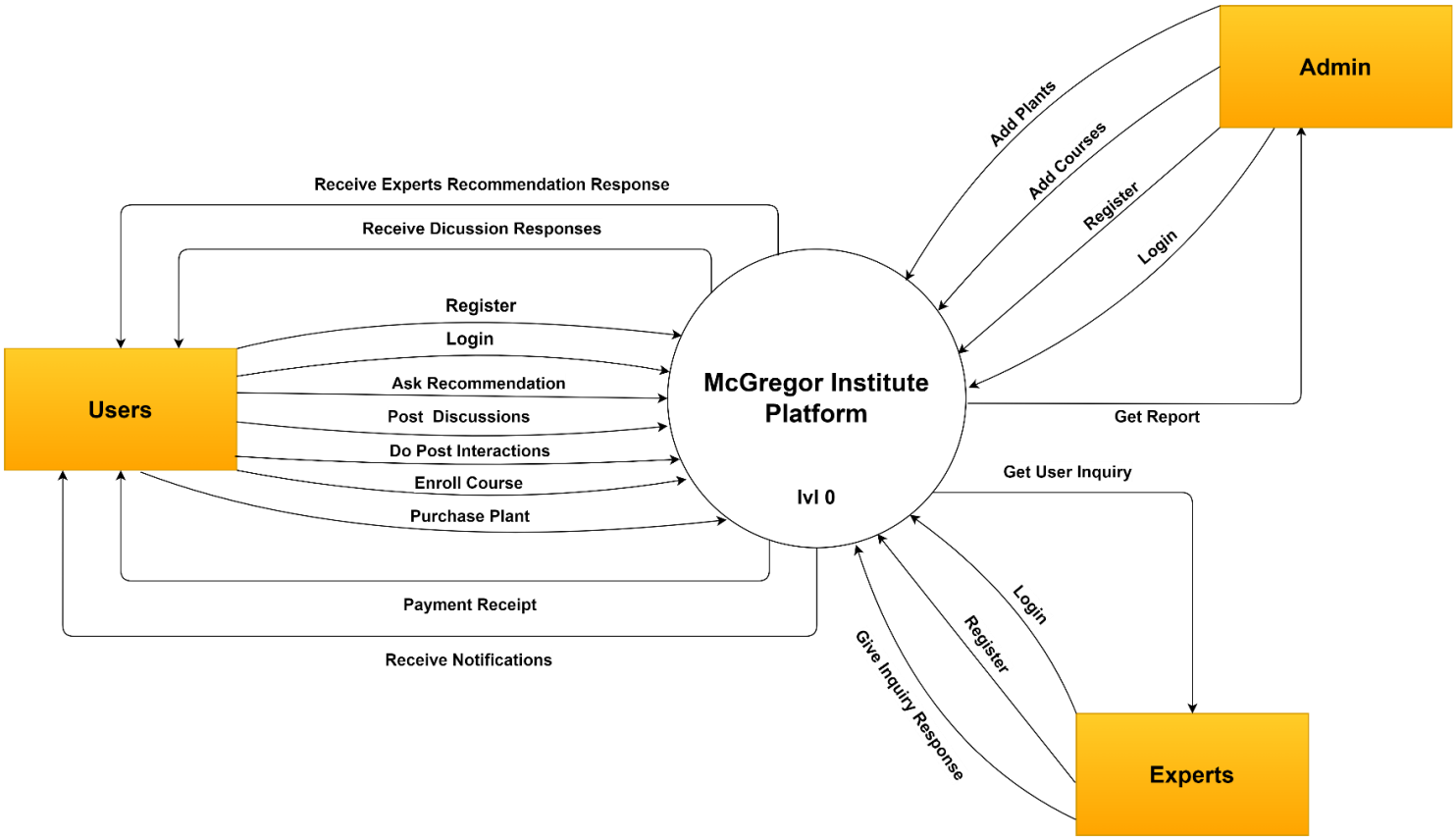


Figure 2: Context Level Diagram

Description:

In our context diagram for the McGregor Institute Platform, three external entities are identified: users, admins, and experts.

Users have the capability to perform the following functions:

- Register in the system
- Log in to the system
- Request recommendations from experts
- Initiate discussions on the forum
- Engage in interactions on forum posts
- Enroll in any available courses
- Purchase plant products
- Make payments for both courses and plant purchases

Admins have the capability to perform the following functions:

- Add plants to the system
- Include course details in the system
- Register in the system
- Log in to the system
- Generate reports

Experts have the capability to perform the following functions:

- Register in the system
- Log in to the system
- Provide inquiry responses

In summary, users, admins, and experts register and log in to the system. Users have ability to enroll in courses, purchase plants, participate in forum discussions, seek recommendations, and complete payments. Admins contribute by adding course and plant details for purchase and generating reports. Additionally, experts engage with users, offering responses to their inquiries

4.1.2 Level 1 Data Flow Diagram

A Level 1 Data Flow Diagram (DFD) serves as a comprehensive illustration that delves into the intricacies of a system's operations. It goes beyond the broad strokes of a context diagram, breaking down high-level processes into detailed subprocesses, elucidating the role of data stores in storing information, and explicitly showcasing the dynamic flow of data between different system components. The use of circles to represent processes, rectangles for data stores, and arrows for data flow ensures a visually intuitive representation. This diagram acts as a crucial intermediate step in system design, providing stakeholders with a more profound insight into the internal mechanics of the system, thus facilitating a more informed discussion about how data is processed and managed within the various components(Van Hentenryck, Saraswat and Deville, 1998).

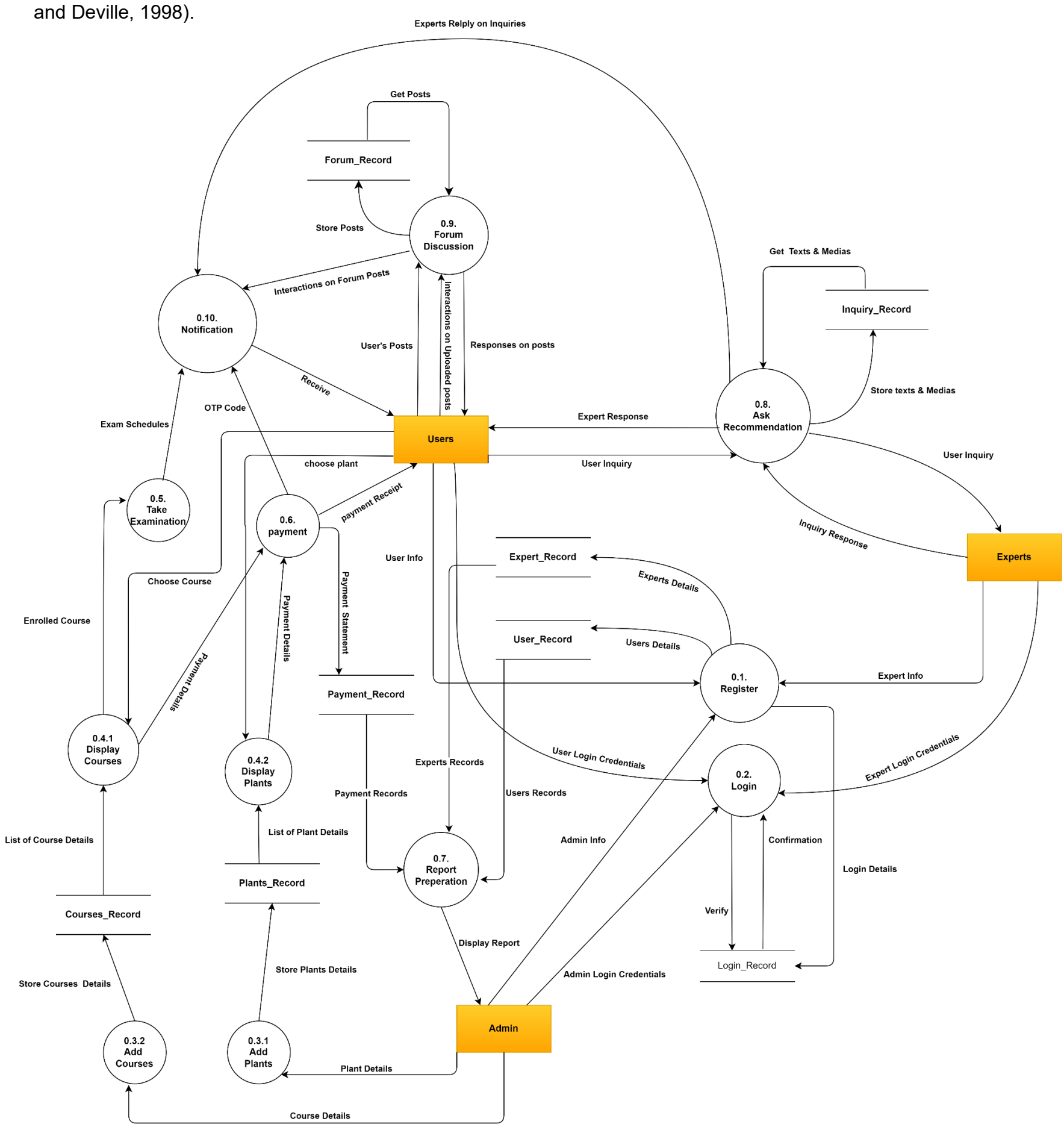


Figure 3: Level one Data Flow Diagram of Overall System

Description:

In our Level 1 DFD for the McGregor Institute Platform, we have delved into the processes to provide a detailed understanding of how our system operates.

- Users, admins, and experts register and log in to the system. User and expert details are stored in the user_record and experts_record databases, respectively. This information is later utilized for report generation, accessible to the admin. Credential information for users, admins, and experts is stored in the database for login validation.

- Admins contribute plant and course details, which are added to the plant_records and course_records databases through the "add plants" and "add courses" processes. The system displays plants and courses through the "display plants" and "display courses" processes, retrieving information from the database.

- Admins can obtain a report through the report preparation process. The report includes information about users, experts, and the payment details made by users.

- Users can select plants, enroll in courses, and make payments. User payment details are stored in the payment_record database, crucial for the admin's report.

- Users can engage in examinations after enrolling in courses and participate in forums, where they can post content and interact with others. Forum information, including user posts, is stored in the forum_record database.

- Users can also seek advice and recommendations from experts, stored in the inquiry_record database. Experts retrieve user inquiries from the inquiry_record database, providing responses and suggestions. These interactions are stored in the inquiry_record database, allowing users to access the responses.

- Users receive notifications for successful payments, exam schedules, exam marks, interactions on their forum posts, and expert replies to inquiries.

4.1.3 Level 2 Data Flow Diagram

A Level 2 Data Flow Diagram (DFD) provides a more detailed view than the Level 1 DFD, breaking down subprocesses into finer tasks. It unveils the internal workings of each process, revealing input-output relationships and data transformations. This level of abstraction helps stakeholders gain a more nuanced understanding of the system's functionalities, serving as a crucial bridge between the overview presented in the Level 1 DFD and the detailed specifications found in subsequent levels of the system design(Van Hentenryck, Saraswat and Deville, 1998).

DFD level 2 of Ask Recommendation :

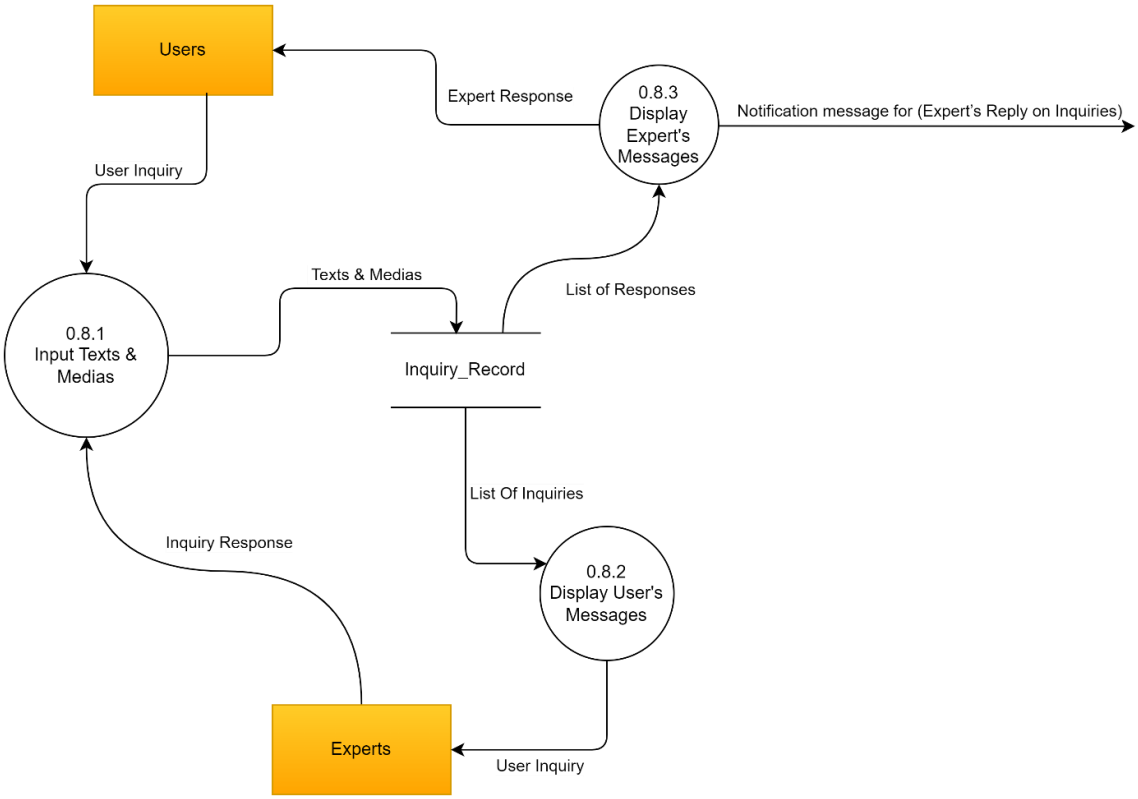


Figure 4: DFD level 2 of Ask Recommendation

DFD level 2 of Forum Discussion:

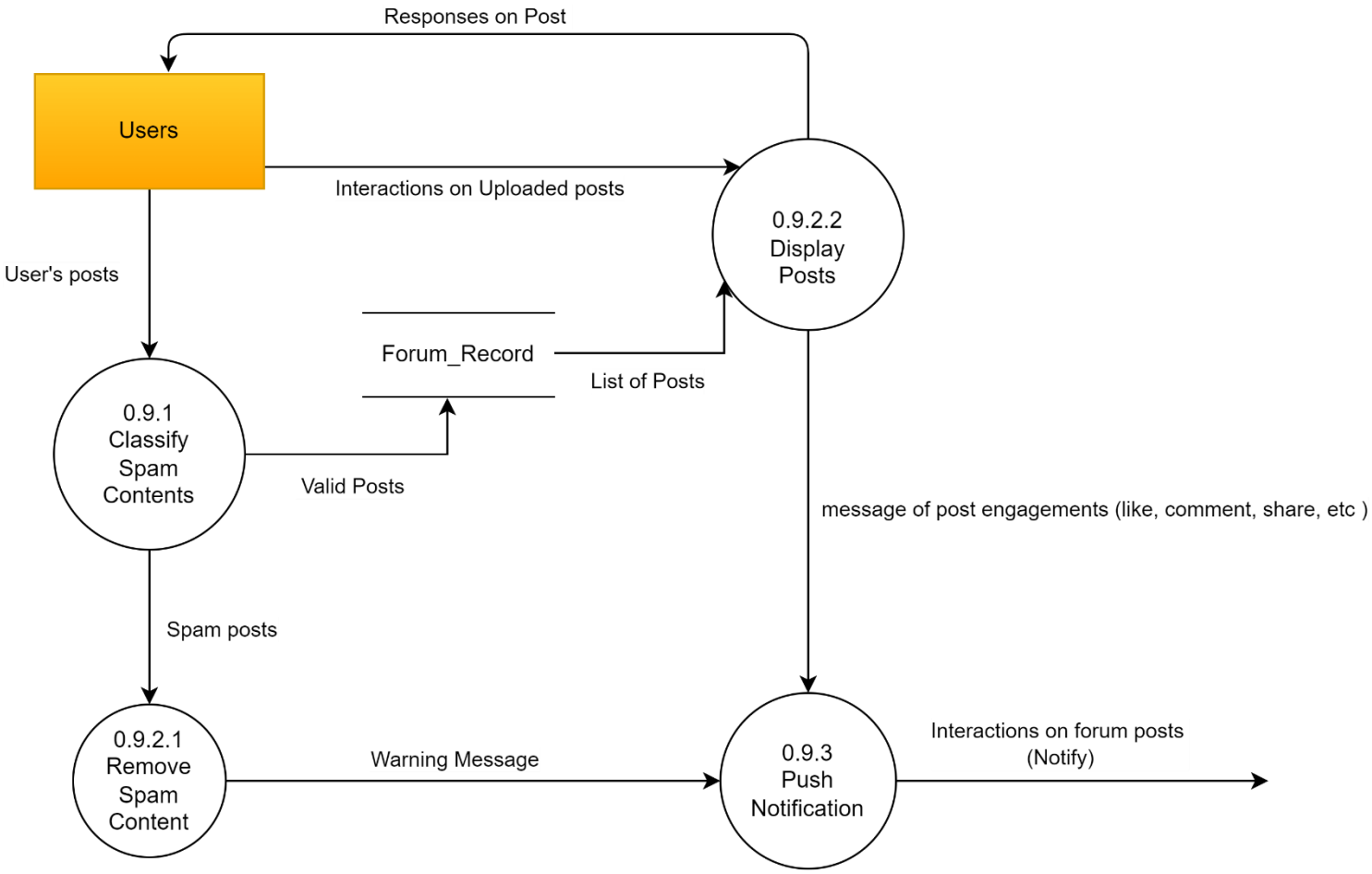


Figure 5: DFD level 2 of Forum Discussion

DFD level 2 of Add Courses:

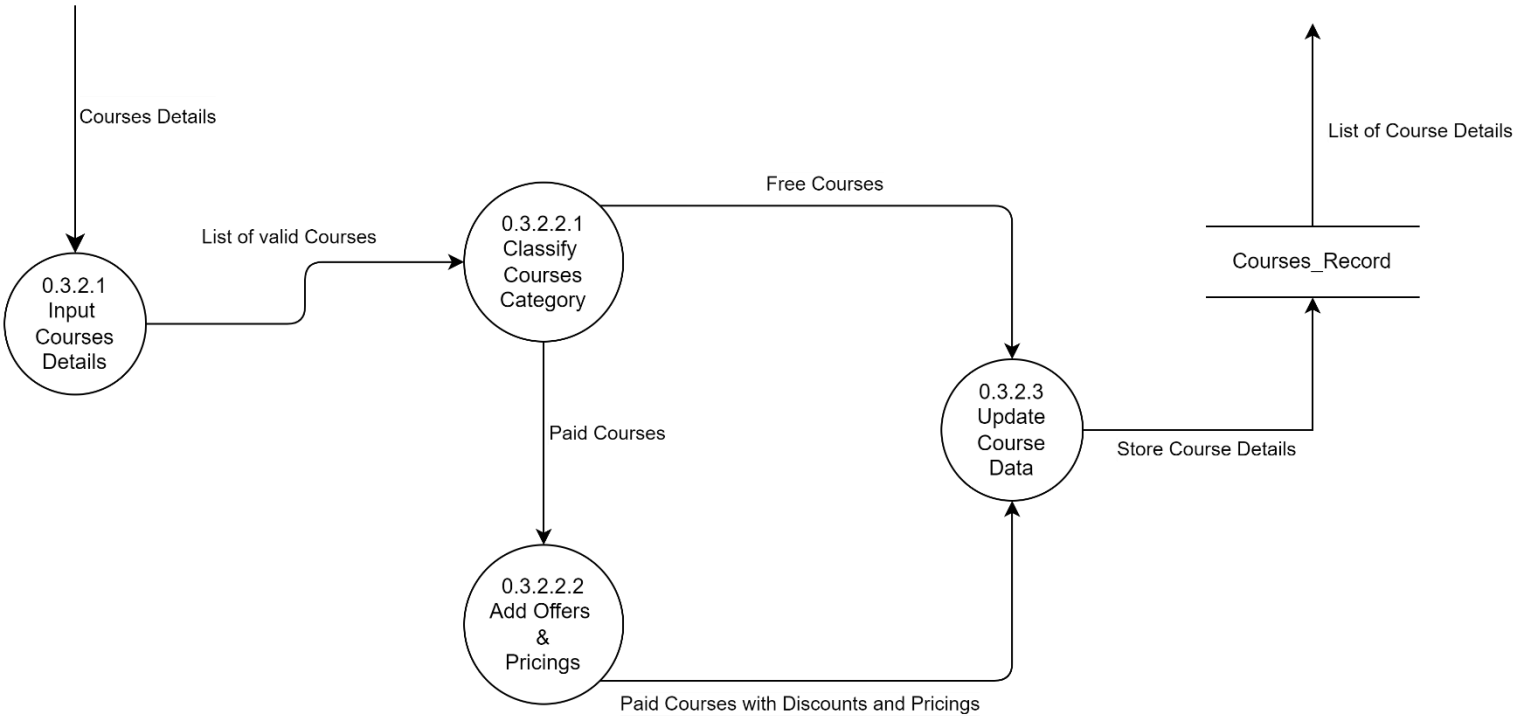


Figure 6: DFD level 2 of Add Courses

4.2 Internal Model Specification

4.2.1 Entity Relationship Diagram (ERD)

ERD stands for Entity-Relationship Diagram. It is a visual representation used in database design to illustrate the relationships between entities (which can be people, objects, concepts, or events) within a system. ERDs use symbols and lines to represent these entities and their relationships. The main components of an ERD include entities (represented by rectangles), attributes (characteristics of entities), relationships (connections between entities), and cardinality (which describes the numerical aspects of the relationships). ERDs help in organizing and understanding the structure of a database, making them a valuable tool in database design and conceptual modeling(Casanova and Amaral de Sa, 1984).

