**Chapter 3: Elicitation**

After discussing the inception part, we need to keep focus on the elicitation part. So this chapter specifies the elicitation part.

## **3.1 Introduction**

Requirements elicitation is a part of requirement engineering that is the practice of gathering requirements from the users, customers, and other stakeholders. We have faced many problems like understanding the problems, problems of making questions for the stakeholders, problems of less communication with the stakeholders for time limitation, problems of volatility. Though it is not too easy to gather requirements within a very short time, we have surpassed these problems in an organized and systematic manner.

## **3.2 Eliciting Requirements**

Inception where Question and Answer approach is used; elicitation makes use of a requirements elicitation format that combines the elements of problem solving, elaboration, negotiation, and specification. It requires the cooperation of a group of end-users and developers to elicit requirements. To elicit requirements, we completed following four works.

* Collaborative Requirements Gathering
* Quality Function Deployment
* Usage Scenarios
* Elicitation work products

### **3.2.1 Collaborative Requirement Gathering**

Many different approaches to collaborative requirements gathering have been proposed. Each makes use of a slightly different scenario. We completed following steps

* The meetings were conducted with the students and discussed with teachers and professionals. They were questioned about their requirements and expectations from the Graduate record verification system
* They were asked about their Graduate record verification system problems.
* At last we selected our final requirement list from the meetings.

### **3.2.2 Quality Function Deployment**

Quality function deployment (QFD) known as a way to represent the “voice of the customer,” is a process for capturing customer requirements and translating them into requirements that can be used by designers, producers, and suppliers. It concentrates on maximizing customer satisfaction from the Software engineering process. With respect to our project the following requirements are identified by a QFD.

**3.2.2.1 Normal Requirements**

Normal requirements consist of objectives and goals that are stated during the meeting with the customers. Normal requirements of our project are:

1. Allow valid user to login and logout
2. Check user validity
3. Feature that will allow dynamic report generation
4. Allow users to generate academic transcript and certificate
5. Efficient and user friendly
6. The user interface of the system would be easy
7. Allow user to view approval of the admin panel
8. Allow user to give approval/rejection (by registrar)
9. Allow user to search Information of students
10. Allow user to track information by using activity log
11. Security issue
12. Error free activity

**3.2.2.2 Expected Requirements**

These requirements are implicit to the system and may be so fundamental that the customer does not explicitly state them. Their absence will be a cause of dissatisfaction. These are –

1. The application shall allow the user to log in based upon a user id and password.
2. The application shall automatically save the current date.
3. The application will allow the users to search, update or delete.
4. The user interface of the system shall be easy to use and shall make the least use of typing by users.
5. The application shall be easily maintainable.
6. The application will be stable.
7. The application will be open for future extension and modification.
8. All the records will be saved to database for future references.
9. The application shall be user friendly
10. The user interface shall make use of input such as drop downs, check boxes and radio buttons as much as possible to avoid invalid and incorrect input.

**3.2.2.3 Exciting Requirements**

These requirements are for features that go beyond the customer's expectations and prove to be very satisfying when present. These are

* 1. The system will enable the user to change user passwords if they forget old ones.
  2. The user interface should provide appropriate error messages for invalid input as well as tool-tips
  3. The system will have activity log for tracking information.

## **3.3 User Scenario**

Graduate Record Verification System

This system will be used for validating graduate students of Institute of Information Technology of University of Dhaka. Here the end users are UGC staffs (Admin, Staff 1, Staff 2), University staffs (Registrar, Data entry operator 1, Data entry operator 2), Graduate students, Stakeholders (Foreign universities, Employers), Payment processing partners.

System admin can add information (name, email address, mobile number, password, and role) about Admin, Registrar, Program officer of Institute/Department, Students, Stakeholder and assign their roles (Admin, Registrar, Program officer of Institute/Department, Students, Stakeholder). System admin can do all operations those defined for other users.

System admin can view activity log, generate report of all things. Activity log keeps track of information change.

(Student profile information depend on Registrar input) Student can log in to the system with his/her email address, password, and registration number. After logging in into the system, he/she can view his/her profile with personal information, academic information, and payment request from stakeholders. Personal information is name, email address, mobile number, present address, permanent address. Academic information is registration number, session, year, institute/department, university name. If any payment request is received, then he/she can pay through external system (PayPal). Response from external system will be recorded in database. Student can edit his/her personal information.

UGC staff can login into the system with email address, password. He/she can add university (name, location, website), university’s registrar with his/her information. UGC staff can generate report with all information (University wise, payment wise).

Registrar will login into the system with his/her email address and password. In Registrar profile, he/she can view his/her personal information (Name, designation, university name), add, edit and delete student information, and add digital signature to verify graduate student information, which are pending for approval. Registrar can generate reports that belong to his/her university.

Institute/Department program officer login into the system with email address, password. In PO profile, he can view his personal information. PO can define number of semester, add course in corresponding semester. PO can insert, update and delete students’ semester wise course results. PO can generate reports that belong to his/her institute/department.

Foreign universities or employers can access the system to validate their candidate students’ information. First, they have to enter student’s name, registration number, university name, mobile number and email address. If students’ information is available in the system, then stakeholder need to provide his/her credentials- name, designation, organizations’ name, contact number and email. After inserting that information, a request will be sent to the corresponding student as a payment request. If the payment is completed, a notification will be sent to Registrar to attach digital signature (signature image) to verify that student’s information. After completing verification, a link will be sent to stakeholders’ email address. Through this link, stakeholder get student has verified information.

## **3.4 Conclusion**

Elicitation phase helped us to understand about the problems of our scopes of the system. This phase also helped us to identify the requirements, negotiate different approaches and specify a preliminary set of solution requirements in an atmosphere that is conducive to the accomplishment of the goal.

# Chapter 4: Scenario Based Modeling

This chapter describes the scenario based model.

## 4.1 Definition of Use case

A use case captures a contract that describes the system behavior under various conditions as the system responds to a request from one of its stakeholders. In essence, a use case tells a stylized story about how an end user interacts with the system under a specific set of circumstances. A use case diagram simply describes a story using corresponding actors, who perform important role in the story and makes the story understandable for the users. The first step in writing a use case is to define that set of “actors” that will be involved in the story. Actors are the different people that use the system or product within the context of the function and behavior that is to be described. Actors represent the roles that people play as the system operators. Every user has one or more goals when using system.

Primary Actor: Primary actors interact directly to achieve required system function and derive the intended benefit from the system. They work directly and frequently with the software.

Secondary Actor: Secondary actors support the system so that primary actors can do their work. They either produce or consume information.

## **4.2 Use Case Diagrams**

Use case diagrams give the non-technical view of overall system.

### **4.2.1 System Description from Level-0 use case:**

After analyzing the user-scenario we found five actors who will directly use the system as a system operator. Primary actors are those who will play action and get a reply from the system whereas secondary actors only produce or consume information.

They all will be our users, who will be using our system to fulfill their procedures.

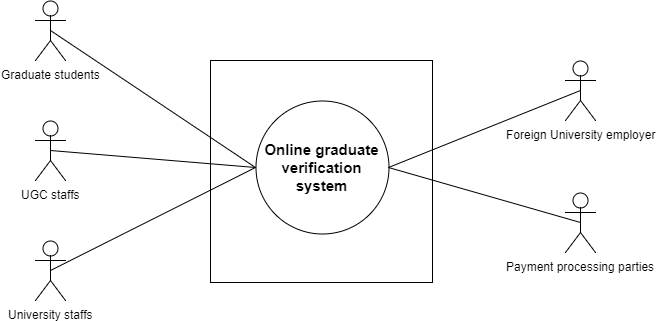


Figure 1: Level-0 Use Case Diagram

### **4.2.2 System Description from Level-1 use case:**

Primary Actor : User

Secondary Actor : System

Goal in Context : To operate the application.

Scenario : The actors of our system have to play different actions and system will reply according to these actions –

Action 1: Enters signup.

Reply 1: Please fill up the required information.

Action 2: Enters the information

Reply 2: Registration successful.

Action 3: Enters username and password.

Reply 3: Sign in successful and shows User Dashboard.

Different user’s action and reply (UGCs, Student, Registrar, Foreign University, Program Officer)

Action: Click add University (UGCs)

Reply: University added (UGCs)

Action: Click add Registrar (UGCs)

Reply: Registrar added (UGCs)

Action: Click add student (Registrar)

Reply: Student added (Registrar)

Action: Click add department (Registrar)

Reply: Department added (Registrar)

Action: Request for student verification (Foreign University)

Reply: Student verified

Action: Click to view student info (Student)

Reply: Show students details information (Student)

Action: Check payment request (Student)

Reply: Show the availability of the request (Student)

Action: Enter report generation (PO, UGCs)

Reply: Report generated (PO, UGCs)

Exception: No Exceptions (if system works correctly by fulfilling requirements)

Exceptions (if) : User is not authenticated.

: System error.

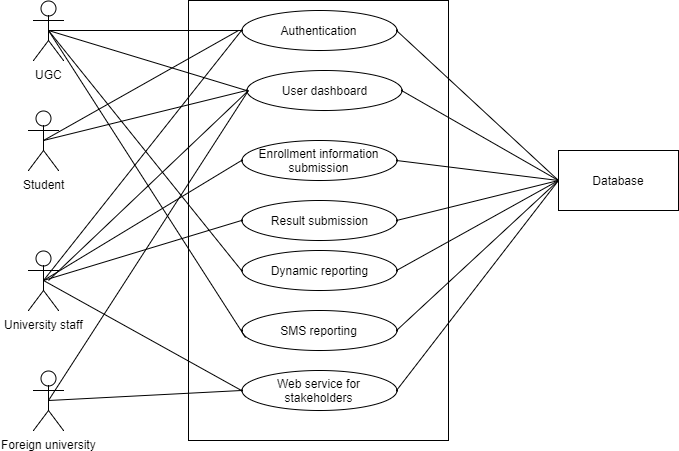


Figure 2: Level-1 Use Case Diagram

### **4.2.3 System Description from Level-1.1 use case:**

Action 1:

* Enters First Name
* Enters Last Name
* Enters email.
* Enter password
* Clicks on **Save.**

Reply 1: Successfully saved.

Action 2: Enters **Change Password.**

Reply 2: Prompts for enter previous password.

Action 3: Enters previous password.

Reply 3: Prompts for new password.

Action 4:

* Enters new password.
* Re-enters new password.
* Enters **Save Changes.**

Reply 4: Password successfully changed.

Action 5: Enters sign out.

Reply 3: Logged out user.

