Software Engineering Laboratory

(UGCA1924)

GIAN JYOTI INSTITUTE OF MANAGEMENT & TECHNOLOGY

PHASE 2, MOHALI

AFFILIATED TO

PUNJAB TECHNICAL UNIVERSITY, JALANDHAR

FOR THE PARTIAL FULFILLMENT FOR QUALIFYING BCA DEGREE

SUBMITTED TO:

Prof. Aakash Gautam

SUBMITTED BY:

Name -Suraj

Class - BCA IV (Sec-B)

Roll No. - 2111840

INDEX

Sr. No.	PRACTICAL'S	Pg.No.	Sign
1	Identify project scope and objective of given problem: a. College automation system. b. Banking Management System.	1-2	
2	Develop software requirements specification for (1 a.) and (1 b.) problem.	3-4	
3	Develop UML Use case model for a problem.	5-6	
4	Develop Class diagrams	7	
5	Represent project Scheduling of above-mentioned projects	8	
6	Use any model for estimating the effort, schedule and cost of software project	9-10	
7	Develop DFD model (level-0, level-1 DFD and Data dictionary) of the project	11-13	
8	Develop sequence diagram	14	
9	Develop Structured design for the DFD model developed	15-17	
10	Develop the waterfall model, prototype model and spiral model of the product	18-20	
11	Explain with reason which model is best suited for the product	21-22	
12	Develop a working protocol of any of two problem	23-24	
13	Use LOC, FP and Cyclomatic Complexity Metric of above-mentioned problem	25	
14	Find Maintainability Index and Reusability Index of above-mentioned problem	26	
15	Using any Case Tool find number of statements, depth and complexity of the prototype	27	

Q1. Identify project scope and objective of given problem:

a. College automation system.

Project Scope:

The College Automation System is an innovative software solution designed to automate and streamline various activities and processes involved in college management. The system should be accessible through a web-based platform that can be accessed by students, faculty, staff, and management. The application should be user-friendly, secure, and scalable, with modules for managing different functions of the college, including but not limited to:

1. Student information management, including enrollment, admissions, course registration, and academic performance tracking.
2. Faculty information management, including hiring, scheduling, course assignments, and evaluations.
3. Course management, including syllabus creation, scheduling, grading, and resource allocation.
4. Attendance management, including tracking, monitoring, and reporting.
5. Examinations and results management, including scheduling, grading, and analysis.

The College Automation System should provide a centralized platform for all stakeholders to manage their tasks and activities efficiently. The system should be capable of generating reports and analytics to aid management in making informed decisions.

Project Objective:

The primary objective of the College Automation System is to enhance the overall efficiency, productivity, and accuracy of college management processes. The system should help colleges to improve their operations by providing a comprehensive and integrated solution for automating various tasks and activities.

The College Automation System aims to provide the following benefits:

1. Improved efficiency: The system should reduce the time and effort required for managing administrative tasks, enabling staff to focus on other important tasks.
2. Improved accuracy: The system should reduce errors and improve accuracy by automating data entry and management, ensuring reliable data for decision-making.
3. Improved communication: The system should improve communication between stakeholders by providing a centralized platform for sharing information, enabling faster and better communication.
4. Improved decision making: The system should provide real-time data and analytics to help management make informed decisions, thereby improving the overall effectiveness of the college.
5. Enhanced student experience: The system should enhance the overall experience of students by providing better access to academic resources, enabling easy communication with faculty and staff, and facilitating better engagement with the college community.

b. Banking Management System.

Project Scope:

The Banking Management System is a comprehensive software solution that automates various activities and processes involved in banking operations, including managing customer information, account information, transactions, loans, deposits, withdrawals, foreign exchange, treasury operations, finance, and administrative tasks. The project scope involves developing a robust and user-friendly web-based application that can be accessed by bank employees and customers. The application should be secure, scalable, and customizable to meet specific business requirements. It should have modules for managing different functions of the bank, such as loan management, deposit management, and treasury operations.

Project Objective:

The objective of the Banking Management System is to streamline banking operations and reduce manual work, thereby improving efficiency, productivity, and accuracy. The system should provide real-time information to management, employees, and customers, enabling them to make informed decisions. In addition, the Banking Management System should provide the following benefits:

1. Enhanced security: The system should have robust security measures to protect sensitive information from unauthorized access and ensure compliance with regulatory requirements.
2. Customizable reporting and analytics: The system should allow for customizable reports and analytics to provide insights into key performance indicators, such as customer acquisition, loan disbursement, and deposit growth.
3. Seamless integration: The system should be seamlessly integrated with other third-party software, such as accounting and CRM software, to streamline business processes.
4. Improved customer experience: The system should provide customers with personalized services, such as online account management, loan application, and transaction tracking, improving customer satisfaction.
5. Efficient loan management: The system should automate the loan management process, including loan origination, underwriting, and servicing, reducing processing time and improving loan portfolio management.
6. Accurate forecasting: The system should use predictive analytics to forecast customer behavior, enabling banks to make informed decisions about product offerings, pricing, and marketing strategies.
7. Cost reduction: The system should reduce operating costs by automating repetitive tasks, such as data entry, and reducing the need for manual intervention.

Q2. Develop software requirements specification for (1 a.) and (1b.) problem.

- **College Automation System**:
  - **Functional Requirements:**

1. Placement management: manage internships, job opportunities, and career guidance services for students, and maintain partnerships with companies and alumni networks.
2. Research management: manage research projects, grants, and publications by faculty and students, and promote interdisciplinary collaboration and innovation.
3. Student management: register, view, update, delete, and search student information, including personal details, academic records, and extracurricular activities.
4. Faculty management: register, view, update, delete, and search faculty information, including personal details, qualifications, courses taught, and research interests.
5. Course management: register, view, update, delete, and search course information, including course code, title, description, prerequisites, and credits.
6. Attendance management: record attendance, view attendance reports, and notify absentees and their parents via email or SMS.
7. Examination management: schedule exams, record results, view exam reports, and generate certificates and transcripts.
8. Hostel management: manage rooms, manage room allocation, view hostel reports, and provide online room booking and cancellation services.
9. Finance management: manage fees, manage expenses, view financial reports, and provide online payment and receipt generation services.

10.Analytics and reporting: provide analytics and reports on student performance, faculty productivity, resource utilization, and institutional effectiveness, and support data-driven decision-making.

  - **Non-functional Requirements:**

1. User-friendly interface with intuitive navigation, responsive design, and customizable preferences.
2. Secure authentication and authorization with multi-factor authentication, encryption, and role-based access control.
3. Scalable architecture with modular components, microservices, and cloud deployment options.
4. Performance: system should handle large volumes of data and multiple simultaneous user requests with minimal latency and response time.
5. Maintenance and support: ensure the system is easy to maintain and support, with automated testing, debugging, and troubleshooting tools, and a dedicated support team.

- **Banking Management System:**
  - **Functional Requirements:**

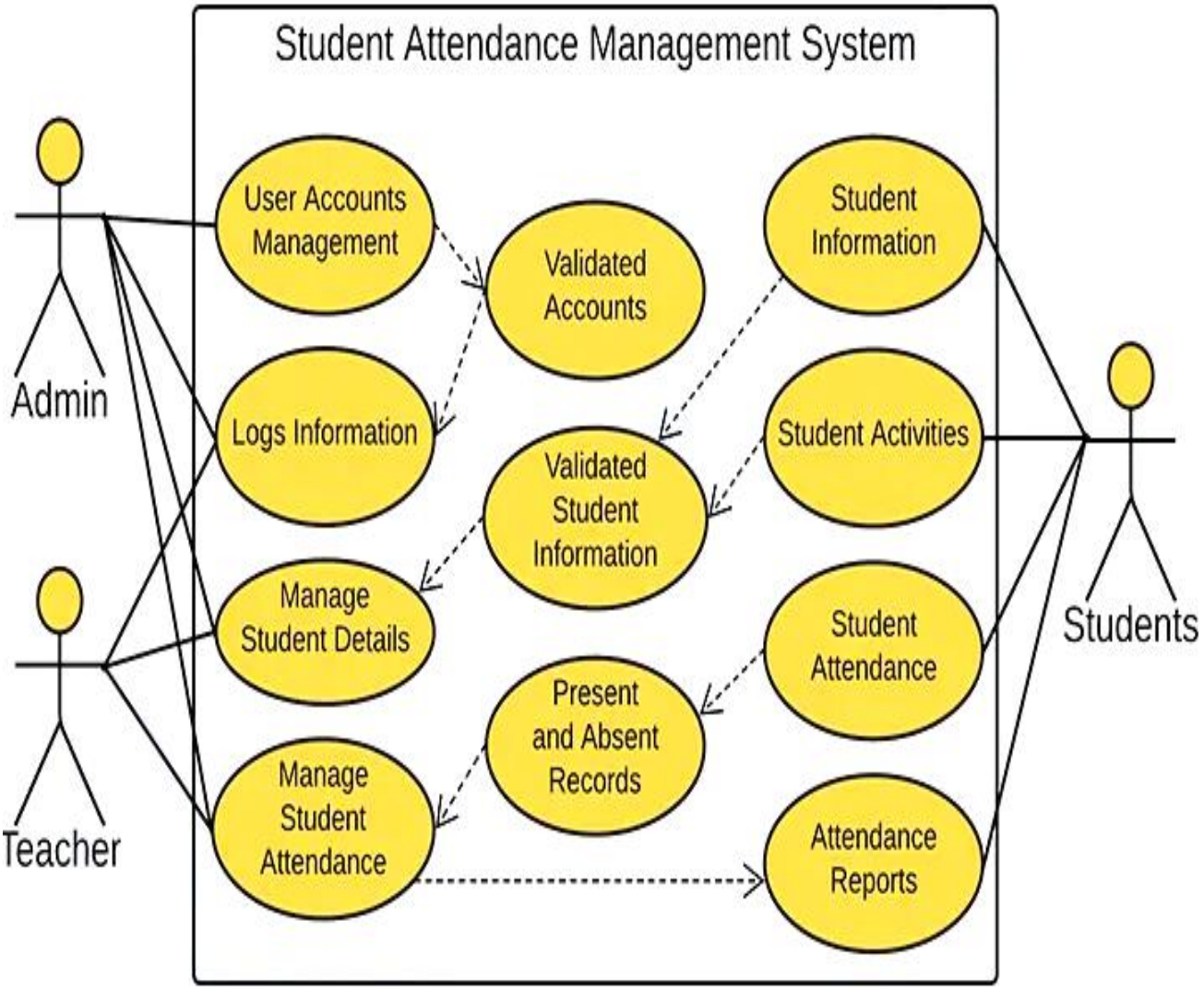
1. Customer management: register, view, update, delete, and search customer information, including personal details, contact information, and transaction history.
2. Account management: open, close, update, and search account information, including account type, balance, interest rates, and terms and conditions.
3. Transaction management: record transactions, view transaction history, and provide real-time notification of account activity through SMS or email.
4. Loan management: manage loan applications, approve or reject loans, view loan reports, and provide online loan payment and prepayment services.
5. Foreign exchange management: manage currency exchange rates, view foreign exchange reports, and provide online currency conversion and remittance services.
6. Branch management: manage branch operations, including staff management, inventory management, and customer service.

