



# **SOFTWARE DESIGN AND ANALYSIS ON PGDIT AUTOMATION SYSTEM**

# **SOFTWARE DESIGN AND ANALYSIS ON PGDIT AUTOMATION SYSTEM**

Submitted to:

Nadia Nahar

Lecturer

Institute of Information Technology

University of Dhaka

Submitted by:

Jarifa Khatun (0710)

Asadullah Hil Galib (0712)

Satyaki Das (0720)

Moumita Asad (0731)

Shafi ul Alam (0735)

3<sup>rd</sup> Year, 6<sup>th</sup> Semester, 7<sup>th</sup> Batch

Session: 2014-2015

Institute of Information Technology

University of Dhaka

Submission Date: 19th November, 2017

# **Letter of Transmittal**

19th November, 2017

Nadia Nahar

Lecturer

Institute of Information Technology

University of Dhaka

Subject: Submission of design document on “PGDIT Automation System”.

Madam,

We, the team, on which the project on “PGDIT Automation System” was assigned, are submitting our report with due respect. We have tried our best for the report. However, it might lack perfection. So, we therefore, hope that you would be kind enough to accept our report and oblige thereby.

Yours sincerely

Jarifa Khatun (0710)

Asadullah Hil Galib (0712)

Satyaki Das (0720)

Moumita Asad (0731)

Shafi ul Alam (0735)

3rd Year, 6th Semester, 7th Batch

Institute of Information Technology

University of Dhaka

Session: 2014-15

## Table of Contents

Chapter 1. Introduction .....	1
Chapter 2. Architectural Design of PGDIT Automation System.....	2
2.1 Representing the System in Context .....	2
2.2 Defining Archetypes.....	4
2.3 Refining the Architecture into Components.....	6
2.4 Describing Instantiations of the System .....	7
2.5 Architectural mapping using data flow.....	7
Chapter 3. Component-Level Design of PGDIT Automation System .....	24
3.1 Identify All Design Classes that Correspond to the Problem Domain .....	24
3.2 Identify All Design Classes that Correspond to the Infrastructure Domain .....	25
3.3 Elaborate All Design Classes that are not Acquired as Reusable Components .....	26
3.3.1 Specify Message Details when Classes or Components Collaborate .....	37
3.3.2 Identify Appropriate Interfaces for Each Component .....	38
3.3.3 Elaborate Attributes and Define Data Types and Data Structures required to Implement Them.....	41
3.3.4 Describe Processing Flow within Each Operation in Detail .....	44
3.4 Describe Persistent Data Sources and Identify the Classes required to Manage Them ....	86
3.5 Develop and Elaborate Behavioral Representations for a Class or Component .....	87
3.6 Elaborate Deployment Diagrams to Provide Additional Implementation Detail .....	93
Chapter 4. User Interface Design .....	94
4.1 Interface Analysis .....	94
4.1.1 User Analysis .....	94
4.1.2 Task Analysis .....	98
4.2 Interface Design Steps .....	103
4.2.1 Define Interface Objects and Actions .....	103
4.2.2 Define Events that will Cause the State of the User Interface to Change .....	131
4.2.3 Depict Each Interface State as It Look to End User .....	138
References .....	164

## List Of Figures

Figure 1 Design Model .....	1
Figure 2 Steps of Architectural Design.....	2
Figure 3 Architectural Context Diagram of PGDIT Automation System .....	3
Figure 4 Archetypes .....	5
Figure 5 Overall Architectural Structure with Top-Level Components .....	6
Figure 6 An Instantiation of the PGDIT Automation System Architecture .....	7
Figure 7 Architectural Mapping .....	8
Figure 8 Level 0 Dataflow Diagram .....	8
Figure 9 Level 1 Dataflow Diagram .....	9
Figure 10 Dataflow Diagram of Authentication and Registration .....	9
Figure 11 Program Chairperson's Dataflow Diagram of Registration .....	9
Figure 12 Teacher's Dataflow Diagram of Registration .....	10
Figure 13 User's Dataflow Diagram of Authentication .....	10
Figure 14 Dataflow Diagram of Admission System.....	11
Figure 15 Applicant's Dataflow Diagram of Admission System .....	11
Figure 16 Exam Controller's Dataflow Diagram of Admission System .....	12
Figure 17 Receptionist's Dataflow Diagram of Admission System .....	12
Figure 18 Scrutinizing team's Dataflow Diagram of Admission System .....	12
Figure 19 Teacher's Dataflow Diagram of Admission System .....	12
Figure 20 Dataflow Diagram of Course Management System .....	13
Figure 21 Program Chairperson's Dataflow Diagram of Course Management System .....	13
Figure 22 Student's Dataflow Diagram of Course Management System .....	13
Figure 23 Teacher's Dataflow Diagram of Course Management System .....	14
Figure 24 Dataflow Diagram of Result Generation System .....	14
Figure 25 Program Chairperson's Dataflow Diagram of Result Generation System .....	15
Figure 26 Exam Controller's Dataflow Diagram of Result Generation System.....	15
Figure 27 Teacher's Dataflow Diagram of Result Generation System.....	16
Figure 28 Student's Dataflow Diagram of Result Generation System.....	16
Figure 29 DFD mapping of Teacher's Dataflow Diagram of Registration .....	16
Figure 30 DFD mapping of Program chairperson's Dataflow Diagram of Registration .....	17
Figure 31 DFD mapping of User's Dataflow Diagram of Authentication .....	17
Figure 32 DFD mapping of Applicant's Dataflow Diagram of Admission System .....	18
Figure 33 Exam Controller's Dataflow Diagram of Admission System .....	18
Figure 34 DFD mapping of Receptionist's Dataflow Diagram of Admission System .....	19
Figure 35 DFD mapping of Scrutinizer's Dataflow Diagram of Admission System .....	19
Figure 36 DFD mapping of Teacher's Dataflow Diagram of Admission System .....	20

Figure 37 DFD mapping of Program Chairperson's Dataflow Diagram of Course Management System.....	20
Figure 38 DFD mapping of Program Chairperson's Dataflow Diagram of Result Generation System.....	21
Figure 39 DFD mapping of Exam Controllers's Dataflow Diagram of Result Generation System	22
Figure 40 DFD mapping of Teacher's Dataflow Diagram of Result Generation System.....	23
Figure 41 DFD mapping of Student's Dataflow Diagram of Result Generation System.....	23
Figure 42 Steps of Component-Level Design.....	24
Figure 43 Analysis classes .....	25
Figure 44 Design classes .....	26
Figure 45 Elaborated Applicant class.....	27
Figure 46 Elaborated User class.....	29
Figure 47 Elaborated ProgramChairperson class.....	<b>Error! Bookmark not defined.</b>
Figure 48 Elaborated Scrutinizer class .....	31
Figure 49 Elaborated Receptionist class .....	32
Figure 50 Elaborated ExamController class .....	<b>Error! Bookmark not defined.</b>
Figure 51 Elaborated Teacher class .....	36
Figure 52 elaborated Student class .....	37
Figure 53 Collaboration Diagram with Messaging of Initiating Admission ....	<b>Error! Bookmark not defined.</b>
Figure 54 Collaboration Diagram with Messaging of Selecting Applicants ....	<b>Error! Bookmark not defined.</b>
Figure 55 Collaboration Diagram with Messaging of Admission Test related Activities.....	<b>Error! Bookmark not defined.</b>
Figure 56 Collaboration Diagram with Messaging of Activities after Admission Test.....	<b>Error! Bookmark not defined.</b>
Figure 57 Collaboration Diagram with Messaging of Approving New Teacher Account .....	<b>Error! Bookmark not defined.</b>
Figure 58 Collaboration Diagram with Messaging of Course Management...	<b>Error! Bookmark not defined.</b>
Figure 59 Collaboration Diagram with Messaging of Initial Result Generation System.....	<b>Error! Bookmark not defined.</b>
Figure 60 collaboration diagram with messaging of supplementary result generation system. .....	<b>Error! Bookmark not defined.</b>
Figure 61 Collaboration Diagram with Messaging of Final Result Generation System .....	<b>Error! Bookmark not defined.</b>
Figure 62 Authentication class.....	38
Figure 63 RollGenerator class .....	39