Exception: No Exceptions

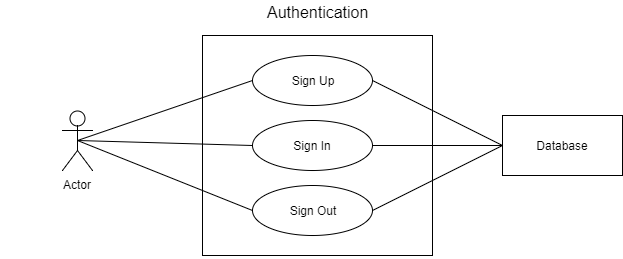


Figure 3: Level-1.1 Use Case Diagram

### **4.2.4 System Description from Level-1.1.1 use case:**

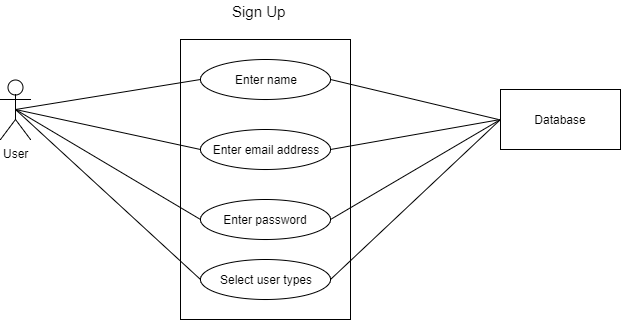


Figure 4: Level-1.1.1 Use Case Diagram

### **4.2.5 System Description from Level-1.1.2 use case:**

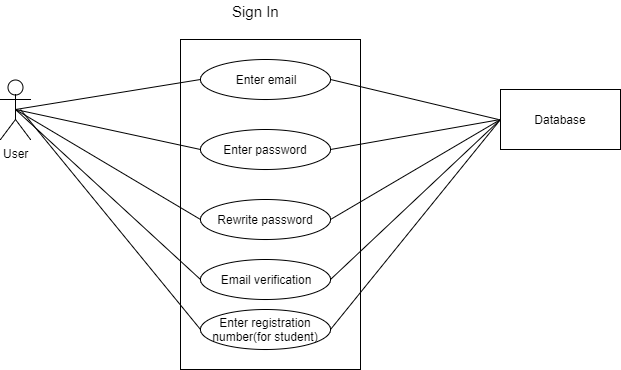


Figure 5: Level-1.1.2 Use Case Diagram

### **4.2.6 System Description from Level-2.1. a. use case:**

### 

Figure 6: Level-2.1. a. Use Case Diagram

### **4.2.6 System Description from Level-2.1. b. use case:**

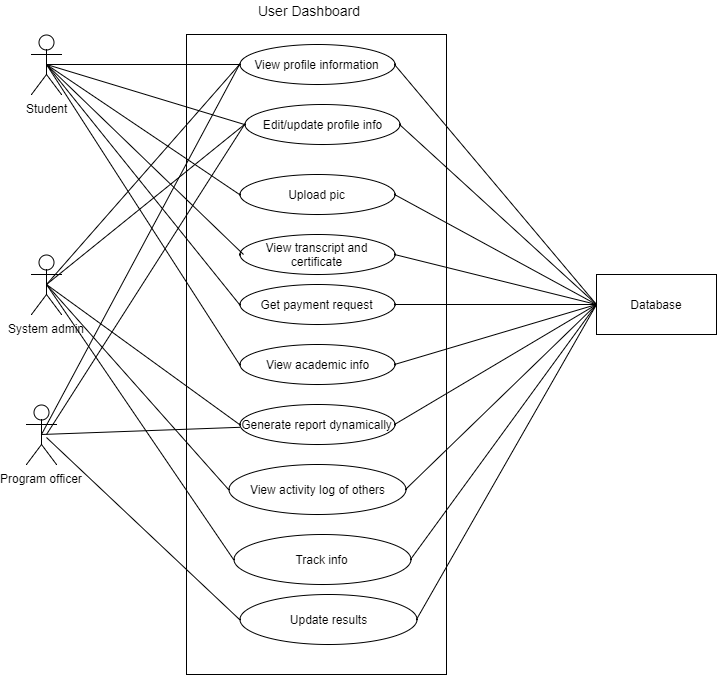


Figure 7: Level-2.1. b. Use Case Diagram

4.3 Activity and Swim lane diagram

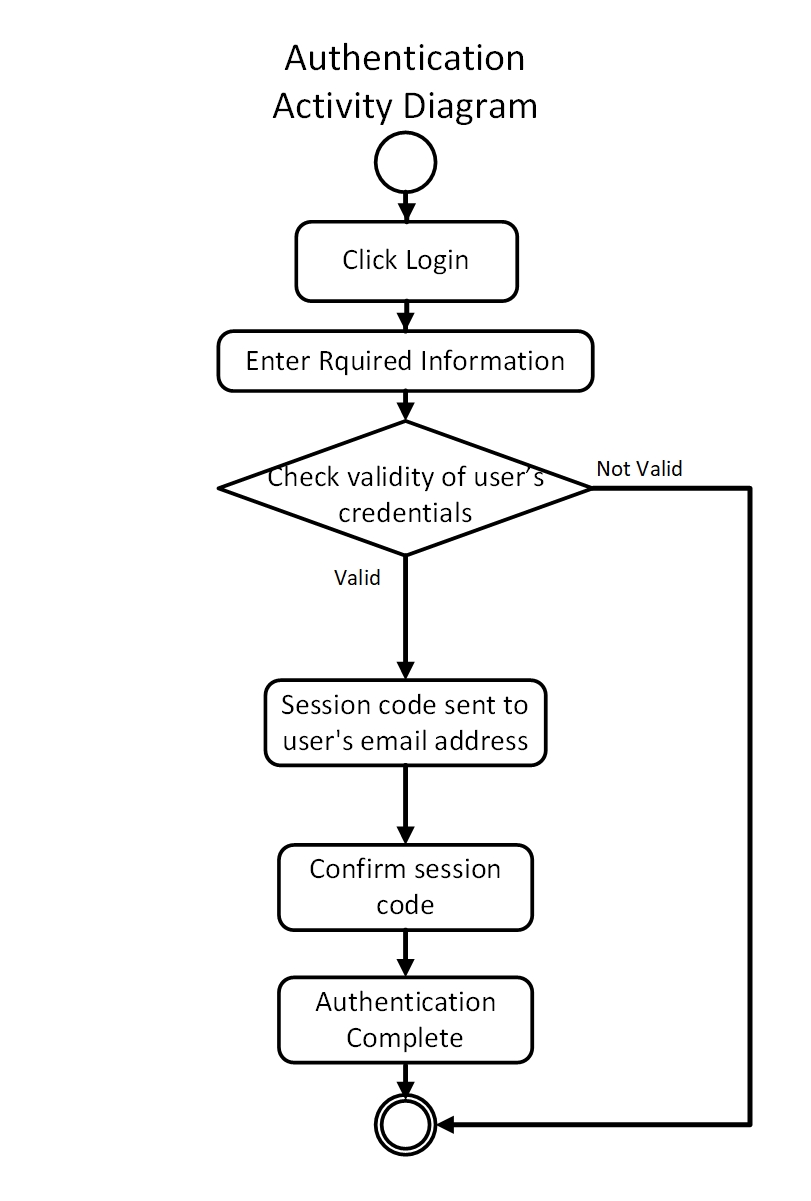
Activity diagramshows the technical view of the system for every use case  
from which we can understand how the system actually works and how the actors interact with the system. Here is the activity diagram for our authentication system.

Figure: Activity for Authentication

**Swim lane** diagram of a specific activity diagram shows the responsibilities of each actor dividing them into lanes. From this diagram we can improve our understanding about how the system works and which actors play what role.

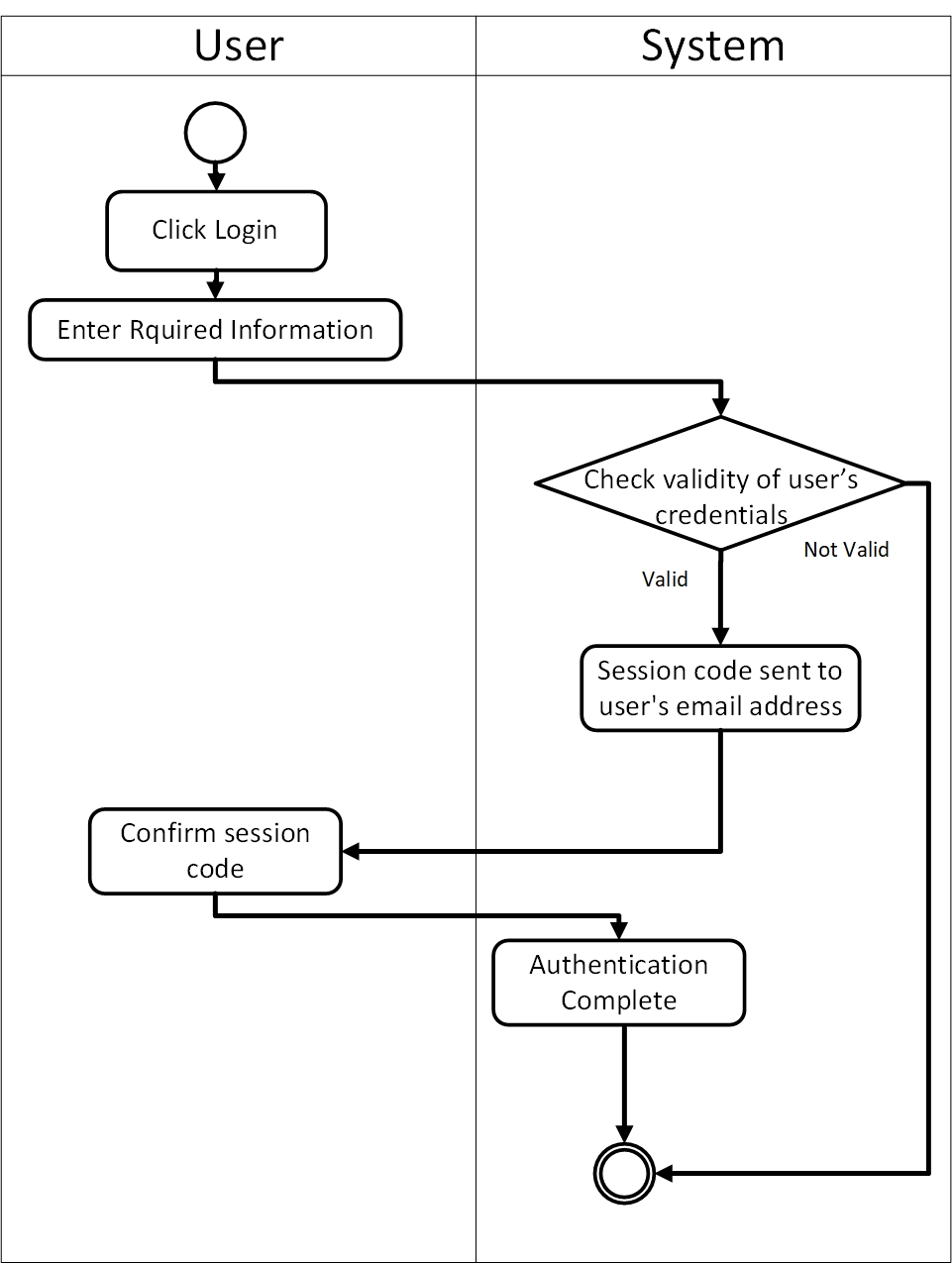
Following is the swim lane for authentication system.

Figure: Swim lane for Authentication

Following is the activity diagram for user graduate students.

