**Software Requirements Specification**

***Version 1.1***

**28th Jan, 2016**

**Academic Registration**

Submitted in partial fulfillment

Of the requirements of

CS 223 Software Engineering

This work is based upon the submissions of the course Software Engineering (CS223).

The students who submitted this team projects were:

1. **Abhinav Rai - B14CS002**
2. **Anurag Sanyal – B14CS007**
3. **Mahendra Meena – B14CS023**
4. **Rishabh Shukla – B14CS028**

# Table of Contents

[Table of Contents i](#_Toc440664050)

[List of Figures ii](#_Toc440664051)

[1.0. Introduction 1](#_Toc440664052)

[1.1. Purpose 1](#_Toc440664053)

[1.2. Scope of Project 1](#_Toc440664054)

[1.3 Constraints 1](#_Toc440664055)

[1.4 Assumptions and Dependencies 1](#_Toc440664056)

[1.3. Glossary 1](#_Toc440664057)

[1.4. References 1](#_Toc440664058)

[1.5. Overview of Document 1](#_Toc440664059)

[2.0. Overall Description 2](#_Toc440664060)

[2.1 System Environment 2](#_Toc440664061)

[2.2 Functional Requirements Specification 2](#_Toc440664062)

[2.2.1 Use case 1 2](#_Toc440664063)

[Use case: 2](#_Toc440664064)

[2.3 User Characteristics 2](#_Toc440664065)

[2.4 Non-Functional Requirements 2](#_Toc440664066)

[3.0. Requirements Specification 3](#_Toc440664067)

[3.1 Functional Requirements 3](#_Toc440664068)

[3.1.1 << Name of the first feature>> 3](#_Toc440664069)

[3.1.2 << Name of the second feature>> 3](#_Toc440664070)

[3.3 Detailed Non-Functional Requirements 3](#_Toc440664071)

[***3.4*** ***Logical Structure of the Data*** 3](#_Toc440664072)

[4.0 Supporting information 3](#_Toc440664073)

[4.1 Table of contents and index 3](#_Toc440664074)

[4.2 Appendixes 3](#_Toc440664075)

# List of Figures

**No table of figures entries found.**

# 1.0. Introduction

## 1.1. Purpose

The purpose of this document is to present a description of Academic Registration System Software. To explain the features, limitations and assumptions of the software.

This document is intended for both the stakeholders and developers of the software.

## 1.2. Scope of Project

This software system will be designed to minimize the time for all stake holders. Stake holders in the project are Students, Faculty, Staff, Admin department, Accounts section. It also aims at removing the discrepancy in the current system. Its motto is to reduce human effort. Informally, it will reduce the average time from 1.5 hour to 10 minutes.

## 1.3 Constraints –

* Network security

## 1.4 Assumptions and Dependencies –

* Given all details of students like roll number, email, gender, category, address and LDAP credentials.
* Payment portal will not be made. Simple Entries in form.
* Library and mess dues will be entered by admin.
* In Phase – 1, we will add all the data manually.
* Time table will not be implemented in Phase – 1.

1.4.1 **Version History – Version 1.1**

|  |  |
| --- | --- |
| Add - Drop Use case | Made separate |
| Language change | Removed words like ‘any’,’everywhere’ |
| New Assumption made | * Data input format added (.xls and txt) * Payment Portal and gateway not in phase 1. |
| Glossary Added | Timetable added |
| UML Diagrams added | Sequence diagram, Activity Diagram, Class Diagram, Use case diagram added. |

## 1.3. Glossary

|  |  |
| --- | --- |
| **Term** | **Definition** |
| Mentor | Member of faculty assigned to each batch who approves registration of student |
| Transcript | Grade sheet of student till last semester |
| Academic Coordinator | Overall Head of Academic Registration Process |
| User | Student, Faculty & Admin Department of IIT Jodhpur. |
| Time Table | A chart in excel format showing starting time of all classes. |
| UI class | User Interface Class |
| Challan | E-receipt generated in a particular format |

## 

## 1.4. References

IEEE. *IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications.* IEEE Computer Society, 1998.