Figure 64 ReportCard class .....	39
Figure 65 RoomHandler class .....	40
Figure 66 HeaderDisplayer class .....	41
Figure 67 PDFHandler class.....	41
Figure 68 verifyUsernameAndPassword() of Authentication.....	44
Figure 69 forgotPassword() of Authentication .....	45
Figure 70 verifyUsernameAndRecoveryCode() of Authentication .....	46
Figure 71 setNewPassword() of Authentication .....	46
Figure 72 applyForAdmission() of Applicant.....	47
Figure 73 checkAvailabilityOfAdmitCard() of Applicant .....	48
Figure 74 generateAdmitCard() of Applicant.....	48
Figure 75 generateQuery() of QueryGenerator.....	49
Figure 76 updateAddress() of User .....	50
Figure 77 updateMobileNo() of User.....	50
Figure 78 updatePicture() of User .....	51
Figure 79 getBlockingNotification() of User .....	51
Figure 80 viewName() of HeaderDisplayer .....	52
Figure 81 viewImage() of HeaderDisplayer .....	52
Figure 82 viewListOfUsers() of ProgramChairperson .....	53
Figure 83 updateFullName() of ProgramChairperson .....	53
Figure 84 updateUserEmail () of ProgramChairperson .....	54
Figure 85 addNewStudentsIntoCourse() of ProgramChairperson .....	54
Figure 86 approveAccountRequest() of ProgramChairperson .....	55
Figure 87 checkFulfillmentOfPayment() of ProgramChairperson .....	56
Figure 88 selectInstructor() of ProgramChairperson.....	57
Figure 89 createStudentAccount() of ProgramChairperson .....	57
Figure 90 viewAccountRequest() of ProgramChairperson .....	<b>Error! Bookmark not defined.</b>
Figure 91 viewCourseWiseResult() of ProgramChairperson .....	58
Figure 92 viewInformationOfOneUser() of ProgramChairperson .....	59
Figure 93 viewListOfCourses () of ProgramChairperson.....	59
Figure 94 checkGradeForSupplementary() of Teacher.....	60
Figure 95 generateResultPdf() of Teacher .....	60
Figure 96 generateGrade() of Teacher .....	61
Figure 97 generateSupplementaryList() of Teacher .....	62
Figure 98 generateSupplementaryResultPdf() of Teacher .....	63
Figure 99 getResultSheet() of Teacher .....	63
Figure 100 notifyForSupplementaryExam() of Teacher .....	64
Figure 101 setCourseMark() of Teacher .....	64

Figure 102 setSupplementaryMark() of Teacher .....	65
Figure 103 updateCourseInformation() of Teacher .....	65
Figure 104 updatePDF() of Teacher .....	<b>Error! Bookmark not defined.</b>
Figure 105 viewCourseInformation() of Teacher .....	66
Figure 106 viewListOfCourse() of Teacher .....	66
Figure 107 viewSupplementaryList() of Teacher .....	67
Figure 108 generateResultPdf() of PDFHandler .....	67
Figure 109 generateSupplementaryResultPdf() of PDFHandler .....	68
Figure 110 updatePdf() of PDFHandler .....	68
Figure 111 uploadFinalResultPdf() of PDFHandler .....	69
Figure 112 getCourseAllocationNotification() of Student .....	<b>Error! Bookmark not defined.</b>
Figure 113 viewCourseDetails() of Student .....	71
Figure 114 viewCourseList() of Student .....	72
Figure 115 generatePdf() of ReportCard .....	73
Figure 116 getMarks() of ReportCard .....	73
Figure 117 addApplicationTimeRange() of Exam Controller .....	74
Figure 118 addPaymentTimeRange() of Exam Controller .....	74
Figure 119 addScrutinizationTimeRange() of Exam Controller .....	75
Figure 120 endAdmissionSession() of ExamController .....	75
Figure 121 lockVivaMarks() of ExamController .....	76
Figure 122 lockWrittenMarks() of ExamController .....	76
Figure 123 updateAvailabilityOfAdmitCard() of ExamController .....	77
Figure 124 updateEnrollmentState() of ExamController .....	77
Figure 125 updateVivaSchedule() of ExamController .....	78
Figure 126 updateWrittenMarks() of ExamController .....	78
Figure 127 viewListOfApplicants() of ExamController .....	79
Figure 128 viewListOfEligibleCandidateAfterFullResult() of ExamController .....	79
Figure 129 viewListOfEnrolledCandidates() of ExamController .....	80
Figure 130 addNewRoom() of RoomHandler .....	80
Figure 131 generateSeatPlan() of RoomHandler .....	81
Figure 132 showListOfRoom() of RoomHandler .....	81
Figure 133 updateRoomCapacity() of RoomHandler .....	82
Figure 134 viewFilteredApplications() of Scrutinizer .....	82
Figure 135 searchByApplicationID () of Scrutinizer .....	83
Figure 136 updateStatusOfEligibility() of Scrutinizer .....	83
Figure 137 assignRollToEligibleApplicants() of RollGenerator .....	84
Figure 138 generateRoll() of RollGenerator .....	85
Figure 139 updateStatusOfPayment() of Receptionist .....	85

Figure 140 viewApplicantInformation() of Receptionist .....	86
Figure 141 DatabaseHandler class.....	86
Figure 142 State Diagram of the Applicant class .....	87
Figure 143 State Diagram of the Authentication class .....	87
Figure 144 state diagram of the Authentication class.....	88
Figure 145 State Diagram of the HeaderDisplayer class .....	88
Figure 146 State Diagram of the ProgramChairperson class.....	89
Figure 147 State Diagram of the ExamController class .....	90
Figure 148 State Diagram of the RoomHandler class.....	91
Figure 149 State Diagram of the Receptionist class .....	91
Figure 150 State Diagram of the Scrutinizer class .....	91
Figure 151 State Diagram of the RollGenerator class.....	91
Figure 152 State Diagram of the Teacher class .....	92
Figure 153 State Diagram of the PDFHandler class .....	92
Figure 154 State Diagram of the Student class.....	93
Figure 155 State Diagram of the ReportCard class.....	93
Figure 156 Deployment Diagram of PGDIT Automation System.....	94
Figure 157 Interface Design Steps .....	103
Figure 158 Home Page .....	138
Figure 159 Registration Page .....	139
Figure 160 About us Page .....	140
Figure 161 Contact us Page.....	141
Figure 162 Program Chairperson:Profile .....	142
Figure 163Program Chairperson: Course Management.....	143
Figure 164 Program Chairperson: Account Requests .....	144
Figure 165 Program Chairperson: View Users .....	145
Figure 166 Teacher: Profile .....	146
Figure 167 Teacher: View Course .....	147
Figure 168 Teacher: Manage Results.....	148
Figure 169 Teacher: Notifications.....	149
Figure 170 Exam Controller: Profile.....	150
Figure 171 Exam Controller: Introduce New Admission.....	150
Figure 172Exam Controller: Update Admission Information .....	151
Figure 173 Exam Controller: Additional Admission Initiation Activities.....	151
Figure 174 Exam Controller: Upload Marks written/VIVA .....	152
Figure 175 Exam Controller: Generate Seat Plan .....	153
Figure 176 Exam Controller: Update Enrollment State .....	153
Figure 177 Exam Controller: Manage Rooms .....	154

Figure 178 Exam Controller: Term Result Management .....	155
Figure 179 Exam Controller: Update Director's Signature .....	155
Figure 180 Exam Controller: Notifications.....	156
Figure 181 Student: Profile .....	157
Figure 182 Student: View Running Courses.....	158
Figure 183 Student: View Courses Not Yet Taken .....	159
Figure 184 Student: View Courses Already Taken .....	160
Figure 185 Student: Notifications .....	160
Figure 186 Scrutinizer: View Applicants .....	161
Figure 187 Receptionist: Profile.....	162
Figure 188 Receptionist: View Applicants .....	163

#### List of Tables

Table 1 Elaborated Attributes of Design classes.....	42
Table 2 Analyzing Teacher .....	95
Table 3 Analyzing Student.....	95
Table 4 Analyzing Program Chairperson.....	96
Table 5 Analyzing Applicant .....	96
Table 6 Analyzing Exam Controller .....	96
Table 7 Analyzing Scrutinizer .....	97
Table 8 Analyzing Receptionist .....	97

## Chapter 1. Introduction

Software design encompasses the set of principles, concepts, and practices that lead to the development of a high-quality product. According to Mitch Kapor, the creator of Lotus 1-2-3, "Design is where you stand with a foot in two worlds—the world of technology and the world of people and human purposes—and you try to bring the two together." Design allows to model the system to be built in such a way that the model can be assessed for quality and improved before code is generated, tests are conducted, and end users become involved in large numbers. Design is the place where software quality is established.

The design model provides detail about software architecture, data structures, interfaces, and components that are necessary to implement the system. Figure 1 shows the elements of design model. This document contains the architectural design, component-level design and interface design of PGDIT Automation System.