  - **Non-functional Requirements:**

1. User-friendly interface with intuitive navigation, responsive design, and customizable preferences.
2. Secure authentication and authorization with multi-factor authentication, encryption, and role-based access control.
3. High availability and reliability with load balancing, failover, backup, and disaster recovery mechanisms.
4. Real-time data processing and reporting with stream processing, data caching, and data visualization tools.
5. Performance: system should handle large volumes of data and multiple simultaneous user requests with minimal latency and response time.
6. Maintenance and support: ensure the system is easy to maintain and support, with automated testing, debugging, and troubleshooting tools, and a dedicated support team.

Q3. Develop UML Use case model for a problem.

STUDENT ATTENDANCE MANAGEMENT SYSTEM



USE CASE DIAGRAM

The use case diagram consists of three main elements: actors, use cases, and associations.

Actors:

Student: The student who uses the system to access their personal and academic information, as well as attendance records.

Teacher: The teacher who uses the system to manage course content, assignments, and attendance records.

Administrator: The administrator who manages the system, creates and manages user accounts, and generates reports.

Use Cases:

Login: The process of logging into the system with valid credentials.

View Student Information: Allows a student to view their personal information and academic records.

Update Personal Information: Allows a student to update their personal information such as their name, email address, and phone number.

View Course Content: Allows a teacher to view the course content they have created and uploaded.

Manage Course Content: Allows a teacher to add or delete course content, such as assignments and quizzes.

Take Attendance: Allows a teacher to take attendance for each class session.

View Attendance Records: Allows a student to view their attendance records for each course they are enrolled in.

Generate Reports: Allows an administrator to generate reports on student attendance and course performance.

Associations:

The Student actor is associated with the Login, View Student Information, and Update Personal Information use cases.

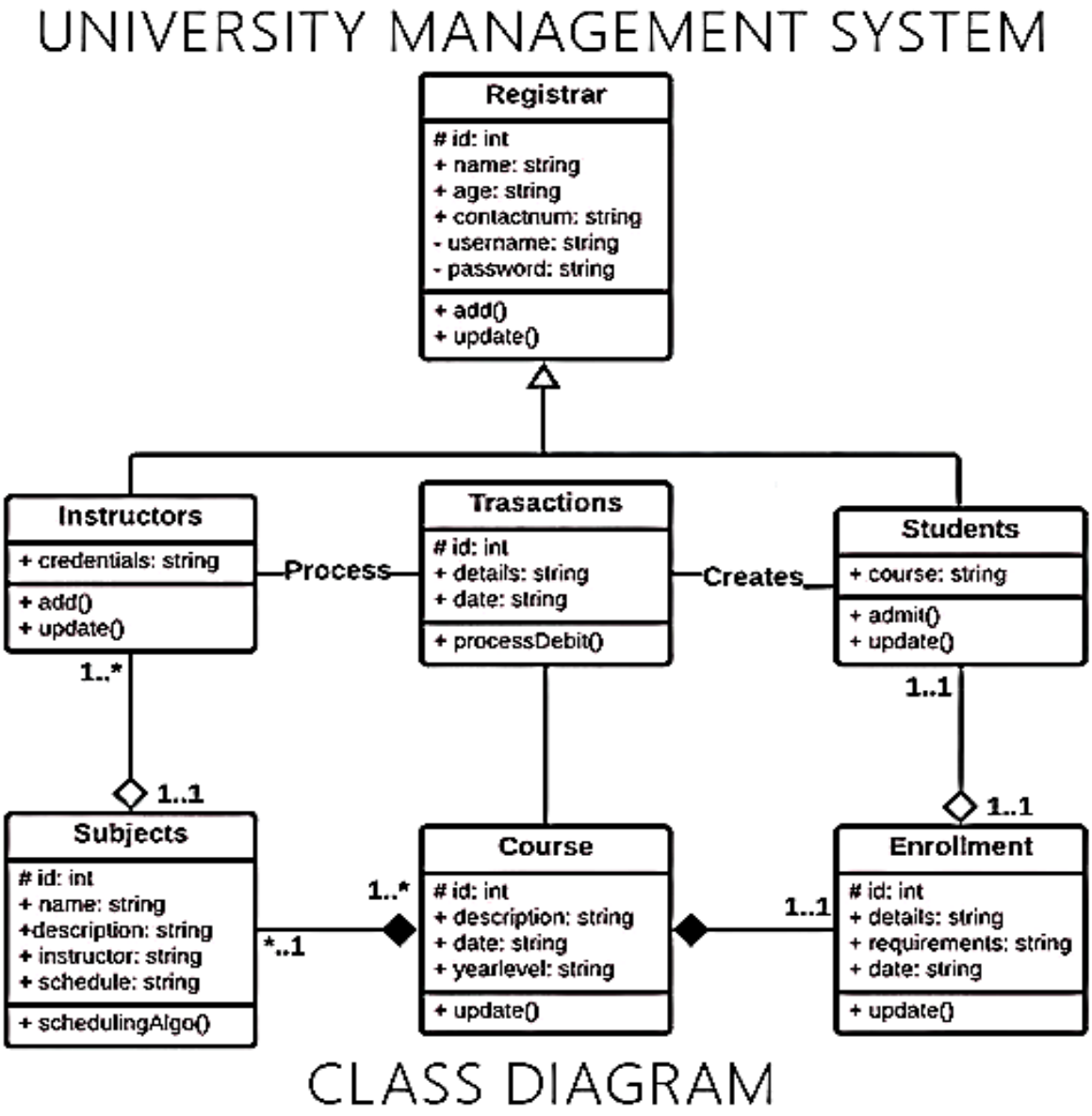
The Teacher actor is associated with the Login, View Course Content, Manage Course Content, Take Attendance, and Generate Reports use cases.

The Administrator actor is associated with the Login and Generate Reports use cases.

Overall, the use case diagram helps to visualize the different actors and their associated use cases in the Student Management System.

Q4. Develop Class diagrams

The Class diagram for University Management System shows the structures of information or data that will be handled in the project. These data or information will be represented by classes. Each of the classes will have its attributes in accord with the methods they will use. So the UML Class diagram was illustrated by a box with 3 partitions the upper part was the name of the class, the middles are the attributes and the bottom is for the methods. The arrows on them represent their relationships with each other.



So the classes that are included in University Management would the **students**, **enrollment**, **instructors**, **subjects**, **courses**, **registrar,** and **transaction**. The mentioned classes were just general. If you want a more complex or wider scope of your University management system, then you can add your desired classes. You must also include the database on your Class Diagram for your system.

Q5. Represent project Scheduling of above-mentioned projects.