In this Entity-Relationship Diagram (ERD), we have key entities such as Expert Record and Admin, each entity have unique attributes like IDs, names, contact details, and login credentials. Both experts and admins log in through the Login Record, which stores login IDs and emails. The Expert Record also interacts with Inquiry Record, which responds for sending inquiries and connects with User Records for providing inquiries response to users. User Records include user details and it connects with Course Record for enrollment in various courses. Users also engage in forums Forum Record and purchase plants Plant Record, which forms many-to-many relationships. Additionally, payments are recorded in the Payment Record, which creates a one-to-one relationship with Users Record.

Business Case Rules for Database Design for our McGregor Institute Platform :

1. **User Management:** The system must manage detailed user information, including basic details such as name, address, contact, and email. This information is crucial for generating reports for the admin. It is applicable not only to users but also to admins and experts.

2. **Forum Involvement:** The system should capture user participation in forums, including posting content and interactions. A user can engage in multiple forums, and a single forum can involve multiple users, fostering the exchange of opinions and ideas.

3. **Enroll Courses:** The system must handle user enrollment in courses, requiring course details such as course ID, name, price, and the associated user. A user can enroll in multiple courses, and a single course can have multiple users involved.

4. **Purchase Plants Processing:** The system needs to manage user purchases of plants, storing details such as plant name, price, description, and the associated user. A user can buy multiple plants, and a single plant may be purchased by multiple users.

5. **Login System:** The system is responsible for login validation for users, admins, and experts. A secure login process is essential for system access, requiring unique login IDs, passwords, and emails for each user. Each user is assigned a distinct set of login credentials.

6. **Inquiry Management:** The system should facilitate the exchange of inquiries between users and experts. Users can send multiple inquiries, and experts can provide responses to inquiries from various users. This interaction is course-specific, with one expert handling responses for multiple inquiries from a single user.

7. **Payment Processing:** The system must manage payment details for users enrolling in paid courses and purchasing plants. Users can utilize various payment methods, but the system should facilitate a one-time payment only. This information is critical for tracking financial transactions within the platform.

4.2.2 Entity Relationship Diagram (ERD) for whole system

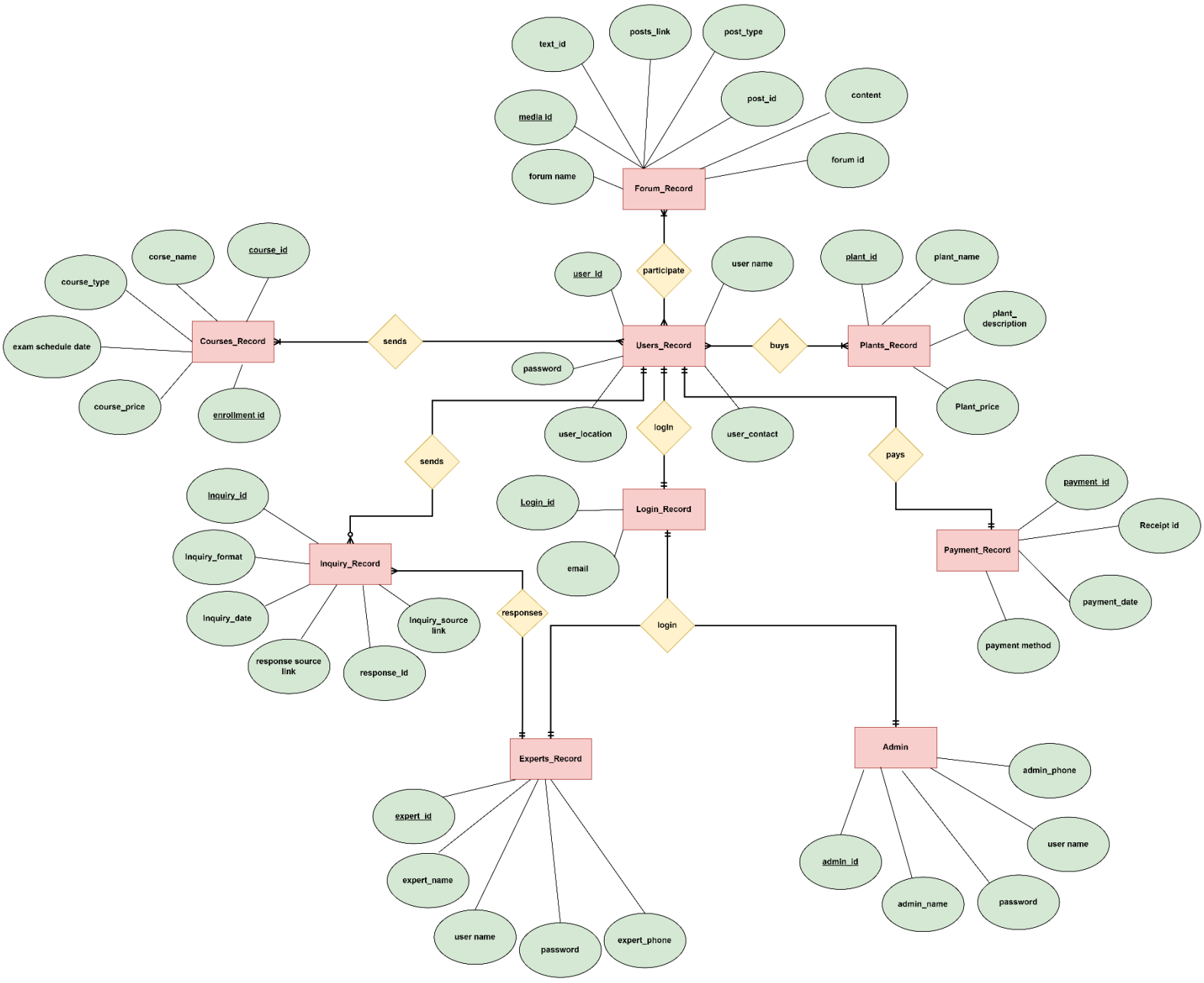


Figure 7: ERD of overall system

Relational Table Diagram for whole system :

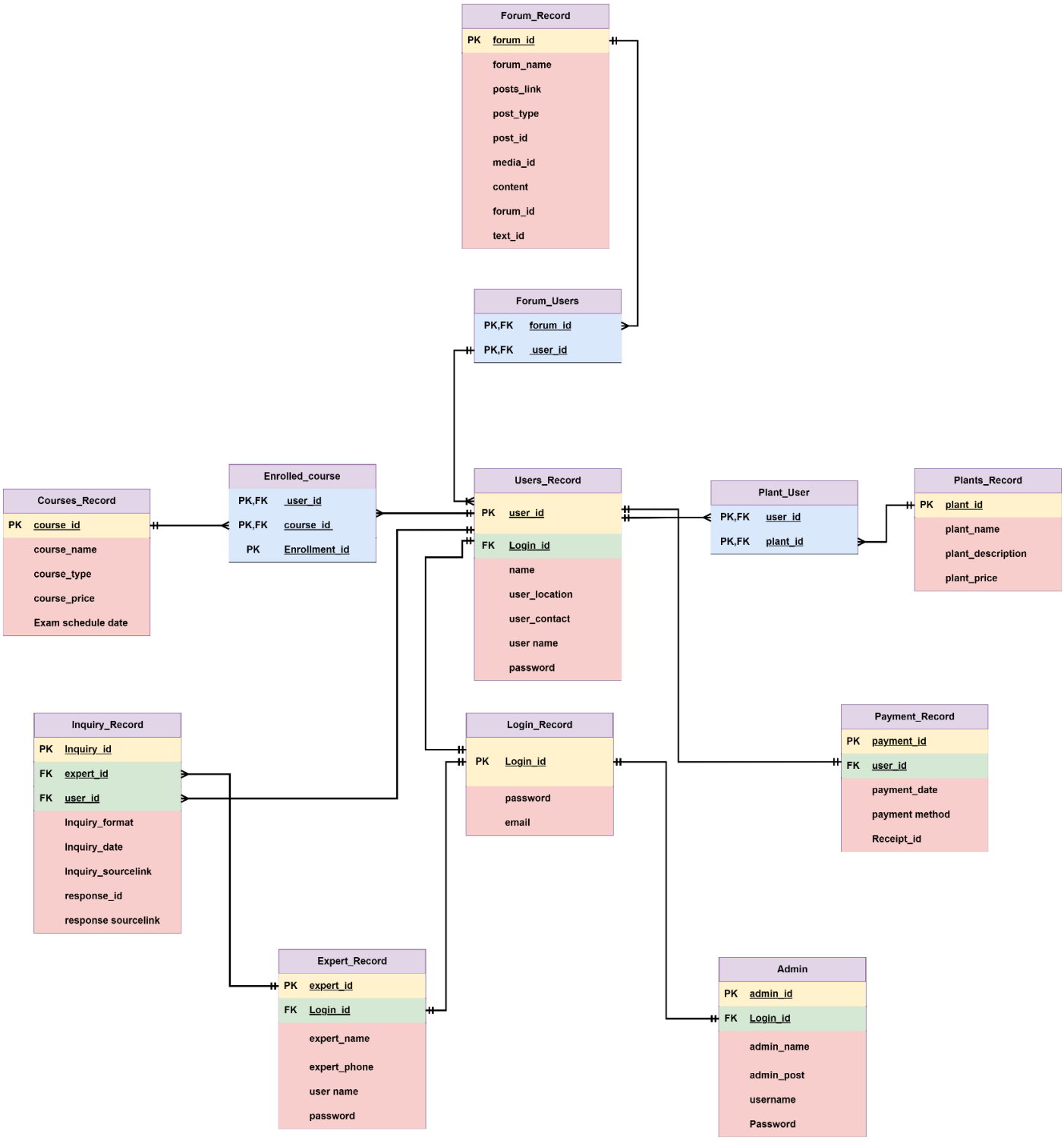


Figure 8: Relational Table Diagram of Overall System

In this relationship diagram, we address many-to-many relationships with bridging tables. For the interaction between users and forums, we introduce Forum Users to connect Users Record and Forum Record. Similarly, to manage enrollments, we use Enrolled Course, which acts as a bridge between Users Record and Course Record, ensuring smooth participation in courses. For plant purchases, Plant User facilitates the many-to-many relationship between Users Record and Plant Record. These bridging entities simplify complex relationships, enhancing the clarity and functionality of the overall system.

4.2.3 Data Dictionary

A data dictionary is a collection of metadata such as object name, data type, size, classification, and relationships with other data assets. Think of it as a list along with a description of tables, fields, and columns. The primary goal of a data dictionary is to help data teams understand data assets (UC Merced Library, n.d.).

Dictionary is an essential component of any relational database because it holds records about objects in the database such as data ownership, data relationship to other objects and other data.

Here are the Notations that are used in Data dictionary:

1. **“+”**: Signifies the combination or concatenation of data items when they are brought together.

2. **“[,]”**: Denotes a selection of data within square brackets, where the listed items separated by commas indicate that any of them can occur.

3. **“()”**: Represents data items enclosed in parentheses, indicating that these items may or may not appear; their presence is variable.

4. **“{}x”**: Indicates data items within curly brackets, suggesting repetition. Here, x represents any number, indicating the frequency of data repetition. The repetition occurs a maximum of x times.

5. **“/ /”**: Resembles coding comments, where anything written between the two asterisks is considered a comment. Comments are explanatory text and do not affect the system's functionality.

1) **Login details:** login id + email + password

   Represents user login information. Login id is an integer, Email is a variable character string (VARCHAR), and Password is also a variable character string.

   - Login id: INTEGER
   - Email: VARCHAR
   - Password: VARCHAR

2) **Confirmation**: command

   Represents a command used for confirming actions or processes within the system.

3) **Verify login record**: command.

   Represents a command used for verifying user login records within the system.

4) **Expert info**: expert id+ expert name + {expert phone}*

Stores information about experts. Expert_id is an integer, Expert name is a variable character string, and Expert phone is a numeric field.

- expert_id: INTEGER
- expert name: VARCHAR
- expert phone: NUMBER

5) **Expert login credentials**:

Stores login credentials for experts. Username is a variable character string, and Password is a variable character string. Username + password

- username: VARCHAR
- password: VARCHAR

6) **Inquiry Response**: Command

Represents a command used for generating responses to user inquiries.

7) **User inquiry**:

Records user inquiries, including unique IDs for inquiries and user IDs.

- inquiry_id: INTEGER
- user_id: INTEGER

8) **Admin Login Credentials**:

Stores login credentials for administrators. Admin_id is an integer, Username is a variable character string, and Password is a variable character string. (admin_id + Username + Password)

- admin_id: INTEGER
- Username: VARCHAR
- Password: VARCHAR

9) **Admin info**:

It contains same parallel information as admin login credentials contains.

10) **User Details**:

Stores user details, including unique IDs, name, contact information, and location.

- user_id + user_name + {user_contact + user_location}*
- user_id: INTEGER
- name: VARCHAR
- user_contact: VARCHAR
- user_location: VARCHAR

11) Expert details:

Contains the same information as expert details for reference.

- Expert details: Records
- Records=expert_id + {expert_name + expert_phone}*
- expert_id: INTEGER
- expert_name: STRING
- expert_phone: VARCHAR

12) Store Text and Medias: text_id + media_id + {Content}

Stores text and media with unique IDs and content information.

- text_id: INTEGER
- media_id: INTEGER
- Content: VARCHAR

13) Get texts and medias:

It also contains same information as store text and medias holds.

14) Expert Response: command

Represents a command used for generating responses from experts.

15) User login credentials:

Stores login credentials for users. User_id is an integer, Username is a variable character string, and Password is a variable character string.

- user_id + username + password
- User_id: INTEGER
- Username: VARCHAR
- Password: VARCHAR

16) Plant details:

Stores information about plants. Plant_id is an integer, Name is a variable Character string, Price is a decimal value, and Description is a variable character String.

- plant_id + Name + Price, Description
- plant_id: INTEGER
- Name: VARCHAR
- Price: DECIMAL
- Description: VARCHAR

17) Course Details:

Stores information about courses. Course_id is an integer, Type is a variable character string, Course name is a variable character string, and Price is a decimal value.

- Course_id + Type + course name + Price
- Course_id: INTEGER
- Type: VARCHAR
- Course name: VARCHAR
- Course name: NUMBER

18) Display Report:

Combines information from payment records, user records, and expert records to Generate a comprehensive display report.

- Payment Record+ User Records + Expert Record

19) Enrolled course:

Records the enrollment of users in courses. Enrollment_id is an integer, User_id is an integer, and Course_id is an integer.

- enrollment + user_id + course_id
- enrollment_id: INTEGER
- user_id: INTEGER
- course_id: INTEGER

20) Payment receipt:

Records information about payment receipts. Payment_id is an integer, User_id is an integer, and Receipt_id is an integer.

- PAYMENT ID + USER ID
- Payment ID: INTEGER
- USER ID: INTEGER
- RECEIPT ID: INTEGER

21) Exam Schedules:

Stores information about exam schedules. Course_id is an integer, and Date is a date type.

- course_id + Date
- course_id: INTEGER
- Date: DATE

22) Notification received: command

23) Responses on posts = Command

24) Store Posts:

Stores information about user posts. Post_id is an integer, and User_id is an integer.

- post_id + user_id
- post_id: INTEGER
- user_id: INTEGER

25) Users Records:

This is the database which contains the same information as user details for reference.

- user_id + user_name + {user_contact + user_location}*
- user_id: INTEGER
- Login id: INTEGER
- user_contact: VARCHAR
- user_location: STRING

26) Forum Records:

Stores information about forum posts. Forum_id is an integer, Posts_link is a variable character string, Post_id is a number, Post_type is a variable character string, and Forum_name is a variable character string.

- forum_id + posts_link + post_id + post_type + {forum_name}*

- forum_id: INTEGER
- Post id: NUMBER
- forum_name: STRING
- posts_link: VARCHAR
- post_type: VARCHAR

27) Course Details:

This is a database which contains the same information as course details for references.

- course_id + course_name + course_type + course_price
- course_id: INTEGER
- course_name: STRING
- course_type: VARCHAR
- course_price: DECIMAL

28) Plant Record:

This is also the database for plant which contains the same information as plant details for reference.

- plant_id + plant_name + plant_description + plant_price
- plant_id: INTEGER
- plant_name: STRING
- plant_description: VARCHAR
- plant_price: DECIMAL

29) Payment Record:

Records information about payments. Payment_id is an integer, Payment_date is a date type, and Payment_method is a variable character string.

- payment_id + payment_date + payment_method
- payment_id: INTEGER
- payment_date: DATE
- payment_method: VARCHAR

30) Login Record:

Records login information for users. Email is a variable character string,

User_name is a variable character string, and Password is a variable character string.

- Email + user_name + password
- Email: VARCHAR
- user_name: STRING
- Password: VARCHAR

31) Inquiry Record: Records

This is also a database which contains the same information as user inquiries for reference.