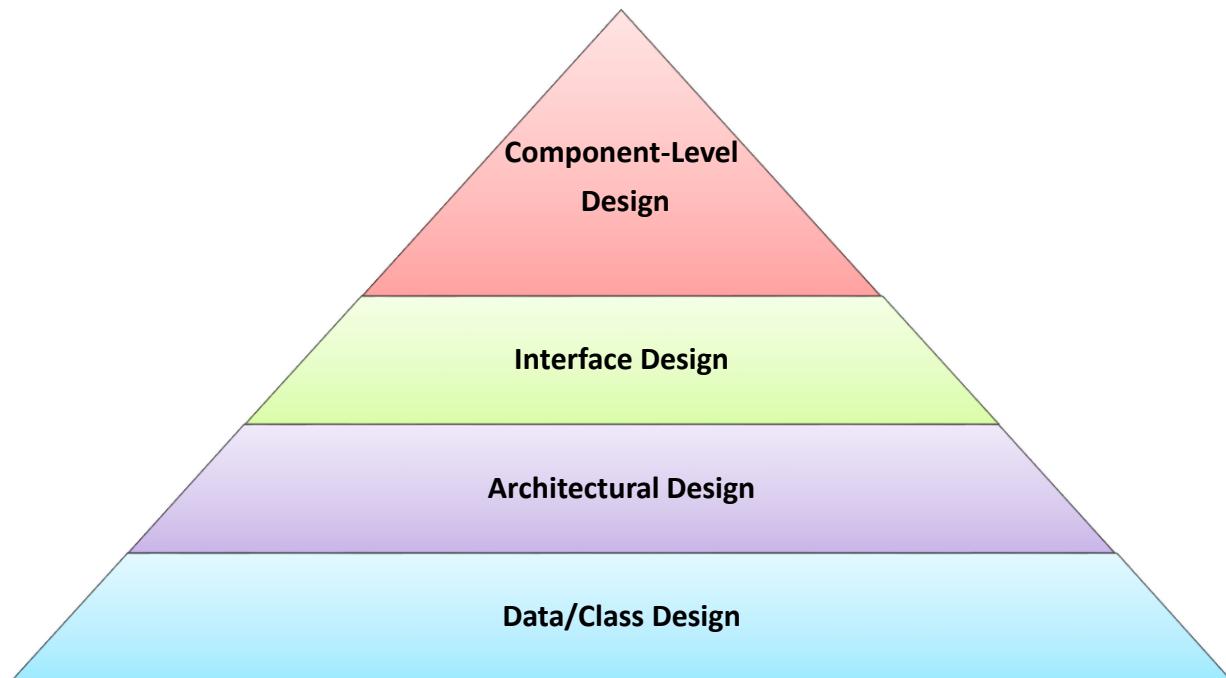


Figure 1 Design Model

## Chapter 2. Architectural Design of PGDIT Automation System

The software architecture of a program is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them. Architectural design represents the structure of data and program components that are required to build a computer-based system. It considers the architectural style that the system will take, the structure and properties of the components that constitute the system, and the interrelationships that occur among all architectural components of a system. Architectural design provides big picture of the system.

Figure 2 shows the steps of Architectural design.

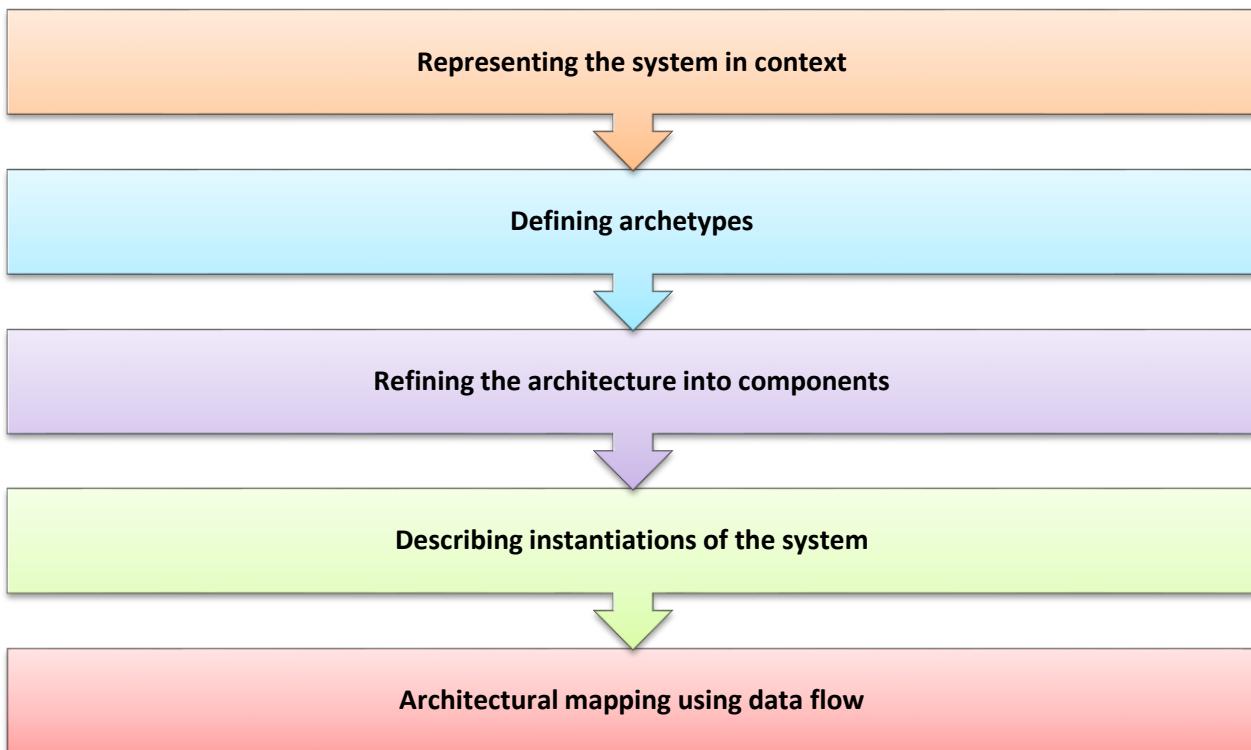


Figure 2 Steps of Architectural Design

### 2.1 Representing the System in Context

Architectural Context Diagram is used to model the manner in which software interacts with entities external to its boundaries. In architectural context diagram systems that interoperate with the target are represented as

- Super-ordinate systems: Use target system as part of some higher level processing scheme

- Sub-ordinate systems: Used by target system and provide necessary data or processing
- Peer-level systems: Interact on a peer-to-peer basis with target system to produce or consume data
- Actors: People or devices that interact with target system to produce or consume data

Each of these external entities communicates with the target system through an interface.