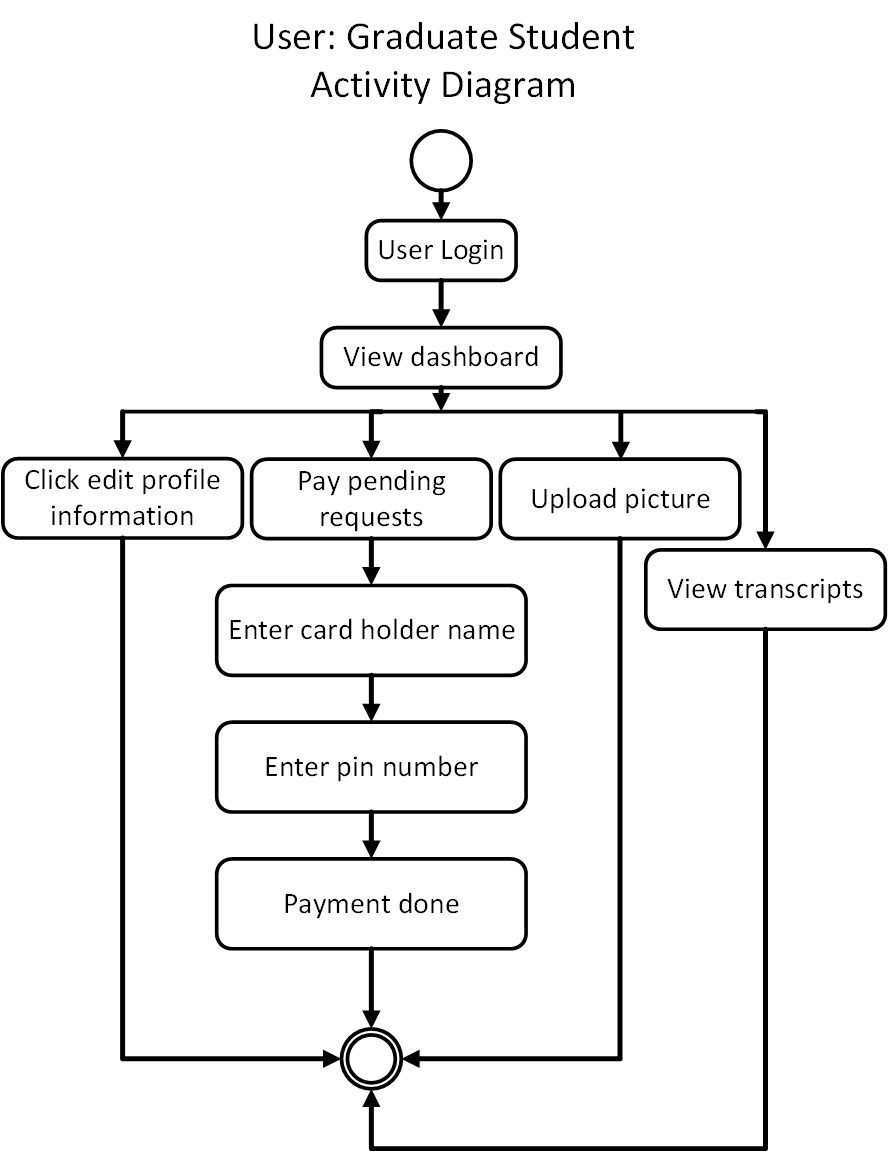


Figure: Activity of Graduate Students

Following is the swim lane diagram for user Graduate students.

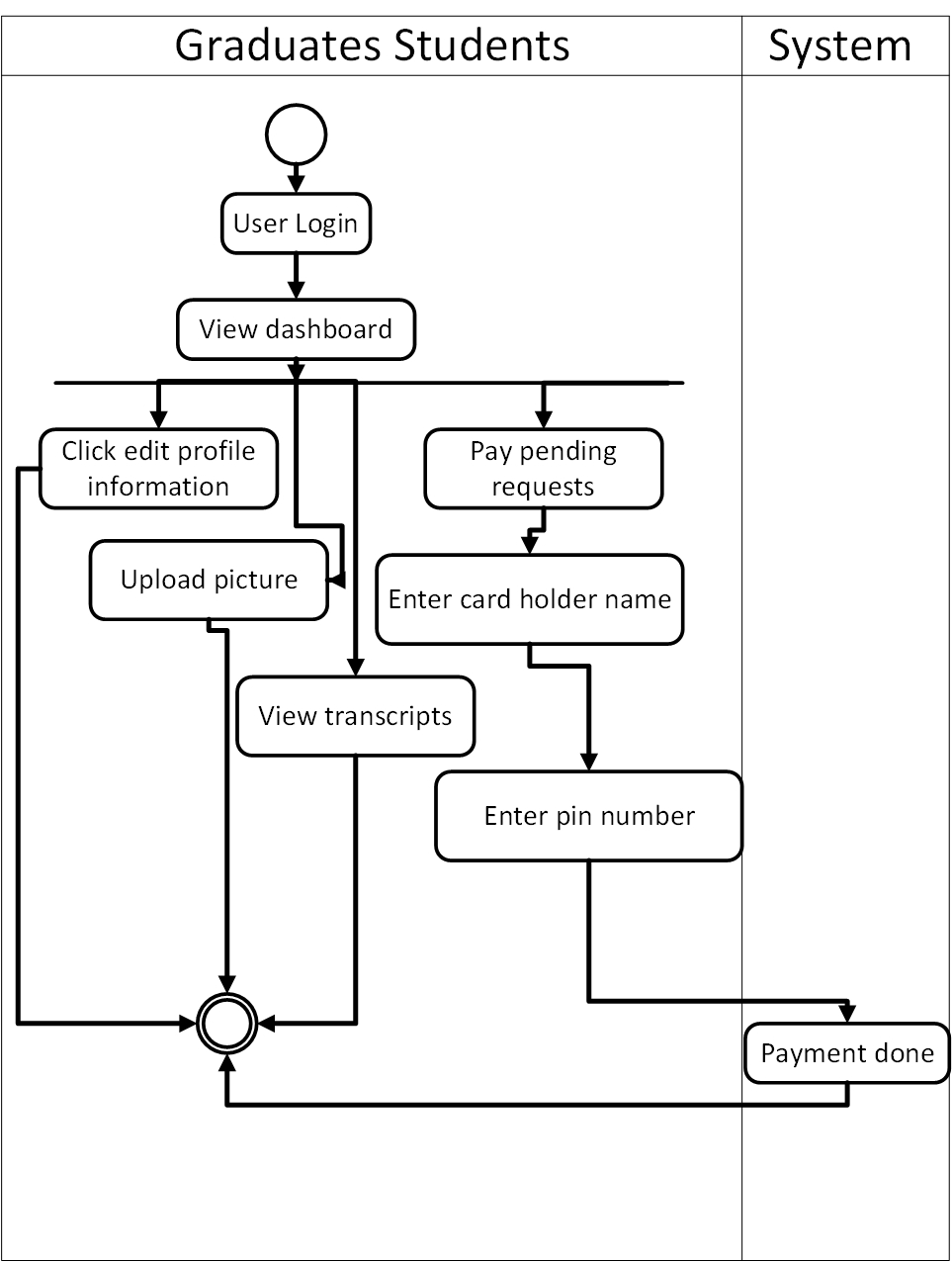


Figure: Swim lane for Graduate Students

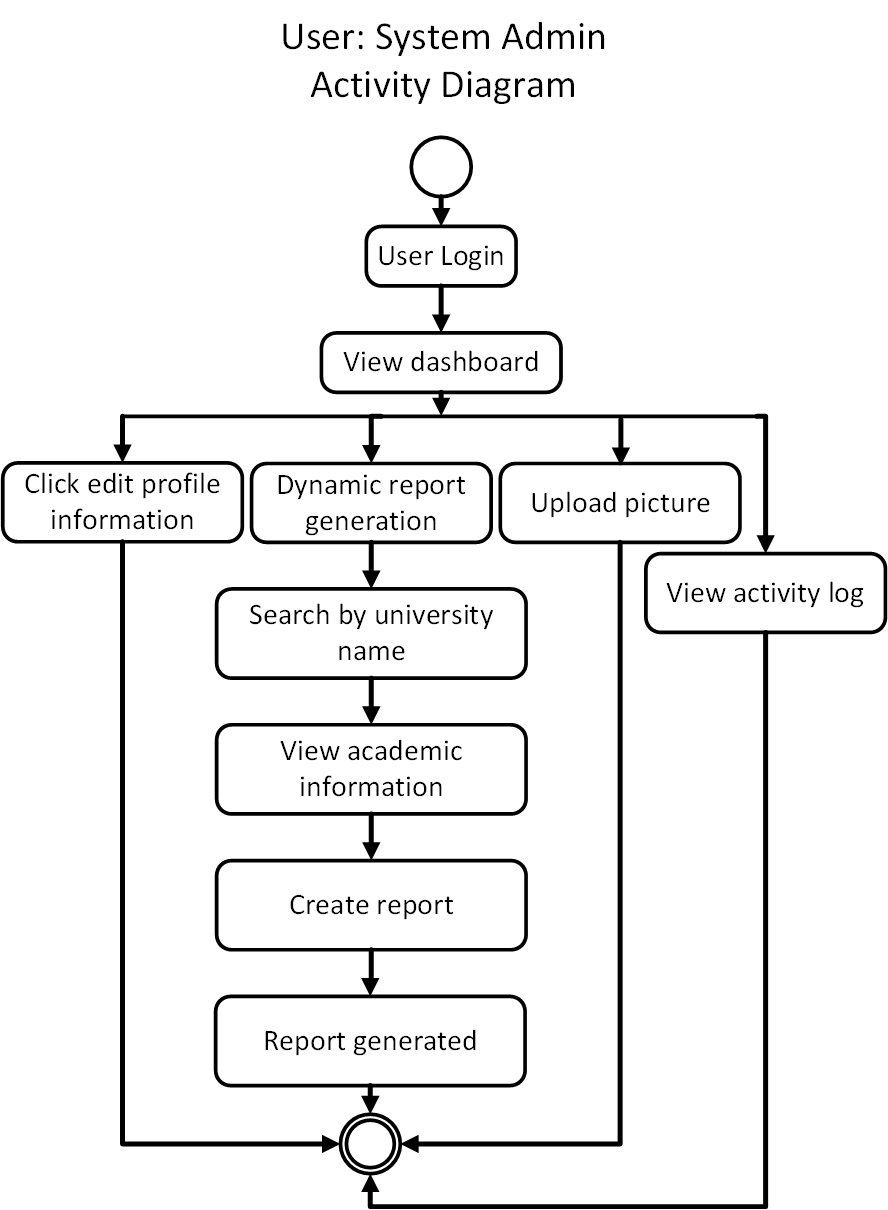
Following is the activity diagram for System Admin.