PROJECT SCHEDULE

PROJECT NAME: College Automation Systeam

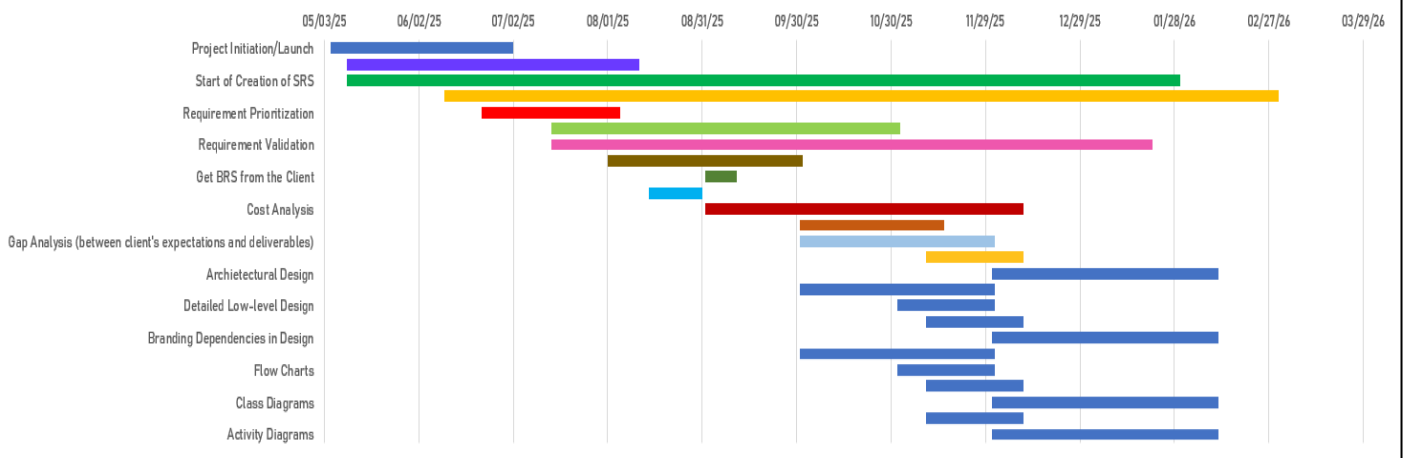
PROJECT MANAGER: Suraj Verma

START DATE: Monday, May 5, 2025

END DATE: Tuesday, February 10, 2026

		TIMELINE			PROJECT NOTES				
	PHASE TITLE	START DATE	END/DUE DATE	DURATION in days	STATUS (In progress, completed, on hold, not started)	BUDGET	HUMAN RESOURCES ASSIGNED/TASK OWNER	RISKS	COMMENTS
Phase - Project Start	Project Conception	01/01/23	01/01/23	1	completed	$ 1,000.00	Amit Bhandari	Very low	
	Project Definition and Initial Planning	01/02/23	01/03/23		completed	$ 2,500.00	Krishan Murari	Low	
	Project Initiation/Launch	05/05/25	07/01/25	58	on hold	$ 300,000.00	Gurinder Singh	Very High	
Phase - Requirment Engineering	Requirement Gathering & Elicitation	05/10/25	08/10/25	93	completed	$ 60,000.00	Amit Bhandari	Intermediate	
	Start of Creation of SRS	05/10/25	25/10/25	265	In progress	$ 5,000.00	Krishan Murari	Intermediate	
	Requirement Categorization	06/10/25	03/01/26	265	In progress	$ 2,500.00	Krishan Murari	Very High	
	Requirement Prioritization	06/22/25	08/04/25	44	In progress	$ 2,500.00	Krishan Murari	Very High	
	Requirement Categorization	07/14/25	11/01/25	111	In progress	$ 5,000.00	Amit Bhandari	Very High	
	ER Diagrams	11/10/25	12/10/25	31	completed	$ 2,500.00	Aakash rawath	Low	
	Activity Diagrams	12/01/25	02/10/26	72	completed	$ 2,500.00	Aakash rawath	Low	
	Requirement Documentation	08/01/25	10/01/25	62	not started	$ 2,500.00	Aakash rawath	Very High	
	Requirement Documentation	08/01/25	10/01/25	62	not started	$ 2,500.00	Aakash rawath	Very High	
	Get BRS from the Client	09/01/25	19/01/25	10	not started		Aakash rawath	Intermediate	
Phase - Analysis	Feasibility Analysis	08/14/25	08/30/25	17	completed	$ 5,000.00	Amit Bhandari	High	
	Cost Analysis	09/01/25	12/10/25	101	completed	$ 3,000.00	Amit Bhandari	High	
	Risk Analysis	10/01/25	11/15/25	46	completed	$ 2,000.00	Krishan Murari	High	
	Gap Analysis (between client's expectations and deliverables)	10/01/25	12/01/25	62	completed	$ 1,000.00	Krishan Murari	High	
	Start of Creation of SRS	11/10/25	12/10/25	31	In progress	$ 2,500.00	Gurinder Singh	Low	
Phase - Design	Archietectural Design	12/01/25	02/10/26	72	not started	$ 2,500.00	Gurinder Singh	Very High	
	High-level Design	10/01/25	12/01/25	62	In progress	$ 5,000.00	Amit Bhandari	Very High	
	Detailed Low-level Design	11/01/25	12/01/25	31	In progress	$ 2,500.00	Krishan Murari	Very High	
	Interface Design	11/10/25	12/10/25	31	In progress	$ 2,500.00	Krishan Murari	Very High	
	Branding Dependencies in Design	12/01/25	02/10/26	72	In progress	$ 2,500.00	Amit Bhandari	Very High	
	User Interface Design	10/01/25	12/01/25	62	completed	$ 5,000.00	Amit Bhandari	High	
	Flow Charts	11/01/25	12/01/25	31	completed	$ 2,500.00	Gurinder Singh	Low	
	Algorithms	11/10/25	12/10/25	31	completed	$ 2,500.00	Aakash rawath	Low	
	Class Diagrams	12/01/25	02/10/26	72	completed	$ 2,500.00	Aakash rawath	Low	

DELIVERY TIMELINE



Q6.Use any model for estimating the effort, schedule and cost of software

One widely used model for estimating effort, schedule and cost of a software project is the COCOMO (Constructive Cost Model) model. COCOMO was developed by Barry Boehm in 1981 and is based on the assumption that software development effort is proportional to the size of the software product and the complexity of the project.

COCOMO has three variants: Basic COCOMO, Intermediate COCOMO, and Detailed COCOMO. The Basic COCOMO model is the simplest and uses a set of linear equations to estimate effort, schedule, and cost based on the size of the project in lines of code (LOC). The Intermediate COCOMO model considers additional factors such as the complexity of the project, the experience of the development team, and the type of software being developed. The Detailed COCOMO model includes even more factors, such as the size and complexity of the database, the number of users, and the use of new or emerging technologies.

here's an example of using the Detailed COCOMO model to estimate the effort, schedule and cost of a software project:

Assumptions:

- Size of the project: 50,000 LOC
- Development team experience: High
- Type of software: Web application
- Database size: Large
- Number of users: 10,000
- Use of new or emerging technologies: No

Using the Detailed COCOMO model, we can estimate the effort, schedule and cost of the project as follows:

Calculate the effort in person-months:

Effort = a * (KLOC)^b * (SF)^E where a = 2.94, b = 1.1 for web application, SF = 1.22 for large database, E = 0.3 for high experience

Effort = 2.94 * (50)^1.1 * 1.22^0.3 = 110.7 person-months

Calculate the duration in months:

Duration = c * (Effort)^d where c = 3.67, d = 0.28 for web application

Duration = 3.67 * (110.7)^0.28 = 14.5 months

Calculate the cost in USD:

Cost = Effort * Hourly rate where Hourly rate = $100 (assumed)

Cost = 110.7 * 160 * $100 = $1,777,920

So, according to the Detailed COCOMO model, the estimated effort for the project is 110.7 person-months, the estimated duration is 14.5 months, and the estimated cost is $1,777,920. Note that this is just an estimate and the actual effort, schedule and cost may vary depending on various factors such as the accuracy of the requirements, the complexity of the project, and the skill level of the development team.

Q7. Develop DFD model (level-0, level-1 DFD and Data dictionary) of the project

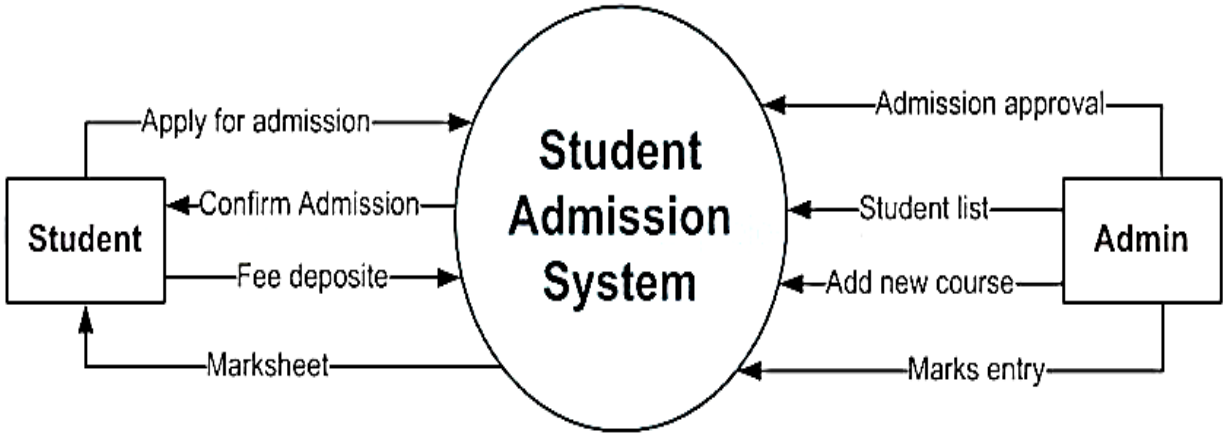
level-0

Context level DFD for Student Admission System

Level 0 is also known as context DFD. This diagram describes the highest level of the student admission system. This level shows the entire system as a single process and its relationship with external entities.