- Records = inquiry_id + inquiry_format + inquiry_date + response_sourcelink +
- response_id + inquiry_sourcelink
- inquiry_id: INTEGER
- inquiry_format: VARCHAR
- inquiry_date: DATE
- response_sourcelink: VARCHAR
- response_id: INTEGER
- inquiry_sourcelink: VARCHAR

4.2.4 Process specifications (Pspecs) for elementary processes

4.2.4.1 Process A

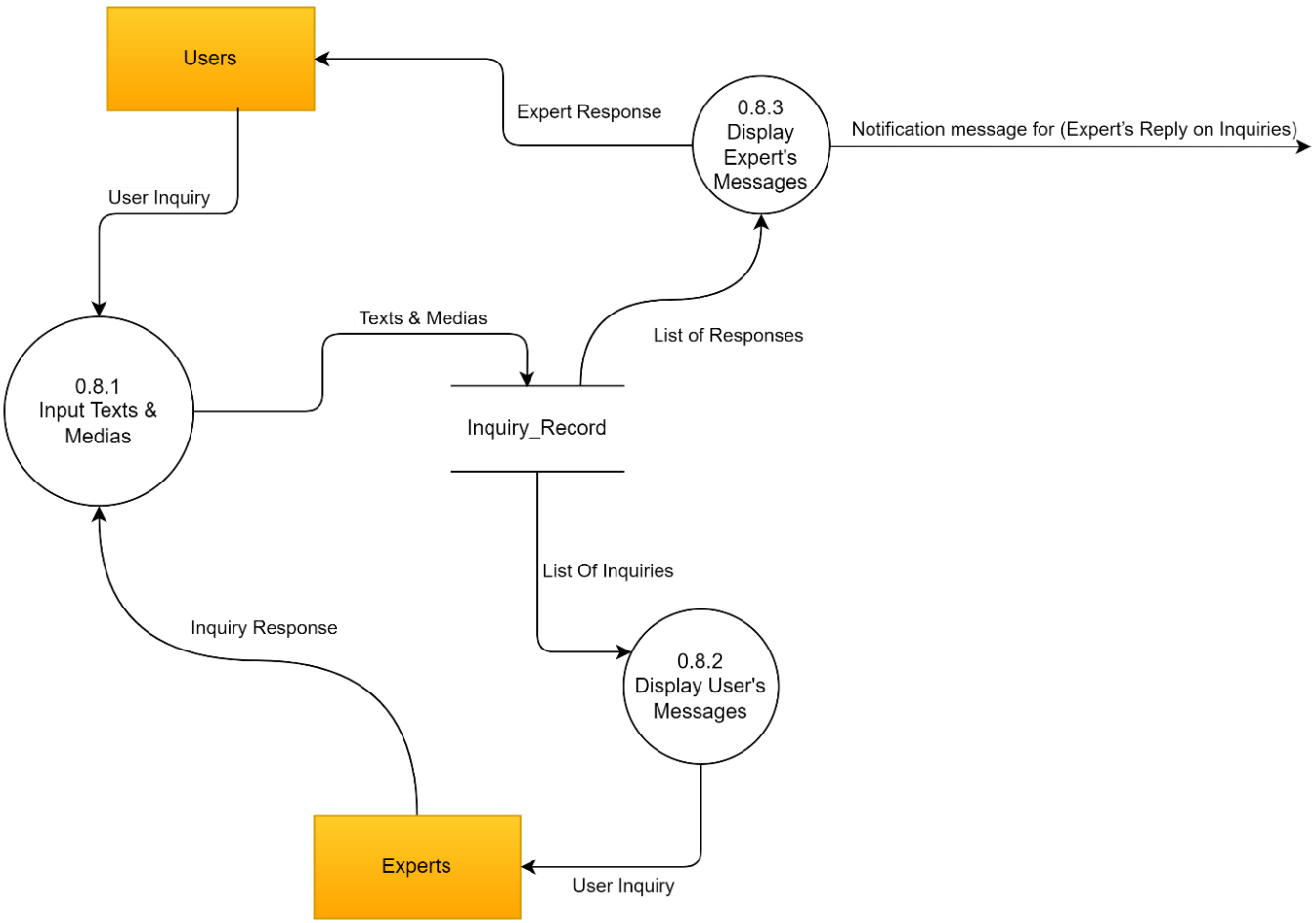


Figure 9: Pspecs of DFD level 2 of Ask Recommendation

Number: 0.8.1

a) ***Model Name :*** Input Texts & Medias

b) ***Description:*** This process takes user input and experts answers, which may be in the form of texts, audio, video, or images and classifies its category then stores it in the database.

c) ***Input Data Flow:*** User Inquiry, Inquiry Response

d) ***Out Data Flow:*** Text & Medias

e) ***Process Logic:*** The logic for this process is as follows:

1. User inquiries and Experts answers, presented in various formats such as texts, audio, video, or images, are received.

2. The inquiry & response message is subsequently categorized based on the type of file, distinguishing between texts, audio, video, and images.

3. Following the classification, the file is stored in the database.

Number:0. 8.2

a) ***Model Name :*** Display User's Messages

b) ***Description:*** This process retrieves user inquiries from the database and displays them to the experts.

c) ***Input Data Flow:*** List of Inquiries

d) ***Out Data Flow:*** user inquiry

e) ***Process Logic:*** The logic for this process is as follows:

1. User inquiries, stored in the database, are retrieved.

2. The inquiry message is displayed to the experts.

Number: 0.8.3

a) ***Model Name :*** Display Expert's Messages

b) ***Description:*** This process retrieves experts responses from the database and displays them to the users.

c) ***Input Data Flow:*** List of Responses

d) ***Out Data Flow:*** Expert Response, Notification message for (Expert's Reply on Inquiries)

e) ***Process Logic:*** The logic for this process is as follows:

1. Experts responses, stored in the database, are retrieved.

2. The response message is displayed to the user.

4.2.4.2 Process B

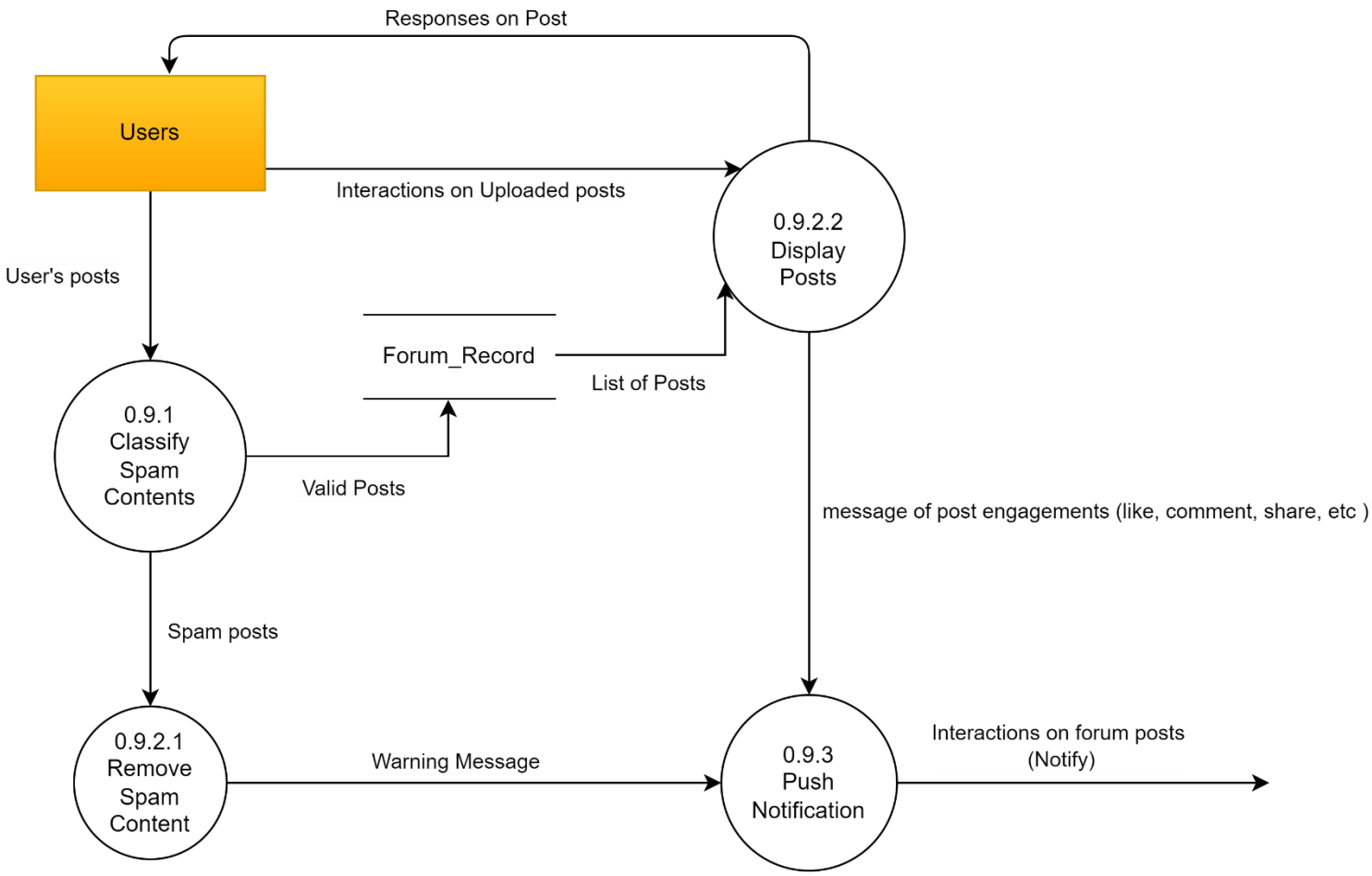


Figure 10: Pspecs of DFD level 2 of Forum Discussion

Number: 0.9.1

a) ***Model Name :*** Classify Spam Contents

b) ***Description:*** This process takes user posts as input, filters them to determine whether the post is spam or not. If spam is found, the post is sent to a further process for removal. If no spam is found, it is stored in the database.

c) ***Input Data Flow:*** User's posts

d) ***Out Data Flow:*** Spam posts, Valid Posts

e) ***Process Logic:*** The logic for this process is as follows:

1. User posts are received as input.

2. The system initiates a machine learning-based filtering process to determine whether the post is spam or not.

3. If the machine learning model identifies the post as spam during the filtering process, it is forwarded to a subsequent process designed for the removal of spam content.

4. If the post is not classified as spam by the machine learning model, It is then stored in the database.

Number: 0.9.2.1

a) ***Model Name :*** Remove Spam Content

b) ***Description:*** This process receives spam posts identified by a machine learning-based filtering process. It removes the spam post and sends a warning message to the user about the invalid content.

c) ***Input Data Flow:*** Spam posts

d) ***Out Data Flow:*** Warning Message

e) ***Process Logic:*** The logic for this process is as follows:

1. The process receives spam posts identified by a machine learning-based filtering process.

2. The system initiates the removal of the identified spam post from the system.

3. A warning message is generated to inform the user about the invalid content in their post.

Number: 0.9.2.2

a) ***Model Name :*** Display Posts

b) ***Description:*** This process retrieves valid non-spam posts stored in the database and displays them on the system for interactions and engagements of other users. If another user interacts with the post, it also generates a notification message for the original user who created that post.

c) ***Input Data Flow:*** Interactions on Uploaded posts, List of Posts

d) ***Out Data Flow:*** Responses on Post , message of post engagements (like, comment, share, etc)

e) ***Process Logic:*** The logic for this process is as follows:

1. The process retrieves valid non-spam posts stored in the database.

2. The retrieved non-spam posts are displayed on the system for interactions and engagements by other users.

3. If another user interacts with a displayed post (e.g., likes, comments, shares), the system captures the interaction.

4. A notification is sent to the post creator, summarizing interactions (likes, comments) and providing details about the engaging user.

Number: 0.9.3

a) ***Model Name :*** Push Notification

b) ***Description:*** This process retrieves warning messages and post interactions messages generated when a user posts spam contents or when someone interacts with a post created by a user. It then sends these messages to the notification module to inform the user through their respective accounts, emails, and phone numbers.

c) ***Input Data Flow:*** Warning Message, message of post engagements (like, comment, share, etc)

d) ***Out Data Flow:*** Interactions on forum posts (Notifications)

e) ***Process Logic:*** The logic for this process is as follows:

1. The Process retrieve warnings messages and post engagements messages related to spam content and user interactions with posts.

2. Forward retrieved messages to the notification module to inform users through their respective platform accounts, email addresses, and phone numbers.

4.2.4.3 Process C

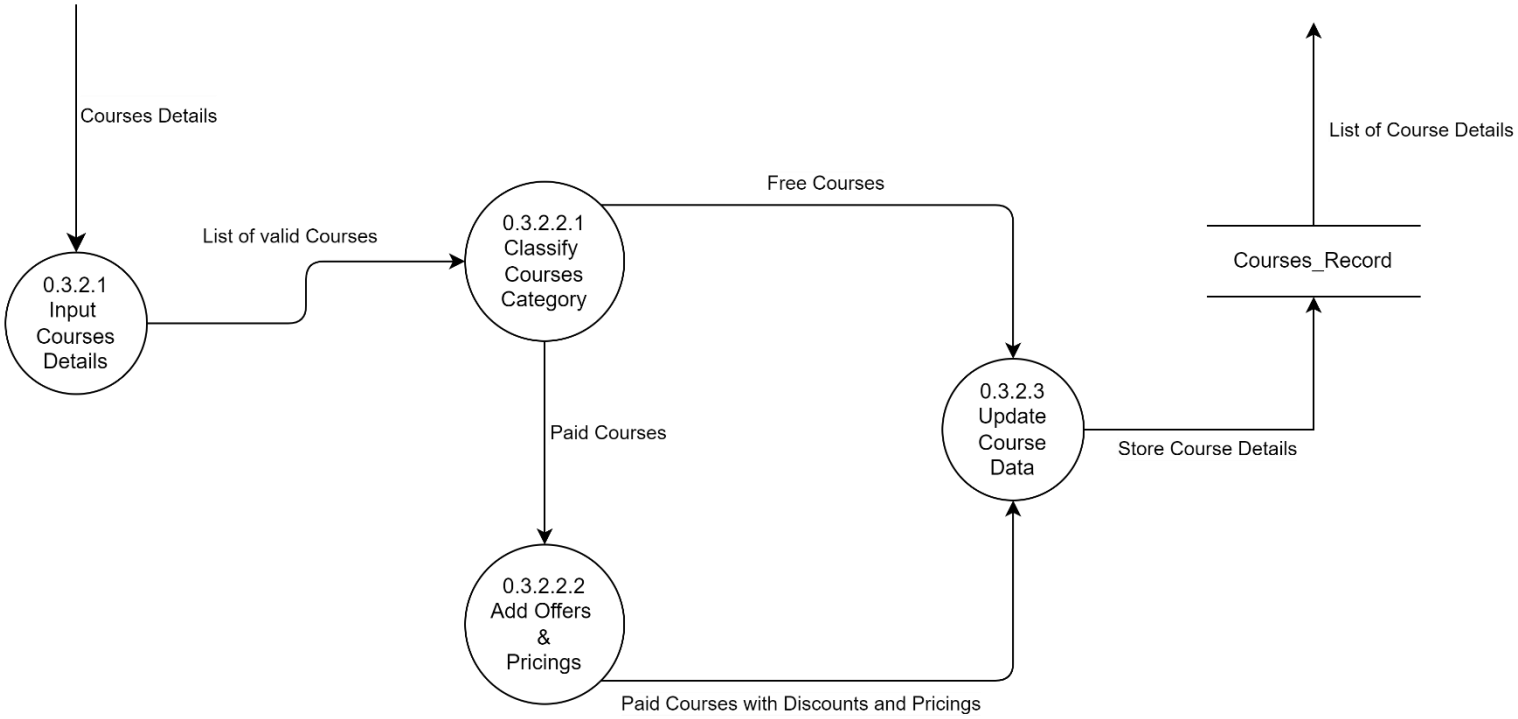


Figure 11: Pspecs of DFD level 2 of Add Courses

Number: 0.3.2.1

a) ***Model Name :*** Input Courses Details

b) ***Description:*** This process takes the course details as input from the admin and validates the course, covering content, structure, and objectives related to botany. After validation, it sends a list of courses to another process for the classification of categories—free or paid courses.

c) ***Input Data Flow:*** Courses Details

d) ***Out Data Flow:*** List of valid Courses

e) ***Process Logic:*** The logic for this process is as follows:

1. This Process Receive course details from the administrator, encompassing content, structure, and objectives.

2. Validate the course contents by performing checks tailored to botany, ensuring alignment with the subject matter and verifying relevant elements.

3. Generate a list of validated courses, forward it to the next process, for category classification.

Number: 0.3.2.2.1

a) ***Model Name :*** Classify Courses Category

b) ***Description:*** This process takes the list of valid courses and classifies them as free or paid courses. If the course is free, it sends its details to be stored in the database directly. If the course is paid, the course details are sent to another process for adding offers and pricings.

c) ***Input Data Flow:*** List of valid Courses

d) ***Out Data Flow:*** Paid Courses, Free Courses

e) ***Process Logic:*** The logic for this process is as follows:

1. This Process Receive the list of valid courses and classify them as free or paid.

2. For free courses, directly store their details in the database.

3. For paid courses, send their details to another process dedicated to adding offers and pricings before storing the comprehensive information in the database.

Number: 0.3.2.2.2

a) ***Model Name :*** Add offers & Pricings

b) ***Description:*** This process deals with courses categorized as paid. It adds offers to the courses based on customer demand and also includes pricings for the courses.

c) ***Input Data Flow:*** paid courses

d) ***Out Data Flow:*** Paid Courses with Discounts and Pricings

e) ***Process Logic:*** The logic for this process is as follows:

1. This Process Receives the courses categorized as paid.

2. After Receiving Courses it add offers to the paid courses, considering customer demand and preferences.

3. It also Incorporate pricings for the courses based on the established offers and customer-oriented considerations.

Number:0.3.2.3

a) ***Model Name :*** Update Course Data

b) ***Description:*** This process gets the details of both free and paid courses and sends this data to the database for storage.

c) ***Input Data Flow:*** Paid Courses with Discounts and Pricings

d) ***Out Data Flow:*** Course Details

e) ***Process Logic:*** The logic for this process is as follows:

1. Gather comprehensive details for both free and paid courses.

2. Integrate and organize the collected information into a unified dataset.

3. Transmit the consolidated course data to the designated database for secure and organized storage, ensuring accessibility and reliability for future use and analysis.

4.3 Design Specification

4.3.1 Structure Chart

A structure chart is a visual representation of the modular structure of a software system. It is a hierarchical chart that partitions the system into different levels of abstraction, allowing for a clear understanding of the system's functionality and organization(Benedusi, Cimitile and De Carlini, 1992).

The key components and concepts associated with a structure chart are as follows:

- **Black Boxes:**

  In a structure chart, modules or components are treated as black boxes. This means that the functionality of each module is known to the users (or other modules that interact with it), but the internal details are hidden.

  Inputs are provided to these black boxes, and the boxes generate appropriate outputs based on their internal processes.

- **Top-Level Modules:**

  Modules at the top level of the structure chart are those that provide the main functionalities of the system. They are also referred to as high-level modules. As we move down the chart, modules become more detailed and specific, breaking down the overall system into smaller, more manageable components.

- **Conditional Call:**

  This represents a scenario where a control module can choose between different sub-modules based on certain conditions. It allows for decision-making within the system.

- **Loop (Repetitive Call of Module):**

  A curved arrow in a structure chart indicates a loop, representing the repetitive execution of a module by a sub-module. This is used to depict scenarios where a certain process needs to be repeated multiple times.

- **Data Flow:**

  Data flow arrows represent the movement of data between modules. They show how information is passed from one module to another. Data flow arrows typically have an empty circle at the end.

- **Control Flow:**

  Control flow arrows represent the flow of control or sequence of execution between modules. They indicate the order in which modules are executed. Control flow arrows typically have a filled circle at the end.

- **Physical Storage:**

> This represents the physical location where information is stored. It could be databases, files, or any other form of storage. In a structure chart, physical storage is not represented as a module but as a separate entity. (geeksforgeeks, 2024)

The structure chart outlines the modular organization of the McGregor Institute of Botanical Training's system. At the top level, the main module oversees key functionalities such as user registration, program enrollment, plant purchase, payment processing, recommendation requests, report preparation, certification exams, forum discussions, and notifications.

Each major function is broken down into sub-modules, which creates a hierarchical structure. For example, the user registration process involves fetching details, validating input, and logging in the user. Plant purchase follows a sequence of actions, including browsing, cart management, bill generation, and payment processing.

The ask recommendation process manages inquiries, expert advice, and user interactions, while forum discussions cover spam classification, user interactions, and post displays. Notifications are integrated across various databases, providing updates on payments, expert responses, forum activities, and exam-related information.

The certification exam process fetched all the input of exam requirements, user participation, result generation, and storage. Report preparation compiles data from user, expert, and payment records, processes it, and presents detailed reports.

From our point of view, this structure chart is a valuable tool for system design, understanding the flow of information, and documenting the McGregor Institute's diverse set of functionalities. It aids in creating a cohesive and user-friendly system for the institute's varied operations.