Figure: Activity for System Admin

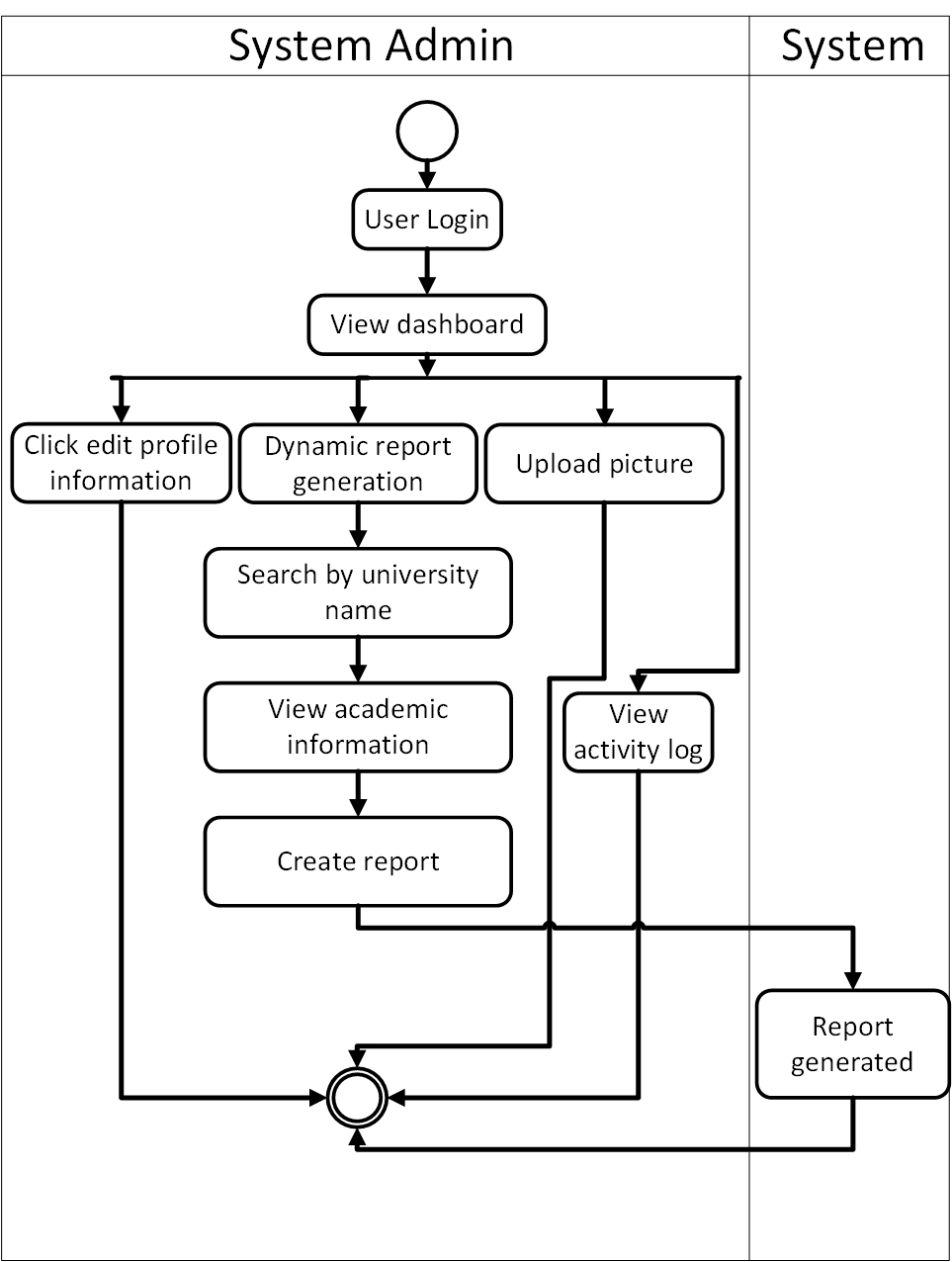
Following is the swim lane diagram for System Admin.

Figure: Swim lane for System Admin

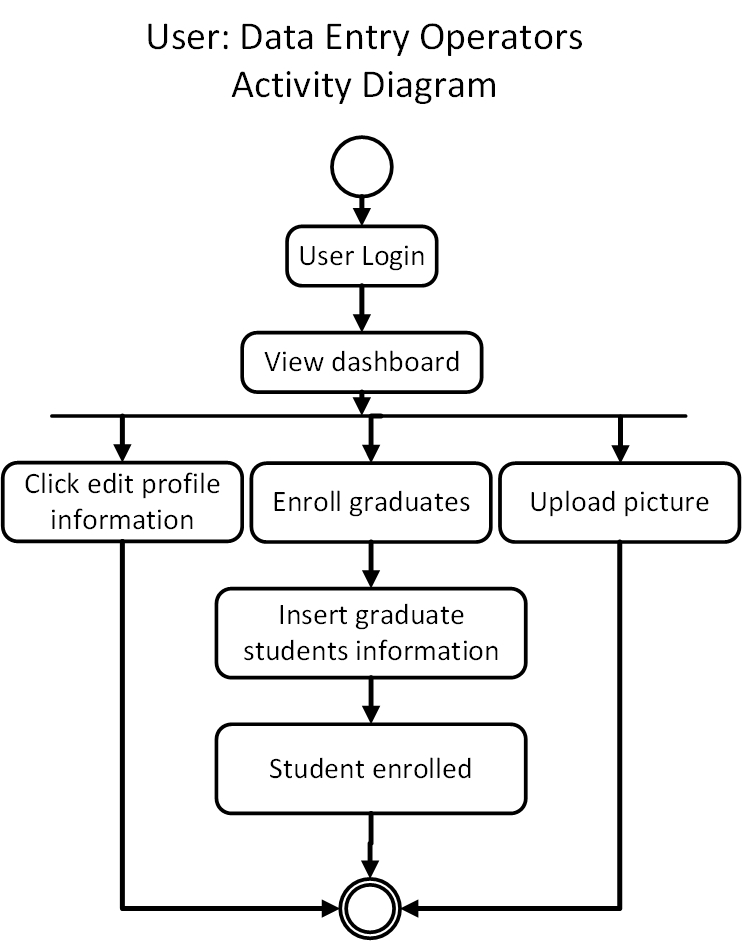
Following is the activity diagram for Data Entry Operators.

Figure: Activity for Data Entry Operators

Following is the swim lane for Data Entry Operators.

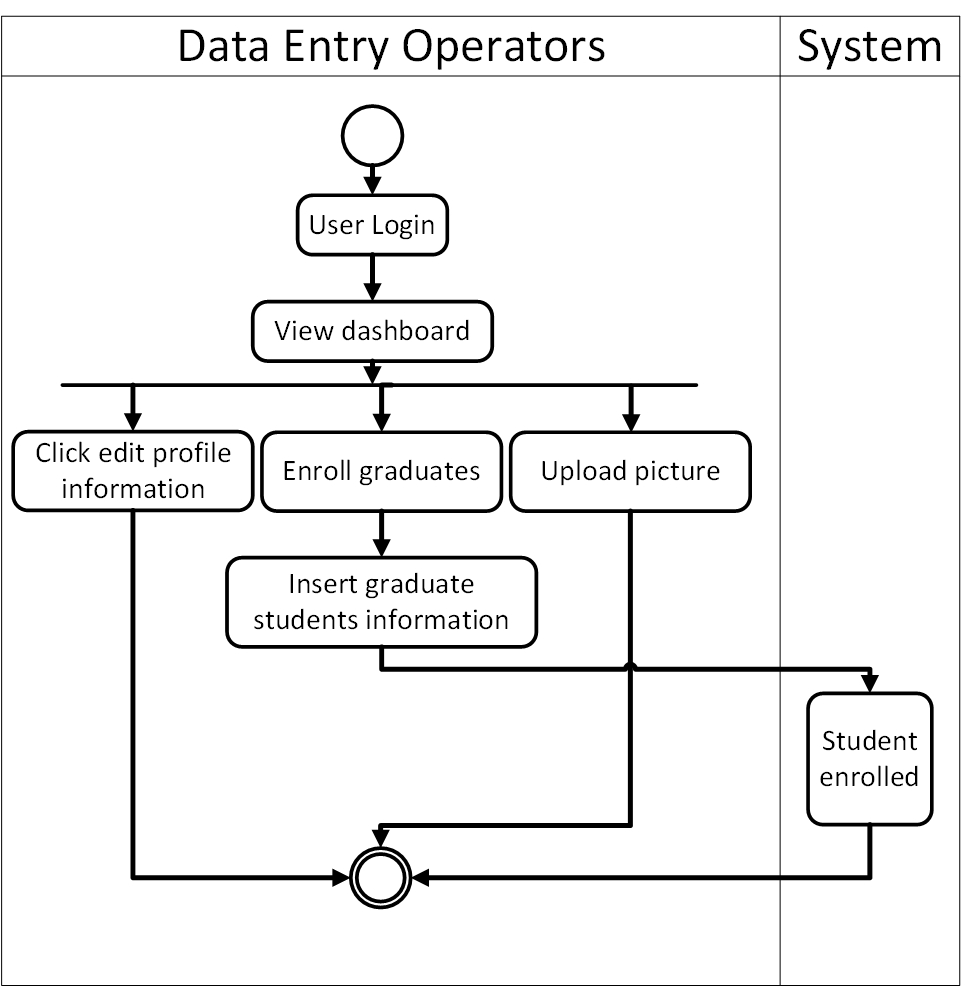


Figure: Swim lane for Data Entry Operators

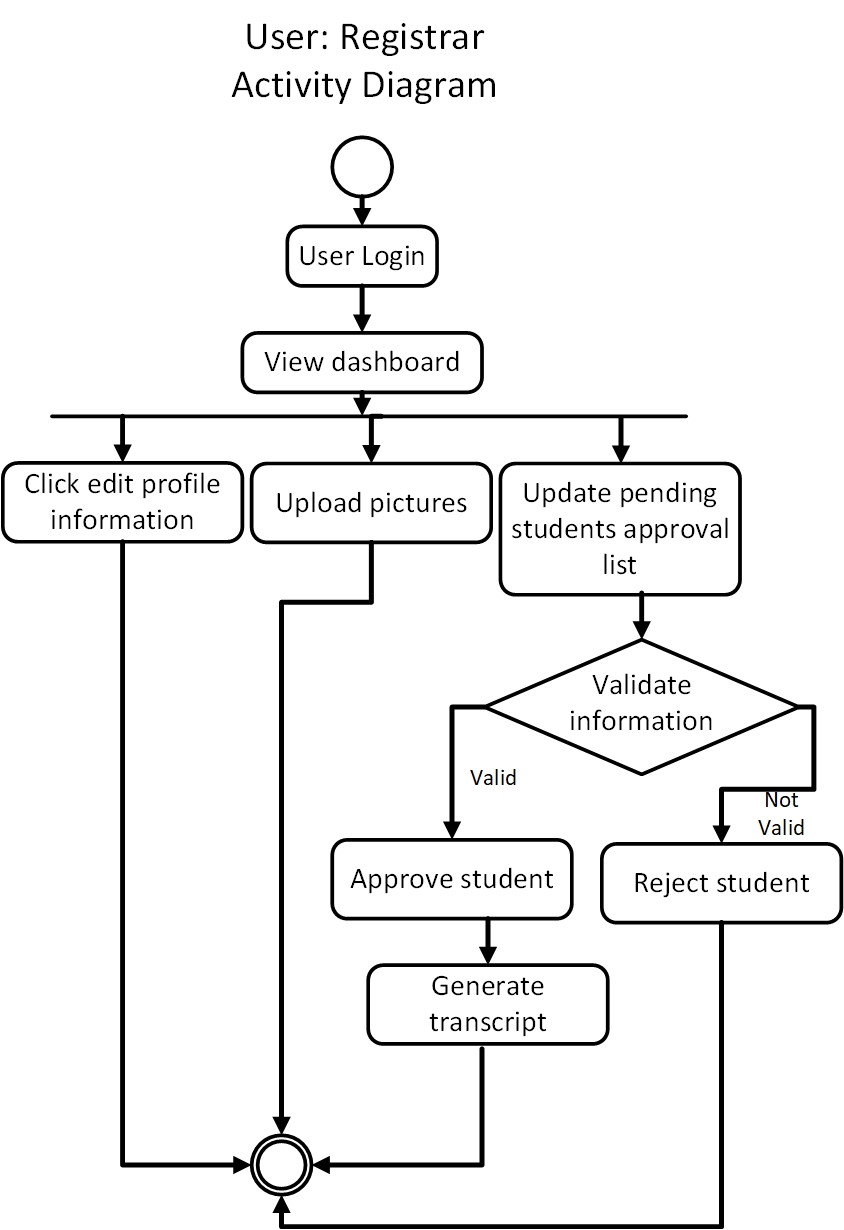
Following is the activity diagram for Registrar.