There are two main external entities in student admission data flow diagram:

- Student
- Admin



Context Level DFD for Student Admission System

Level 0 DFD for Student Admission System

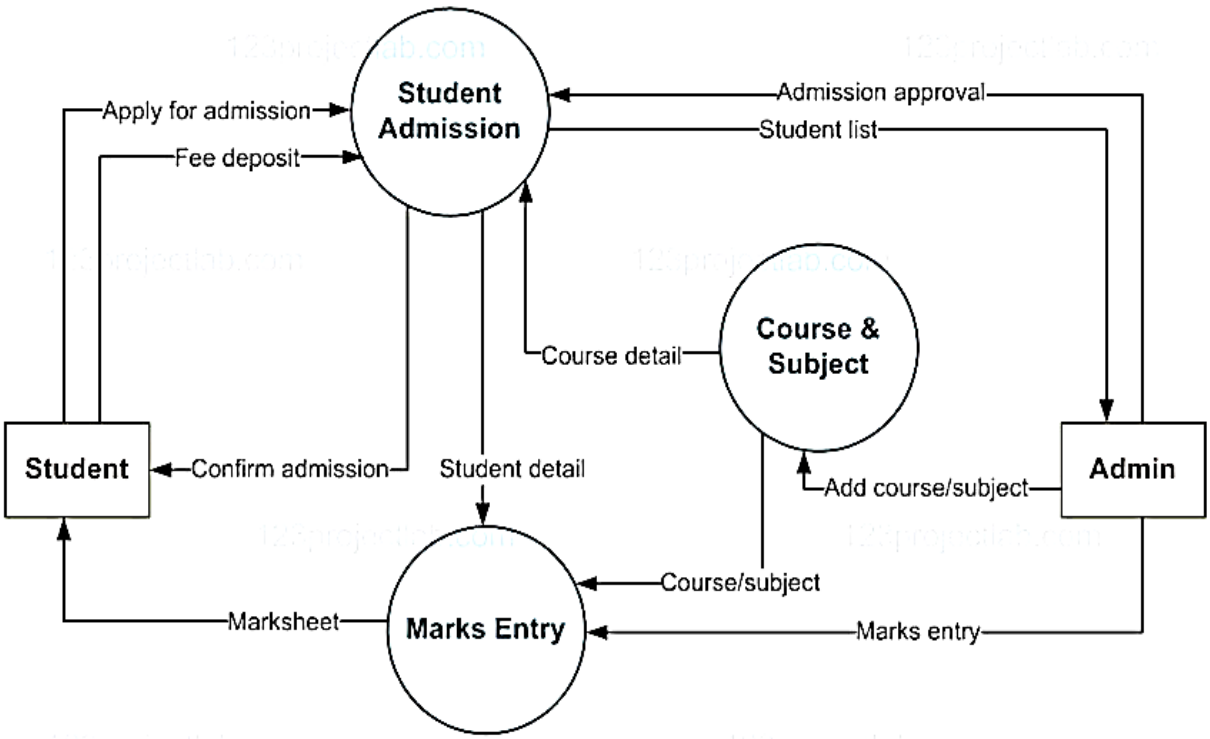
Level 1 DFD for Student Admission System

Level 1 DFD gives a more detailed view of the system as compared to level 0. In this level the single process is divided into multiple sub processes. This level also shows how data flows between sub processes. Student Admission system level 1 DFD has following sub processes:

Student Admission: This process takes all student records as input and maintains this record for future use. Admin gives approval on student admission applications. After approval, students can submit fee.

Course & Subject: This process maintains a record of all courses and modules of each course. Admin can add new courses and modules. If there is any change in curriculum then he can change or add a new module.

Marks Entry: As the name indicates, this process is related to maintaining student marks. Admin enters each student's marks obtained in the examination. Each student mark sheet is prepared and students can view this and download.



Level-1 DFD for Student Admission System

Level 1 DFD for Student Admission System

Data dictionary for DFD For Student Admission System

The data dictionary provides a description of each field used in the Student admission system diagram. It contains the following details:

1. StudentID: A unique identification number for each student. Data type is text with a field size of 8 characters.
2. First_Name: The first name of the student. Data type is text with a field size of 20 characters.
3. Last_Name: The last name of the student. Data type is text with a field size of 20 characters.
4. Year Group: The year level the student is in. Data type is number with a field size of 2 digits.
5. Date_of_Birth: The date of birth of the student. Data type is date/time with a format of DD/MM/YYYY and a field size of 10 characters.
6. Student_Image: A profile photo of the student. Data type is image with a format of .jpg and a field size of 10 characters.
7. School_Team: The coloured team the student is assigned to. Data type is text with a field size of 10 characters.

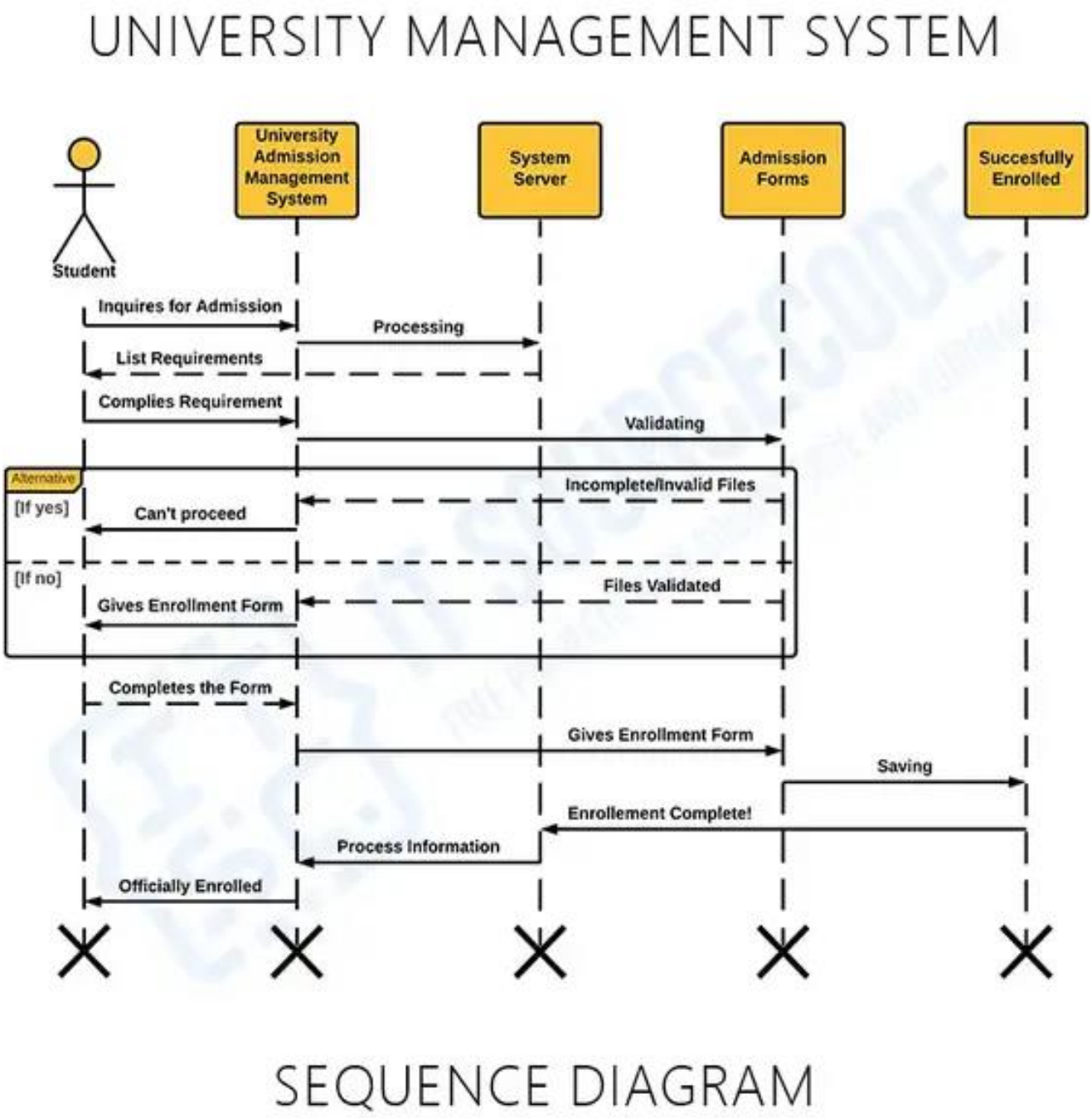
This data dictionary provides a clear understanding of each field used in the Student admission system diagram, which helps in the design, development, and maintenance of the system.

Data Dictionary: Student Details

Field Name	Data Type	Data Format	Field Size	Description	Example
StudentID	Text	XNNNNNNN	8	Unique Identification for each Student	S1234567
First_Name	Text		20	Student's First Name	Connor
Last_Name	Text		20	Student's Surname	Thomas
Year_Group	Number	##	2	Year level the student is in	8
Date_of_Birth	Date/Time	DD/MM/YYYY	10	Date the Student was born	14/08/2005
Student_Image	Image	.jpg	---	Profile photo of the Student	---
School_Team	Text		10	Coloured team student assigned	Blue

Q8. Develop sequence diagram

The designed sequence diagram illustrates the series of events that occurs in the University Management System. In this illustration, the actors are represented by a stick man, and the transactions or classes are represented by objects. It will give you a clear explanation of the behavior of a University Management System in terms of processing the flow of instructions.