## 1.5. Overview of Document

The rest of the document is designed in the following way:

# 2.0. Overall Description

## 2.1 System Environment

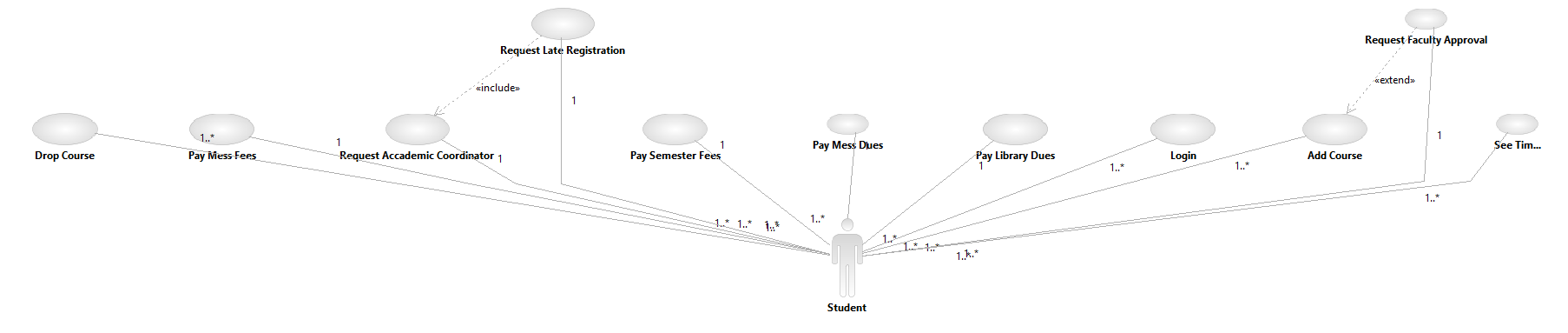
<< Keep blank for the time being >>

## 2.2 Functional Requirements Specification

. << Keep blank for the time being >>

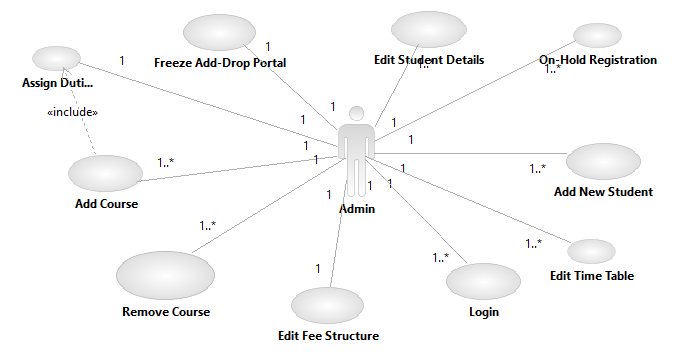
### 2.2.1 Use case 1

#### Use case: Student

**Diagram:**

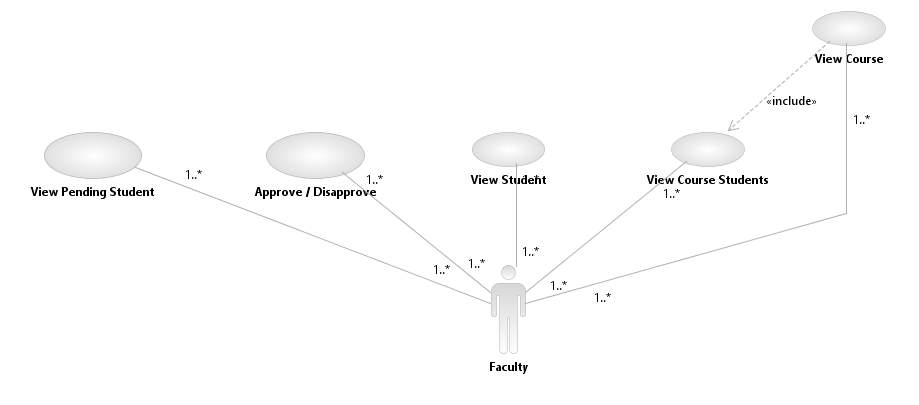
#### Use case: Admin

**Diagram:**

****

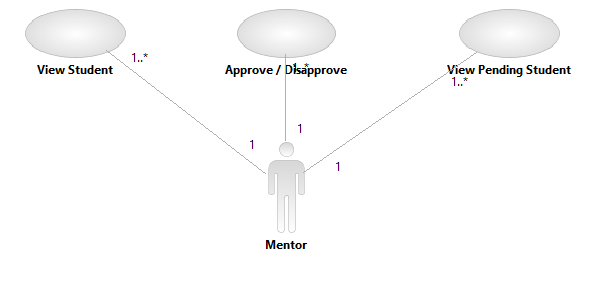
#### Use case: Faculty

**Diagram:**

****

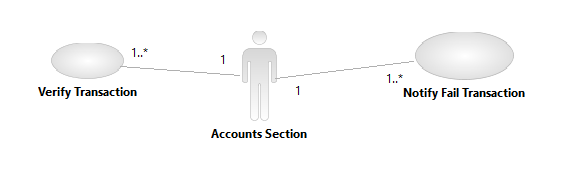
#### Use case: Mentor

**Diagram:**

****

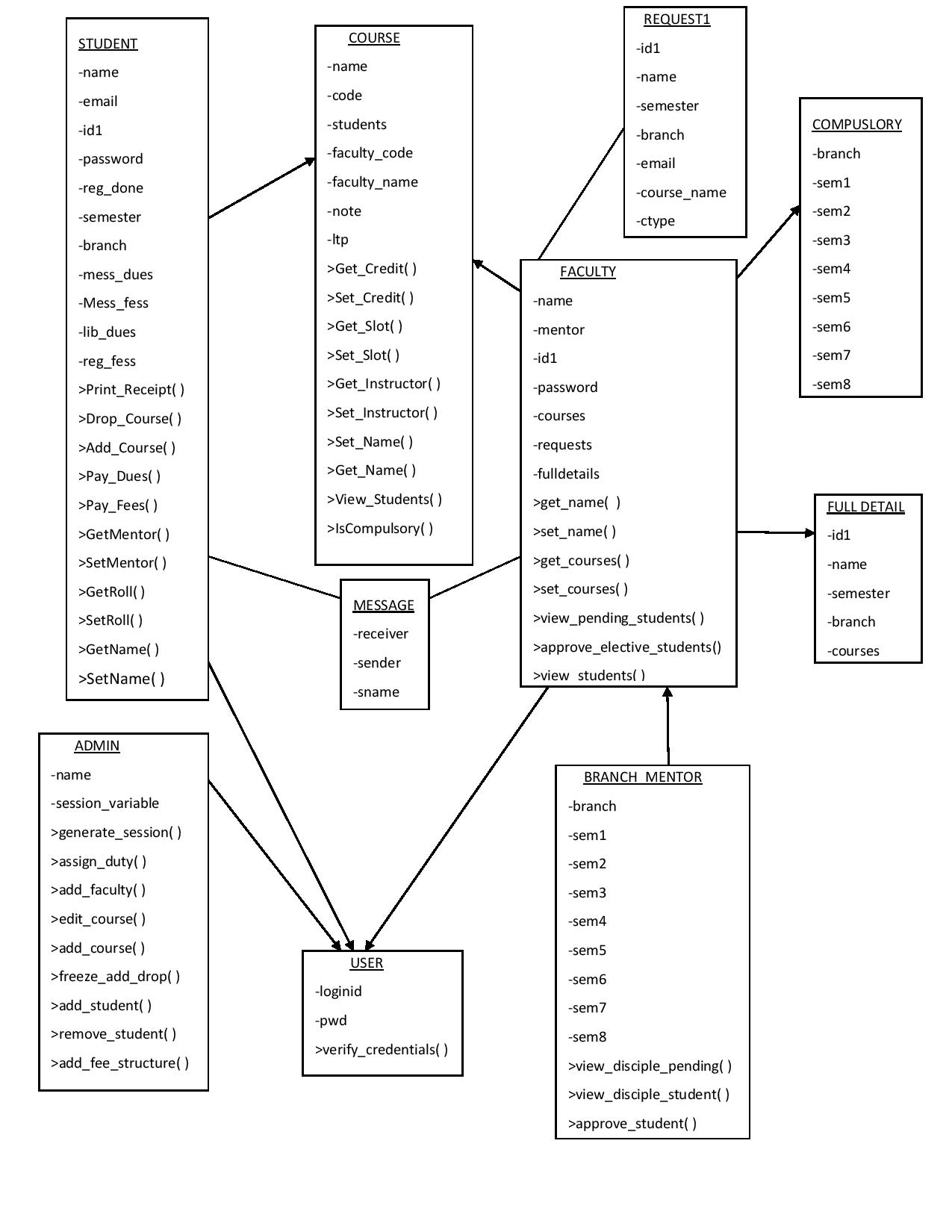