Figure: Activity diagram for Registrar

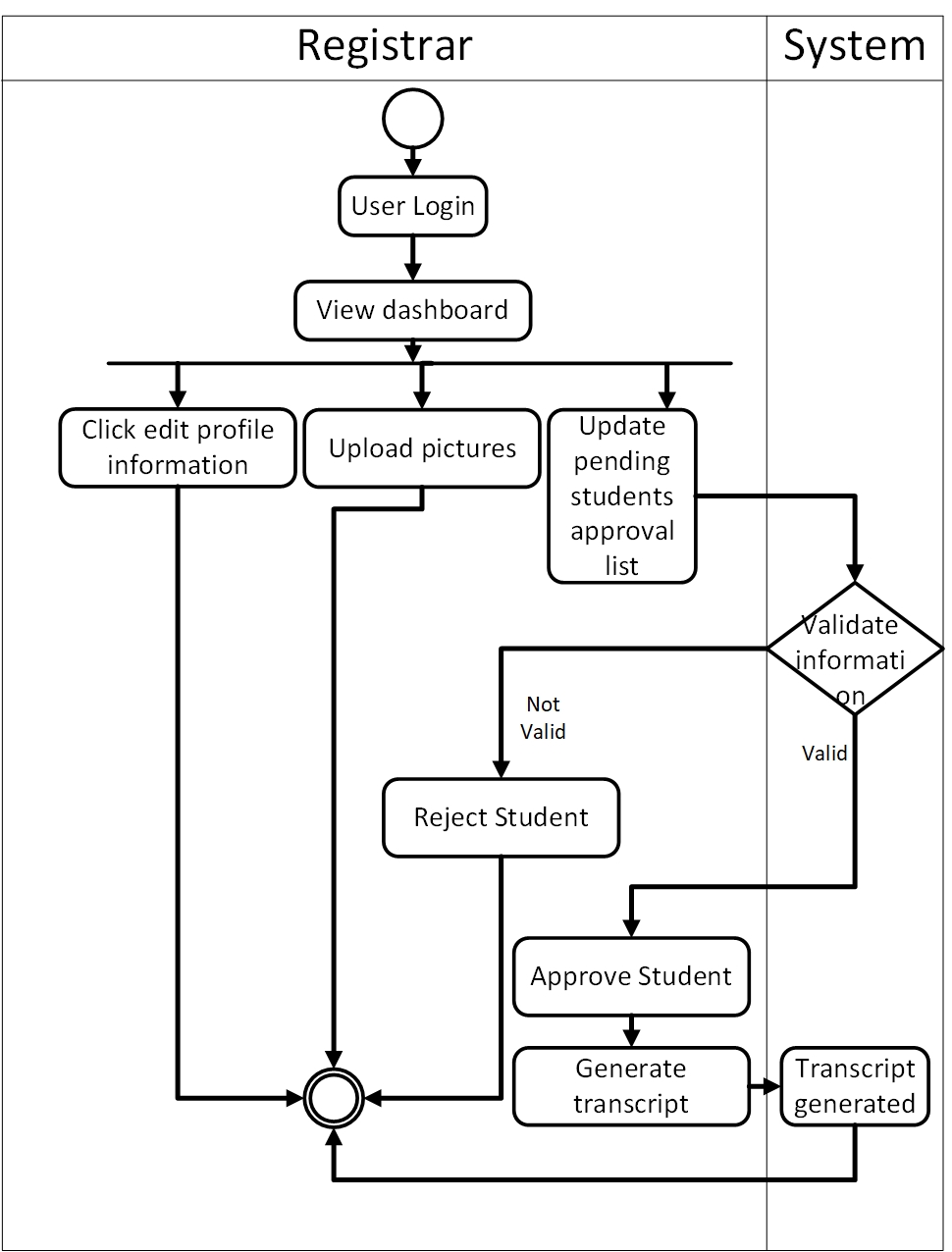
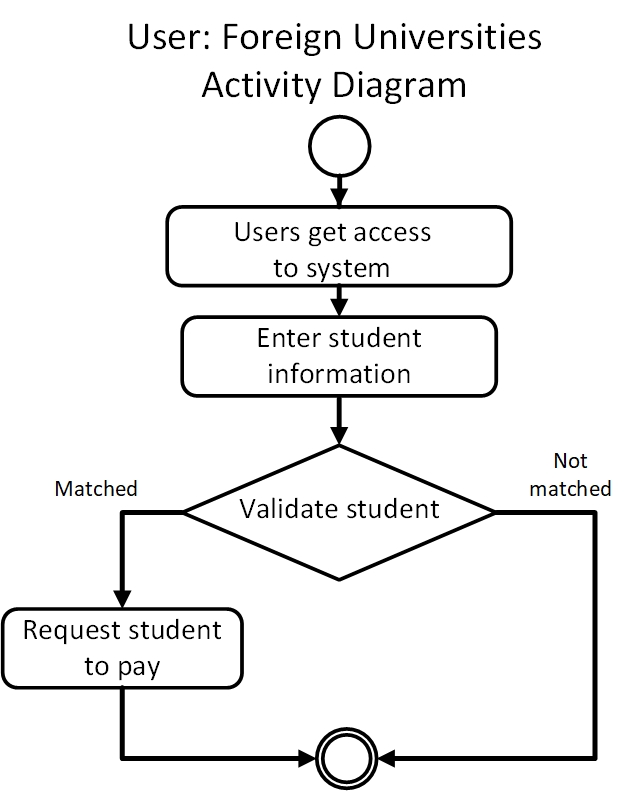
Following is the swim lane for registrar..

Figure: Swim lane for Registrar

Following is the activity diagram for foreign universities.



## 

Figure: Activity for Foreign Universities

Following is the swim lane diagram for Foreign Universities.

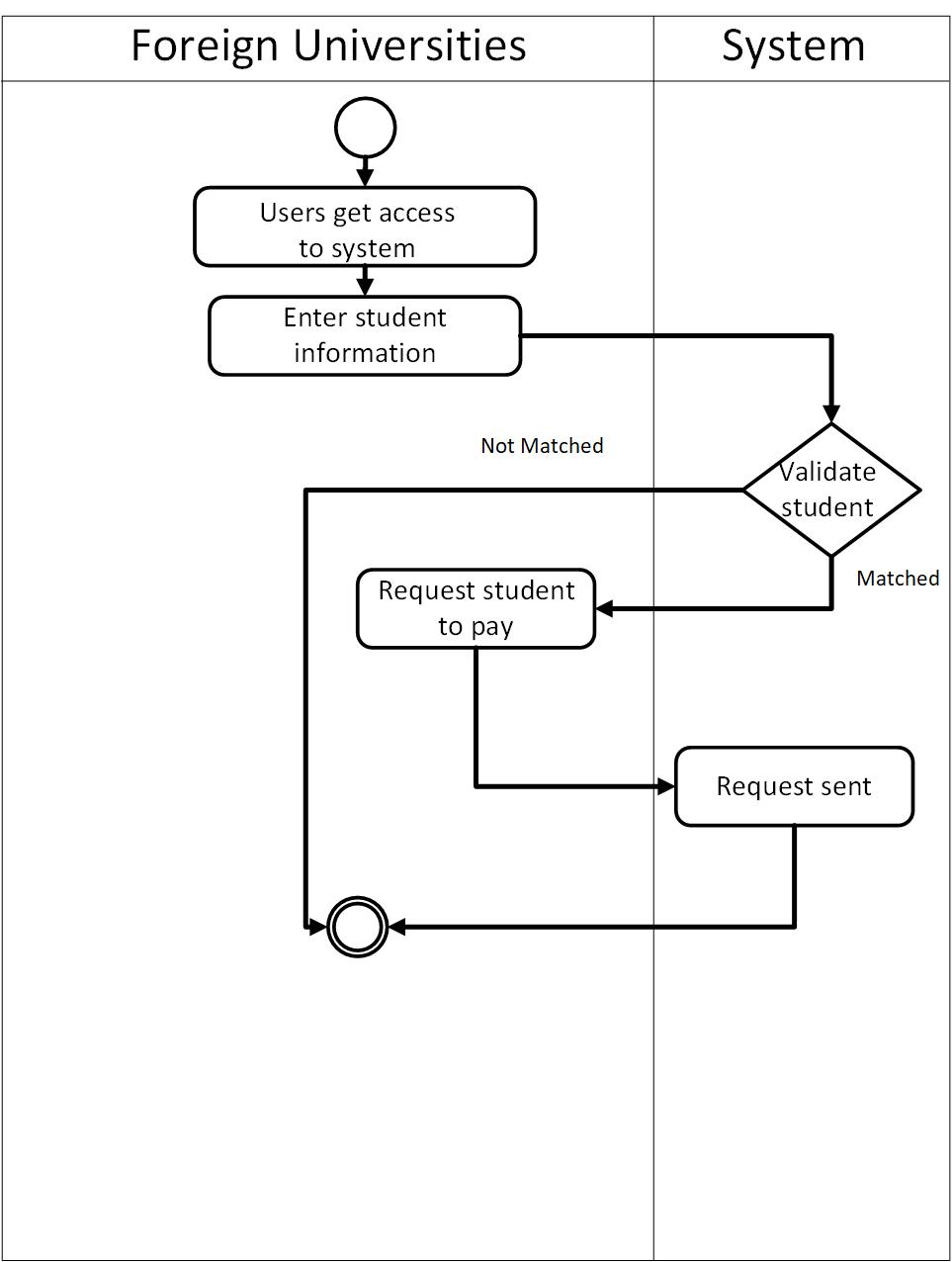


Figure: Swim lane for Foreign Universities

**Chapter 6**

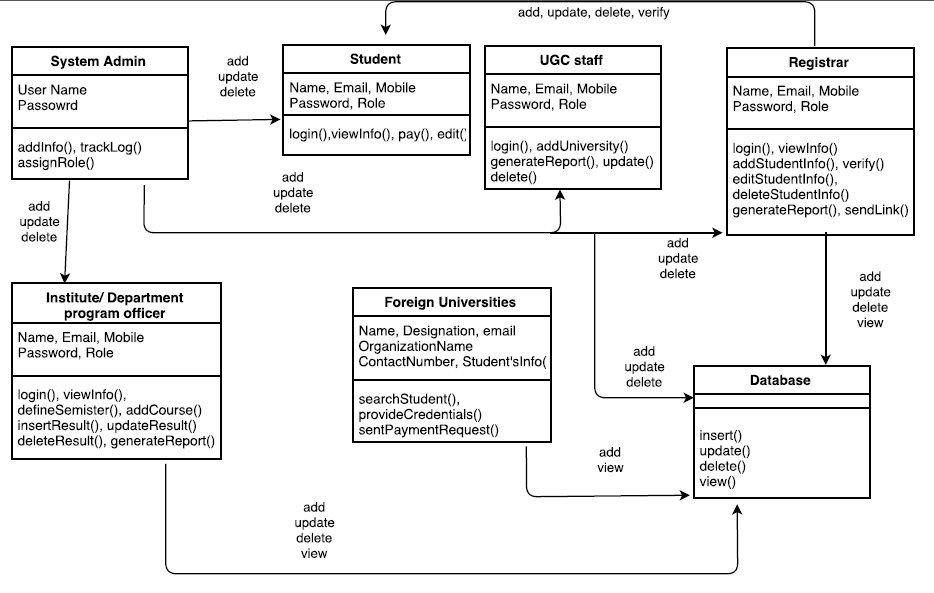
**Class-Based Model**

**6.1 Introduction**

Class-based modelling represents the objects that the system will manipulate, the operations that will be applied to the objects, relationships between the objects and the collaborations that occur between the classes that are defined.