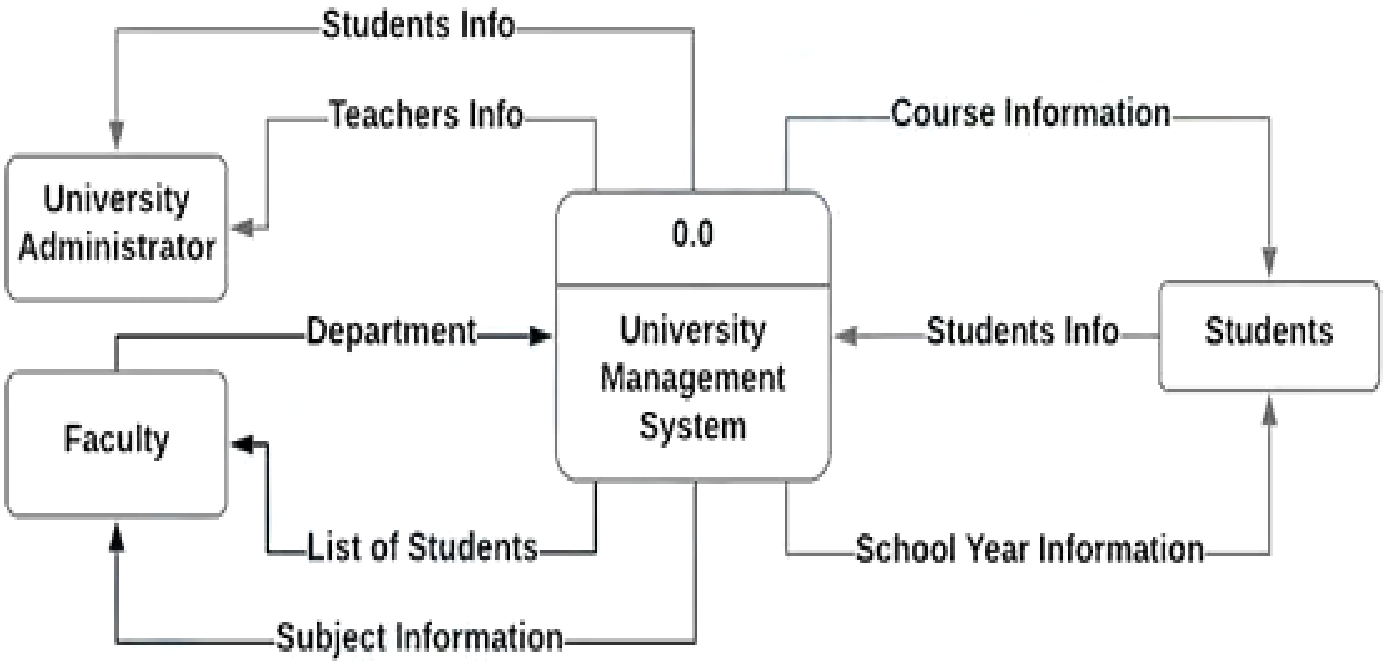
University Management System Sequence Diagram

This designed sequence diagram can show programmers and readers the sequence of messages between the actor and the objects.

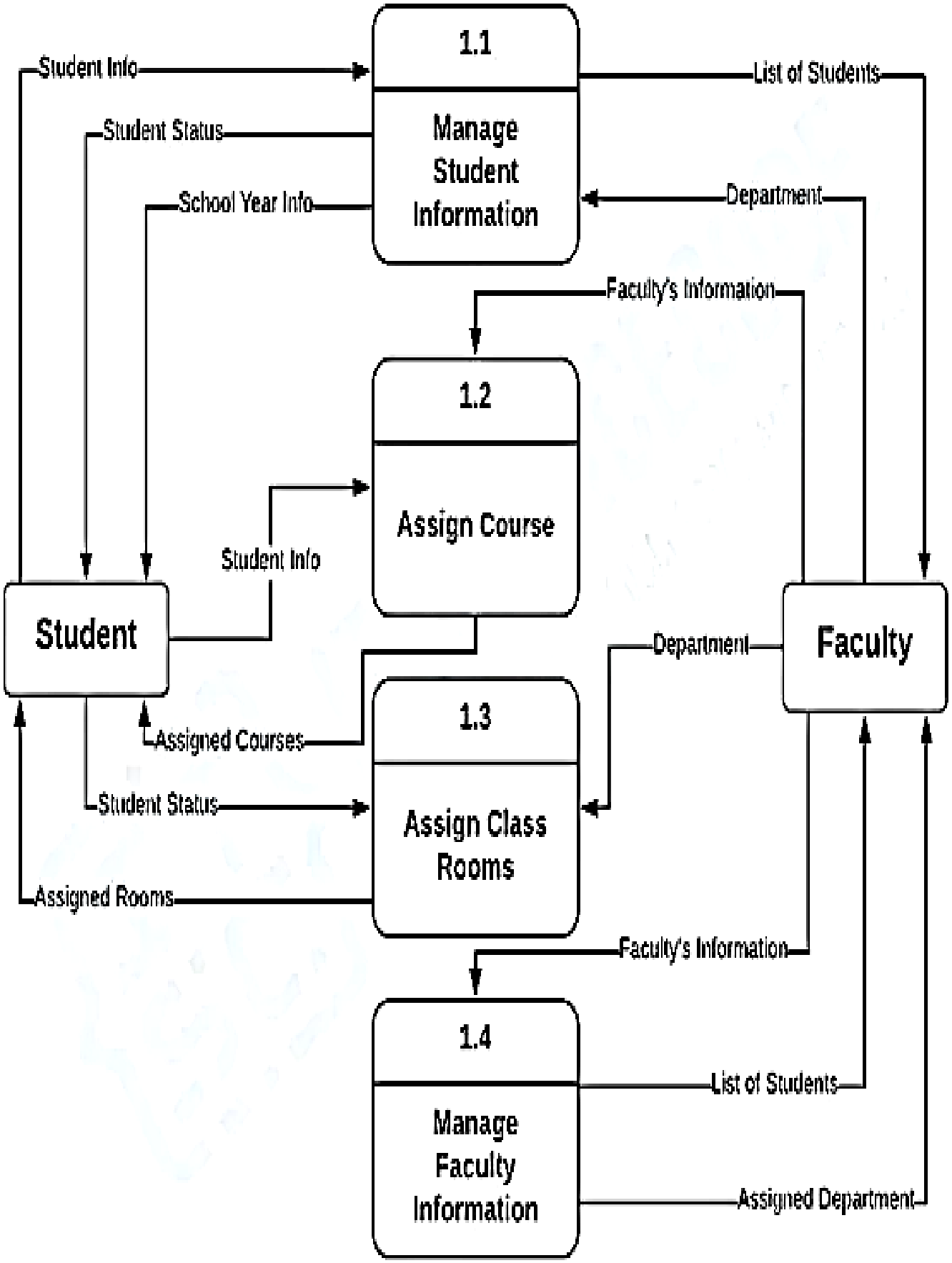
Q9. Develop Structured design for the DFD model developed

University Management System DFD

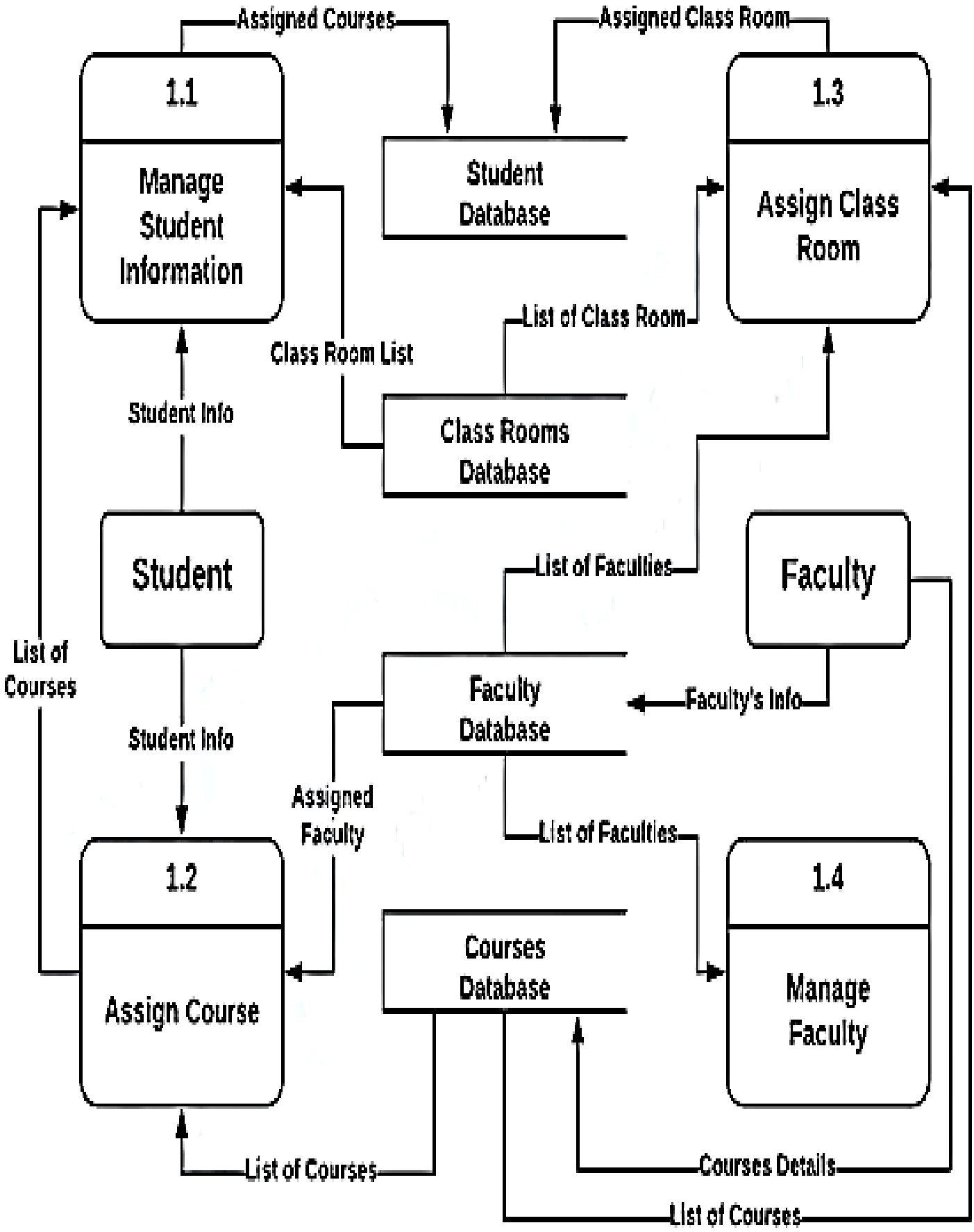
In an addition to the diagrams that will be a big help in doing your Project is the Data Flow Diagram. It does not belong to the UML diagrams but it also helps in knowing more about the University Management System. So the DFD serves as an illustration of the systems data handling. It has levels 0, 1, and 2.