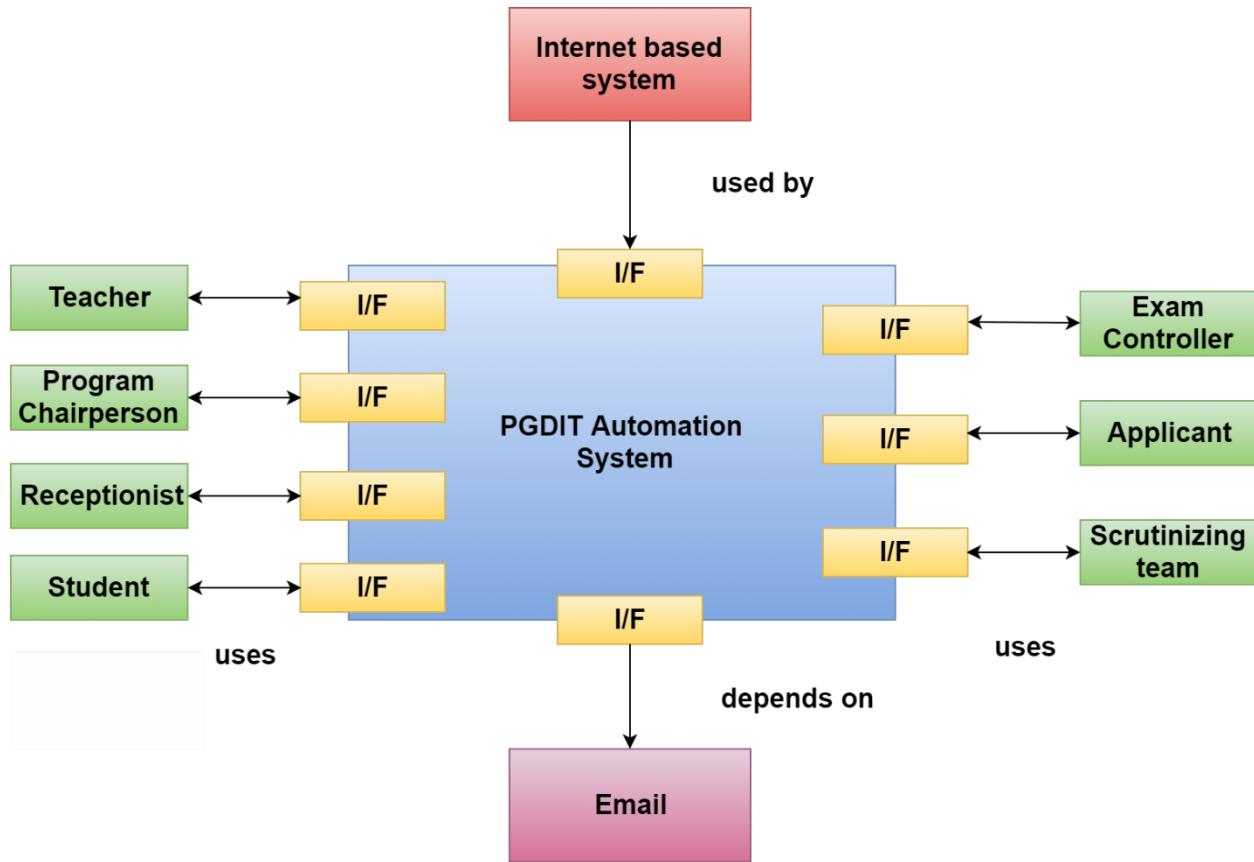


Figure 3 Architectural Context Diagram of PGDIT Automation System

Figure 3 shows architectural context diagram of PGDIT Automation System. The Internet based system is the superordinate system here. Teacher, program chairperson, exam controller, scrutinizing team, student, applicant, and receptionist are actors that are both producers and consumers of information used or produced by the PGDIT Automation System. Finally, email is used by the system and is shown as subordinate to it.

## 2.2 Defining Archetypes

An archetype is a class or pattern that represents a core abstraction that is critical to the design of an architecture for the target system. They represent stable elements of the architecture. Archetypes can be derived by examining the analysis classes defined as part of the requirements model. The archetypes of PGDIT Automation System are:

- Teacher
- ProgramChairperson
- ExamController
- Scrutinizer
- Student
- Applicant
- Receptionist

The archetypes, their attributes, methods and relationships are illustrated in figure 4.

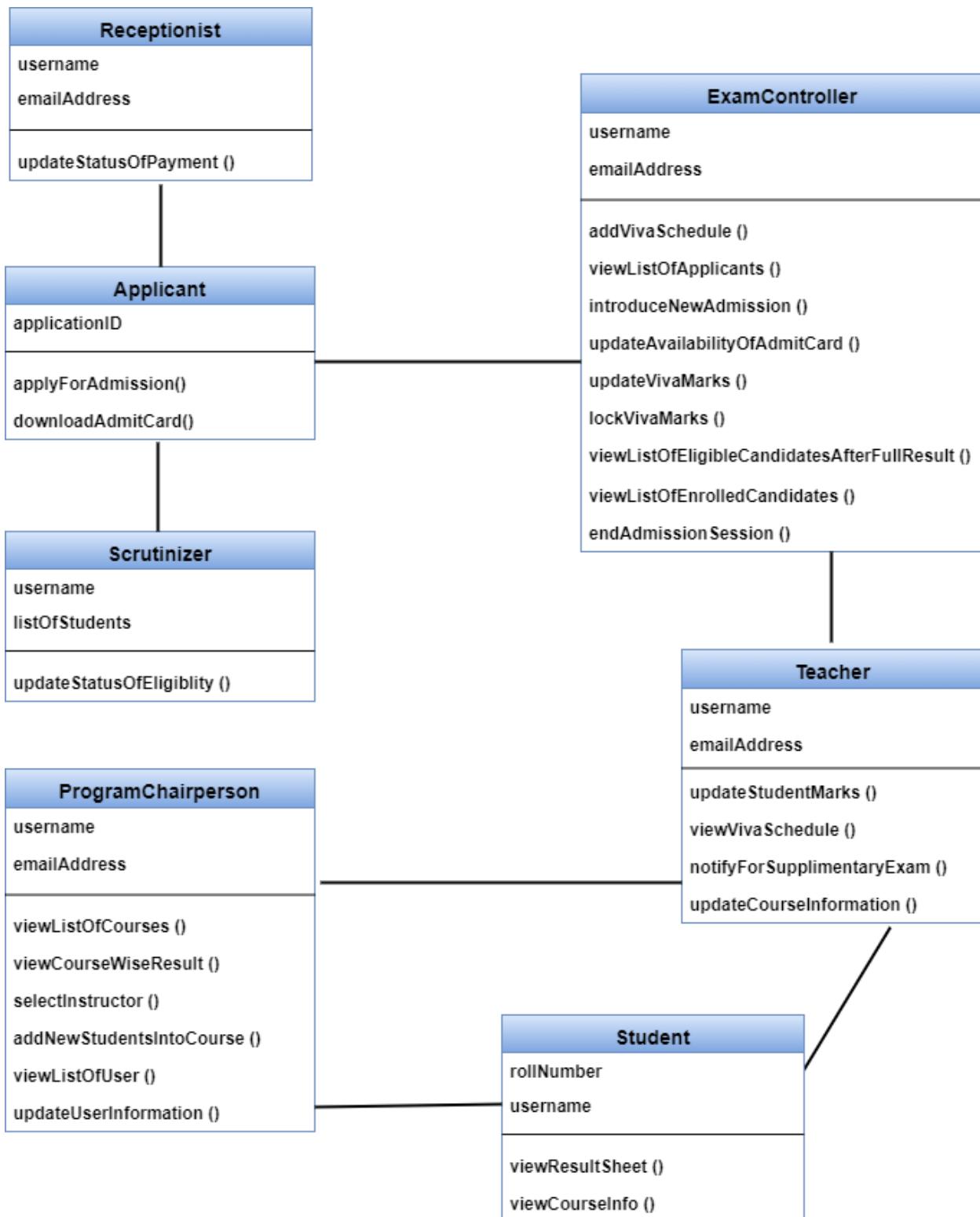


Figure 4 Archetypes

## 2.3 Refining the Architecture into Components

In this step software architecture is refined into components. These components can be derived from various sources

- ✚ Application Domain provides application components.
- ✚ Infrastructure Domain provides design components like design classes.
- ✚ The interfaces in the ACD imply one or more specialized components that process the data that flow across the interface.

PGDIT Automation System consists of the following top-level components:

- ✚ User Account Management: manages works related to user account.
- ✚ Admission: handles admission system related activities.
- ✚ GUI: manages graphical user interface.
- ✚ Course Result Management: handles activities related to course allocation and publishing result.

Figure 5 shows overall architectural structure for PGDIT Automation System with top-level components.