**6.2 Class Diagram**

We have shown here how the classes interact together to accomplish certain goal.



**Chapter 7**

**Flow-Oriented Model**

**7.1 Introduction**

Although data flow-oriented modeling is perceived as an outdated technique by some software engineers, it continues to be one of the most widely used requirements analysis notations in use today.

**7.2 Data flow diagram**

A data flow diagram (DFD) is a significant modeling technique for analyzing and constructing information processes. DFD literally means an illustration that explains the course or movement of information in a process. DFD illustrates this flow of information in a process based on the inputs and outputs. A DFD can be referred to as a Process Model.



Figure 7.1: Level 1 Data Flow Diagram



Figure 7.2: Level 1.1 Data Flow Diagram



Figure 7.3: Level 1.2 Data Flow Diagram



Figure 7.4: Level 1.3 Data Flow Diagram



Figure 7.5: Level 1.4 Data Flow Diagram



Figure 7.6: Level 1.5 Data Flow Diagram



Figure 7.7: Level 1.6 Data Flow Diagram

**Chapter 8**

**Behavioral Model**

**8.1 Introduction**

Behavior modeling is also referred to as State modeling, State machines and State transition matrix. Behavior modeling is when one thinks of his ideas in terms of states and transitions. This requires both identifying all of the interesting states of being that software or its components are likely to be in. And also, at a high level, abstracting what events are likely to cause software or its components to change between states of being.

**8.2 Sequence Diagram**



Figure: Sequence Diagram

**Design**

**Architectural Diagram**

**1. Architectural Design for OOP:**

1.1 Representing the system in context

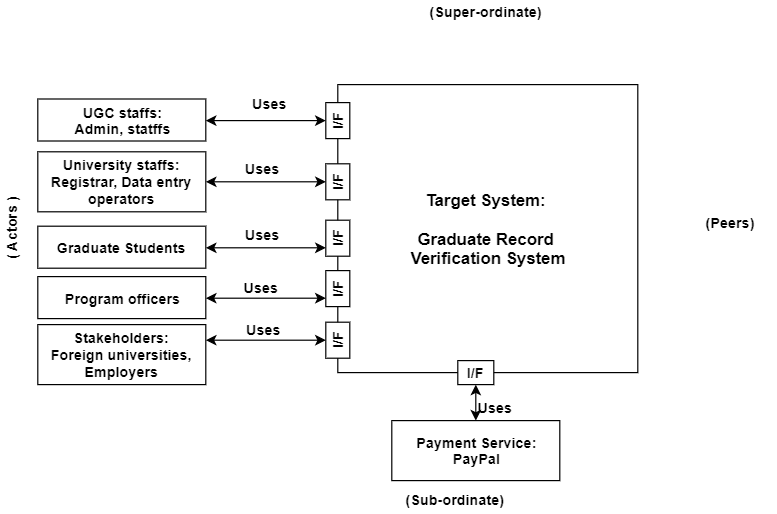


Figure 1.1.1: Representing the system in context

**1.2 Define Archetypes**

1. User creation
2. Authentication
3. Viewing activity log
4. Report generation
5. Viewing and editing personal information
6. Viewing academic information
7. Viewing payment request from stakeholders
8. Paying through PayPal
9. Adding universities information and its registration’s details
10. Adding, removing, updating student information
11. Adding digital signature to verify students
12. Adding course and semester
13. Insertion of semester wise result
14. Validating candidate student’s information
15. Sending payment request
16. Sending notification to attach digital signature
17. Sending link to stakeholder

**1.3 Refining architecture into components**

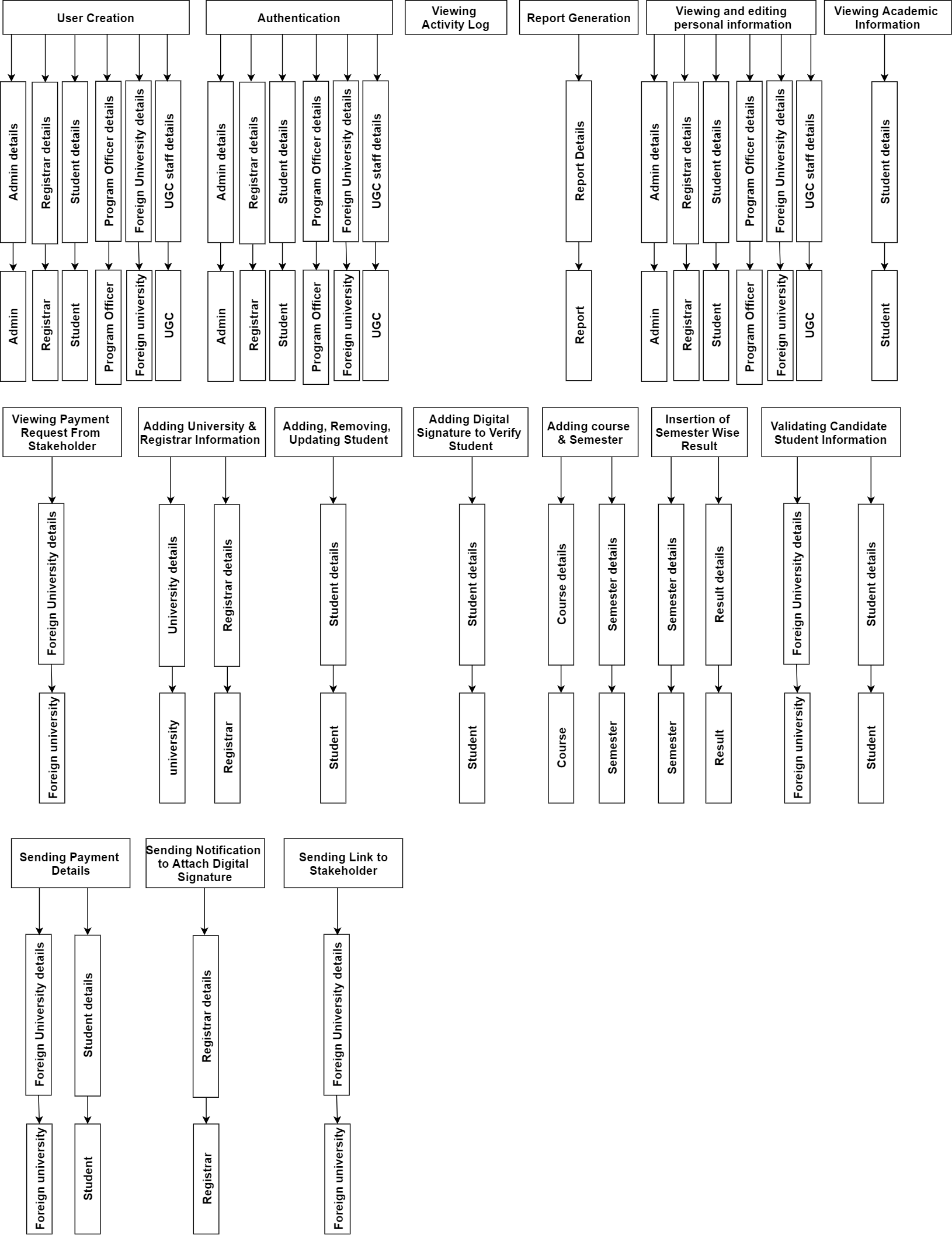
**Components:**

1. Admin details, registrar details, student details, program officer details, foreign university details, UGC staff details.
2. Admin details, registrar details, student details, program officer details, foreign university details, UGC staff details.
3. Activity log details
4. Report details
5. Admin details, registrar details, student details, program officer details, foreign university details, UGC staff details.
6. Student details
7. Foreign university details
8. Student details
9. University details, registrar details
10. Student details
11. Student details
12. Course details, semester details
13. Result details, semester details
14. Foreign university details
15. Student details, foreign university details
16. Registrar details
17. Foreign university details

**Classes:**

1. Admin, Registrar, Student, Program Officer, Foreign University, UGC staff, DAL
2. Admin, Registrar, Student, Program Officer, Foreign University, UGC staff, DAL
3. DAL
4. Report, DAL
5. Admin, Registrar, Student, Program Officer, Foreign University, UGC staff, DAL
6. Student
7. University
8. Student
9. Registrar, University
10. Student
11. Student
12. Course, Semester
13. Result, Semester
14. Foreign University, Student
15. Student, Foreign University
16. Registrar
17. Foreign University

**1.4 Describing Instantiation of the system**

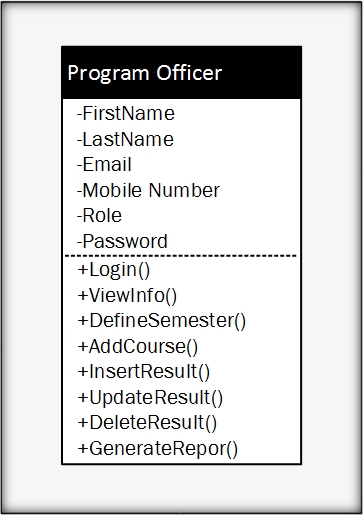
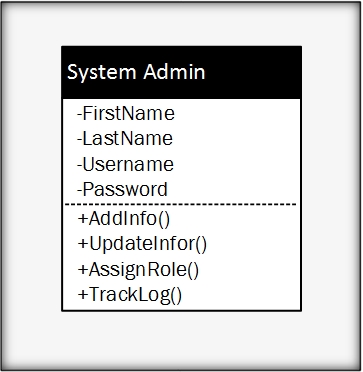
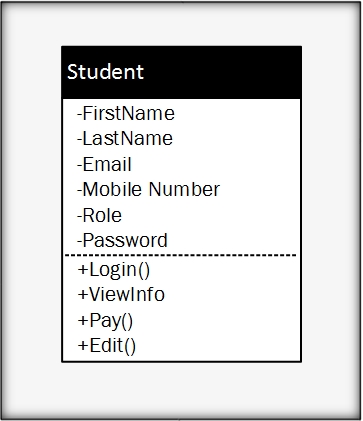