University Management System DFD Level 0



University Management System DFD Level 1



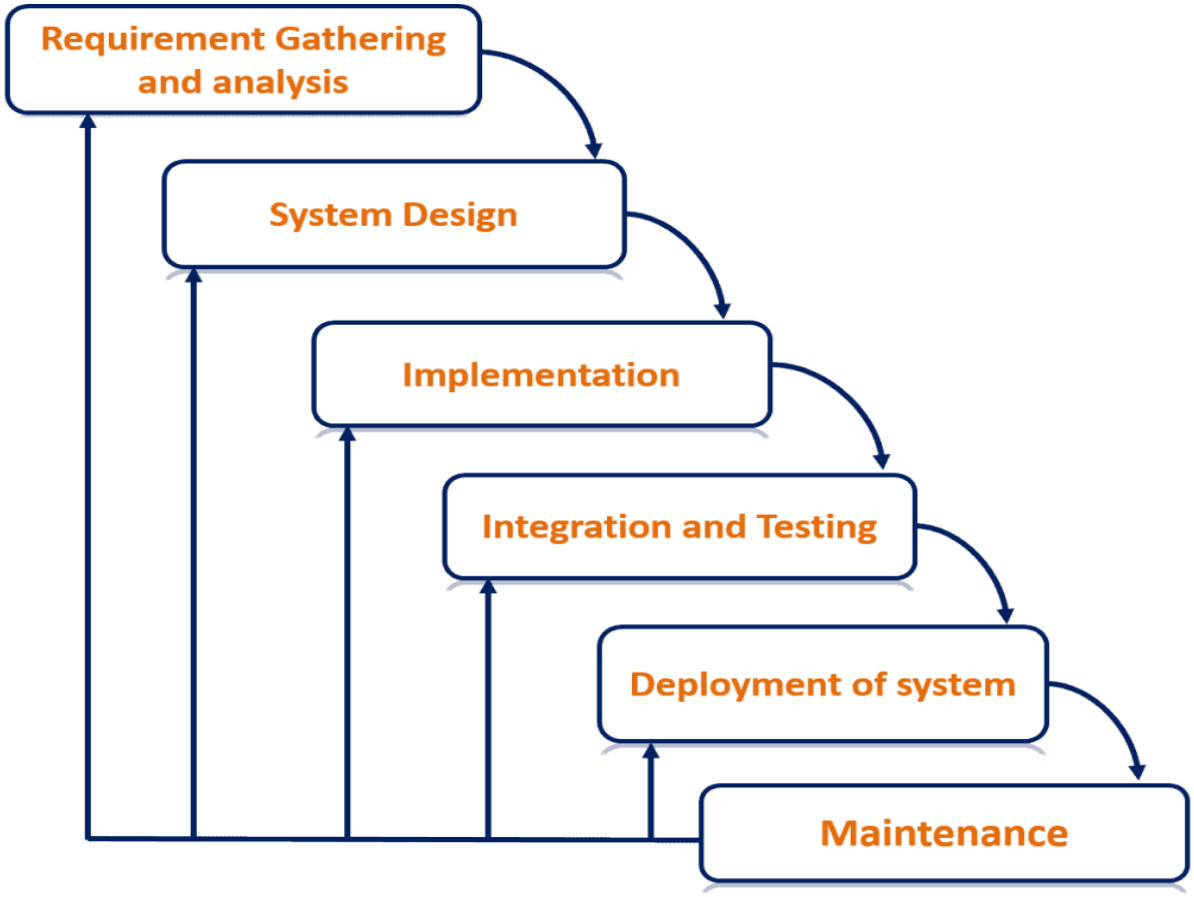
University Management System DFD Level 2

10. Develop the waterfall model, prototype model and spiral model of the product.

1. Waterfall Model

The waterfall model is a linear sequential approach to software development, where each stage of the process flows downwards like a waterfall. The stages in the waterfall model are executed in a linear order and each stage must be completed before proceeding to the next. The stages are:

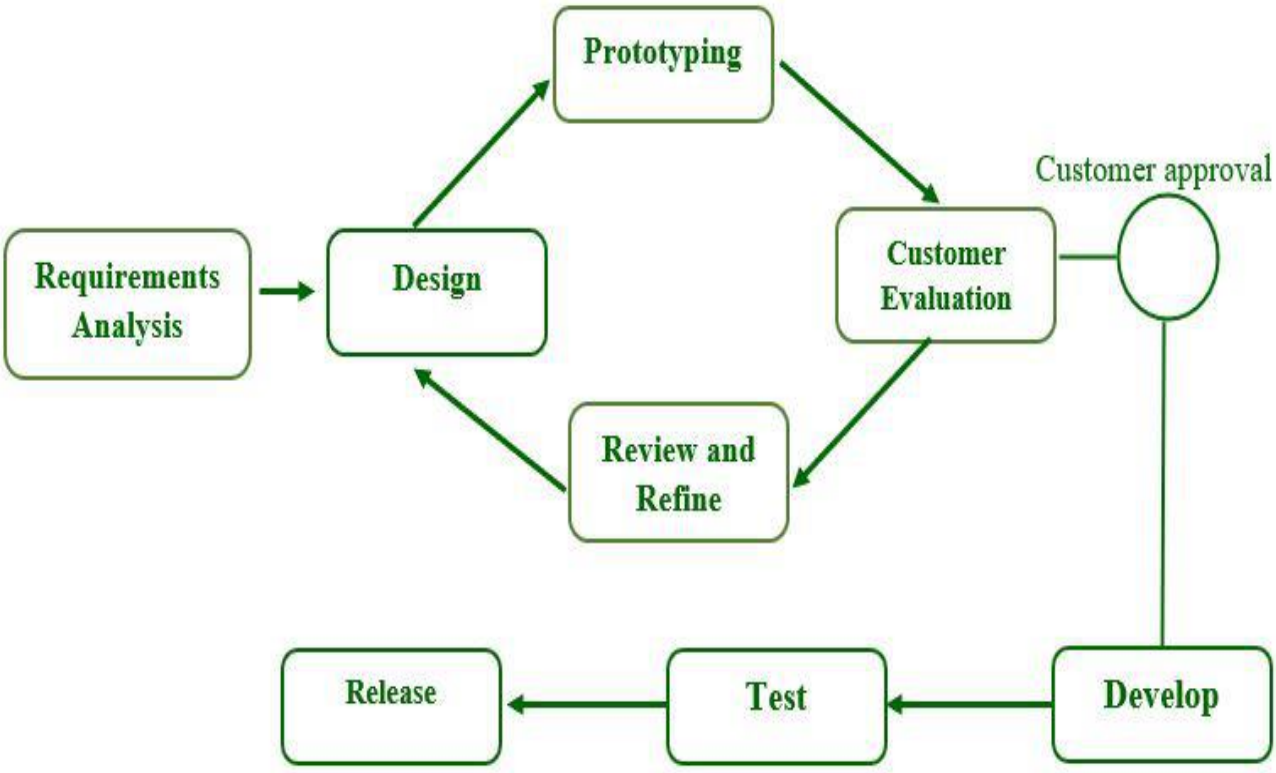
1. Requirements gathering: In this stage, the requirements of the college automation system are gathered from stakeholders, including students, faculty, and administrators.
2. Analysis: The requirements are analyzed, and a detailed understanding of the system's functionalities is developed.
3. Design: The system design is developed, including the architecture, user interface, and database design.
4. Implementation: The system is developed based on the design specifications.
5. Testing: The system is thoroughly tested to ensure that it meets the functional and non-functional requirements.
6. Deployment: The system is deployed to production.
7. Maintenance: Ongoing maintenance and support is provided to ensure the system remains functional.