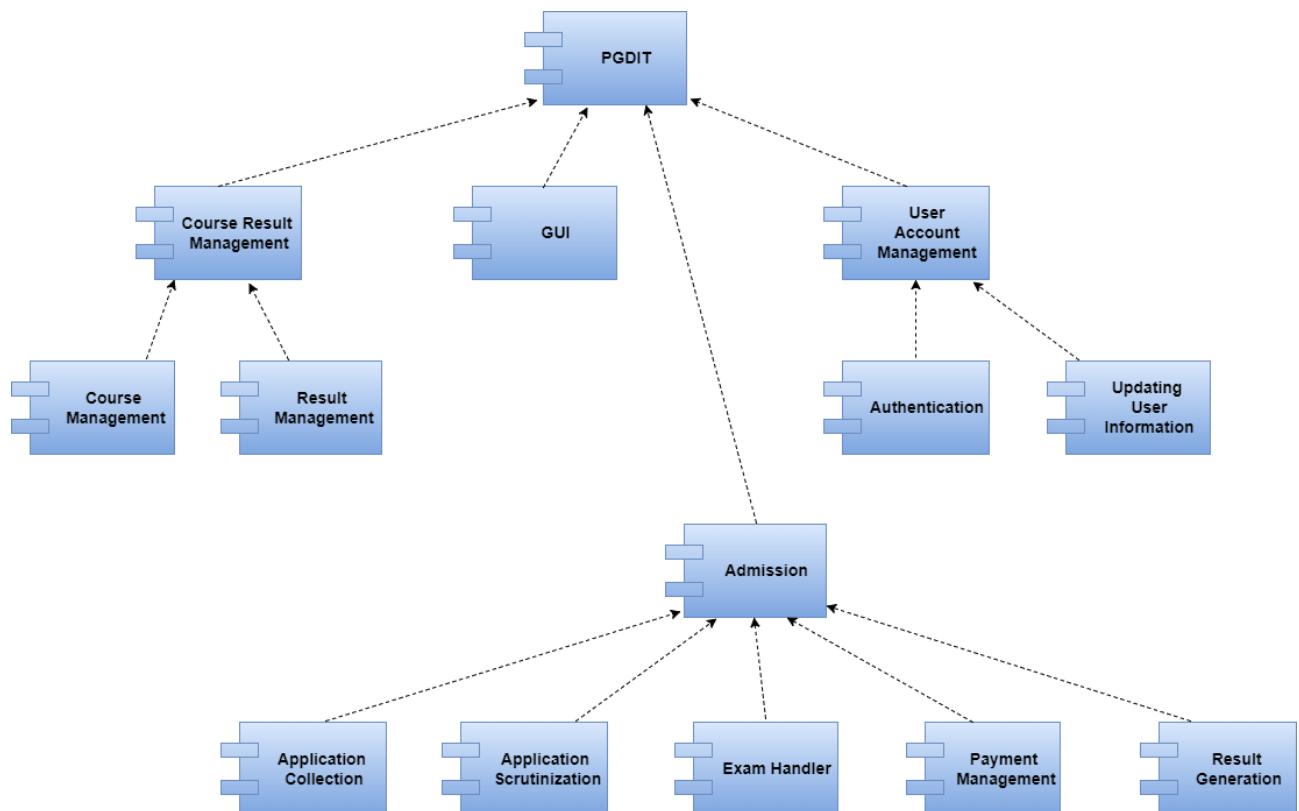


Figure 5 Overall Architectural Structure with Top-Level Components

## 2.4 Describing Instantiations of the System

Although the major software components have been identified, further refinement is still necessary. To accomplish this, an actual instantiation of the architecture is developed.

Figure 6 illustrates an instantiation of the PGDIT Automation System architecture. Components shown in Figure 5 are elaborated to show additional detail.

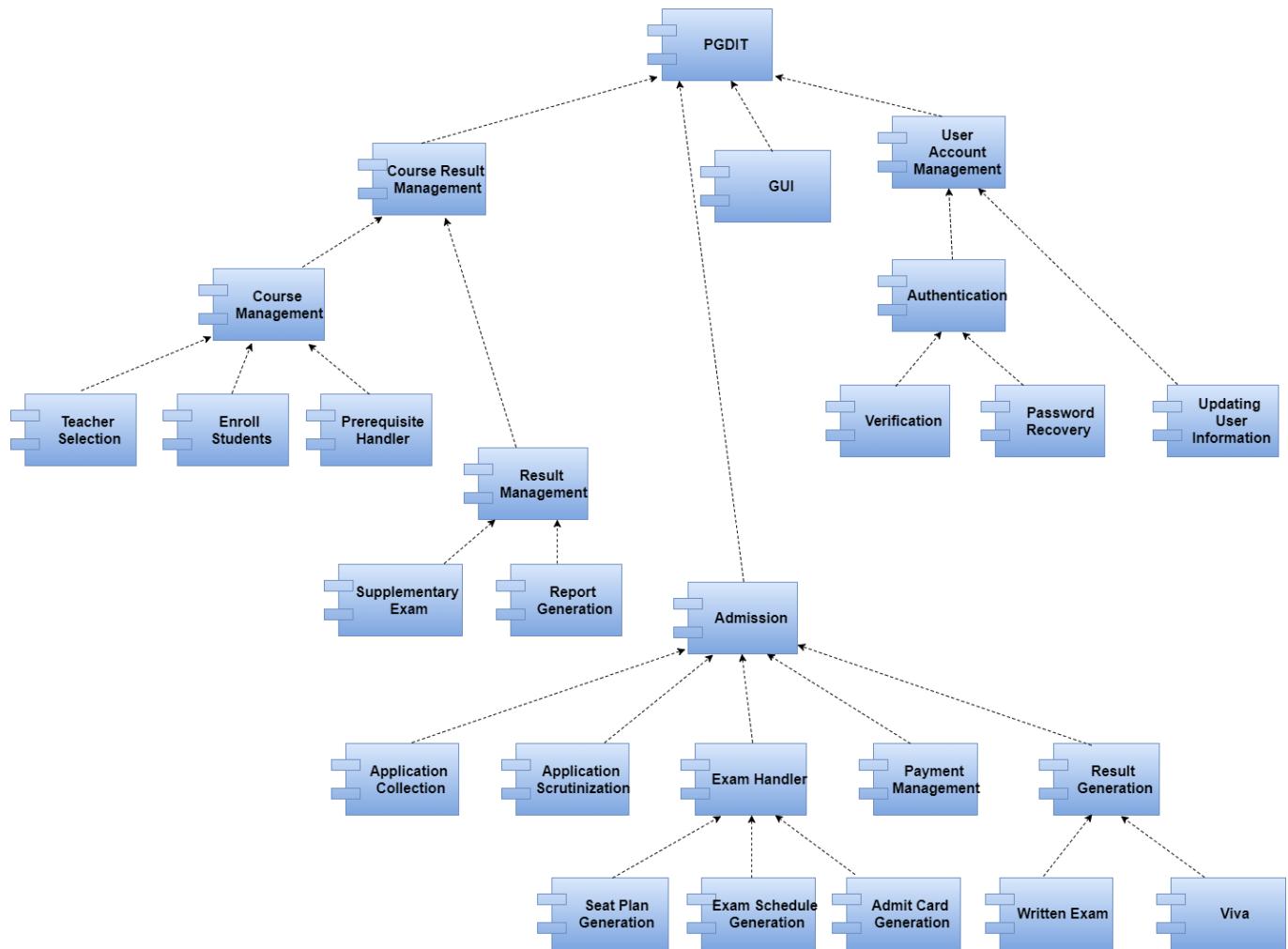


Figure 6 An Instantiation of the PGDIT Automation System Architecture

## 2.5 Architectural mapping using data flow

A mapping technique called structured design is often characterized as a data flow-oriented design method because it provides a convenient transition from a data flow diagram to software architecture. The transition from information flow to program structure is accomplished as part of a six step process:

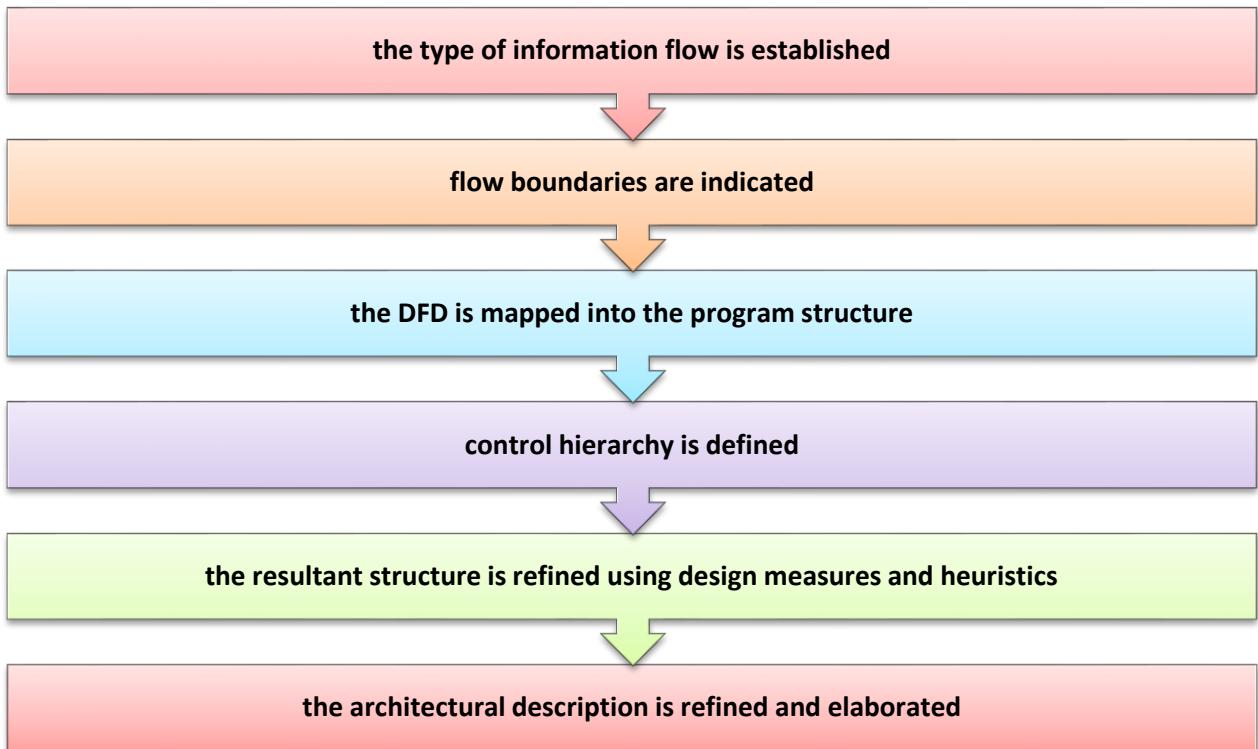


Figure 7 Architectural Mapping

The dataflow diagrams of PGDIT Automation System are shown in figure 8-\*.