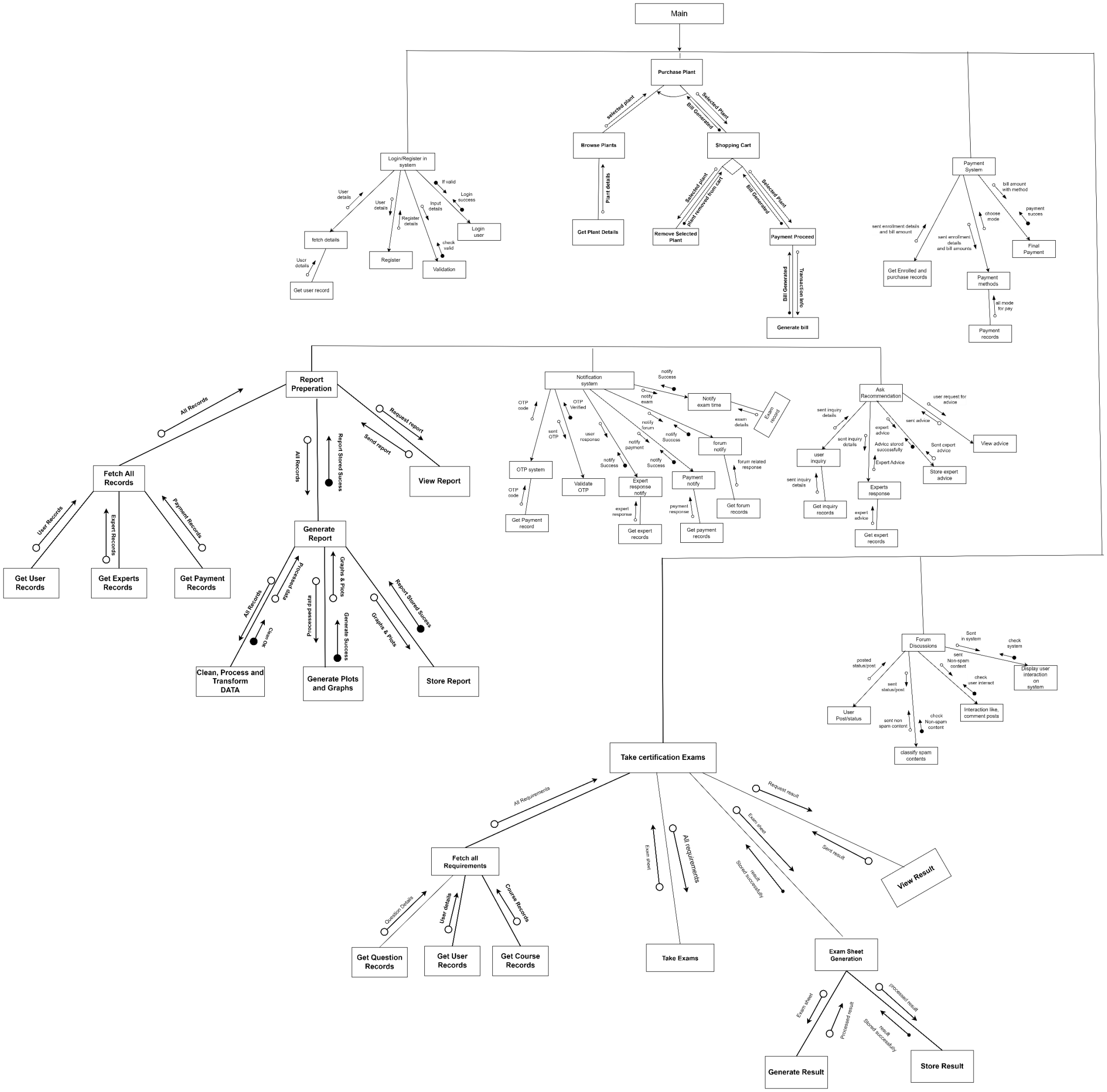


Figure 12: Structure chart for the whole system.

4.4 Assignment Diary

4.4.1 Assumptions

The assumption for database to store data are as follows:

a) Admin Record:

- Admin needs to login to the system to get access to the report preparation interface.
- Admins have access to all system functionalities.
- Admins can update, add, or remove information related to users, experts, forums, courses, plants, payments, logins, and inquiries also generate report.

b) Expert Record:

- Experts are individuals with specialized knowledge in plants and horticulture.
- Experts can provide recommendations in response to user inquiries which can be text/messages and media files.
- Experts can engage in discussions on forums and share their insights.

c) Users Record:

- Users need to register and login first in order to enroll in courses and purchase plants except seeing displayed items like course and plants.
- Users can join various programs, purchase plants, make payments, enroll in courses, participate in forums, and receive notifications.

d) Forum Records:

- Forums serve as a platform for users and experts to discuss plant-related topics.
- Users can create posts, comment, and upvote others' contributions.
- Forums are categorized into different topics for organized discussions.

e) Course Record:

- Courses are available for users to enroll in, both paid and unpaid.
- Users can take certification exams after fulfilling prerequisites i,e choosing. courses.
- Course details, including type, name, and price, are recorded.

f) Plant Records:

- Various plant varieties are available for users to view and purchase.
- Users can add plants to their carts and buy them in bulk.
- Plant details, such as name, description, and price, are recorded.

g) Payment Records:

- Payments are recorded for plant purchases, course enrollments, and other transactions.
- Payment details, including date and method, are stored for financial tracking.

h) Login Records:

- Login records help verify the authenticity of users and admins.
- Admins and users have login credentials recorded for system access.

i) Inquiry Records:

- Users can submit inquiries to experts.
- Inquiries include details like inquiry format, date, response source link, and inquiry source link.
- Experts respond to inquiries, and the interaction details are recorded.
- Inquiries include sending text, message, and media files to experts by user.

4.4.2 Inconsistencies

Inconsistencies refer to conflicts, contradictions, or discrepancies within a system's specifications, requirements, design, or implementation. These inconsistencies can lead to misunderstandings, errors, and potentially serious issues during the development process or in the deployed software.

1. **User Role Constraints:**

   Different user roles (e.g., administrator, expert, regular user) have distinct access levels and permissions within the system.

2. **Plant and Course Selection Constraint:**

   Users must choose at least one plant or course during the selection process, and the system prompts users if selections are missing.

3. **Post Content Constraints:**

   Media content uploaded by users follows acceptable use policies and may be subject to review.

4. **Data Privacy Constraints:**

   User personal information, especially sensitive data like passwords, is stored securely and follows data protection regulations.
   Access controls are in place to restrict unauthorized access to sensitive user data.

5. **Unique Constraints:**

   All primary key fields (e.g., user_id, forum_id) are unique, ensuring the integrity of the data.

6. **Foreign Key Constraints:**

   Foreign key constraints are enforced to maintain referential integrity between related tables.

7. **Data Validation Constraints:**

   Validation checks are in place to ensure that data types (e.g., INTEGER, VARCHAR) match the defined schema.

8. **Password Security Constraint:**

   Passwords must meet security requirements, such as a minimum length and inclusion of alphanumeric characters.

9. **Payment Amount Constraint:**

   Payment amounts must be positive and represent valid currency values.

10. **Enrollment Constraints:**

    Users can enroll in courses only if the courses are currently available and have open slots.

4.4.3 Group members and responsibility

The group member description along with the roles and responsibilities are tabulated below:

Group Member	Responsibilities
Apil Thapa (Project Manager)	• Individual task completion for " Take Certification Exam". • Work on the data dictionary for the entire system. • Oversee the completion of the System's Data Flow Diagram (DFD). • Complete the Entity-Relationship Diagram (ERD) for the entire system.
Miraj Deep Bhandari (Data analyst)	• Individual task completion for " Report Preparation". • Researched and write on the topic of Structure Chart for the documentation. • Help in implementation of Structure Chart to other members as well. • Completed project charter.
Jenish Katuwal (Backend developer)	• Individual task completion for "Make Payment.

	• Assist in the completion of context level diagram. • Created meeting log. • Help in creating main report.
Rijan Poudel (Front-end developer)	• Individual task completion for" Join the program". • Assist in the completion of Relational diagram. • Research and write on the topic of Process Specification. • Research and writing on Structure Chart topics for documentation.
Arbit Bhandari (UI/UX Designer)	• Individual task completion for " Purchase Plant". Complete the Software Requirements Specification (SRS). • System report documentation. • Assist in the completion of the assignment diary.

Table 4: Group members and responsibility table

4.4.4 Group Meetings

Meeting log -1

Day – 1 **DATE 12-12-2023**

Start Time – 01:00 **End Time 02:30**

Location : Brit Cafe

Topic Discussed : On day one, we got the whole team together for a brainstorming session. We dive straight into the coursework question, dissecting it to uncover the essentials. What was expected of us, what tools and resources we needed, and how we could pull it all off within the deadline. By the end of the session, we had a proper understanding of the course work and a clear roadmap to success. First, the meeting truly set the stage for our collaborative journey.

Accomplishment : Identifying potential challenges in our coursework was crucial, from drawing Data Flow Diagrams (DFD) to Writing Software Requirements Specifications (SRS) and other topics. We took some time to think about which is the software that would be the most helpful for our project. Thanks to our cooperative group, we sorted through these challenges, determining the right approaches and solutions. Ultimately, we successfully figured out how to complete our tasks within the given time period.

Problem : So, after all that planning, it was time to start and get to work. But two problems arise. First, who would do which part of the course work and the second problem was how we would communicate with each other through which medium these two were the major issues.

Tasks for Next Meeting : Gathering as much information about how to start introduction and SRS.

Meeting log -2

Day – 2 **DATE 13-12-2023**

Start Time – 11 :00 **End Time 01:00**

Location : Learning Zone.

Topic Discussed : We talked about writing introductions, sharing tips and tricks for the best start and each of us presented our ideas for the give questions SRS, then discussed them together to choose the best approach.

Accomplishment : Started documentation and completed both Introduction and on the second day SRS.

Problem : At the beginning SRS was confusing to all of us, then we met our tutor then it was clear that how should we approach and finally we solved the problem by discussing in the group.

Tasks for Next Meeting : Dept research of Data Flow Diagram.

Meeting log -3

Day – 3 **DATE 14-12-2023**

Start Time – 01 :00 **End Time 03:00**

Location : Chiya Chautari.

Topic Discussed : On the third day , we discussed the issue of DFD after clearing some doubts by our tutor. We explained the concept to each other to make sure everyone was on the same page. Then, we each started building DFD.

Accomplishment : We are finally able to make DFD of different levels starting from zero as per the course work requirements.

Problems : As we level of coursework keep increase it was hard to take everyone on the same page though we some how managed it. As DFD level 0, 1 and 2 have been completed but we were not sure about it so we ask our tutor.

Task For Next Meeting : Group was divided into two portions one should collect more details and information regarding DFD and for other new task were given to do research on the topic of ERD.

Meeting log - 4

Day – 4 **DATE 15-12-2023**

Start Time – 1 :00 **End Time 2:00**

Location : Alumni Block.

Topic Discussed : On this day, we wrapped up DFD, but something fell off. So, we checked with our module leader, who showed us in the right direction. With DFD cleared, we straight dive into ERD for next step.

Accomplishment : DFD was completed, and we started making rough ERD diagram using DFD as a reference .

Problems : It was hard to choose the best software to make ERD and everyone was exhausted from the work load.

Task For Next Meeting : Continuing research on ERD and followed by Data Dictionary and Process Specification.

Meeting log - 5

Day – 5 **DATE 16-12-2023**

Start Time – 10 :00 **End Time 2:00**

Location : Discussion Room.

Topic Discussed : On day fifth, we discuss about last ERD, and we clear each other's misconceptions and doubts of data dictionary and process specification so that how should we approach it.

Accomplishment : We have completed both ERD and data dictionary and process specification at the same time with the help of our instructor. He also corrected us in many points.

Problems : Timing was the major issue as we put our time management skills to the ultimate test, balancing all the coursework and working simultaneously on different tasks.

Task For Next Meeting : Research and started preparing Structure chart and assignment diary.

Meeting log - 6

Day – 6 **DATE 16-12-2023**

Start Time – 9 :00 **End Time 1:00**

Location : Kumari Block.

Topic Discussed : Diving deep into research, we brainstormed with our team about using structured charts and assignment diaries effectively. We explored all the pros and cons and considered different research approaches.

Accomplishment : Completed both structured charts and assignment diaries and shared knowledge with all the team members . Later, consulted with our tutor as well as module leader.

Problems : As the deadline approached, individual task burdens intensified, leading to a decline in inter-team communication and a subsequent reduction in the coursework's overall pace.

Task For Next Meeting : With everything evenly distributed, we gathered for a final, thorough review to ensure all tasks were completed and polished.

Meeting log - 7

Day – 7 **DATE 18-12-2023**

Start Time – 9 :00 **End Time 12:00**

Location : Kumari Block.

Topic Discussed : Each member contributed their completed documentation sections, which we then seamlessly merged into one master file. A thorough review confirmed everything was in its proper place and looking sharp.

Accomplishment : Seeking a final stamp of approval, we presented our work to the module leader, who guided us toward perfection. With his expert insights, we confidently submitted the coursework as a team.

Problems : Juggling multiple coursework commitments alongside team management proved challenging, requiring us to implement effective strategies to stay on track.

Task For Next Meeting : As we have completed all our task so there is no meeting for next task.

5. Individual task

5.1. Environmental model specification

5.1.1.a Context Level diagram of Make a Payment (Jenish Katuwal 22068751)

Payment processing in our system is done jointly by the user and the system. The user first gives the system his or her information/details, after which the system saves the information in the user database, validates the information, performs a security check, and allows the user to request payment. The system then gives the user the option to choose a payment method (debit, credit, or PayPal). If the user's selected option is accepted, they can proceed with making a payment, and an invoice will be issued following the payment. The user will receive a receipt along with a message confirming the transaction.

This diagram illustrates the payment procedure at the context level (Level - 0 DFD). The user and system data flow is depicted in the diagram. This is a basic understanding of the system's operation.

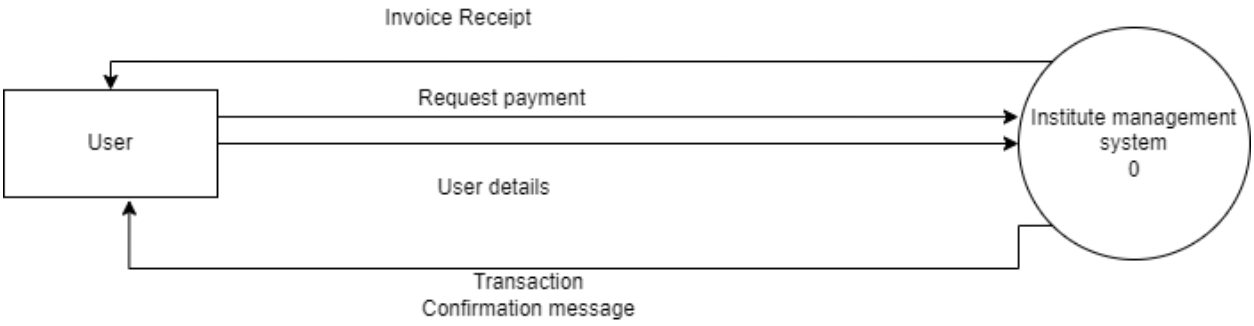


Figure 13: Level -0 dfd (Make Payment)

5.2 Internal model specification for the system

5.2.1.a Level - 1 DFD of Make a Payment

More information on making payments is included in the Level 2 DFD Diagram than in the Level 0 DFD Diagram. Before completing a payment, the System must save the user's details in the system. Once the payment has been requested, the System must record or store all payment information in a database to be reviewed later. The user will be provided with an invoice receipt and confirmation of the transaction.

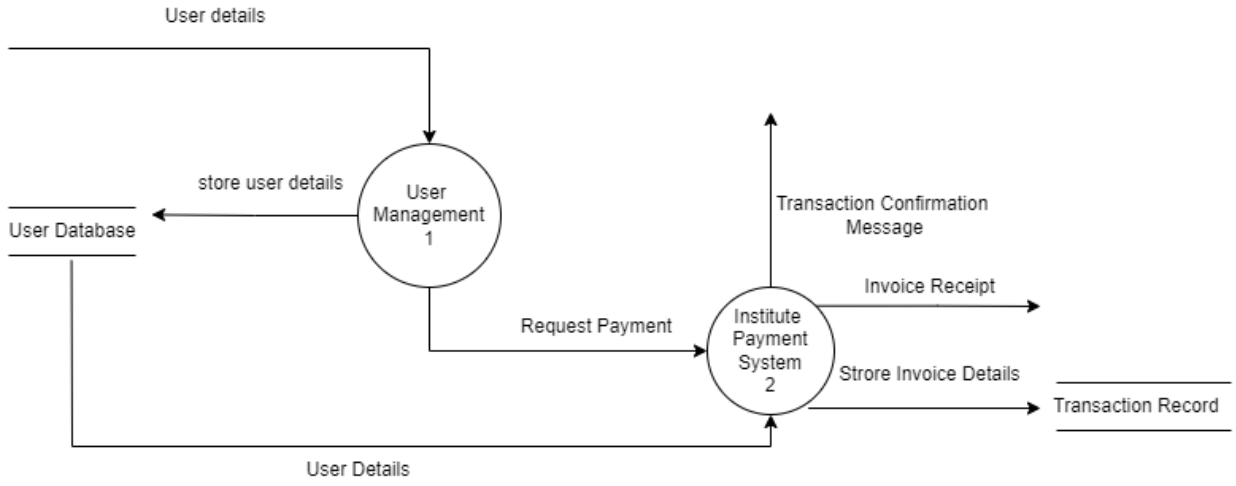


Figure 14: Level-1 dfd (Make Payment)

5.2.2.a Level – 2 DFD of Make a Payment

The entire payment procedure is shown in the level 2 DFD. User data that is entered by the user and kept in the system's database is retrieved in the first stage. Next, a security check and user data validation are carried out by the system. Afterward, the user is presented with a payment method choice by the system, which includes PayPal, credit, and debit cards. An invoice will be sent out after the user's payment is received, provided that the option they choose is approved. Along with a message verifying the transaction, the user will receive a receipt.

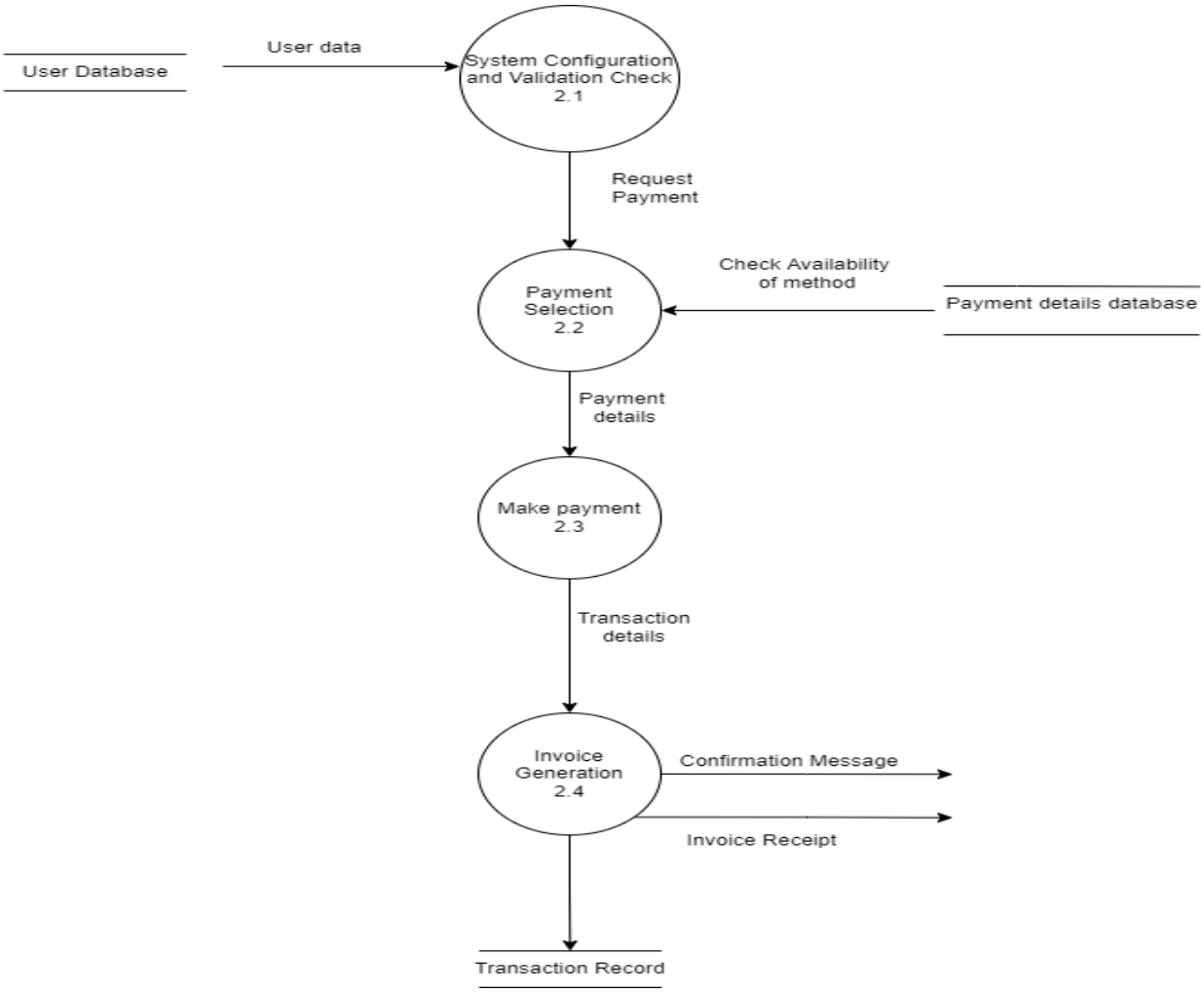


Figure 15: Level -2 dfd (Make Payment)

5.3 Design specification

5.3.1.a Structure Chart of Make Payment

The structure chart of the "Make Payment" illustrates complete data flow between different functions. The Make Payment feature in our software handles both user registration and payment processing, guaranteeing a safe and easy user experience. As they explore making payments, the system registers their details securely on the database. The system conducts strict data validation and security checks as users choose transactions, guaranteeing the dependability of the payment procedure. If there is a payment option, consumers make the payment request with ease. After payment, our software quickly generates an invoice and provides users with a comprehensive invoice receipt, ensuring a safe and effective payment process.