2. Prototype Model:

The prototype model is an iterative approach to software development, where a prototype of the system is developed, and feedback is gathered from stakeholders to refine and improve the system. The stages in the prototype model are:

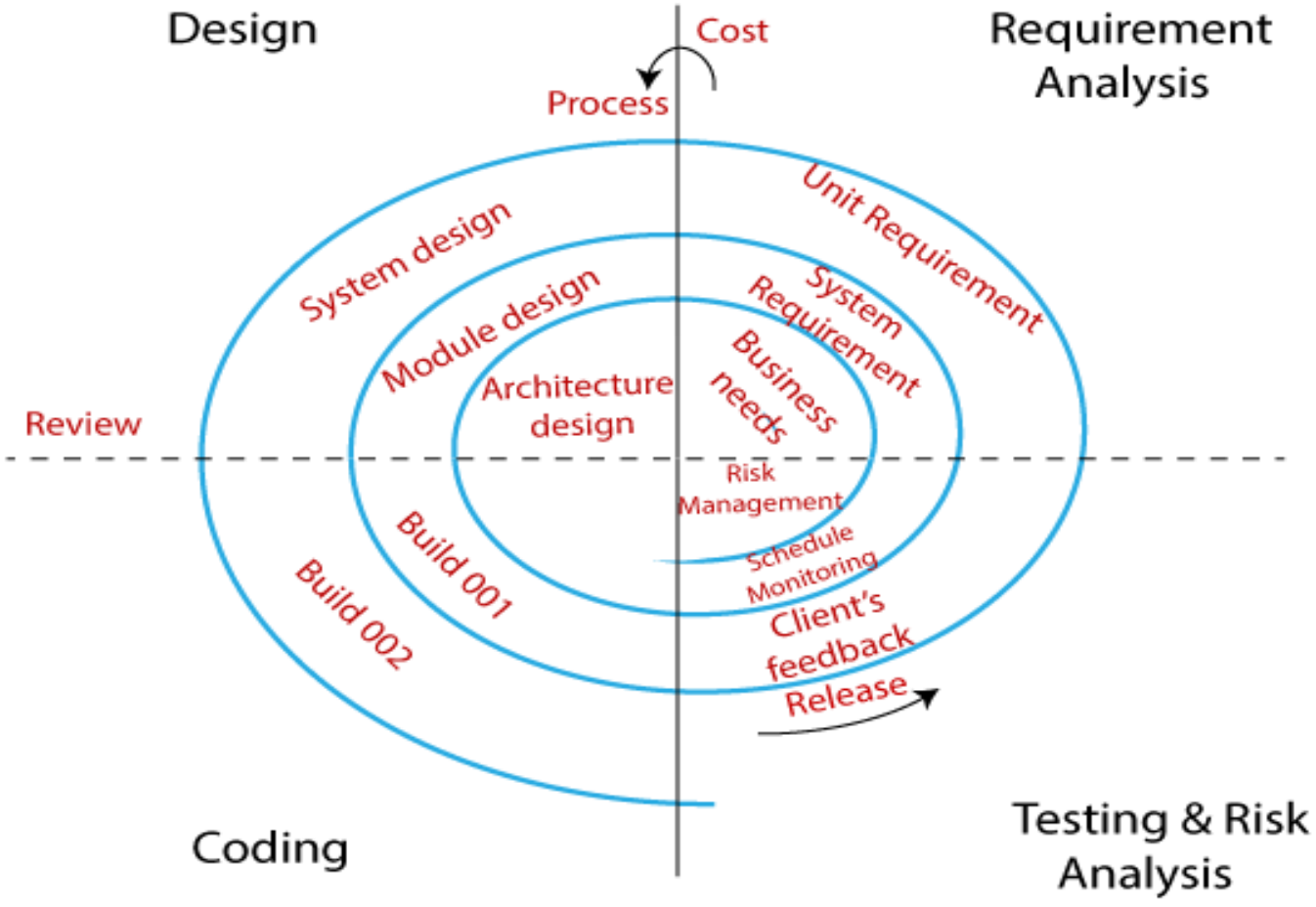
1. Requirements gathering: In this stage, the requirements of the college automation system are gathered from stakeholders, including students, faculty, and administrators.
2. Design: A basic design of the system is developed based on the requirements.
3. Prototype development: A working prototype of the system is developed.
4. Feedback gathering: Feedback is gathered from stakeholders on the prototype.
5. Refinement: Based on the feedback, the prototype is refined and improved.
6. Repeat steps 4 and 5 until the final system is developed.



3. Spiral Model:

The spiral model is a risk-driven model that emphasizes flexibility and adaptability. The model consists of four main phases, which are executed in a spiral pattern:

1. Determine objectives: In this stage, the objectives of the system are defined.
2. Risk analysis: Risks associated with the development of the system are identified.
3. Development: The system is developed based on the objectives and risk analysis.
4. Testing: The system is thoroughly tested to ensure that it meets the functional and non-functional requirements.
5. Evaluate: The progress of the project is evaluated, and feedback is gathered from stakeholders.
6. Repeat steps 2-5 until the final system is developed.



11. Explain with reason which model is best suited for the product

Spiral Model: The Spiral Model is an iterative approach that combines elements of both the Waterfall and Prototype models. This model is best suited for complex and long-term projects like the College Automation System, where frequent feedback from customers and stakeholders is required.

Spiral Process Model

The Spiral Model involves the following stages:

1. Planning: In this stage, the project is planned, and requirements are gathered.
2. Risk Analysis: In this stage, potential risks are identified and analyzed. Strategies are developed to mitigate these risks.

3. Prototype: In this stage, a basic prototype is developed based on the requirements and risk analysis. This prototype can be used to test and refine the requirements.
4. Design: In this stage, a detailed design of the system is created based on the requirements and feedback from stakeholders.
5. Implementation: In this stage, the system is built and tested.
6. Evaluation: In this stage, the system is evaluated, and feedback is gathered from stakeholders.

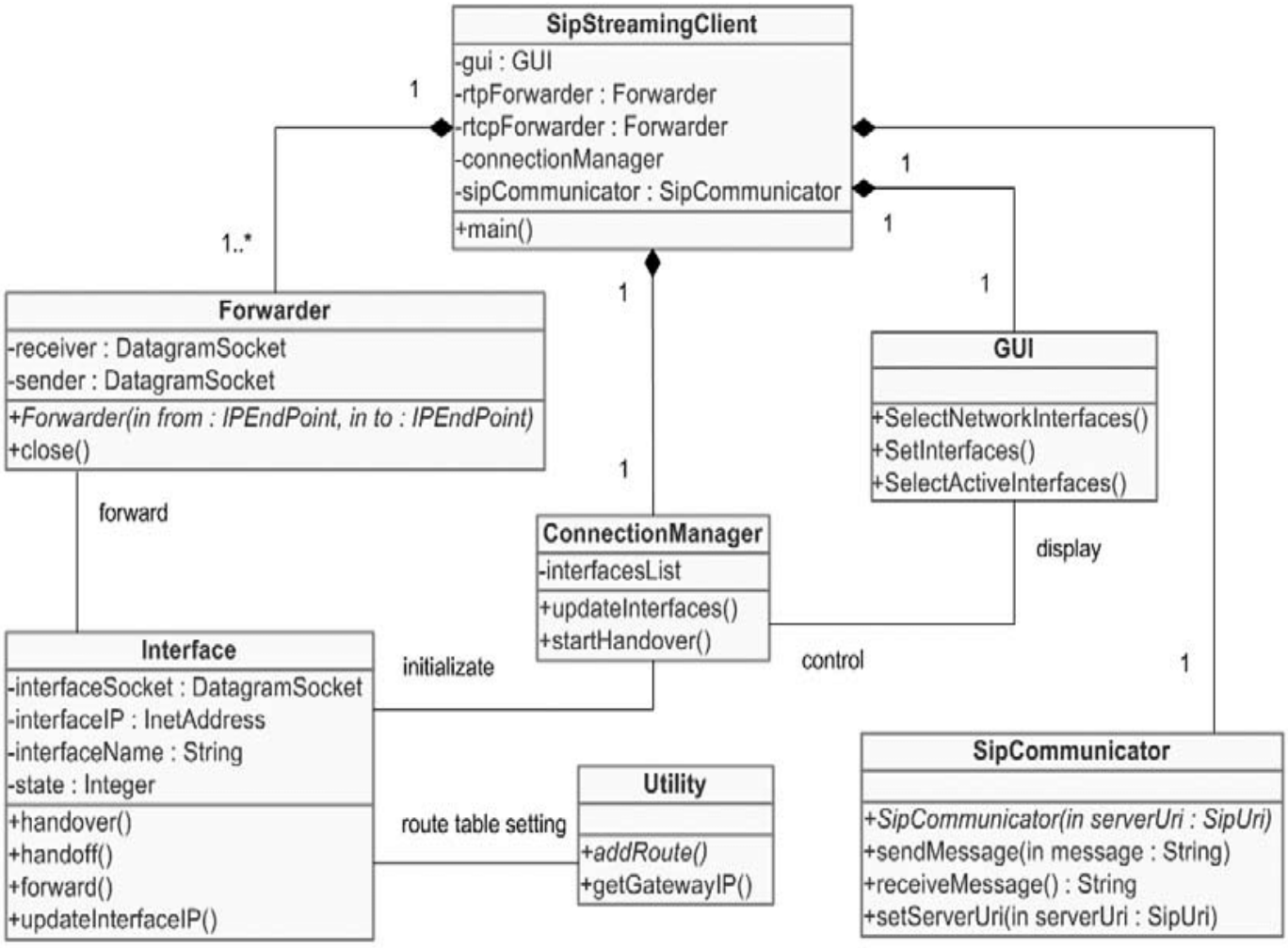
For a College automation system, the Spiral model may also be a good fit. It allows for a more flexible approach to development than the Waterfall model, but it also includes risk analysis and planning to help mitigate potential issues. The iterative nature of the model allows for frequent feedback from customers and stakeholders, which can be used to adjust the development process as needed. The Spiral model is suitable for a complex and long-term project like college automation system development.