#### Use case: Accounts Section

**Diagram:**

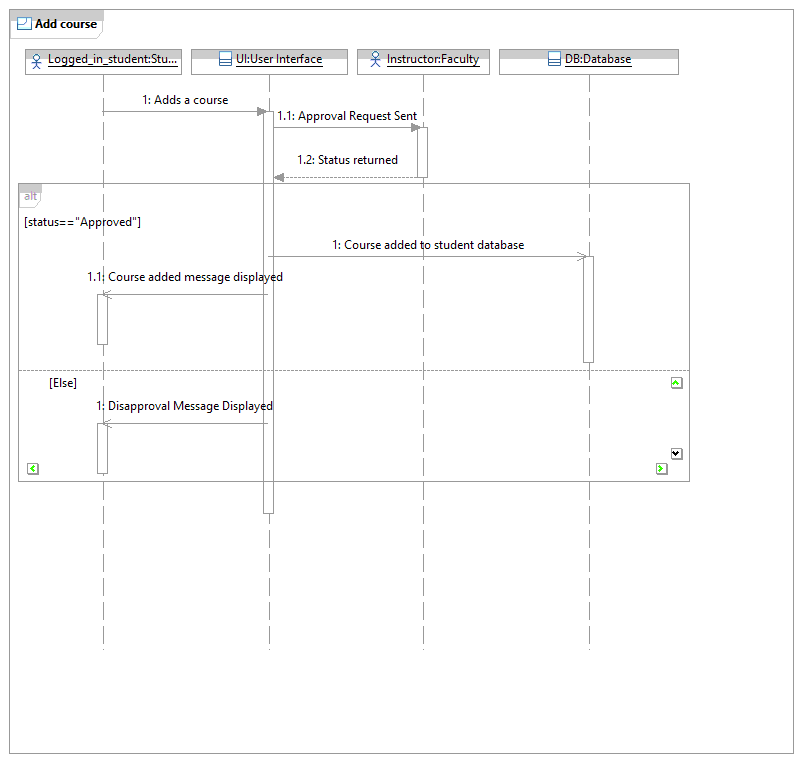
****

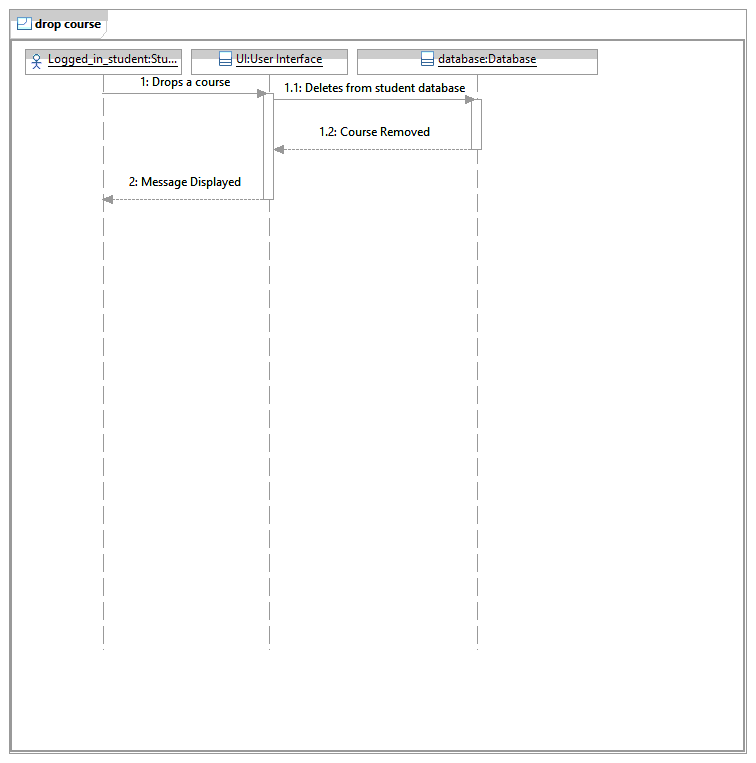
**Diagram:**

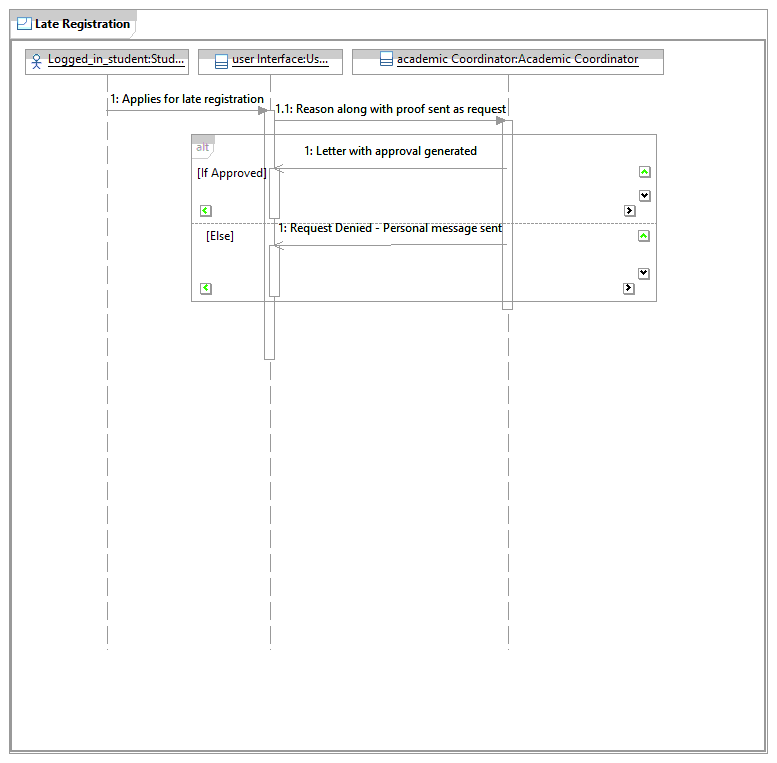
**Class Diagrams:-**

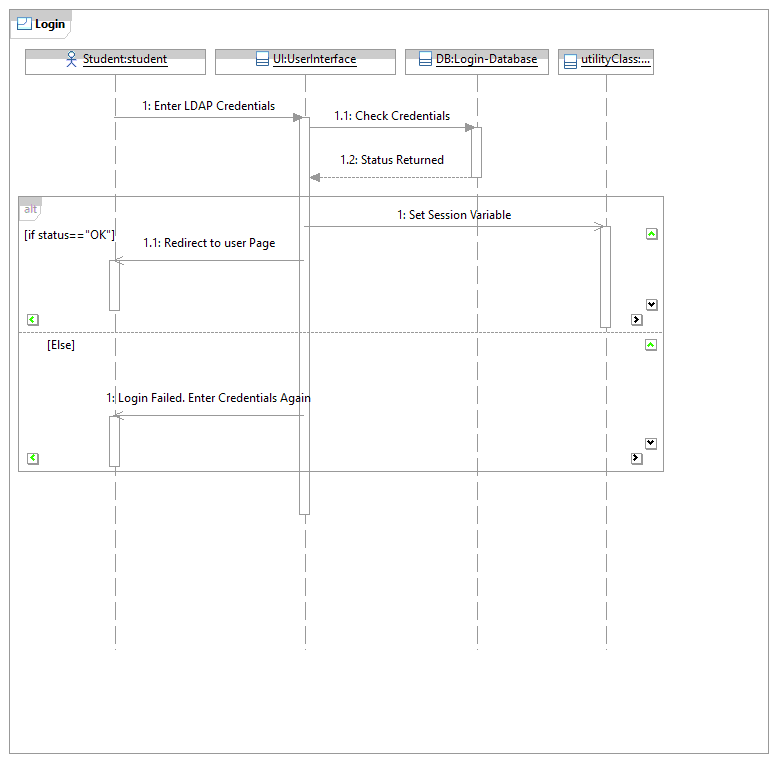
****

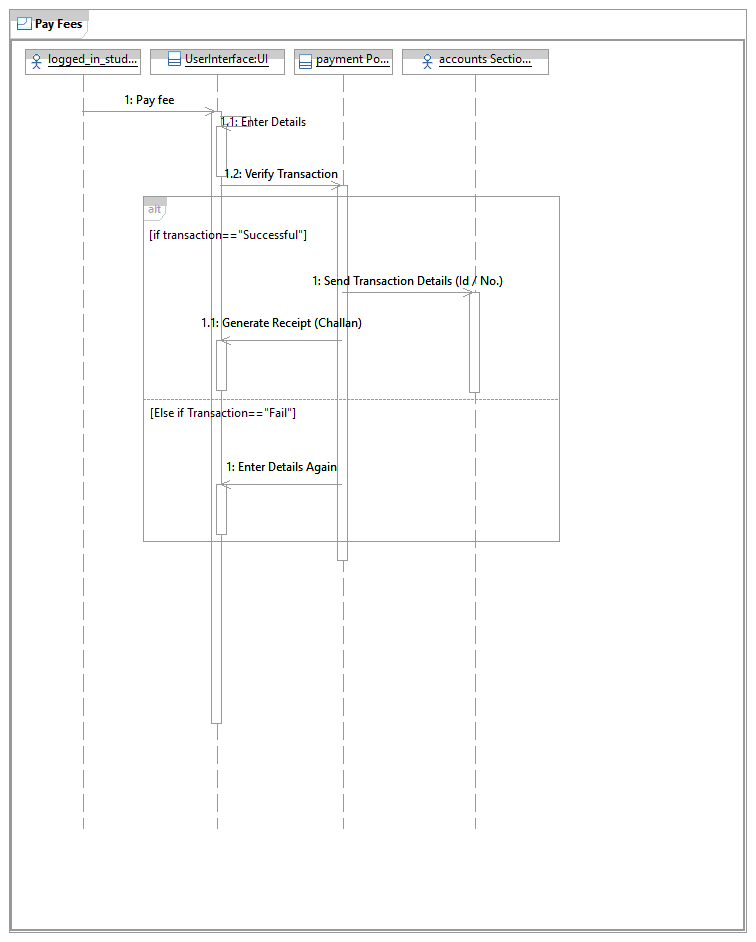
**Sequence Diagram**

****

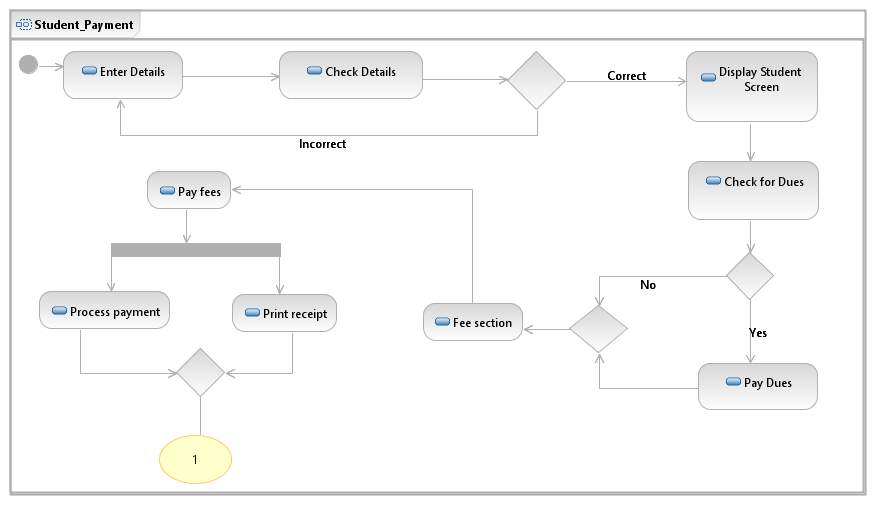
****

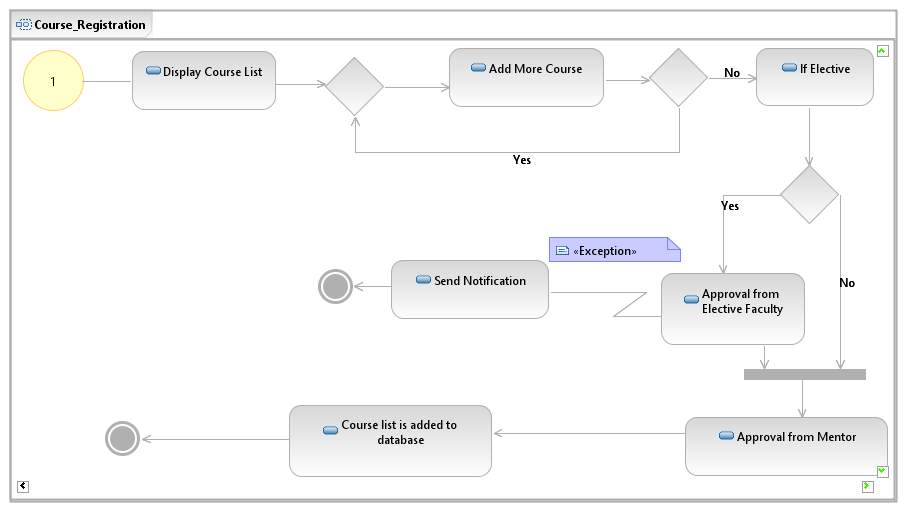






**Activity Diagrams:-**

****

****

**Brief Description**

**Initial Step-By-Step Description**

## 2.3 User Characteristics

User must know how to operate computer and basics of internet.

## 2.4 Non-Functional Requirements

* Internet Facility