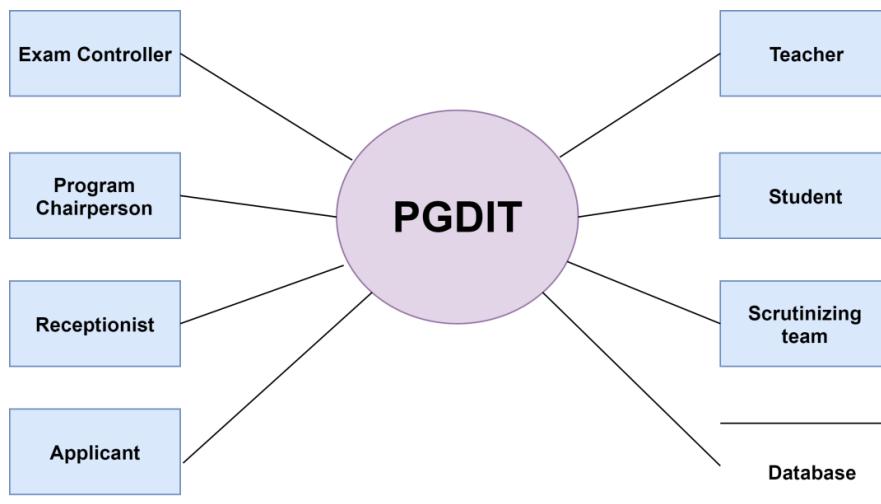


Figure 8 Level 0 Dataflow Diagram

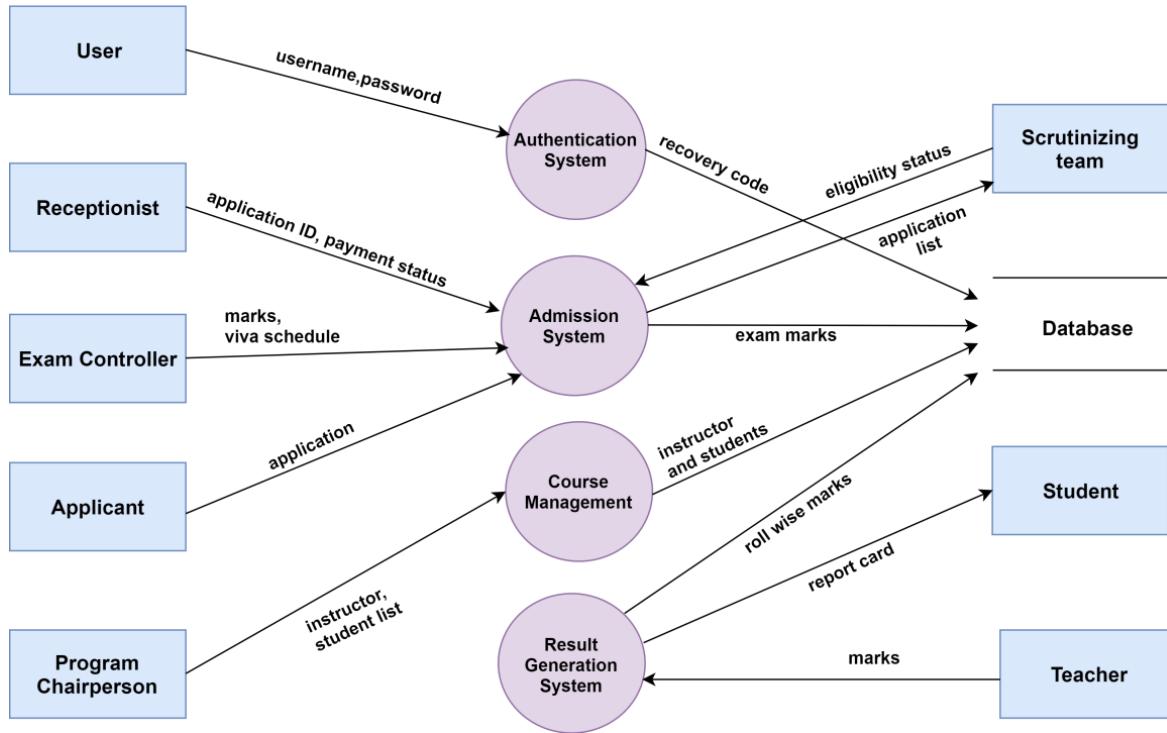


Figure 9 Level 1 Dataflow Diagram

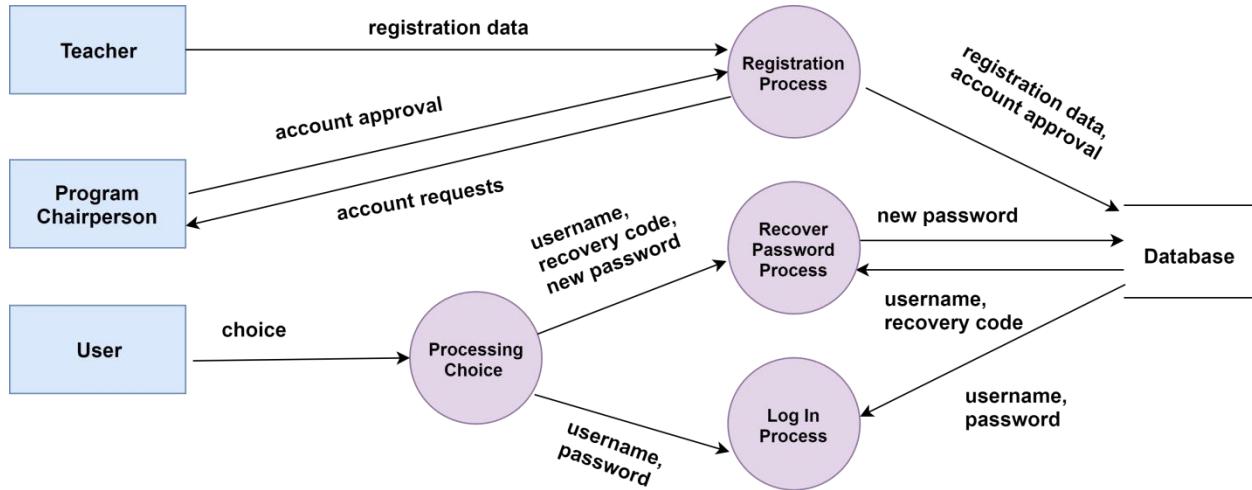


Figure 10 Dataflow Diagram of Authentication and Registration

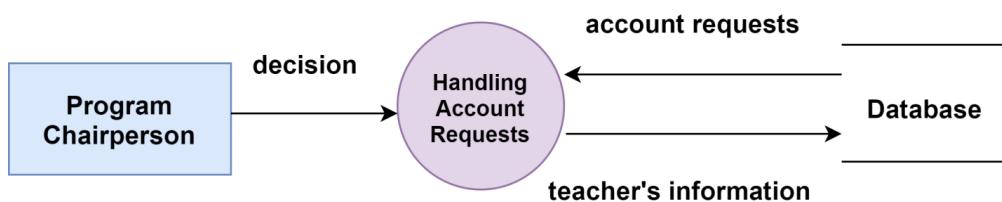


Figure 11 Program Chairperson's Dataflow Diagram of Registration

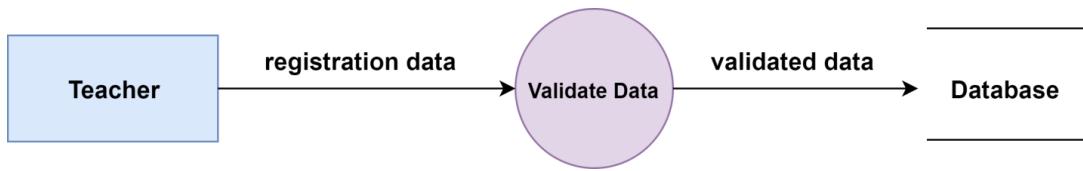