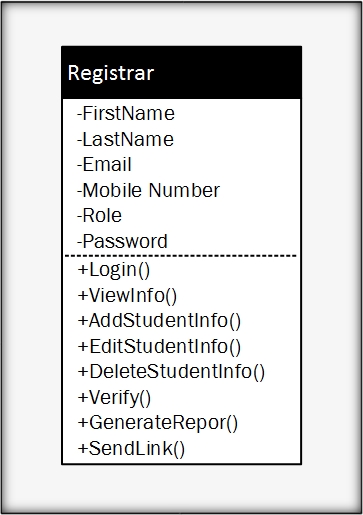
**Component Level Design**

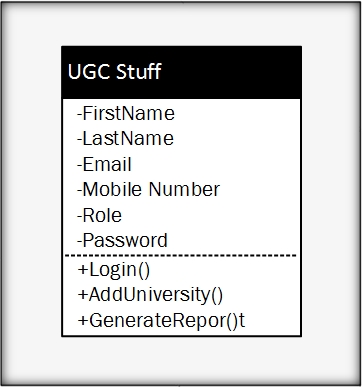
**1 Analysis Class**

This is the first step of object-oriented approach in component level design. In this section, all design classes that correspond to the problem domain as defined in the analysis model and architectural model are identified. The analysis classes for this project are:

1. System Admin
2. Student
3. UGC Stuff
4. Registrar
5. University/Department Program Office
6. Foreign Universities
7. Database







1.1 Design Components

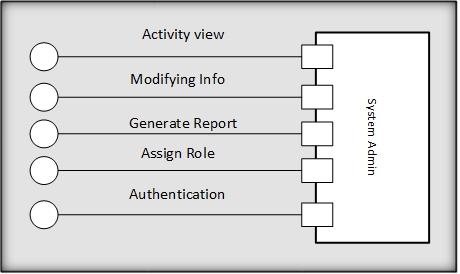


Figure 1: Design component of System Admin Class

Figure 2: Design component of Student Class

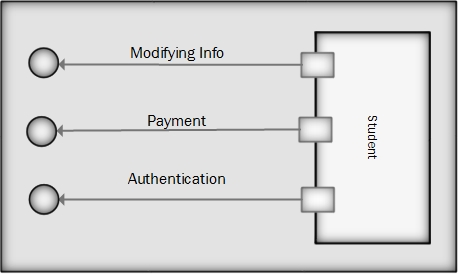
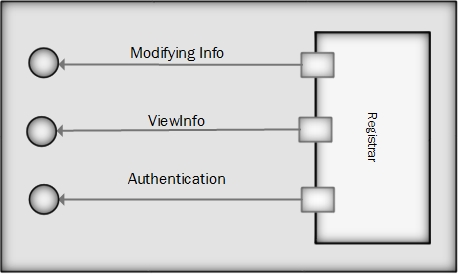


Figure 3: Design Component of Registrar Class

Figure 4: Design Component of Program Office

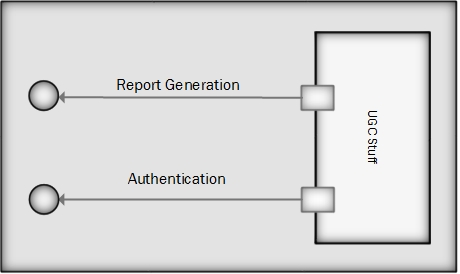
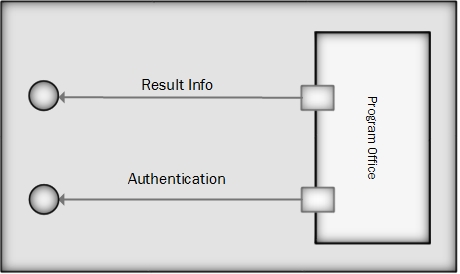


Figure 5: Design Component of UGC Stuff Class



**2. Infrastructure Class**

In this section, all design classes that correspond to the infrastructure domain are identified and listed. These classes are usually not present in the analysis or architectural models. These classes include GUI components, operating system components, data management components, networking components, etc. For OGRVS, the infrastructure classes are listed below:

1. Report
2. Course
3. Semester
4. Result
5. DAL
6. University

**3 Elaborated Design Class**

The elaborated design classes are the following:

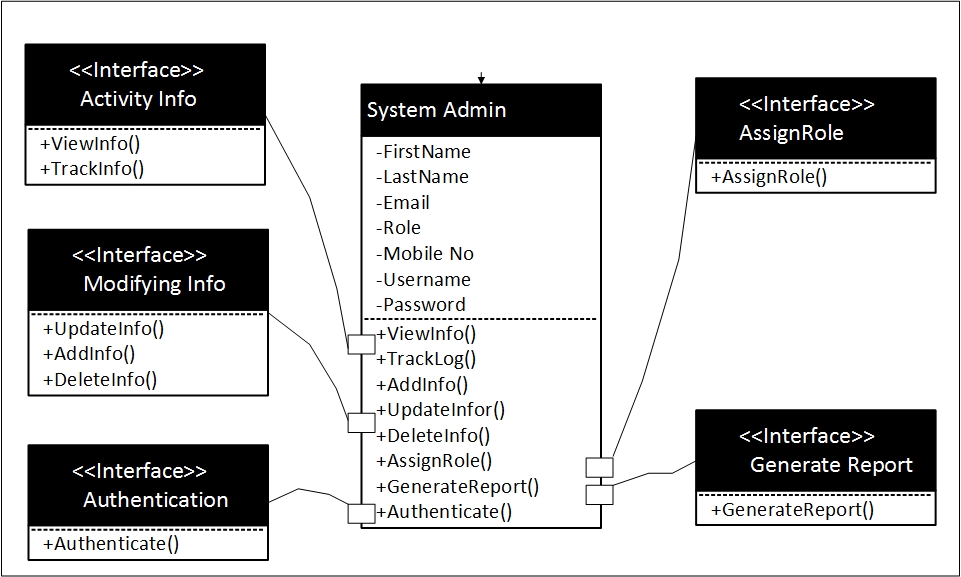


Figure 6: Elaborated Design of System Admin Class

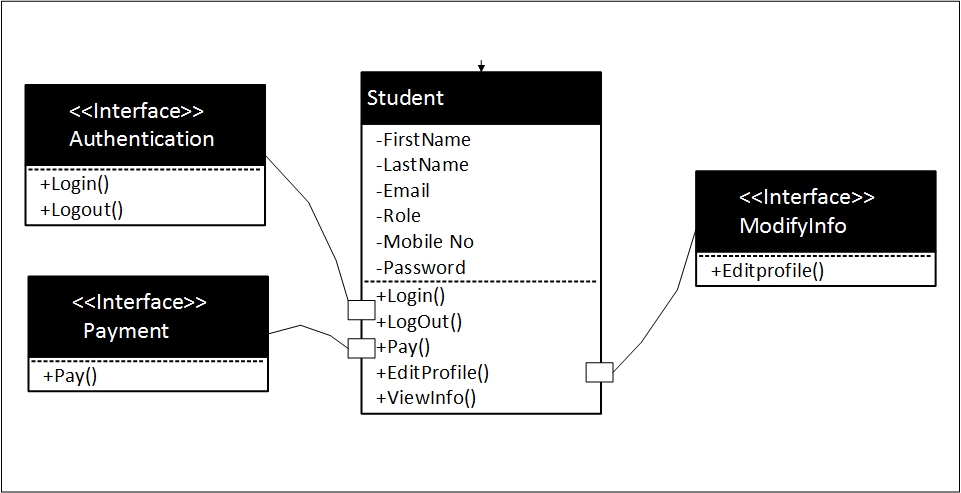


Figure 7: Elaborated Design of Student Class

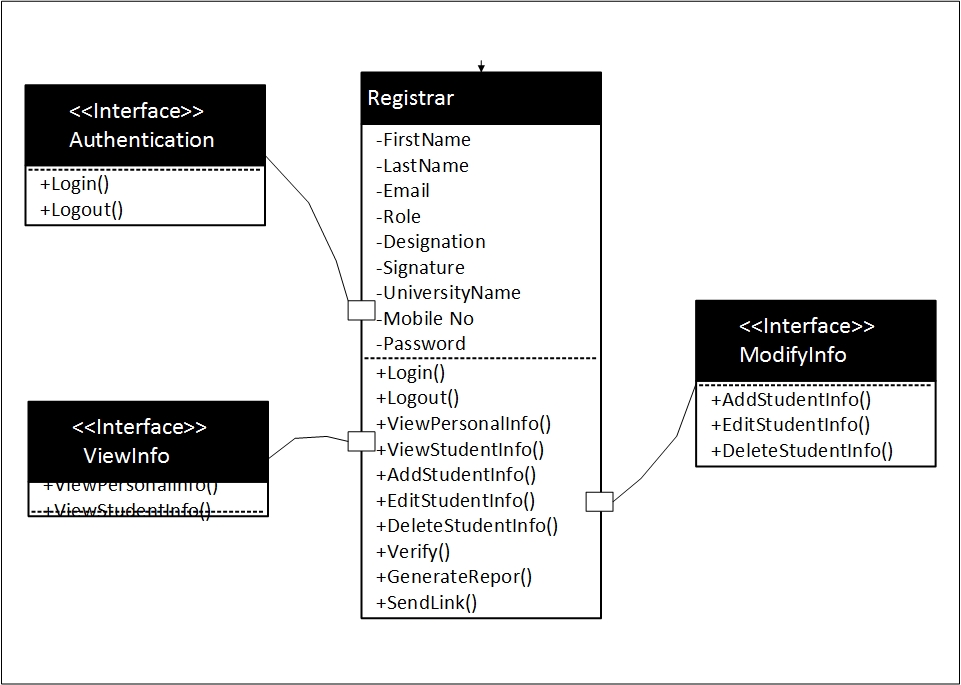


Figure 8: Elaborated Design of Registrar Class

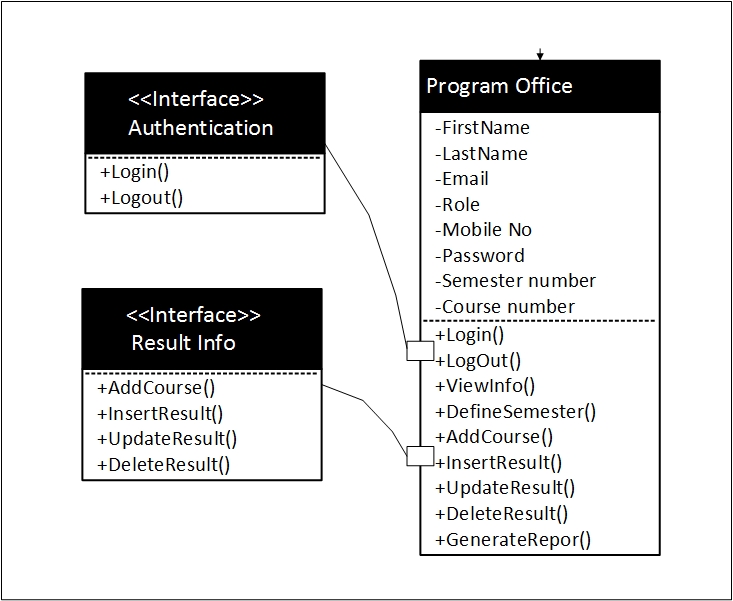
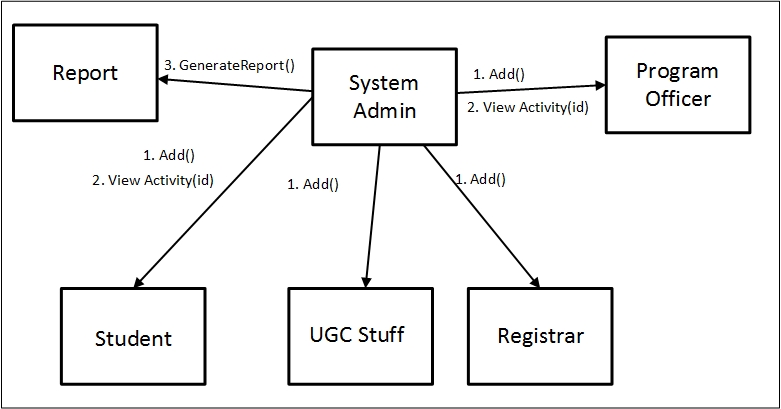