* Secured Server
* Database

3.0. Requirements Specification

## 3.1 Functional Requirements

**Student Use Case**

3.1.1 – Add / Drop Course ---------------!!!! Change it

|  |  |
| --- | --- |
| **Use Case Name** | Add/Drop Course |
| **Trigger** | Student Decides to add/drop a particular course |
| **Precondition** | Student is present in the database. He is a registered user.  If the course is not among compulsory courses student, needs faculty approval from specific faculty of that course. |
| **Basic Path** | 1. Student selects the course  2. Click the add/drop button |
| **Alternative Paths** |  |
| **Postcondition** | The course is added/dropped from course list |
| **Exception Paths** |  |
| **Other** | He has to send the online request for Elective course to specific faculty. The course only gets added if the specific faculty approves it. |

3.1.2 – Elective Course Approval

|  |  |
| --- | --- |
| **Use Case Name** | Elective Course Approval. |
| **Trigger** | Student decides to add an elective course. |
| **Precondition** | Student is present in the database. |
| **Basic Path** | 1. A notification is sent to the faculty for approval. 2. Transcript of the person is also shown to the concerned faculty. |
| **Alternative Paths** | On hardcopy – Take approval from concerned faculty and re-submit the application. |
| **Postcondition** | Student is approved or disapproved by the faculty. |
| **Exception Paths** |  |
| **Other** |  |

3.1.3 – Pay Mess and Semester Fee

|  |  |
| --- | --- |
| **Use Case Name** | Pay Mess and Semester Fee |
| **Trigger** | Student decides to pay the fee |
| **Precondition** | Student is present in the database |
| **Basic Path** | 1.Select the mode of payment  2.Enter details  3.Pay the Fee |
| **Alternative Paths** |  |
| **Postcondition** | A receipt with transaction id is generated |
| **Exception Paths** |  |
| **Other** | Details of his transactions are saved in the database and will be given to Accounts section in .xls format.  Will only be activated if Mess and Library dues are completed. |