Figure 12 Teacher's Dataflow Diagram of Registration

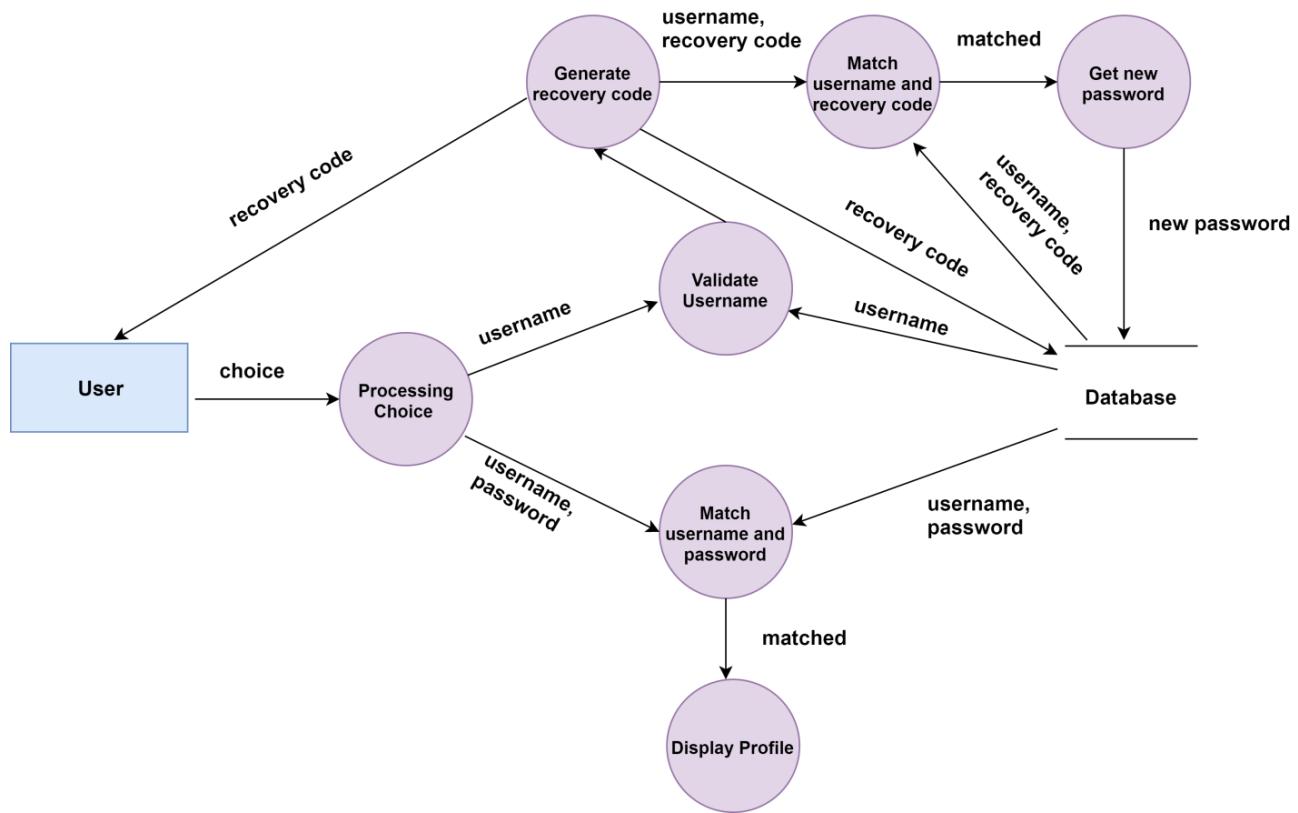


Figure 13 User's Dataflow Diagram of Authentication

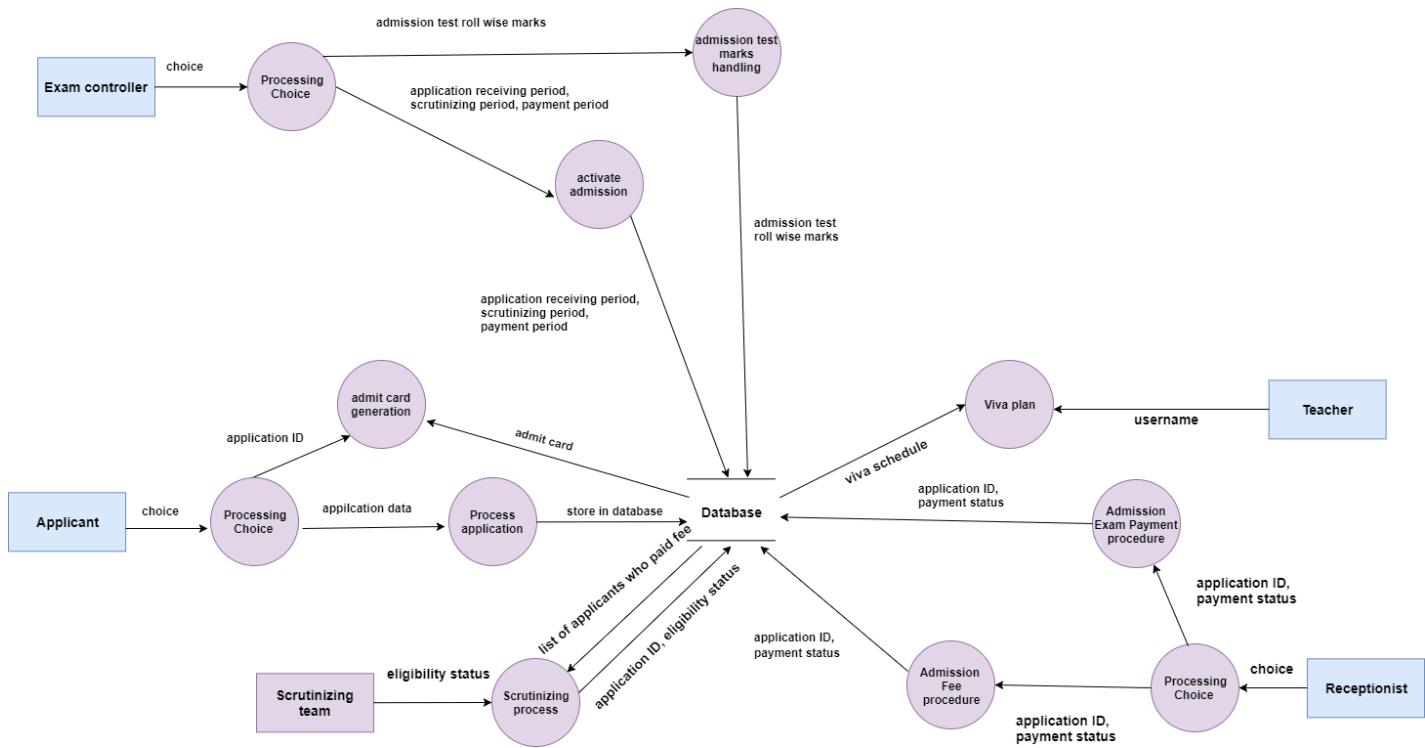


Figure 14 Dataflow Diagram of Admission System

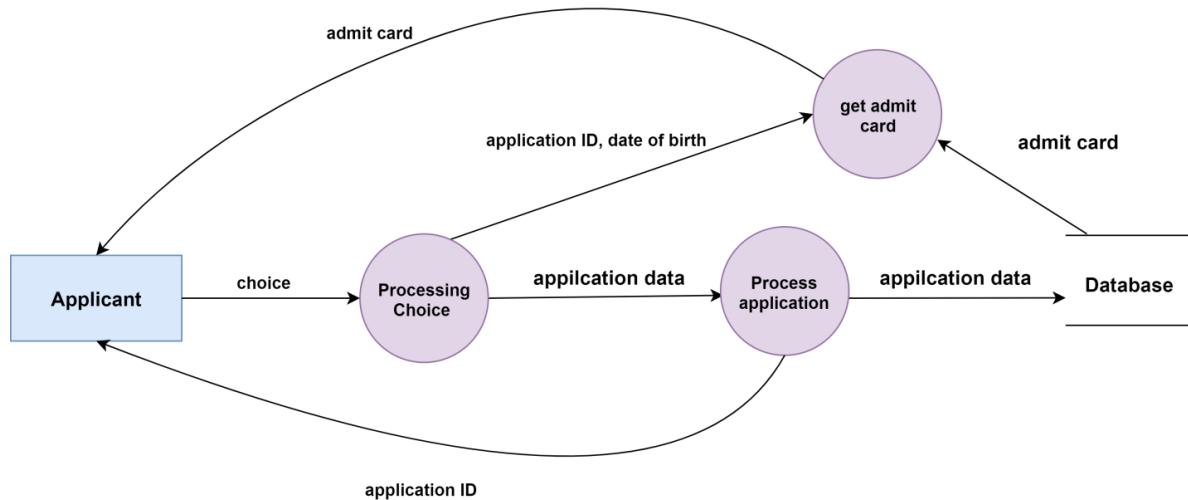


Figure 15 Applicant's Dataflow Diagram of Admission System

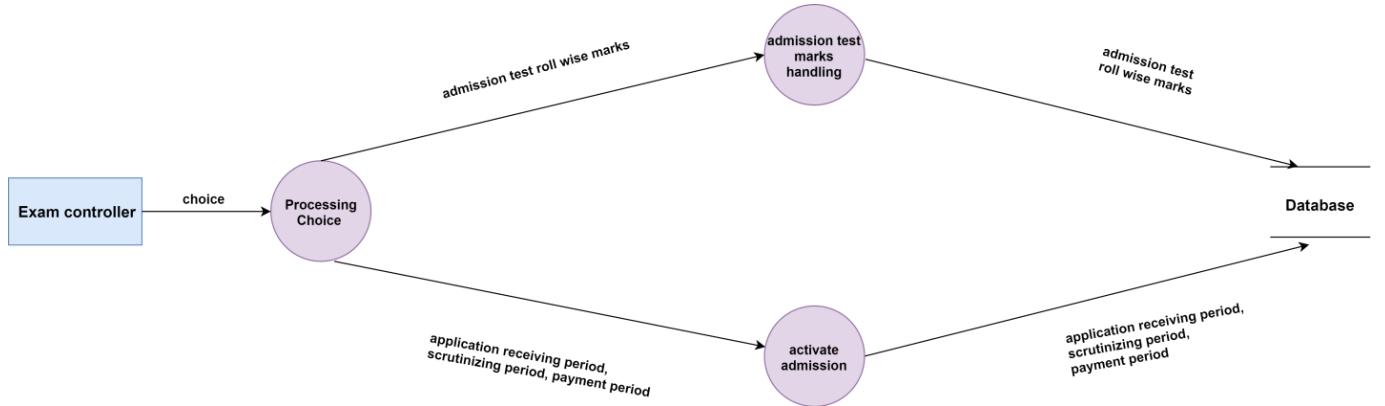


Figure 16 Exam Controller's Dataflow Diagram of Admission System