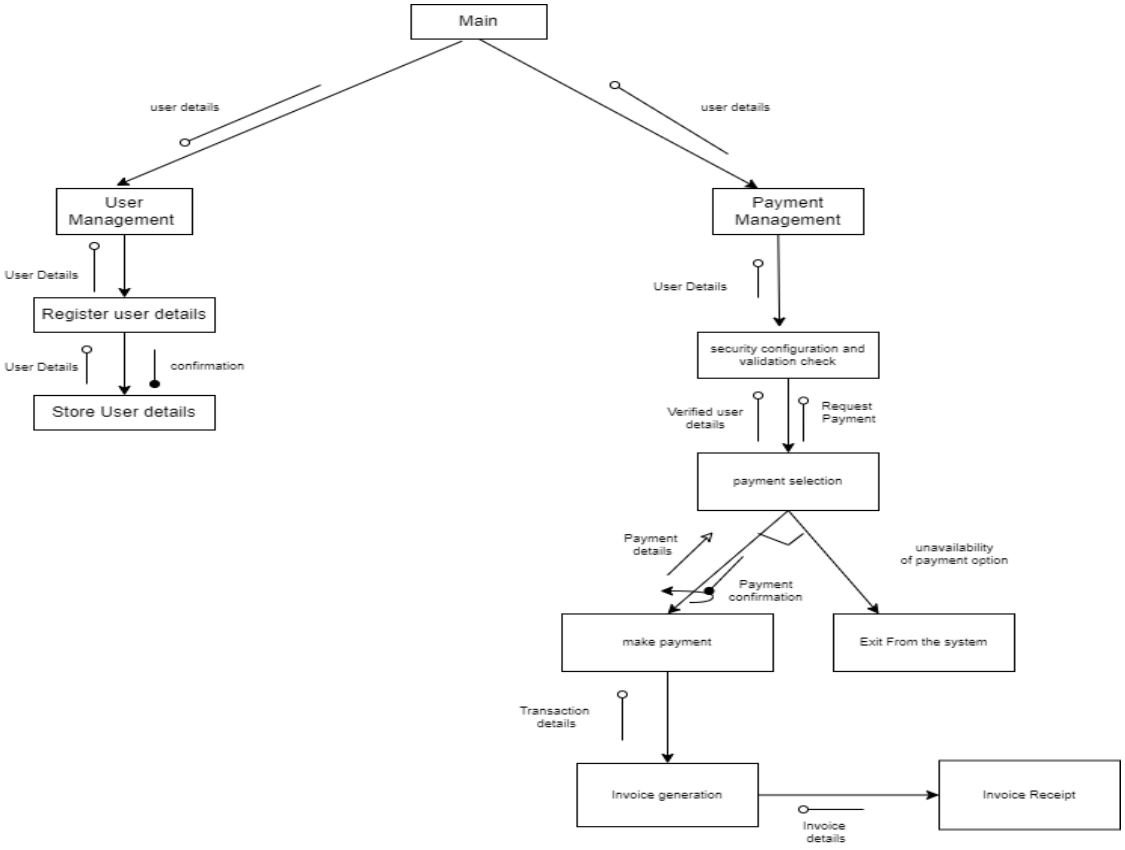


Figure 16: Structure Chat of Make a Payment

5.3.2.a Model Specification of Make Payment

Name	Make Payment
Purpose	The main goal is to make the payment procedure easier for the user. It verifies that the transaction is completed successfully and gathers the required payment information.
Pseudo code	DO IF User_Input = Gather_user_input() DISPLAY Payment Selection Option ELSE DISPLAY login_page WHILE user select payment method VALIDATE user payment details VERIFY Security of the system IF Payment is successful PROCESS Payment_Deduction RECORD Transaction_Details GENERATE Inoice_Receipt DISPLAY Confirmation_Message ELSE DISPLAY Payment_Failure_Message
Output Parameters	Confirmation_Message, Payment_Failure_Message

Global Variable	Database(DB)
Local Variable	User_Input, user payment details, Invoice_Receipt, Transaction_Details
Called By	User Interface for Payment Process

Table 5: Model Specification of Make Payment

INDIVIDUAL TASK	
NAME	**Arbit Bhandari**
LONDONMET-ID	**220681111**
SECTION	**L2C8**

5.1.1.b Context Level Diagram of Purchase Plant

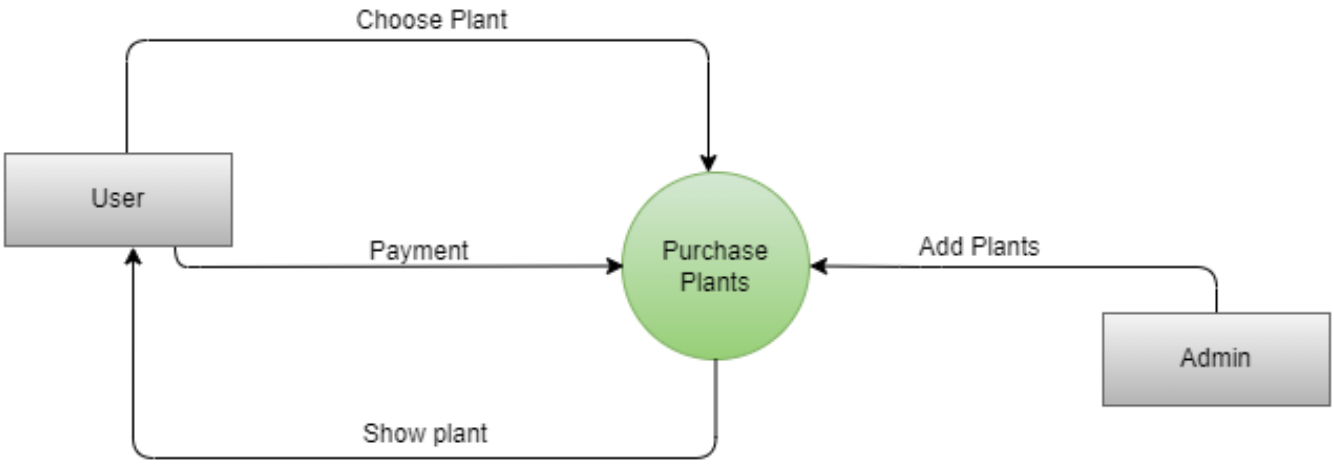


Figure 17: Context Diagram of Purchase Plants

Description:

In our context level diagram, the system displays the plants. Users can then choose the plants and proceed to make payments. Additionally, administrators have the capability to add plants to the system, allowing users to view the newly added plants.

The context level diagram includes the following features:

- **Plant Display:** The system showcases various plants to users.
- **User Interaction:** Users have the option to choose specific plants from the displayed selection.
- **Payment Process:** Users can proceed to make payments for their selected plants.
- **Administrator Functionality:** Administrators possess the capability to add new plants to the system.
- **User Visibility:** Newly added plants by administrators are made visible to users for viewing.

5.2.1.b Level 1 DFD of Purchase Plant

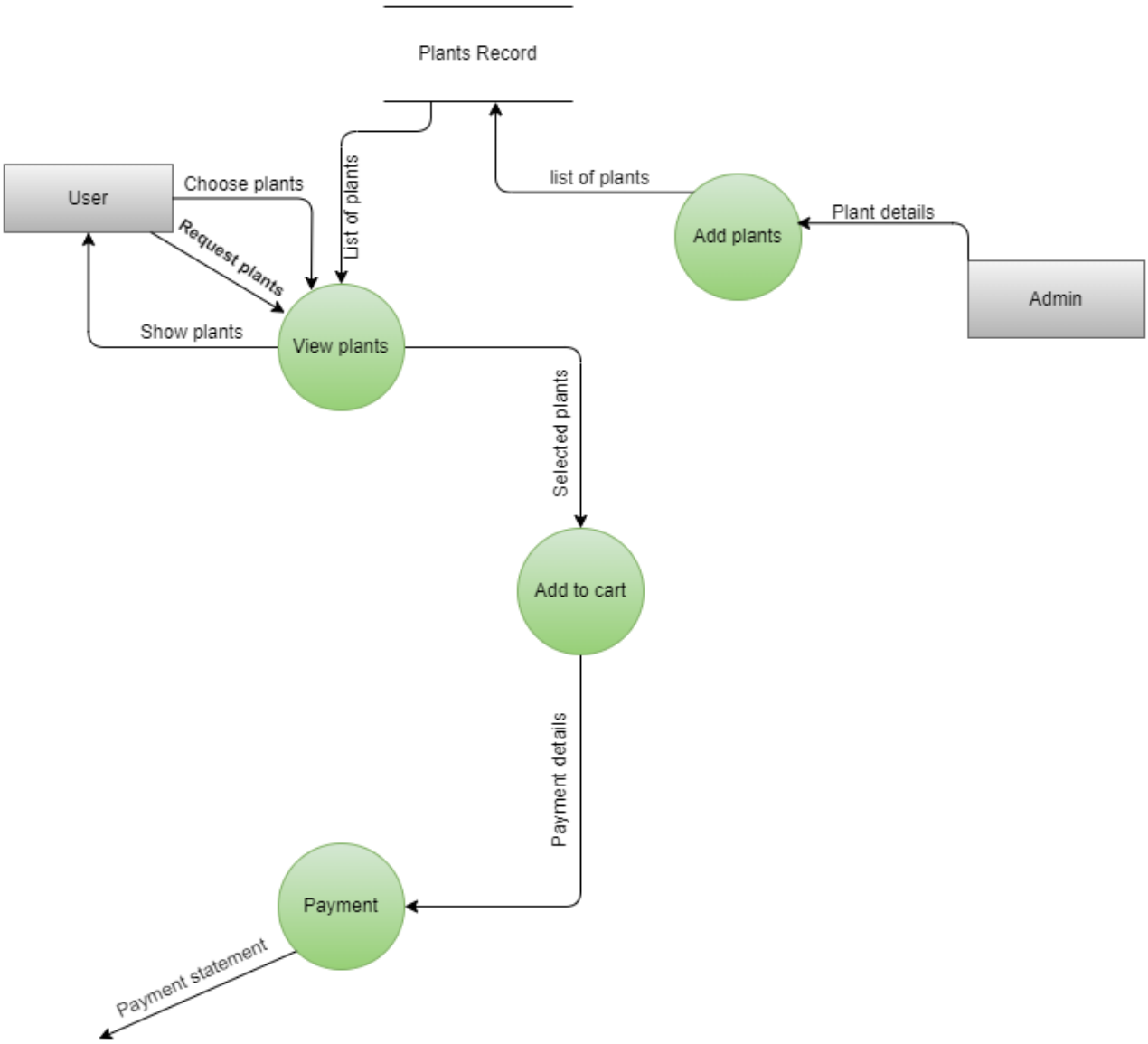


Figure 18: Level 1 DFD of Purchase Plants

Description:

In the first level of the Data Flow Diagram (DFD), the process begins with a user requesting information about plants. The user then retrieves plant details through the "View Plants" process and proceeds to choose specific plants. Once the plant selection is made, the chosen items enter the "Add to Cart" process. After being added to the cart, the payment details are forwarded to the "Payment Process," where the payment information is stored in the database.

Similarly, in the administrative flow, the admin provides plant details, initiating the "Add Plants" process. The list of plants generated in this process is directed to the "Plants Records" for storage. Simultaneously, the list is also sent to the "View Plants" process, allowing users to access and view the updated collection of plants.

5.2.2.b Level 2 DFD of Purchase Plant

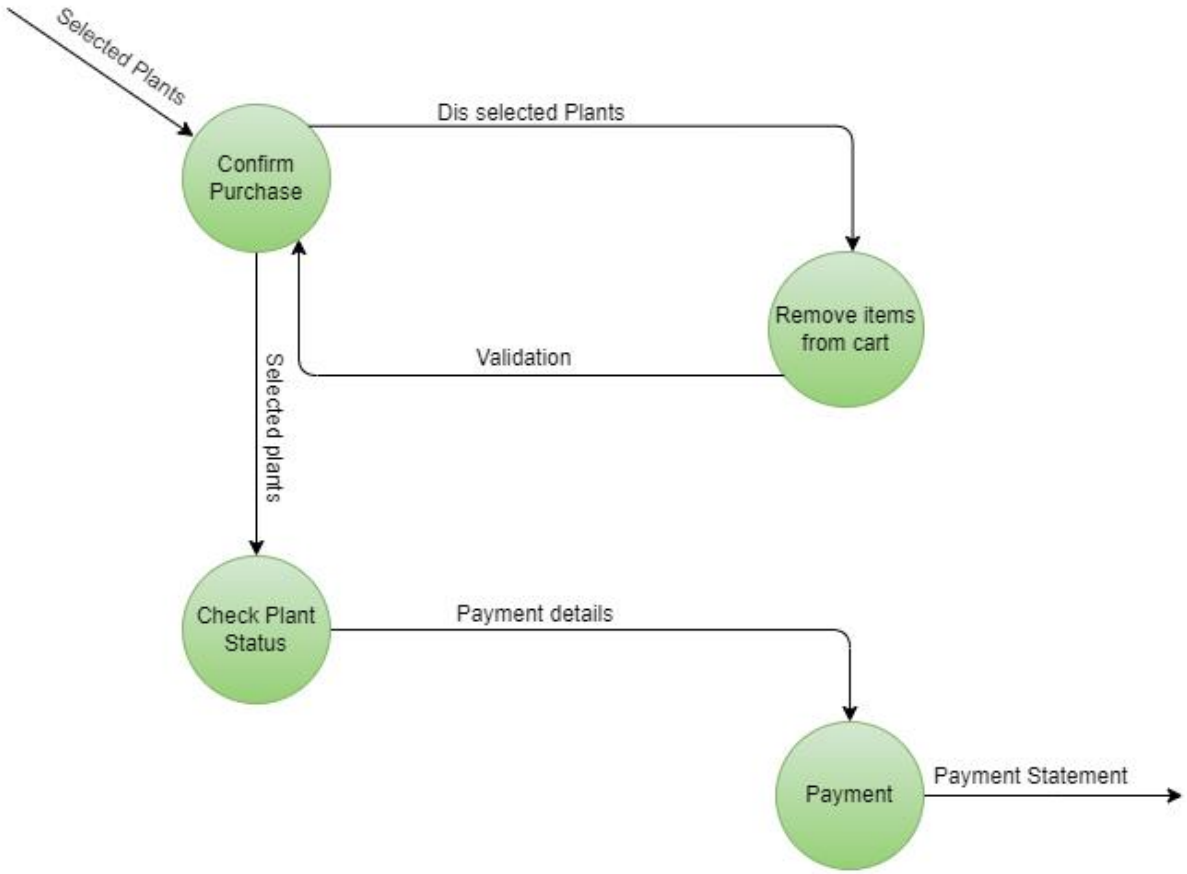


Figure 19: Level 2 DFD of Purchase Plants

Description:

In the Level 2 Data Flow Diagram (DFD), the flow continues from the selection of plants. After the user selects plants, they move to the "Confirm Purchase" subprocess. Within the "Confirm Purchase" subprocess, if the user confirms the selection, the chosen plants proceed to the "Check Plants Status" process. In this step, the system verifies the status of the selected plants, prompting the user to provide payment details. The payment details then move to the "Payment Process.

Conversely, if the user decides to deselect plants during the "Confirm Purchase" subprocess, the plant details are removed from the cart. This removal is facilitated by the "Remove from Cart" subprocess, which also includes a validation step to ensure the accuracy and legitimacy of the deselection process.

5.3.1.b Structure Chart of Purchase Plant

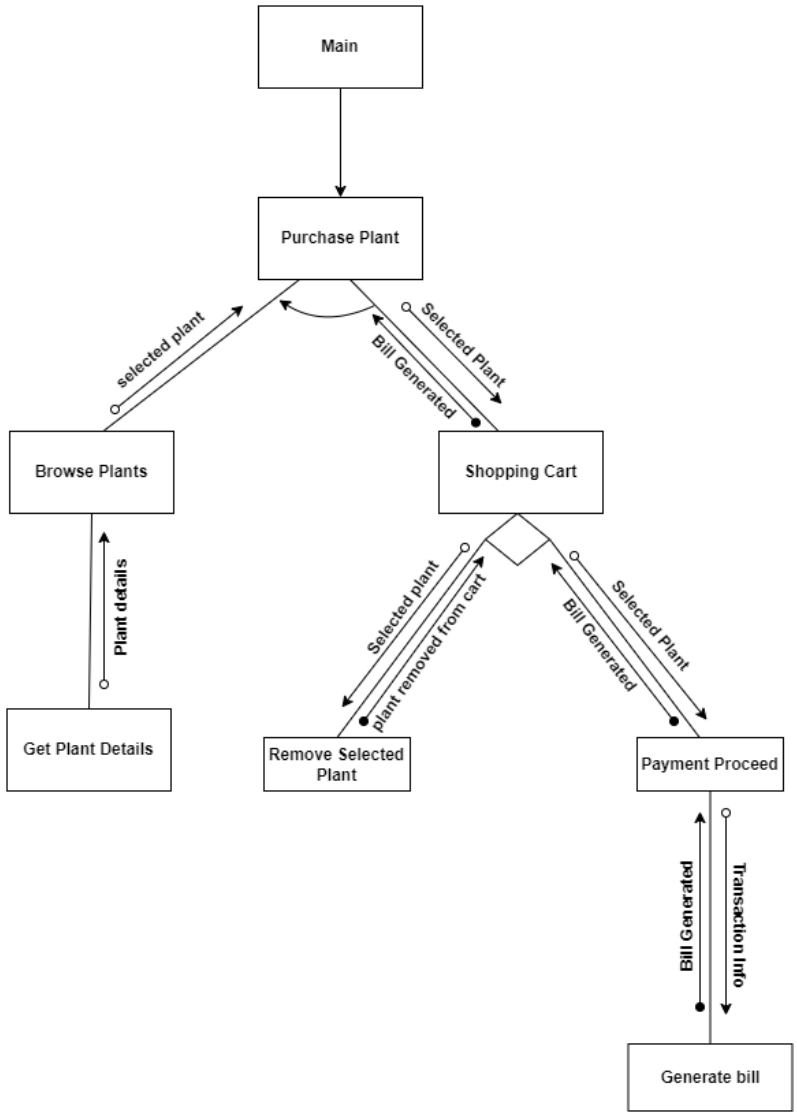


Figure 20: Structure Chart of Purchase Plants

Description:

The structure chart is a graphical representation of the program's structure. It shows the hierarchy of the modules in the program, and the dependencies between them. The main module is at the top of the chart, and the other modules are below it. The arrows between the modules show the dependencies between them. For example, the "Purchase Plant" module depends on the "Browse Plants" module and the "Get Plant Details" module.

The "Browse Plants" module is responsible for displaying the plants that are available for purchase. The "Get Plant Details" module is responsible for getting the details of a specific plant. The "Purchase Plant" module is responsible for purchasing a plant. The "Shopping Cart" module is responsible for keeping track of the plants that have been selected for purchase. The "Payment" module is responsible for processing the payment for the plants that have been selected for purchase. The "Generate Bill" module is responsible for generating a bill for the plants that have been purchased.

The structure chart shows that the program is divided into a number of modules. This makes it easier to understand the program and to make changes to it. The structure chart also shows the dependencies between the modules. This information can be used to identify potential problems in the program.