3.1.4 – Pay Mess Dues

|  |  |
| --- | --- |
| **Use Case Name** | Pay **Mess** **Dues** |
| **Trigger** | Paying of Mess dues of the previous semester. |
| **Precondition** | 1. Student is present in database  2. Student has standing dues  3. We have the details of all the students in Excel. |
| **Basic Path** | 1.Select the mode of payment  2.Enter details  3.Pay the Fee |
| **Alternative Paths** |  |
| **Postcondition** | A receipt with transaction id is generated |
| **Exception Paths** |  |
| **Other** |  |

3.1.5 – Pay Library Dues

|  |  |
| --- | --- |
| **Use Case Name** | Pay Library Dues |
| **Trigger** | Student decides to clear his library dues |
| **Precondition** | 1.Student is present in database  2.Student has standing dues  3. List of students with dues is given in excel sheet. |
| **Basic Path** | 1.Select the mode of payment  2.Enter details  3.Pay the Fee |
| **Alternative Paths** |  |
| **Postcondition** | A receipt with transaction id is generated |
| **Exception Paths** |  |
| **Other** |  |

3.1.6 – Request for late registration

|  |  |
| --- | --- |
| **Use Case Name** | Request for late registration |
| **Trigger** | Student feels the need to apply for late registration. |
| **Precondition** | 1.Student is present in database |
| **Basic Path** | 1.Student clicks on the late registration tab  2. Student specifies the reason for late registration and attaches the relevant document. |
| **Alternative Paths** |  |
| **Postcondition** | The request is sent to the Academic Coordinator along with the documents. |
| **Exception Paths** |  |
| **Other** | His Registration form will only be retrieved if his application is passed by Academic Coordinator. |

3.1.7 – View time table

|  |  |
| --- | --- |
| **Use Case Name** | View time table |
| **Trigger** | Student wants to see the course slot |
| **Precondition** | 1.Student is present in database |
| **Basic Path** | 1.Click the View Time Table button |
| **Alternative Paths** | Decode the Email. |
| **Postcondition** | Time table is shown |
| **Exception Paths** |  |
| **Other** |  |

3.1.8 – View fees Structure

|  |  |
| --- | --- |
| **Use Case Name** | View fees structure |
| **Trigger** | Student decides to view the fees structure |
| **Precondition** | 1.Student is present in database |
| **Basic Path** | 1. Click the View Fees Structure button. |
| **Alternative Paths** |  |
| **Postcondition** | Fees structure is shown |
| **Exception Paths** |  |
| **Other** |  |

**Faculty Use Case**

3.1.9 – View Course students

|  |  |
| --- | --- |
| **Use Case Name** | View course students |
| **Trigger** | Faculty decides to see the students registered in the course |
| **Precondition** | 1.The course is present in the database |
| **Basic Path** | 1.Select the course  2.Click the View Students button |
| **Alternative Paths** |  |
| **Postcondition** | A list of students registered in the course is generated |
| **Exception Paths** |  |
| **Other** | Specific Details of students are shown like email, name, etc. |

3.1.10 – Approve / Disapprove a student by Elective Faculty

|  |  |
| --- | --- |
| **Use Case Name** | Approve/Disapprove a student by Elective Faculty. |
| **Trigger** | Faculty decides to approve/disapprove a student for a specific course. |
| **Precondition** | 1.Student is present in database  2.The course is present in the database  3.Student has applied for the course |
| **Basic Path** | 1.Faculty takes the decision regarding student  2.Click Approve/Disapprove button |
| **Alternative Paths** |  |
| **Postcondition** | A notification is sent to the student as per the decision of faculty |
| **Exception Paths** |  |
| **Other** | His transcript is also shown to the faculty. |

3.1.11 – Approve/Disapprove a student by Faculty Mentor

|  |  |
| --- | --- |
| **Use Case Name** | Approve/Disapprove a student by Faculty Mentor. |
| **Trigger** | Mentor decides to approve/disapprove a student for a specific course. |
| **Precondition** | 1.Student is present in database  2.The course is present in the database  3.Student has applied for the course |
| **Basic Path** | 1.Mentor takes the decision regarding student  2.Click Approve/Disapprove button |
| **Alternative Paths** |  |
| **Postcondition** | A notification is sent to the student as per the decision of mentor |
| **Exception Paths** |  |
| **Other** |  |

3.1.12 – View Student

|  |  |
| --- | --- |
| **Use Case Name** | View Student |
| **Trigger** | Mentor wants to see the full details of a student |
| **Precondition** | 1.Student is present in database |
| **Basic Path** | 1.Click the View Student button |
| **Alternative Paths** |  |
| **Postcondition** | Details of the student is shown |
| **Exception Paths** |  |
| **Other** |  |