Figure 9: Elaborated Design of Program Office Class



Figure 10: Elaborated Design of UGC Class

3.1 Collaboration Details

In this section, the collaboration between classes will be designed. Following is the diagrams:

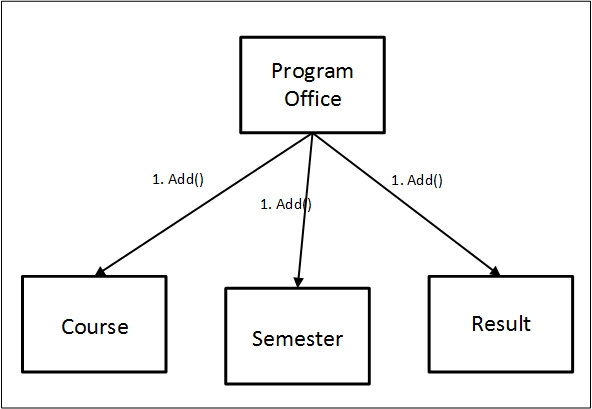
Figure 11: Collaboration Details of System Admin Class

Figure 12: Collaboration Details of Program Office

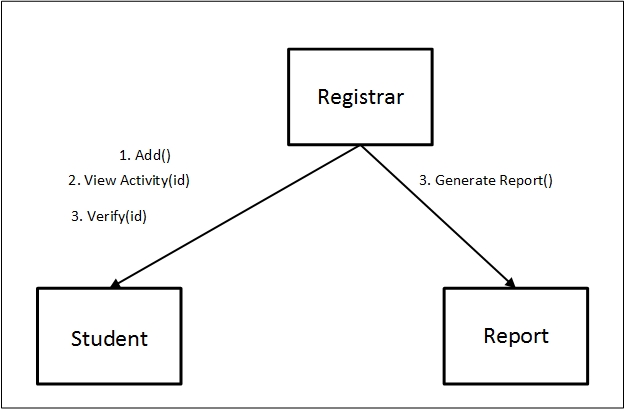


Figure 13: Collaboration Details of Registrar Class

## 3.2 Appropriate Interfaces

There is no necessity to divide the classes in subclasses as they exhibit sufficient cohesion. So there is no need to define appropriate interfaces.

## 3.3 Elaborate Attributes

Elaborate attributes for System **Admin class**:

First-name: string = “name1” {contains name value – Rubaiyat, Tasnim, Farzana}

Last-name: string = “name2” {contains name value – Mumu, fabiha, Muna}

Email: email = “null” {Contains value – [rubaiyat@gmail.com](mailto:rubaiyat@gmail.com), tasnim@hotmail.com}

Role: string = “null” {contains name value – student, admin, PO}

Username: string = “username1” {All Characters}

Password: string= “password1” {All Characters}

Mobile Number: Number = “088 01716724859” {contains name value – 088-01711467599}

Elaborated attributes for **Student class**:

First-name: string = “name1” {contains name value – Rubaiyat, Tasnim, Farzana}

Last-name: string = “name2” {contains name value – Mumu, fabiha, Muna}

Email: email = “null” {Contains value – [rubaiyat@gmail.com](mailto:rubaiyat@gmail.com), tasnim@hotmail.com}

Role: string = “role1” {contains name value – student, admin, PO}

Password: string= “null” {All Characters}

Mobile Number: Number = “088 01716724859” {contains name value – 088-01711467599}

Elaborated attributes for **Registrar class**:

First-name: string = “name1” {contains name value – Rubaiyat, Tasnim, Farzana}

Last-name: string = “name2” {contains name value – Mumu, fabiha, Muna}

Email: email = “null” {Contains value – [rubaiyat@gmail.com](mailto:rubaiyat@gmail.com), tasnim@hotmail.com}

Role: string = “null” {contains name value – student, admin, PO}

Password: string= “password1” {All Characters}

Mobile Number: Number = “088 0000000000” {contains name value – 088-01711467599}

University-name: String= “null” {Contains name value – Dhaka University, Chittagong university}

Designation: string = “null” {Contains name value- Manager, Senior manager}

Elaborated attributes for **Program Office** class:

First-name: string = “name1” {contains name value – Rubaiyat, Tasnim, Farzana}

Last-name: string = “name2” {contains name value – Mumu, fabiha, Muna}

Email: email = “null” {Contains value – [rubaiyat@gmail.com](mailto:rubaiyat@gmail.com), tasnim@hotmail.com}

Role: string = “null” {contains name value – student, admin, PO}

Password: string= “password1” {All Characters}

Mobile Number: Number = “088 0000000000” {contains name value – 088-01711467599}

Semester number: Int = “1” { automatically increments – 1,2,3,….. }

Course number: Int = “1” { automatically increments – 1,2,3,….. }

Elaborated attributes for **UGC Stuff** class:

First-name: string = “name1” {contains name value – Rubaiyat, Tasnim, Farzana}

Last-name: string = “name2” {contains name value – Mumu, fabiha, Muna}

Email: email = “null” {Contains value – [rubaiyat@gmail.com](mailto:rubaiyat@gmail.com), tasnim@hotmail.com}

Role: string = “null” {contains name value – student, admin, PO}

Password: string= “password1” {All Characters}

Mobile Number: Number = “088 0000000000” {contains name value – 088-01711467599}

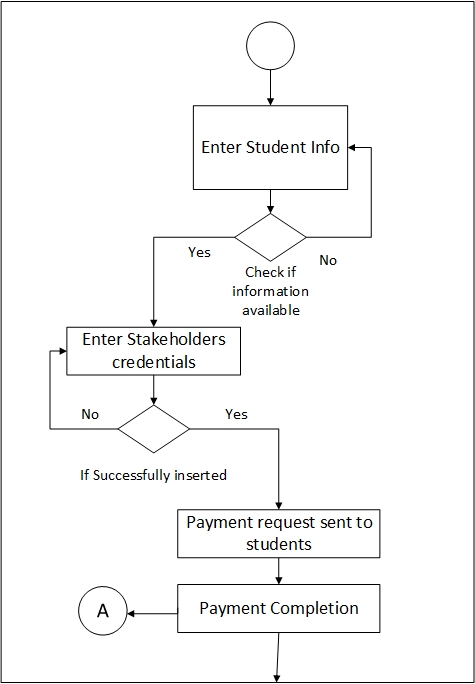
University-name: String= “null” {Contains name value – Dhaka University, Chittagong university}

University-address: String = “null” {All Characters}

University-website: String = “null” {All Characters – www.du.ac.bd}

3.3 Describe Processing Flow

The processing flow of the verification system is described below:



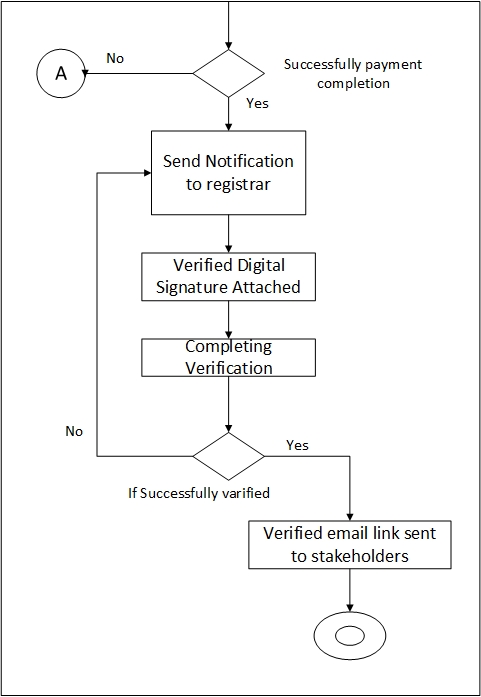


Figure 14: Processing Flow of Verification System

4. Persistent Data

The persistent data sources (databases and files) and the classes required to manage them are following:

**Persistent data source**: Database

**Classes to manage data source**: Database classes for each entity

**Step 5:** Develop or elaborate behavioral representation for a class or component

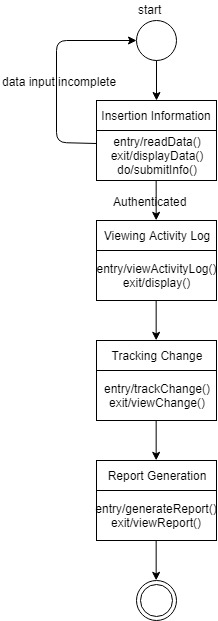


Figure: State chart fragment for “System Admin”

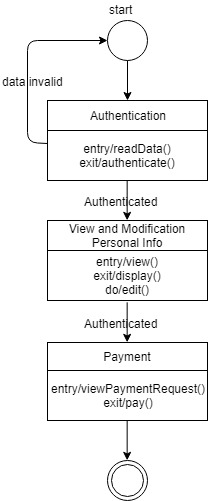


Figure: State chart fragment for “Student”

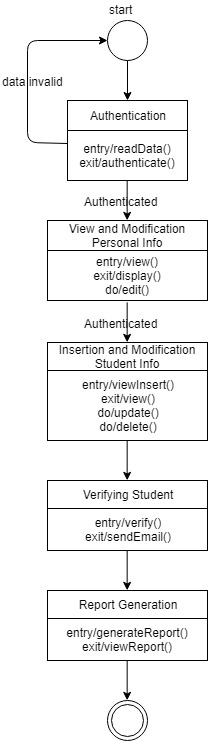


Figure: State chart fragment for “Registrar”

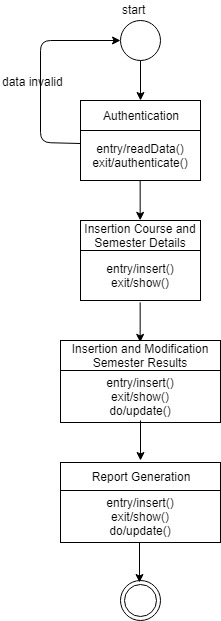


Figure: State chart fragment for “Program Office

6. Elaborative Deployment

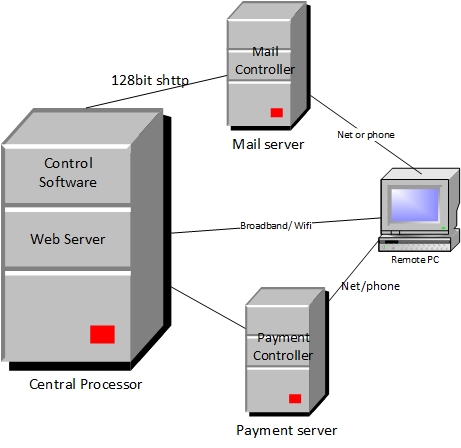
****

Figure 15: Elaborative Deployment of OGRVS