5.3.2.b Module Specification (MSpecs) of Purchase Plant

PURPOSE	The purpose of this module is to help users purchase the selected plants and provide functionality such as adding to the cart, removing from the cart, and accepting payment after purchasing the plant.
PSEUDOCODE	// Retrieve plant details from the database var **plants_details** = DB.get_plant_records() // Retrieve user details from the database var **user_details** = DB.get_user_records() // Display available plants and let the user select some var **selected_plants** = show_plants(plants_details) // Process and enhance the selected plants for purchase var **processed_plants** = process_plants(selected_plants) // Add the processed plants to the shopping cart var **cart_added_plants** = add_to_cart(processed_plants)

	// Check if the user wants to remove any plants from the cart if (module_wants_to_remove_plants) { **remove_plants_from_cart**(cart_added_plants) } else { // Proceed with payment using the shopping cart and user details var **payment_details** = proceed_payment(cart_added_plants, user_details) // Display the final bill to the user **show_bill**(payment_details) }
INPUT PARAMETERS	plants_details, user_details, selected_plants, processed_plants, cart_added_plants,
OUTPUT PARAMETERS	remove_plants_from_cart, show_bill,
GLOBAL VARIABLE	DB

CALLS	• get_user_records • get_plant_records • show_plants • payment_details • process_plants • add_to_cart, • remove_plants_from_cart • proceed_payment • show_bill
CALLED BY	MAIN

Table 6: Module Specification (MSpecs) of Purchase Plant

INDIVIDUAL TASK	
NAME	**Miraj Deep Bhandari**
LONDONMET-ID	**22067814**
SECTION	**L2C8**

5.1.1.c Context Level Diagram of Report Preparation:

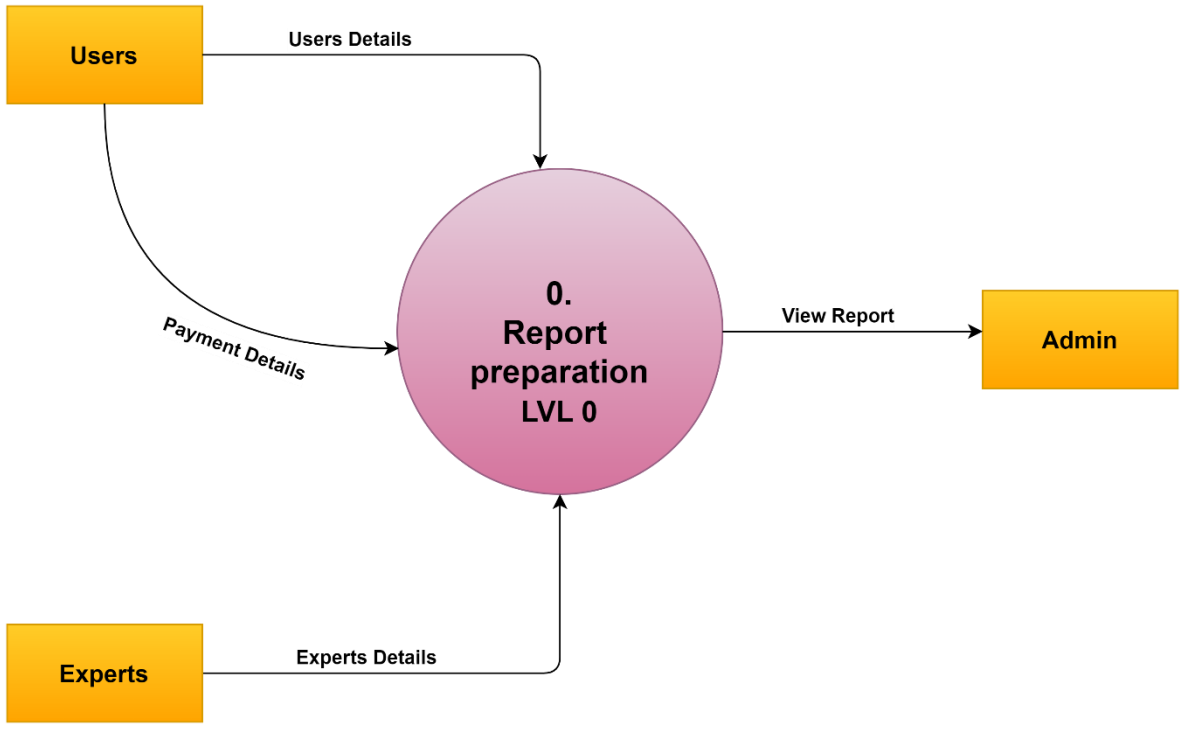


Figure 21: Context Level Diagram of Report Preparation

Description:

The Context Diagram, often referred to as the Level 0 Data Flow Diagram, provides a high-level overview of how a system functions. The diagram depicted above plays a pivotal role in generating a comprehensive report on the system's activities, which is then presented to the system administrator. Gathering the necessary information for this report involves crucial details from users, such as their personal information, specifics about experts, and details regarding the payments made by users. Essentially, the diagram acts as a visual representation, offering a simplified understanding of the fundamental interactions and data flows within the system. This aids in efficiently preparing detailed reports for the system administrator.

This Level 0 Data Flow Diagram performs the following key functions:

From Users:

- Collects user details
- Gathers payment details made by users when enrolling in courses or purchasing plants

From Experts:

- Collects details about experts

To Administrator:

- Presents the generated report

5.2.1.c Level 1 DFD of Report Preparation:

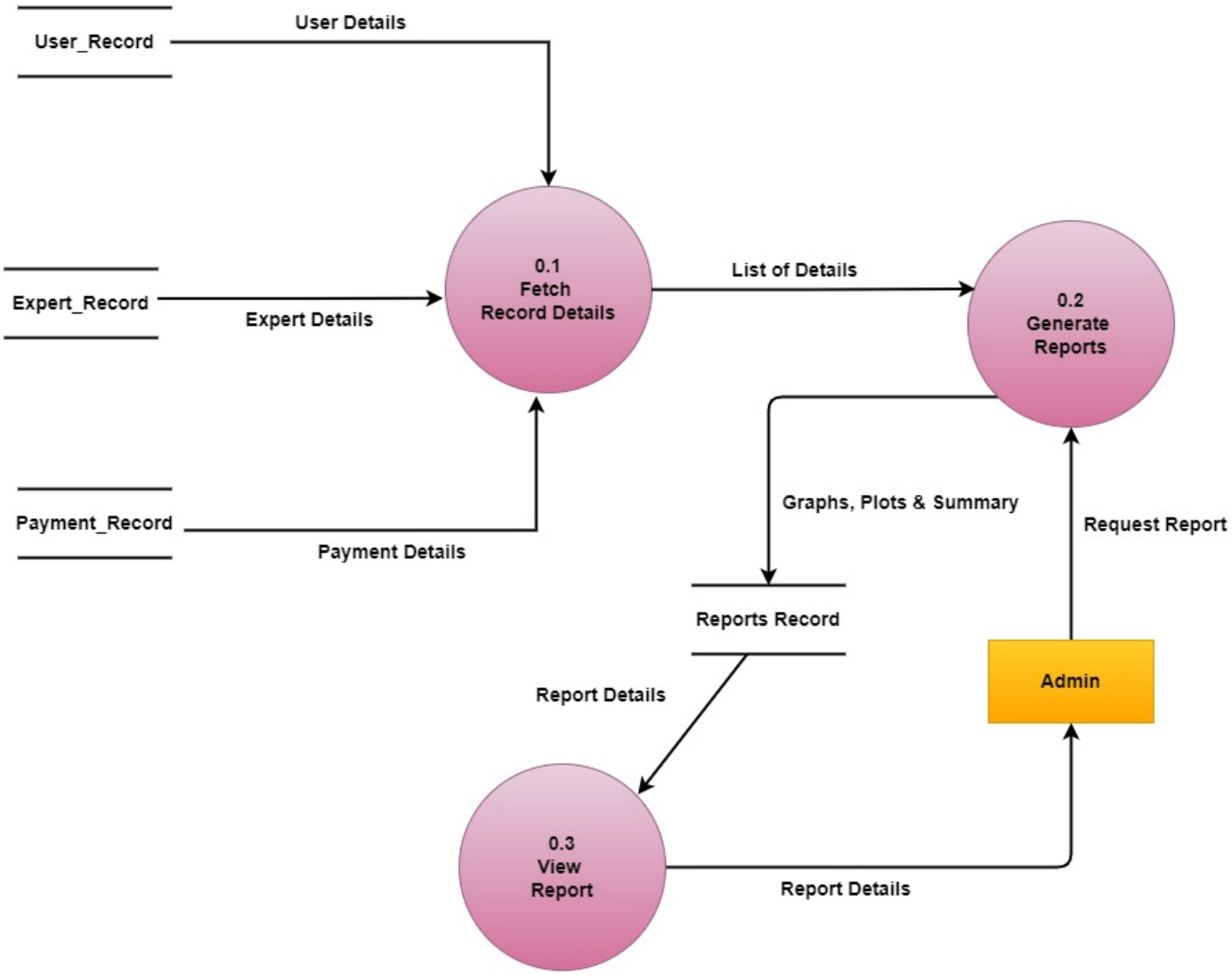


Figure 22: Level 1 DFD of Report Preparation

Description:

Level one data flow diagram is the sub-process of the Context Diagram. In this system, the procedure for handling real cases is defined. Here, In our Level 1 data flow diagram *user details, expert details*, and *payment details* are retrieved from the **user_record, expert_record,** and **payment_record** databases, respectively. These details are then processed in the *fetch record details process*, where they are combined and sent to another process called "*generate reports*."

When the admin requests a report, the graphs, plots, and summary are sent from the "*generate report*" process and stored in the **reports_record.** The "*view report*" process retrieves records from the **reports_record** database. Finally, the admin views the report through the "*view report*" process.

Level 1 Data Flow Diagram performs the following key functions :

From Users:

- Retrieves user details from the user_record database.

From Experts:

- Retrieves expert details from the expert_record database.

From Payments:

- Retrieves payment details from the payment_record database.

Processing:

- The fetch record details process combines user details, expert details, and payment details.

To Generate Reports:

- Sends the combined details to the "generate reports" process.

From Generate Reports:

- When the admin requests a report, the process sends graphs, plots, and a summary.

To Store Reports:

- The generated content is stored in the reports_record.

From View Reports:

- The "view report" process retrieves records from the reports_record database.

To Administrator:

- The admin views the report through the "view report" process.

5.2.2.c Level 2 DFD of Report Preparation

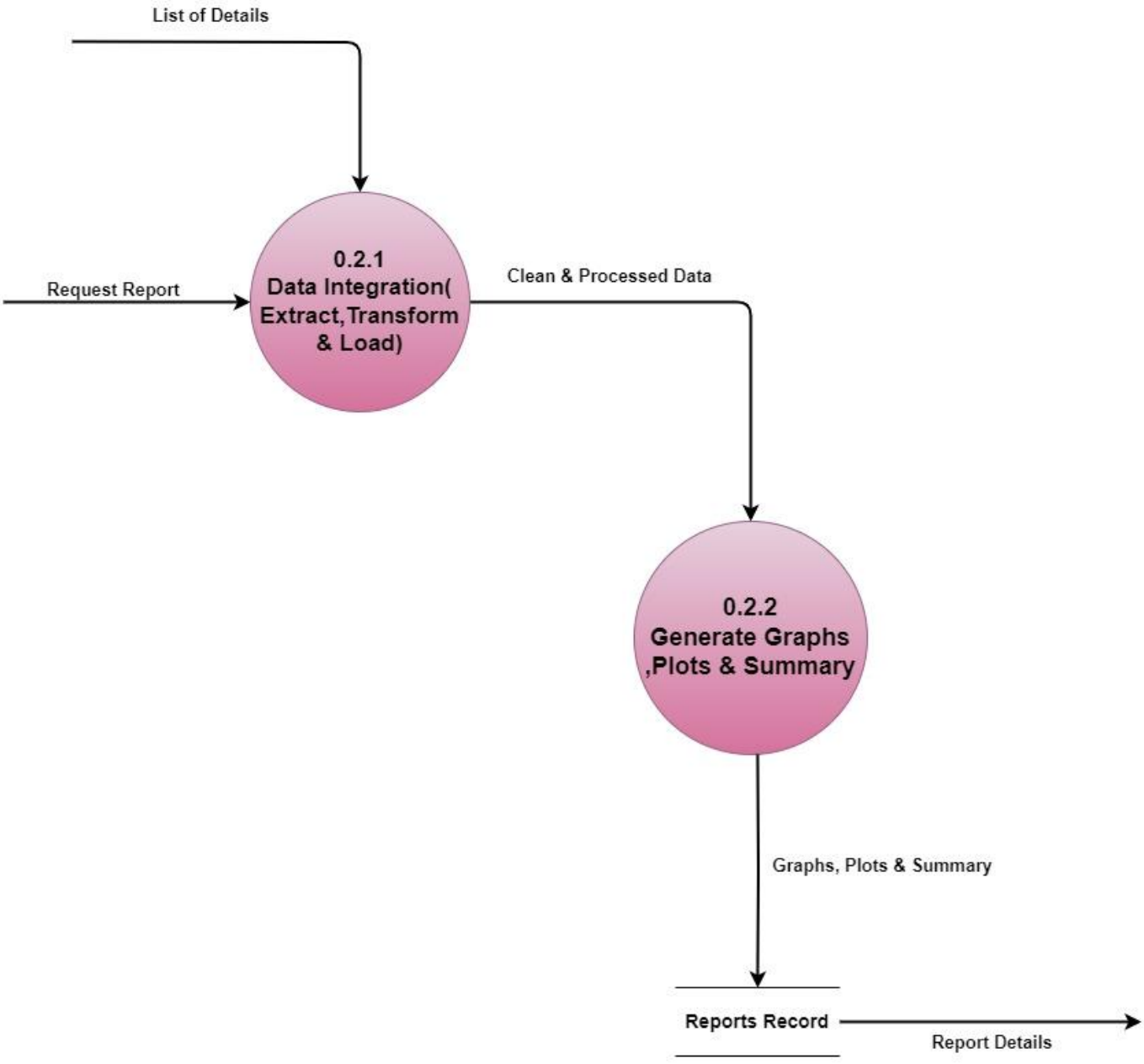


Figure 23: Level 2 DFD of Report Preparation

Description:

Level 2 is the sub-process in which the system describes the data inflow and outflow. It represents the final data entity process. The diagram above illustrates the Level 2 DFD (Data Flow Diagram) for the "Generate Report" process. When the user triggers the "Generate Report" process, all the combined details of users, experts, and payments undergo a data integration sub process, which includes the ETL process (Extract, Transform, and Load).

After the ETL process, the cleaned and processed data is sent to the "Generate Plots, Graphs & Summary" sub process. This sub process utilizes machine learning tools that provide diverse analysis with graphs, plots, and summaries. The generated graphs, plots, and summary are then stored in the Reports_Record database. The details of the report can be obtained by accessing this reports_records database through another process named "View."

Data Inflow and Outflow in Level 2 DFD (Generate Report):

Trigger Event:

- User initiates the "Generate Report" process.

Data Integration Subprocess (ETL Process):

- Combines user details from the user_record.
- Combines expert details from the expert_record.

- Incorporates payment details from the payment_record.

- Executes the ETL process (Extract, Transform, and Load).

Cleaned and Processed Data:

- The result of the ETL process is cleaned and processed data.

Generate Plots, Graphs & Summary Subprocess:

- Receives cleaned and processed data.
- Utilizes machine learning tools for diverse analysis.
- Generates graphs, plots, and a summary.

Machine Learning Tools:

- Applied within the "Generate Plots, Graphs & Summary" subprocess.

Stored in Reports_Record Database:

- The generated graphs, plots, and summary are stored in the Reports_Record database.

Accessing Report Details:

- Another process named "View" allows access to the details stored in the reports_records database.

5.3.1.c Structure Chart of Report Preparation:

A structure chart in software engineering is a diagrammatic representation that illustrates the organization and hierarchy of different modules or components in a software system. It is a top-down modular decomposition technique used during the design phase of software development to represent the structure of a system.

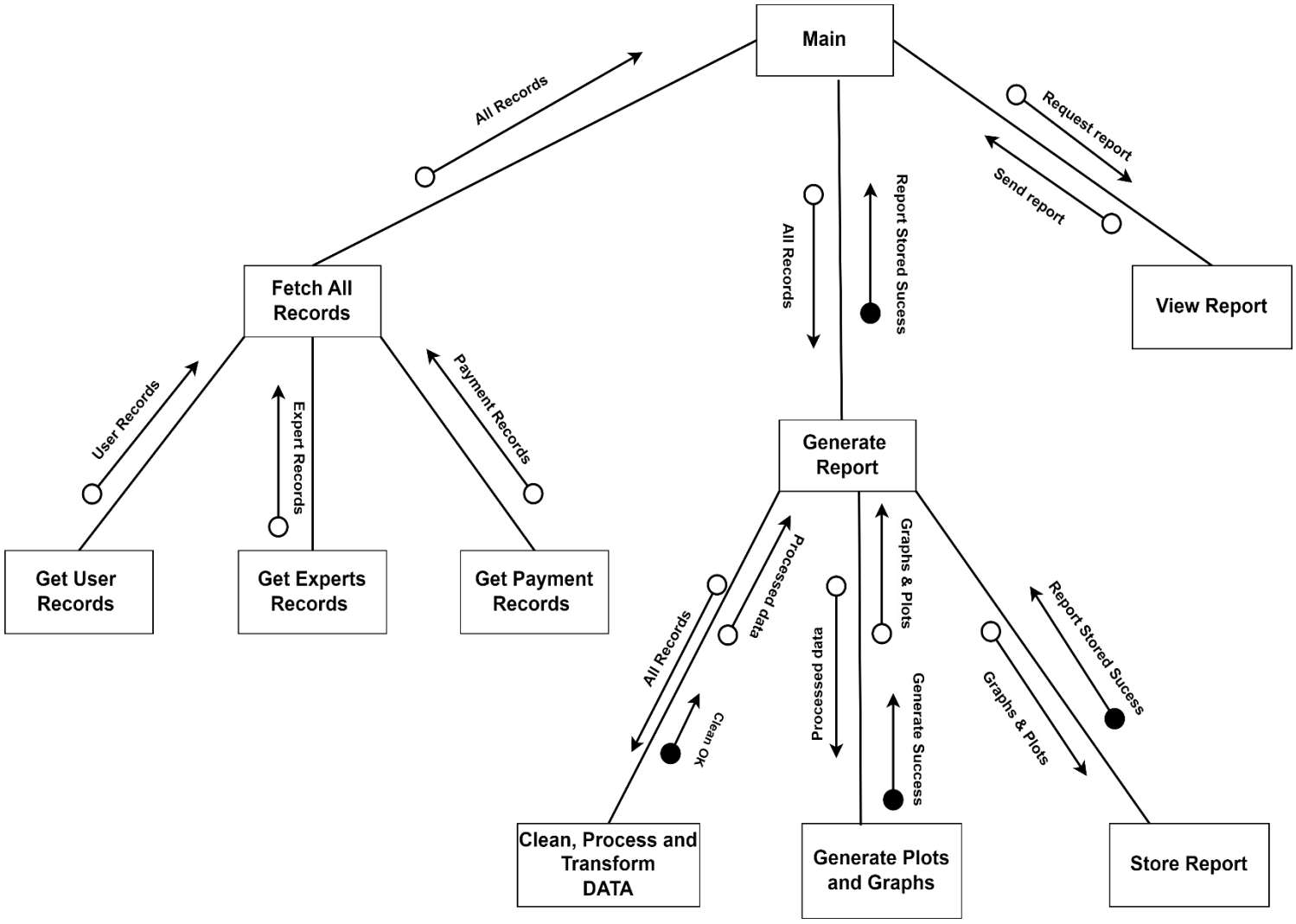


Figure 24: Structure Chart of Report Preparation

Description:

In the above figure, the structure chart of report preparation is shown. The user records, expert records, and payment records are received from the "Get User Records," "Get Experts Records," and "Get Payment Records" modules, respectively. All these records are called by the "Fetch All Records" module. This module confirms and combines all the records. The main module calls "Fetch All Records" and sends all the records to the "Generate Report" module. The "Generate Report" module will have three sub-modules.

The first module, named "Clean, Process, and Transform Data," performs data preprocessing. After preprocessing, the preprocessed data is called by the "Generate Plots and Graphs" module. This module uses machine learning tools that generate plots, graphs, and summaries. The "Store Report" module calls these graphs, plots, and summaries and stores them in the database.

The "View Report" module receives the admin request and provides the generated report from the database to the admin .