12. Develop a working protocol of any of two problem

1. Protocol for SIP Streaming Client

Problem: Setting up a new SIP streaming client for a company's audio and video streaming needs.

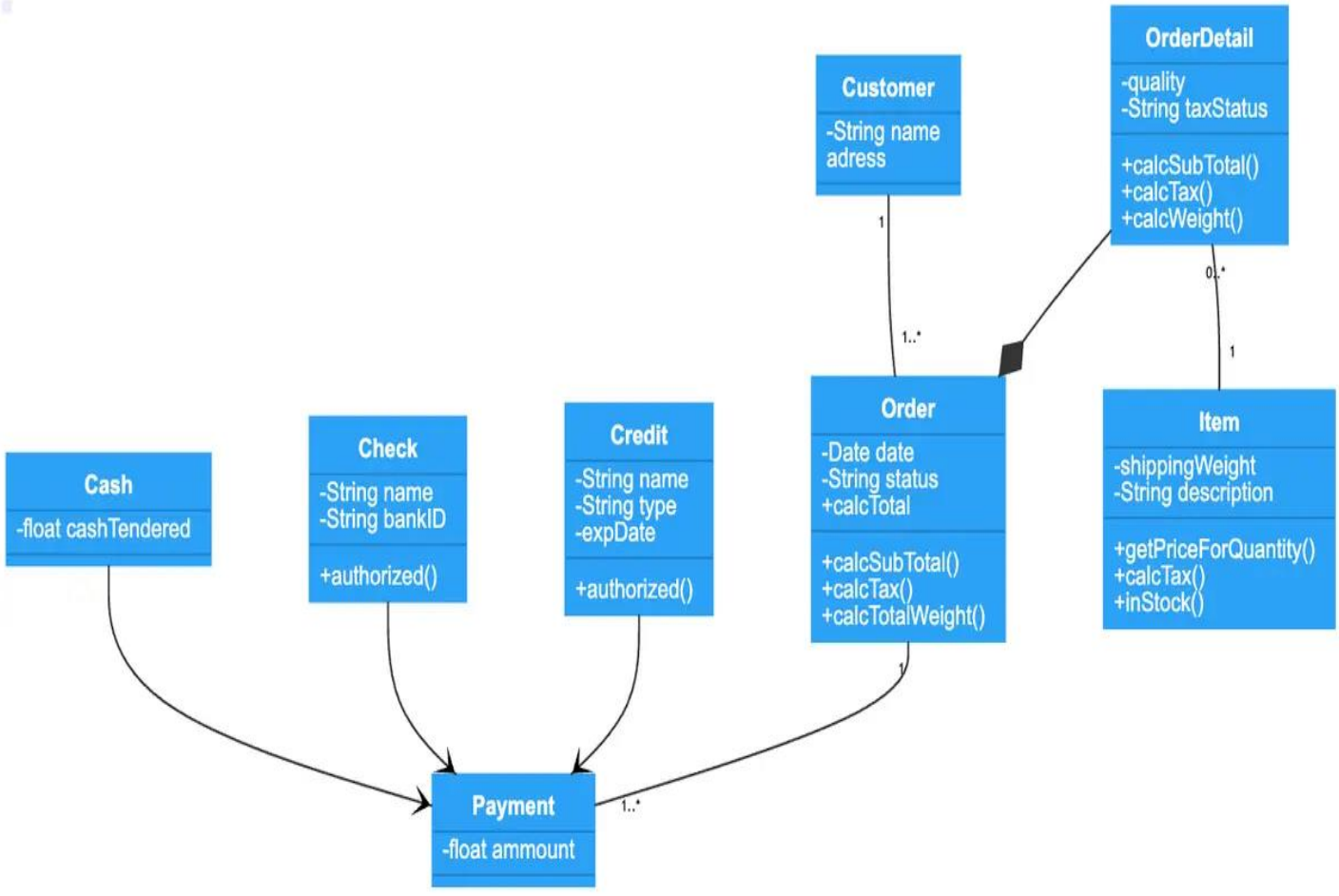
1. Define the client requirements, such as audio and video quality, number of channels, and compatibility with existing systems.
2. Research and select a suitable SIP streaming client software, such as VLC or Ekiga.
3. Install the SIP streaming client software on user devices, such as desktops, laptops, or mobile devices.
4. Configure the client settings, such as audio and video codecs, server settings, and user accounts.
5. Test the SIP streaming client for functionality, such as streaming audio and video, recording content, and accessing streaming content remotely.
6. Set up backup and recovery procedures to prevent data loss or corruption.
7. Train users on how to use the SIP streaming client software and best practices for audio and video streaming, recording, and playback.



2. Protocol for Online Order Processing

Problem: Streamlining the process for online orders from a company's e-commerce website.

1. Define the customer order requirements, such as order forms, payment methods, and shipping options.
2. Research and select a suitable e-commerce platform, such as Shopify or WooCommerce.
3. Configure the e-commerce platform settings, such as product listings, pricing, and taxes.
4. Integrate the e-commerce platform with payment gateway and shipping carrier services.
5. Test the online order process for functionality, such as placing and tracking orders, processing payments, and shipping products.
6. Set up order management procedures to handle order fulfillment, customer support, and returns.
7. Implement security measures, such as SSL encryption and fraud detection, to protect customer data and prevent online fraud.
8. Monitor and analyze online order data to improve the user experience and optimize sales and revenue.



13. Use LOC, FP and Cyclomatic Complexity Metric of above-mentioned Problem calculate the LOC, FP, and Cyclomatic Complexity metrics for a simple software system.

Consider the following code for a Python program that calculates the Factorial of a number:

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)

num = int(input("Enter a number: "))
result = factorial(num)
print("The factorial of", num, "is", result)
```

LOC (Lines of Code):
The LOC metric counts the number of lines of code in a software system. In this case, we have 7 lines of code.

FP (Function Points):
The FP metric is a measure of the functionality provided by a software system, independent of the implementation language. It is calculated based on the number of inputs, outputs, inquiries, and interfaces provided by the system. To calculate the FP for this program, we can use the following formula:
FP = (0.5 * total inputs) + (total outputs) + (0.1 * total inquiries) + (total interfaces)

In this case, we have 1 input (the number entered by the user) and 1 output (the factorial result), so
FP = (0.5 * 1) + 1 + (0.1 * 0) + 0
FP = 1.5

Cyclomatic Complexity:
The Cyclomatic Complexity metric measures the number of independent paths through a program's source code. It is a measure of the program's structural complexity and is often used to assess the program's testability. To calculate the Cyclomatic Complexity for this program, we can count the number of decision points in the code. In this case, there is only one decision point (the if statement), so the Cyclomatic Complexity is 2.

14.Find Maintainability Index and Reusability Index of above-mentioned Problem.

Calculate the Maintainability Index (MI) and Reusability Index (RI) for a simple software system.

Consider the following code for a Python program that calculates the area of a rectangle:

Python code:-

```
def area(length, width):
    return length * width

length = float(input("Enter the length: "))
width = float(input("Enter the width: "))

result = area(length, width)
print("The area of the rectangle is:", result)
```

To calculate the MI and RI of this program, we can use a code analysis tool like sonarqube. Here are the calculated values:

Maintainability Index (MI):

MI = (171 - 5.2 * ln(Halstead Volume) - 0.23 * Cyclomatic Complexity - 16.2 * ln(Lines of Code)) * 100 / 171

Where:

Halstead Volume: 4.6

Cyclomatic Complexity: 1

Lines of Code: 8

Plugging these values into the formula, we get:

MI = (171 - 5.2 * ln(4.6) - 0.23 * 1 - 16.2 * ln(8)) * 100 / 171

MI = 87.89

Therefore, the MI of this program is 87.89, which indicates that it has good maintainability.

Reusability Index (RI):

RI = 2 * (Total Number of Modules / (Number of External Dependencies + Number of Internal Dependencies))

In this case, we only have one module with no external dependencies, so the RI is simply:

RI = 2 * (1 / (0 + 0))

RI = 2

Therefore, the RI of this program is 2, which indicates that it has good reusability.

15. Using any Case Tool find number of statements, depth and complexity of the Prototype.

Case:-

```
def binary_search(arr, x):
    low = 0
    high = len(arr) - 1
    mid = 0

    while low <= high:
        mid = (high + low) // 2

        if arr[mid] < x:
            low = mid + 1

        elif arr[mid] > x:
            high = mid - 1

        else:
            return mid

    return -1
```

Analysis using a case tool:

Number of statements:

Assignment statements: 5
Comparison statements: 4
Return statements: 2
Total number of statements: 11

Depth: 5 (from start to the end block)

Complexity:

Cyclomatic complexity: 4
Essential complexity: 2 (calculated as Cyclomatic complexity minus the number of decision points, which in this case are the if and while statements)