**Admin Use case**

3.1.13 – Freeze Add/Drop

|  |  |
| --- | --- |
| **Use Case Name** | Freeze Add/Drop |
| **Trigger** | Admin decides to freeze add/drop option |
| **Precondition** | Expiration of last date of add/drop option |
| **Basic Path** | 1.Click the Freeze button |
| **Alternative Paths** |  |
| **Postcondition** | Add/Drop option is freezed.  No student is able to add/drop a course |
| **Exception Paths** |  |
| **Other** |  |

3.1.14 – Add a new course

|  |  |
| --- | --- |
| **Use Case Name** | Add a new course |
| **Trigger** | Admin decides to add a new course |
| **Precondition** |  |
| **Basic Path** | 1.Click the Add New Course button  2.Enter details of the course  3.Click Submit button |
| **Alternative Paths** |  |
| **Postcondition** | The new course is added to the course list. |
| **Exception Paths** |  |
| **Other** |  |

3.1.15 – Remove Course

|  |  |
| --- | --- |
| **Use Case Name** | Remove course |
| **Trigger** | Admin decides to remove a particular course |
| **Precondition** | 1.Course is present in the database |
| **Basic Path** | 1.Click the Remove Course button.  2.Select the particular course. |
| **Alternative Paths** |  |
| **Postcondition** | The course is removed from the database. |
| **Exception Paths** |  |
| **Other** |  |

3.1.16 – Add/Edit Fees structure

|  |  |
| --- | --- |
| **Use Case Name** | Add/Edit Fees structure |
| **Trigger** | Admin decides to add/edit the fees structure |
| **Precondition** |  |
| **Basic Path** | 1.Click the Add/Edit button  2.Make the changes |
| **Alternative Paths** |  |
| **Postcondition** | Fees structure is added/edited |
| **Exception Paths** |  |
| **Other** |  |

3.1.17 – Add time table

|  |  |
| --- | --- |
| **Use Case Name** | Add time table |
| **Trigger** | Admin wants to add a time table |
| **Precondition** |  |
| **Basic Path** | 1.Click the Add Time Table button  2. Upload Time table. |
| **Alternative Paths** |  |
| **Postcondition** | Time table is added |
| **Exception Paths** |  |
| **Other** |  |

3.1.18 – Add new student

|  |  |
| --- | --- |
| **Use Case Name** | Add new student |
| **Trigger** | A new batch has arrived. |
| **Precondition** | Addition by Admin. |
| **Basic Path** | 1.Click the Add Student button  2.Enter details of the student |
| **Alternative Paths** | Manually add. |
| **Postcondition** | The new student is added to the database. |
| **Exception Paths** |  |
| **Other** |  |

3.1.19 – Edit / Delete Student

|  |  |
| --- | --- |
| **Use Case Name** | Edit / Delete student |
| **Trigger** | Student has completed his programme/ Student left IITJ. Data Changed of a student. |
| **Precondition** | 1.Student is present in database |
| **Basic Path** | 1.Click the Delete Student button  2.Select the student. |
| **Alternative Paths** |  |
| **Postcondition** | The particular student is deleted from the database. |
| **Exception Paths** |  |
| **Other** |  |

**Accounts Section Use Case**

3.1.20 – Verify transaction

|  |  |
| --- | --- |
| **Use Case Name** | Verify transaction |
| **Trigger** | Account Section wants to verify transaction of all student after payment |
| **Precondition** | 1.Transaction database is available. |
| **Basic Path** | 1. Click generate excel option.  2. Send the Excel file to accounts section. |
| **Alternative Paths** | Manually – By hand. |
| **Postcondition** | List of failed transaction is shown  In case of no failed transaction, All Transaction Successful message is shown |
| **Exception Paths** | Check Manually by hand with the database printed sheets. |
| **Other** |  |

3.1.21 – Print Receipt and challan of a student

|  |  |
| --- | --- |
| **Use Case Name** | Print Receipt and challan of a student |
| **Trigger** | Student has to submit the hard copies of payment receipt |
| **Precondition** | 1.Student is present in database  2.No dues remaining  3.Fees is paid |
| **Basic Path** | 1.Click the Print Receipt button |
| **Alternative Paths** |  |
| **Postcondition** | Receipts are printed.­­­ |
| **Exception Paths** |  |
| **Other** | Automatically Challan is filled with payment id and other details. |

## 3.3 Detailed Non-Functional Requirements -

## A fully setup working server will be required to host the application. A database application like MongoDB will be required on server to store database. Server should be able to handle all the users. Internet facility is required.

### ***3.4 Logical Structure of the Data***

<< Keep this blank for the time being>>

# 4.0 Supporting information

## 4.1 Table of contents and index

## 4.2 Appendixes