5.3.2.c Module Specification (MSpecs) of Report Preparation:

PURPOSE	The purpose of this module is to generate a report of users and experts, and it assists in displaying the report to the admin. To create the report, it collects data from users records, experts records, and payment records made by the users.
PSEUDOCODE	**DO** // Retrieve user records from the database ***var users_details*** *= DB.get_user_records()* // Retrieve expert records from the database ***var experts_details*** *= DB.get_expert_records()* // Retrieve payment records from the database ***var payments_details*** *= DB.get_payment_records()* // Combine all relevant records for comprehensive analysis ***var all_data*** *= fetch_all_data(users_details,experts_details, payments_details)*

	// Clean, process, and transform the combined data to generate insights ***var processed_data*** *=clean_process_transform_data(all_data)* // Generate informative graphs and plots based on the processed data. ***var visual_data*** *= generate_graphs_plots(processed_data)* // Display a comprehensive report with insightful graphs and plots ***var final_report*** *= send_report(visual_data)* *Display(final_report)* **END DO**
INPUT PARAMETERS	users_details, experts_details, payments_details, all_data, processed_data, visual_data
OUTPUT PARAMETERS	final_report
GLOBAL VARIABLE	DB
CALLS	• get_user_records • get_expert_records • get_payment_records • fetch_all_data • clean_process_transform_data • generate_graphs_plots • send_report.
CALLED BY	MAIN

Table 7: Module Specification (MSpecs) of Report Preparation

Individual Task (Join the program Rijan Paudel)

5.1.1.d Context Level Diagram of Join the Program

The process “Join The Program” allows users/students to join the program or enroll in the course. There are different type of courses provided i.e Post Graduate, Undergraduate and Short term certification course. The courses are both paid and unpaid. To join program student needs to provide student details through registration which is verified by the admin. Student needs to select the suitable program for them. After the course selection, if the course is paid they need to make payment and if course is unpaid they can register or enroll without payment. So, they will get enrolled after successful payment. Student also get different course materials and certificates after enrollment.

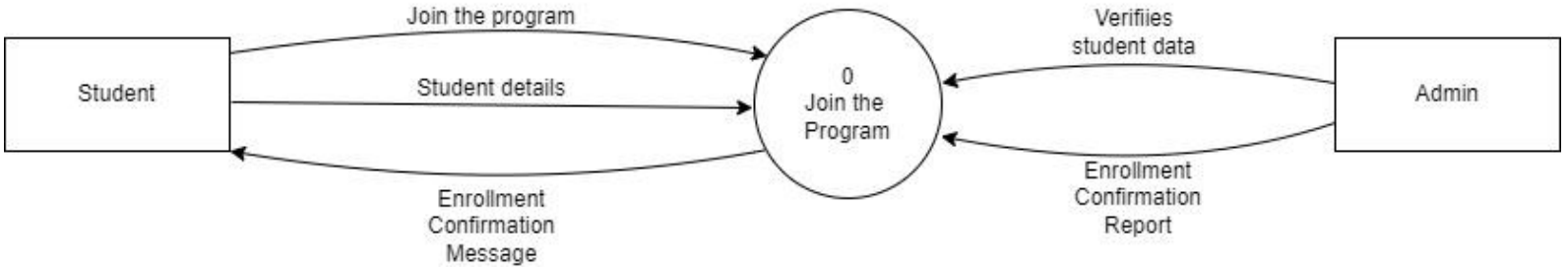


Figure 25: Context level diagram for Join the program.

Description

This is a context level diagram for the process Join The Program, which show very baisc idea of enrollment process. The digram shows that to join the program student provides their details to admin, which will be verified by admin. Then admin will provide enrollment confirmation report to the system. So, students can get enrollment confirmation message.

5.2.1.d Level - 1 DFD of Join the Program

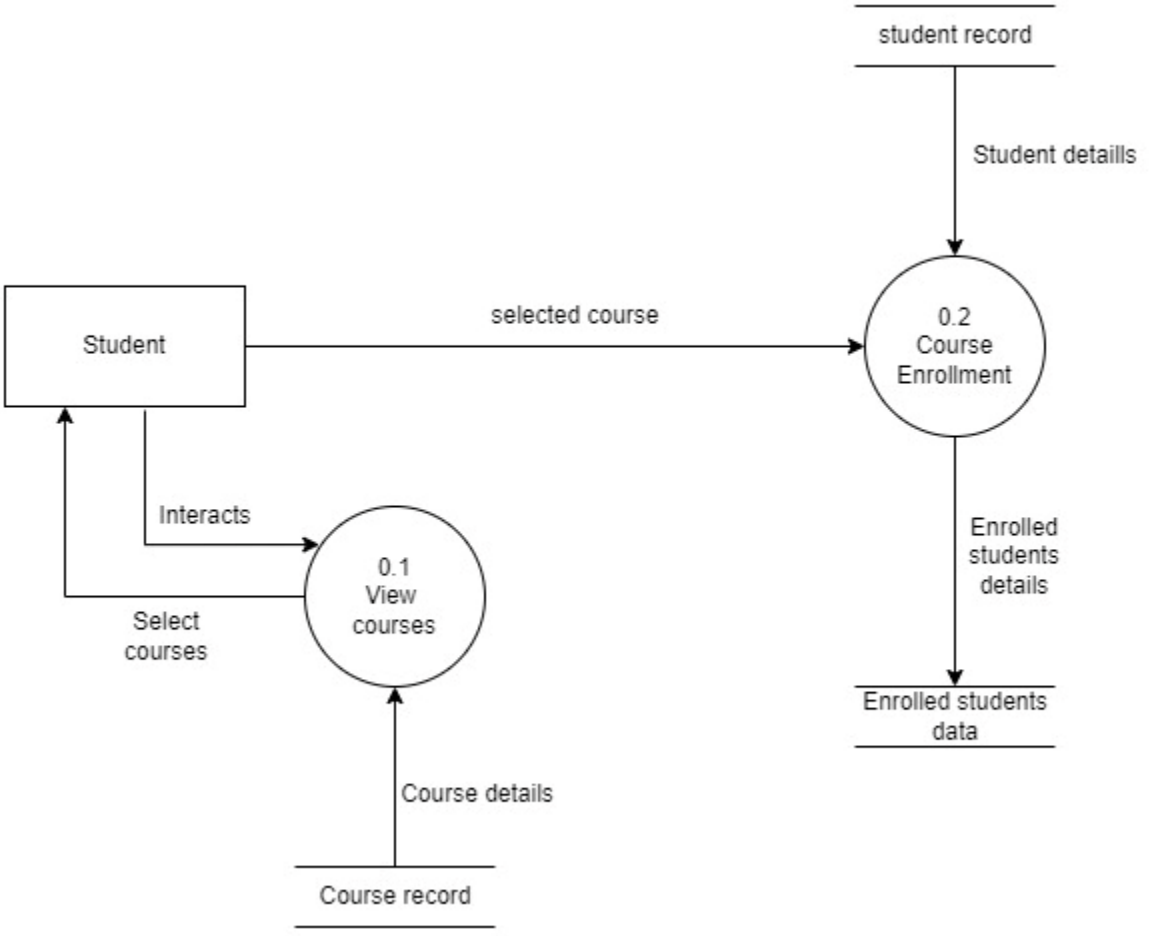


Figure 27: Level 1 DFD of Join the program.

Description

In this Level 1 dfd, the process of course enrollment or join the program is shown in more detailed way. At first student needs to interact with the system to view the details of available courses provided in the institue. Students get that course details through course record database. After viewing the course details, students selects the appropriate course for them which goes to course enrollment process. The course enrollment process gets the student details from student record database. After getting the information of student and their selected course, students gets enrolled in the program. So, course enrollment process stores the enrolled student details in the enrolled students database.

5.2.2.d Level 2 – DFD of Join the Program

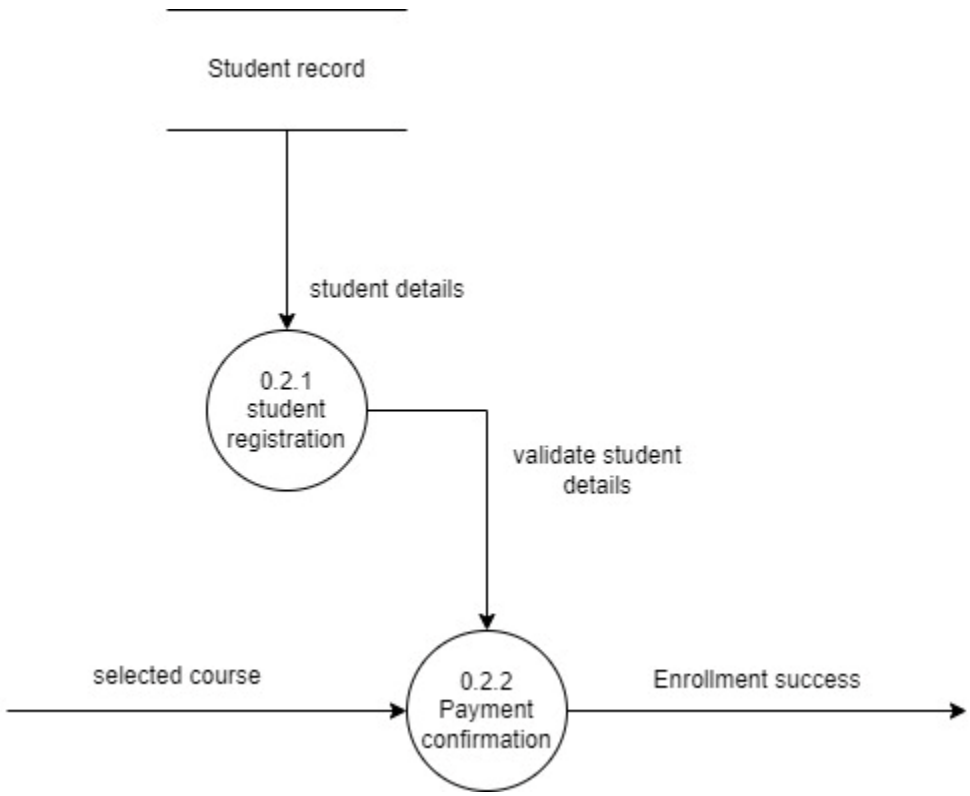


Figure 28: Level 2 DFD of Join the Program.

Description

This Level 2 dfd shows the detailed process of enrollment of students or the process of joining the program. In this dfd, the process Course Enrollment is futher divided into more details. In this process, student needs to register in the system through database of student details. After the registration, student details are validated and it goes for course selection process. Also, the course selection process gets the data of selected course. After the course selection process successfully enrolls the students after payment confirmation.

5.3.1.d Structure Chart of Join the Program

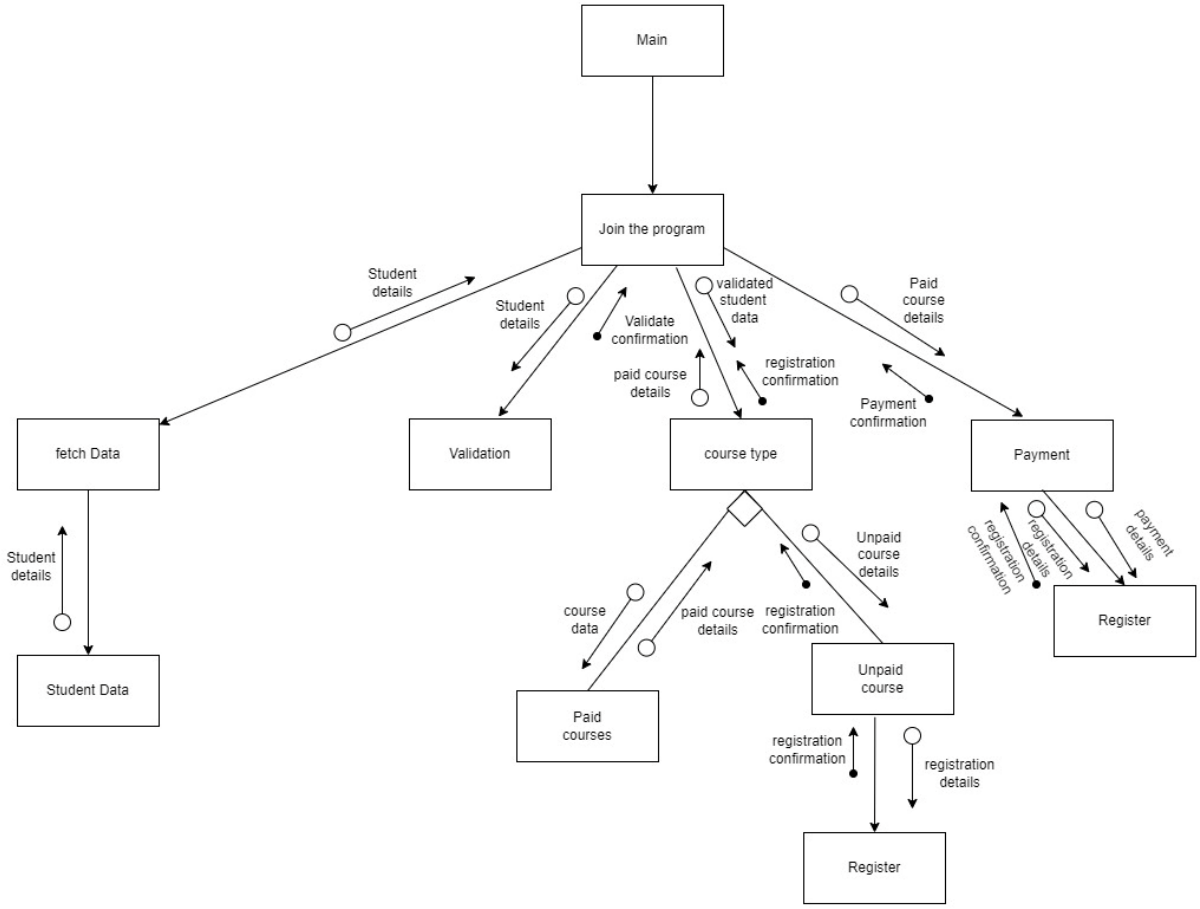


Figure 29: Structure Chart for Join the program.

Description

The structure chart shows the decomposition of the main program into smaller, more manageable modules. The main program is responsible for initializing the system and then calling the join the program module. The join the program module is responsible for enroll the students in the program or course. The join the program module has three sub-module Fetch Data, Validation, Course type and Payment. First it recieves student details from it's sub-module Student data and sends it to Join the program module. Then, student details are sent for validation in sub-module "Validation". That sub-module sends validation confirmation to module Join the program. Then that module sends validated student data to it's sub-module Course type. The sub-module course type sends data of paid course details which is got from the sub-module Paid course. The course type sub-module has two sub-modules in descision box i.e Paid Courses and Unpaid courses. The student gets registered directly if the course is unpaid and needs to confirm payment if the course is paid.

5.3.2.d Module Specs of Join the Program

Name	**Join The Program**
Purpose	The purpose of the Join the Program module is to facilitate user enrollment in both paid and unpaid undergraduate and postgraduate courses. This module aims to streamline the process of joining and participating in the educational programs offered by the institute.
Pseudo Code	**DO** // Get student details from the database var student_data = DB.get_student_details() //validate student data var validate = validation(student_data) // Display available courses var selected_courses = show_courses(validate) //chooses between paid and unpaid courses **IF (PAID COURSES)** **THEN** Paid course_details=Sent course_data(validate) Payment=payment(Paid course_details) Display(“payment confirmation”) **END IF** **ELSE** Unpaid_courses=unpaid courses_details(validate) Register(Unpaid_courses) Display(“validation confirmation”) **END ELSE** **END DO**
Input Parameters	student_data

OUTPUT PARAMETERS	Enrolled_course
GLOBAL VARIABLE	DB
CALLS	Validation, show_courses, Sent course_data, payment, unpaid courses_details, Register, Display
CALLED BY	MAIN

Table 8: Module Specs for Join the program.

INDIVIDUAL TASK	
NAME	**Apil Thapa**
LONDONMET-ID	**22067753**
SECTION	**L2C8**

A short brief explanation about Take certification Exams:

The Certificate Exams system is designed to allow students to take exams to assess their knowledge in specific courses. Users have the capability to view their exam results, and admin play a important role in managing the exam content by gathering user exam details and adding questions into the system. The system include various databases, such as the User Record Database, Course Record, Question Record, and Result Record Database. Users participate in exams through the Exam System, and the generated exam sheets move to the Generate Result Process. The results are then stored in the Result Record Database. Users can access their results through the View Result Process, which retrieves information from the database and accommodates result requests. Additionally, admin have the ability to interact with the results for reporting purposes. In the more advanced process, an Input Exam Requirements Process collects necessary details to set up exams, while the Conduct Exam Process involves both admin and users actively participating in the exams, with exam sheets subsequently moving to the Generate Result Process. This structured system ensures a smooth flow from exam setup to result generation, providing a comprehensive approach to managing and assessing student performance in certificate exams.

5.1.1.e Context level diagram of Take Certification Exams:

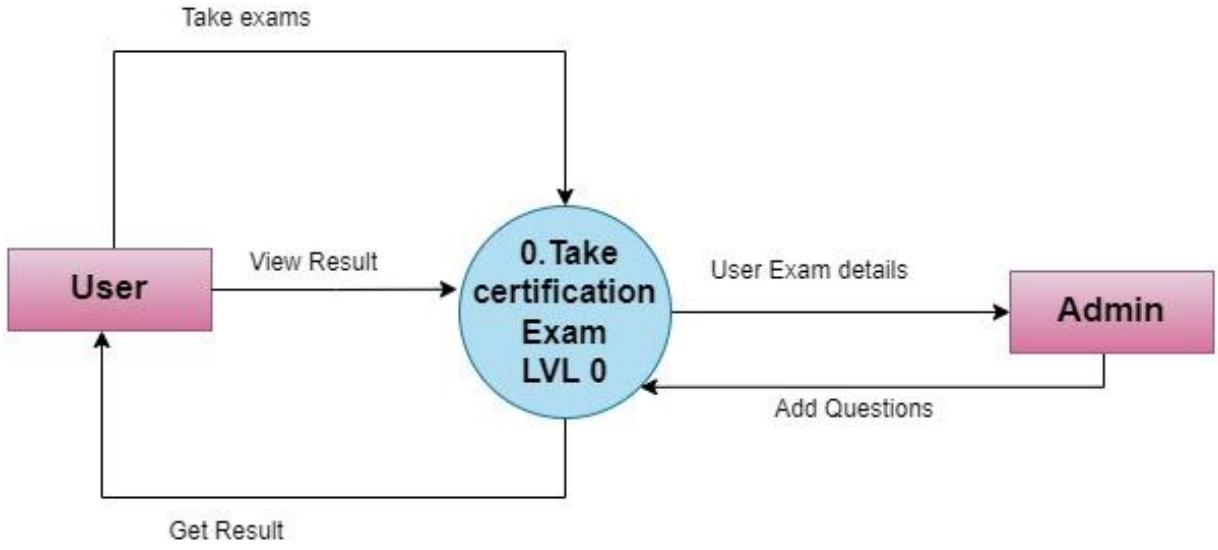


Figure 30:Context level diagram of Take Certification Exams

The above figure suggests the context level diagram of the process, which shows a simplified overview of a system where users engage in the process of taking exams. Initially, users initiate the "Take Examination" process, in which users actively participate in answering questions and completing their exams within the system.users have the convenience of immediately viewing their results and can also request a more detailed outcome from the system, including scores and additional feedback. Simultaneously, an admin, acts as a manager, is involved in the process. The administrator receives information about the exams taken by users and holds the which is responsible of contributing to the system's content by adding new questions. This interaction ensures a cycle where users engage with exams, receive prompt feedback, and admin for system effectiveness enhance the system by introducing fresh and relevant questions. The context level diagram provides a high-level understanding of the key participants and their interactions in this exam-oriented system.

Description:

In our context diagram for the Take certification Exams, two external entities are identified: users and admins.

Users: Individuals who takes exams

Admins: System administrators

Admins have the capability to perform the following functions:

- Add Questions to the system
- Get Exam details

Users have the capability to perform the following functions:

- Take exams
- View result
- Get result

In summary, The Certification Exam System involves two main entities - Users and Admins. Admins can add questions to the system and retrieve exam details. On the other hand, Users can take exams, view their results, and retrieve detailed exam results.

5.2.1.e Level 1 DFD Take Certification Exams

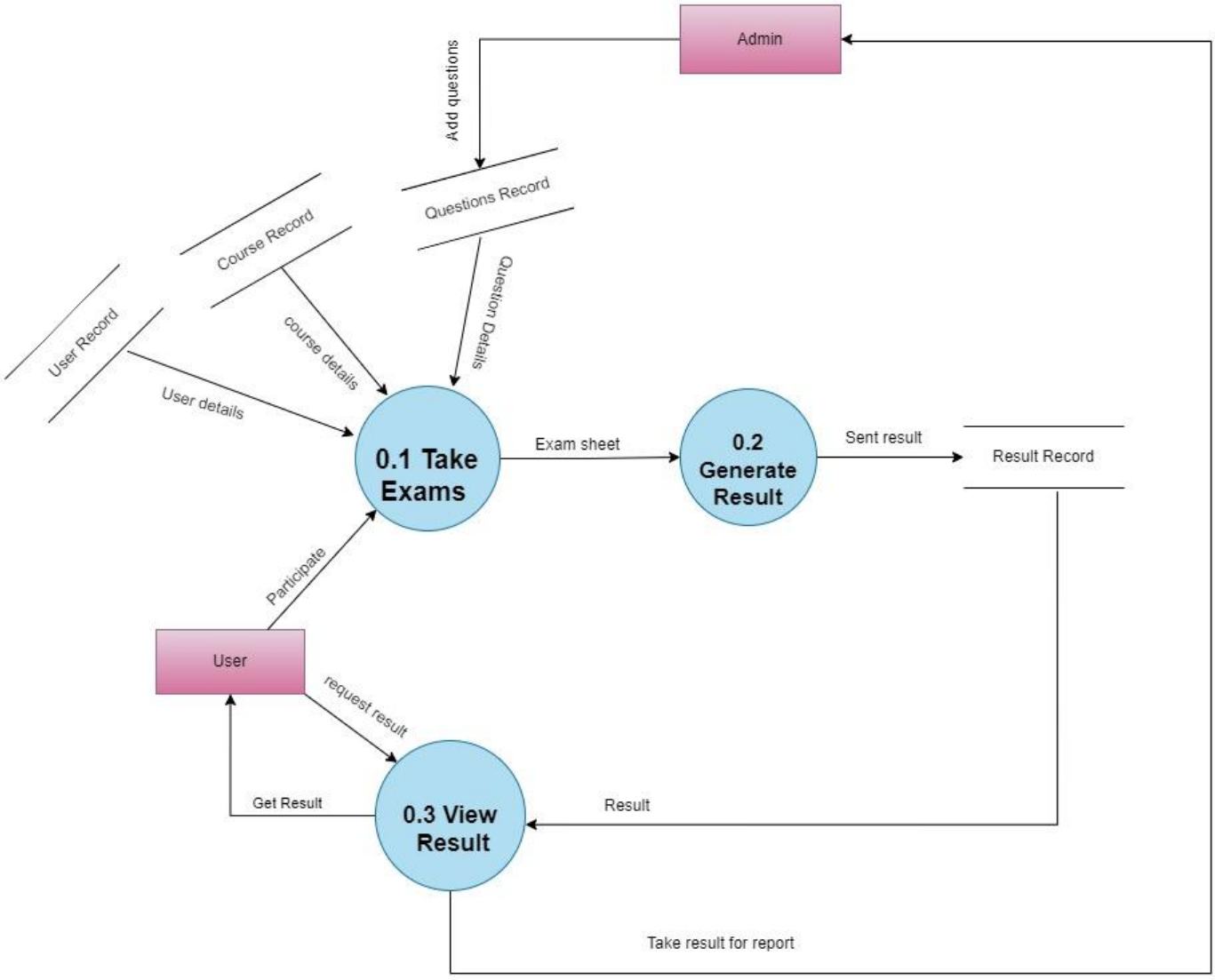


Figure 31: Level 1 DFD Take Certification Exams

Description:

User Engagement:

Data from Users: Users actively participate in exams, providing input during the exam-taking process. This input includes answers to questions presented in the exams.

Admin Management:

Data to Admin: The users' exam details, including their answers, are collected and stored in the system. Admins contribute to the system by adding questions to the Question Record Database and managing exam content.

Database Interactions:

User Record Database: This database stores information about users, including details related to their engagement in exams.

Course Record Database: Contains information about the specific courses for which exams are conducted.

Question Record Database: Admins contribute questions to this database, which are later used in generating exams for users.

Result Record Database: This database plays a pivotal role in storing the results generated by the system.

User Exam Participation: Users take part in exams, providing answers to questions.

Generate Result Process: The system processes the user responses, generates exam results, and stores them in the Result Record Database.

View Result Process: Users can access their exam results through this process, gaining insights into their performance.

Admin Reporting:

Admins utilize the stored results for reporting purposes, enabling them to assess user performance and make informed decisions.

In summary, the flow of data begins with user engagement in exams, moves through admin management and contribution to the system, and is then stored and processed in various databases. The Exam System ensures a structured and effective approach to conducting and managing certificate exams by facilitating smooth interactions between users and administrators

5.2.2.e Level 2 DFD of Take Certification Exams

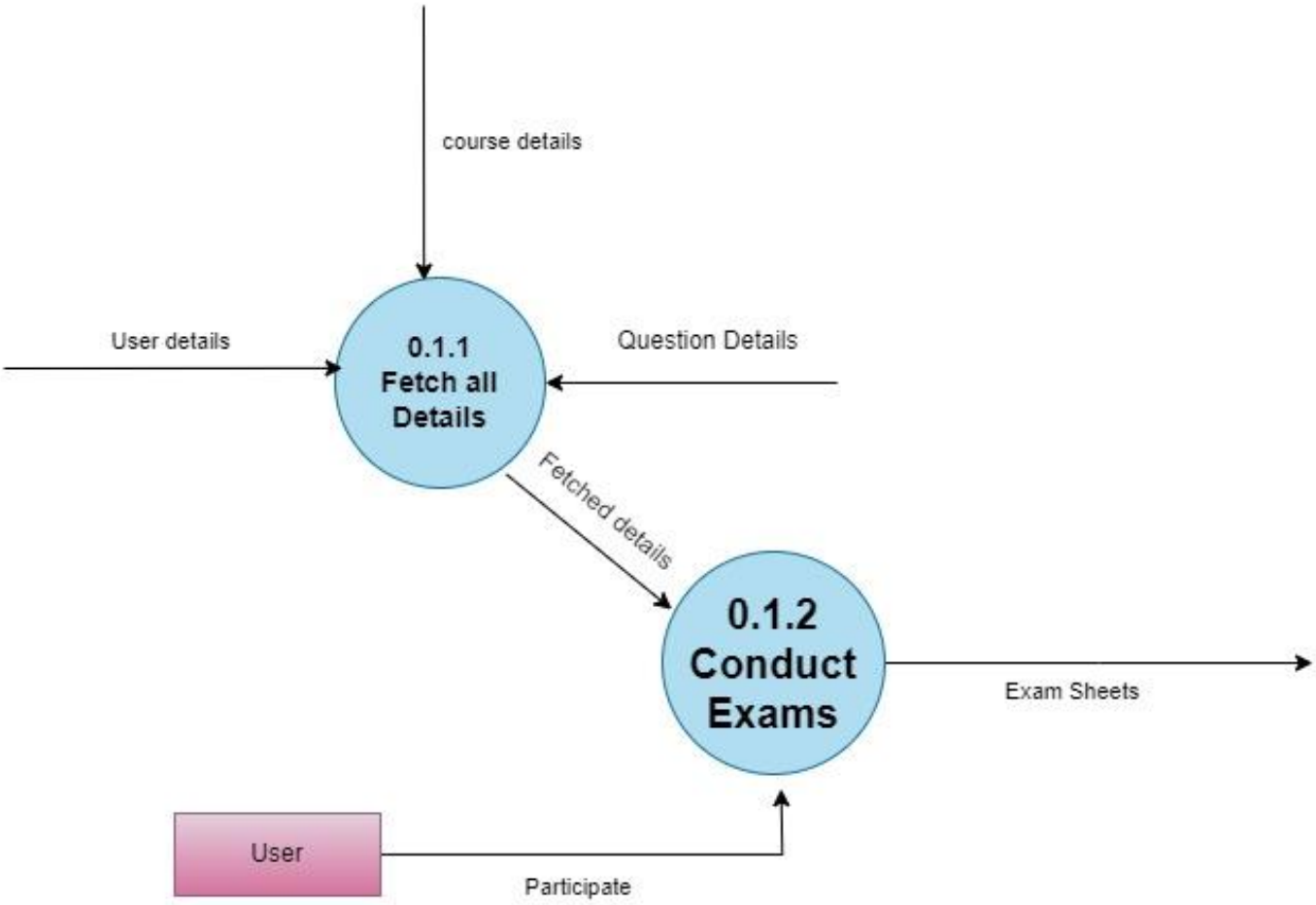


Figure 32: Level 2 DFD of Take Certification Exams

Description:

User Engagement:

Data from Users (Students): Students actively engage in taking exams to assess their understanding of specific courses, providing answers during the exam.

Admin Management:

Data to Admins: Admins continue to play a crucial role in managing exam content. They contribute to the system by defining exam requirements, including user information, course specifics, and questions for setting up exams.

Fetch all details:

This process is introduced at the second level to fetch essential details for exam setup. It gathers information such as user details, course specifics, and questions. This ensures that exams are tailored to the unique needs of users and courses.

Conduct Exam Process

Both admins and users actively participate in this process, contributing to a collaborative and comprehensive exam experience. Admins play a role in overseeing and managing the exam, while users provide their responses.

Following the exam, generated exam sheets move ahead in another process.in that process. The system processes the responses and generates detailed results, maintaining the flow established in the first level.

Structured Approach:

The second-level diagram provides a more detailed and beneficial system, highlighting a structured approach. This approach benefits both administrators and users in the assessment and learning process.

In summary, the data flow at the second level involves users actively participating in exams, all details required for exam is Fetched here, and a collaborative exam experience in the Conduct Exam Process. The system maintains its structured approach and provides detailed results through the Generate Result Process, ultimately benefiting both administrators and users in the assessment and learning process.

5.3.1.e Structure chart of Take Certification Exams:

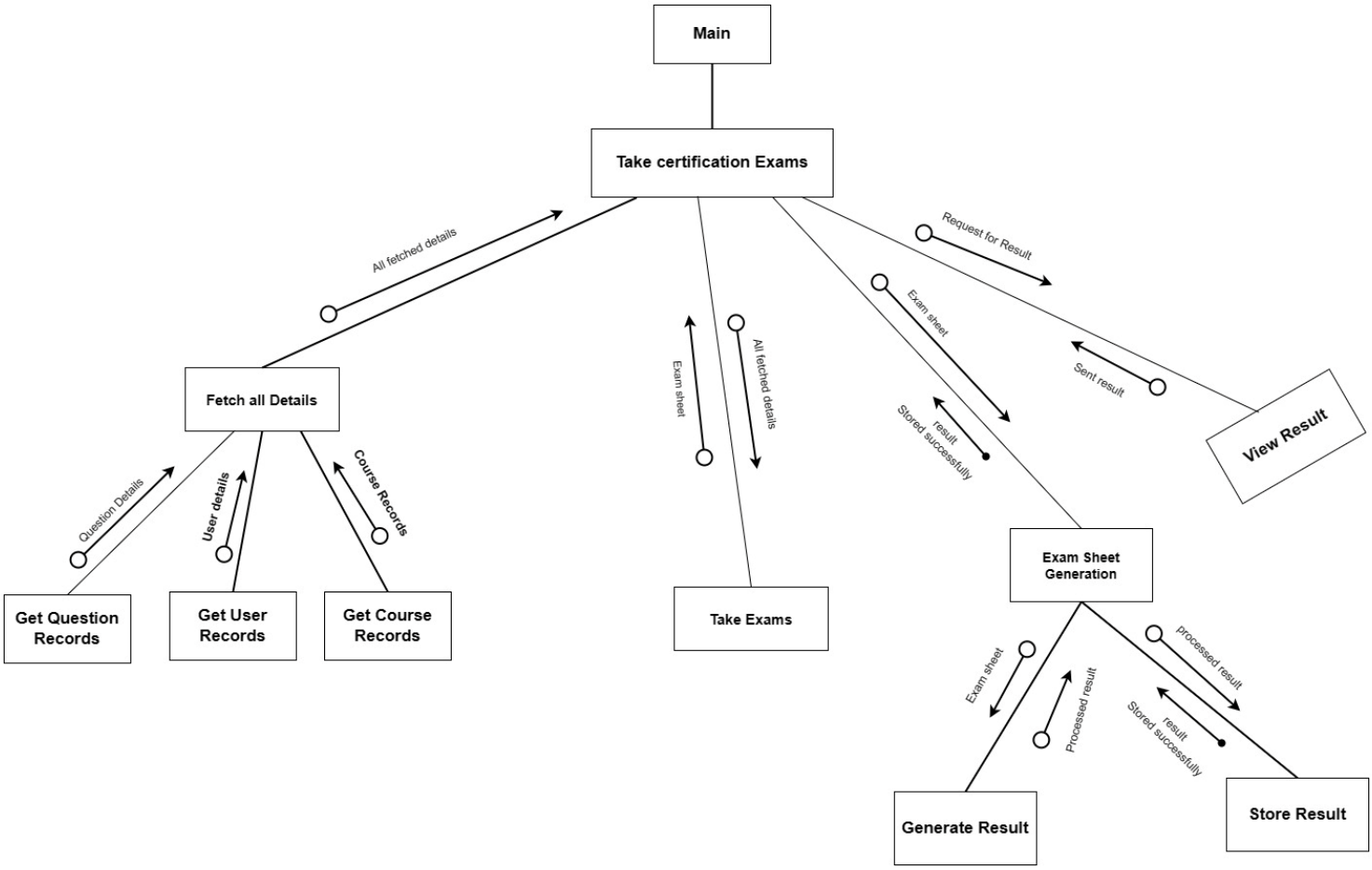


Figure 33: Structure chart of Take Certification Exams

Structure chart of whole system:

Description:

Fetch all details:

Get Question Records Module: Retrieves question details from the database.

Get User Records Module: Retrieves user details from the database.

Get Course Records Module: Retrieves course details from the database.

Integration in "Take Certification Exam" Module: All retrieved exam requirements are integrated back into the main "Take Certification Exam" module.

Certification Exam Process:

Take Certification Exam Module: Gathers all exam requirements from the sub-function. Users actively participate in the exam in the "Take Exams" module.

Return to "Take Certification Exam" Module: The generated exam sheet returns to the main "Take Certification Exam" module.

Result Generation and result store module:

Generate Result Module: The generated exam sheet is sent to this module, which prepares the exam results.

Store result module: This module store all generated result.

Result Viewing and Confirmation:

View Result Module: Exam results are directed to this module, allowing users to view their results.

Confirmation Signal: A confirmation signal is triggered by the user upon successfully viewing the results.

In summary, the process begins with the fetch all details sub-function, which gathers necessary details from different modules (question records, user records, and course records). The flow then moves through the main "Take Certification Exam" module, involving user interaction in the "Take Exams" module. The generated exam sheet proceeds to the "Generate Result" module, and the results are sent back to the "Take Certification Exam" module. The process concludes with users viewing their results in the "View Result" module, triggering a confirmation signal for a comprehensive and user-friendly certification exam experience. The interconnected flow ensures a smooth and well-structured process.

5.3.2.e Module Specs of Take Certification Exams

NAME	Take Certification Exams
PURPOSE	The module aims to optimize user engagement in certification exams by efficiently managing exam details, questions, and user information. It ensures a seamless process from input to result generation which enhance the overall user experience.
PSEUDOCODE	***DO*** *// Retrieve question records from the database* *Var question =DB.get_question_records()* *// Retrieve user records from the database* *Var user =DB.get_ user_records()* *// Retrieve course records from the database* *Var course=DB.get_ course_records()* *// fetched all details for taking exams* *Var fetched_data=fetch_all_data(users,*

	question, course) *//take exams* *Var exams=take exams(fetched_data)* *//sent exam sheet for result* *Var sheet=exam sheet generation(exams)* *//generate result* *Var sent_result=generate_result(sheet)* *//store result* *Var store =store_result(sent_result)* *//show final result to user using display function* *Var final_result=View Result(store)* *Display(final_result)* ***END DO***
INPUT PARAMETERS	users,question, course
OUTPUT PARAMETERS	Final_result
GLOBAL VARIABLE	DB
CALLS	• *fetch_all_data*

	• *take exams* • *exam sheet generation* • *generate_ result* • store _result • *View Result*
CALLED BY	MAIN

Table 9: Module Specs of Take Certification Exams

6. Summary:

We had succesfully developed a comprehensive software system for McGregor Institute of Botanical Training, an established institute in Ireland operating in Nepal for seven years.the institute aimed was to enhance its offerings by introducing short-term certification courses in horticulture, selling diverse plant varieties, and fostering a community platform for plant enthusiasts. This system was designed to streamline user interactions, including program enrollment, plant purchases, payments, expert recommendations, report preparation, certification exams, and forum participation. The utilization of concepts such as Data Flow Diagrams (DFD), Structure Chart, Entity Relationship Diagram (ERD), and various other software engineering principles, particularly the structured approach by Yourdon, formed the foundation for the system's development.

To complete this project,we consists of 5 students, who collaboratively tackled the project under the guidance of our respected tutor.Throughout the process, the team engaged in discussions, allowing for solving all the challenges and the formulation of logical assumptions where necessary. The incorporation of diverse perspectives and individual responsibilities within the team contributed to the successful completion of this task. Team members worked hard to create a project charter, Software Requirement Specification (SRS), environmental and internal model specifications, design specifications, and an assignment diary, which ensured a comprehensive and well-structured outcome.We are applying all structured software engineering concepts, such as Data Flow Diagrams, Structure Charts, and Entity Relationship Diagrams, to design a system that fulfill the institute needs. we collaboratively approached the project, utilizing a structured approach for creating data flow diagram(DFD).

Throughout the software creation process, our team encountered various challenges and issues. To address these, we raised open communication within the group for discussing concerns and brainstorming solutions collectively. we had carried out different extensive research such as consulting different articles to gain insights and alternative perspectives which enables us to find effective solutions to the identified issues.In tackling specific problems related to the creation of structure charts and Data Flow Diagrams (DFD), we sought guidance from our respected tutor. Additionally, we engaged in discussions with other team members, for diverse perspectives to deep dive into the challanges. This collaborative approach allowed us to identify potential solutions and ensure that the design components, including structure charts and DFDs, were clarified and optimized for the McGregor Institute's software system.

7. References

- Ali, N. and Lai, R. (2015). A method of software requirements specification and validation for global software development. *Requirements Engineering*, 22(2), pp.191–214. doi:https://doi.org/10.1007/s00766-015-0240-4.

- Benedusi, P., Cimitile, A. and De Carlini, U. (1992). Reverse engineering processes, design document production, and structure charts. *Journal of Systems and Software*, 19(3), pp.225–245. doi:https://doi.org/10.1016/0164-1212(92)90053-m.

- Benslimane, Y., Cysneiros, L.M. and Bahli, B. (2007). Assessing critical functional and non-functional requirements for web-based procurement systems: a comprehensive survey. *Requirements Engineering*, 12(3), pp.191–198. doi:https://doi.org/10.1007/s00766-007-0050-4.

- Casanova, M.A. and Amaral de Sa, J.E. (1984). Mapping Uninterpreted Schemes into Entity-Relationship Diagrams: Two Applications to Conceptual Schema Design. *IBM Journal of Research and Development*, 28(1), pp.82–94. doi:https://doi.org/10.1147/rd.281.0082.

- Guibijar, B.L. (2018). Data Flow Diagram DFD in Developing Online Product Monitoring System OPMS of DTI. *International Journal of Trend in Scientific Research and Development*, Volume-2(Issue-6), pp.1–7. doi:https://doi.org/10.31142/ijtsrd18394.

- Joshi, R.D. (2019). Software development for reliable software systems. *Journal of Systems and Software*, 3(2), pp.107–121. doi:https://doi.org/10.1016/0164-1212(83)90024-9.

- Kopp, A. and Orlovskyi, D. (2017). Development of tools to support data flow diagrams analysis process. *ScienceRise*, 12(1), pp.48–53. doi:https://doi.org/10.15587/2313-8416.2017.118799.
- Shi, M. (2010). Documenting Software Requirements Specification: A Revisit. *Computer and Information Science*, 3(1). doi:https://doi.org/10.5539/cis.v3n1p17.
- UC Merced Library (n.d.). *What Is a Data Dictionary?* | *UC Merced Library*. [online] library.ucmerced.edu. Available at: https://library.ucmerced.edu/data-dictionaries.
- Van Hentenryck, P., Saraswat, V. and Deville, Y. (1998). Design, implementation, and evaluation of the constraint language cc(FD). *The Journal of Logic Programming*, 37(1-3), pp.139–164. doi:https://doi.org/10.1016/s0743-1066(98)10006-7.
- Yeates, D. (1994). Software project management. *International Journal of Project Management*, 12(1), p.57. doi:https://doi.org/10.1016/0263-7863(94)90010-8.