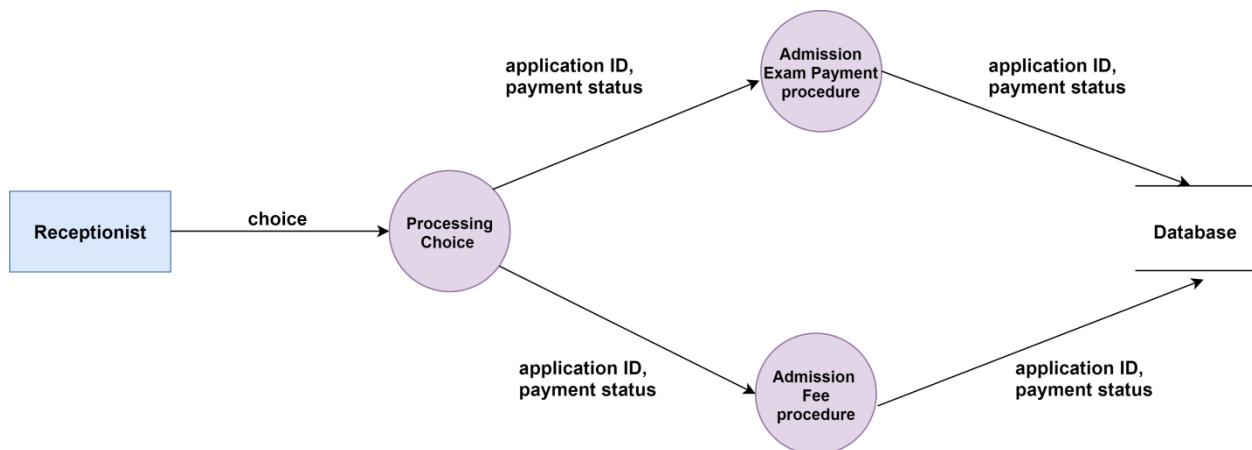


Figure 17 Receptionist's Dataflow Diagram of Admission System

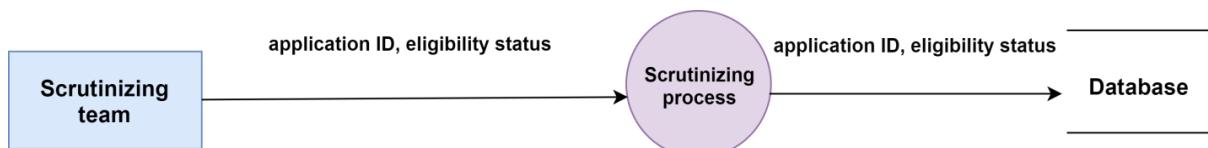


Figure 18 Scrutinizing team's Dataflow Diagram of Admission System

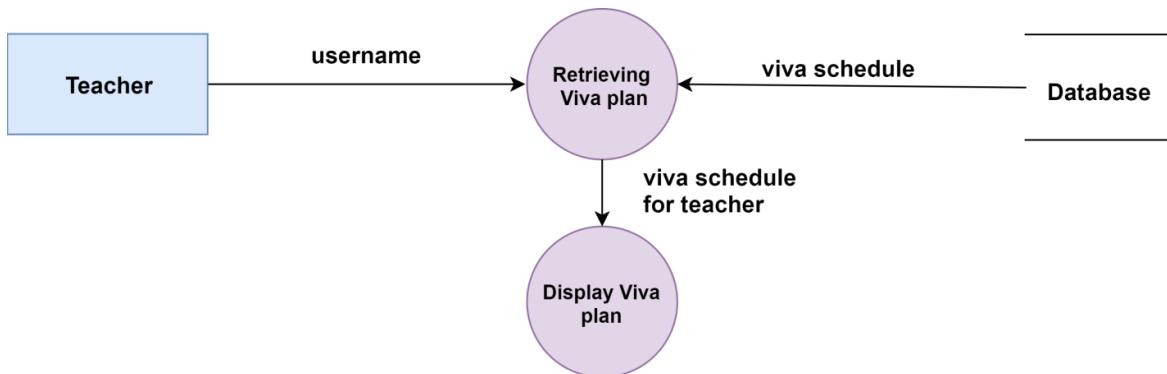


Figure 19 Teacher's Dataflow Diagram of Admission System

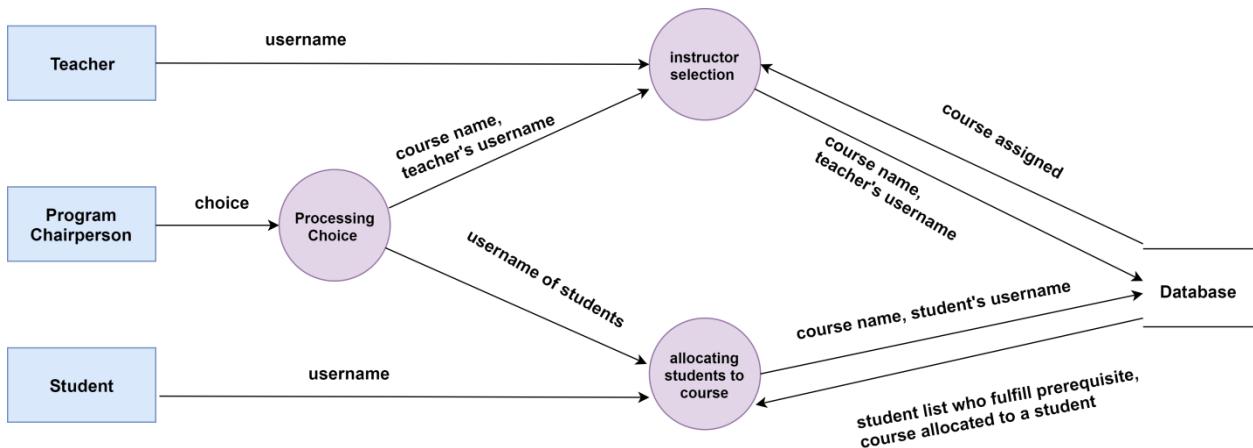


Figure 20 Dataflow Diagram of Course Management System

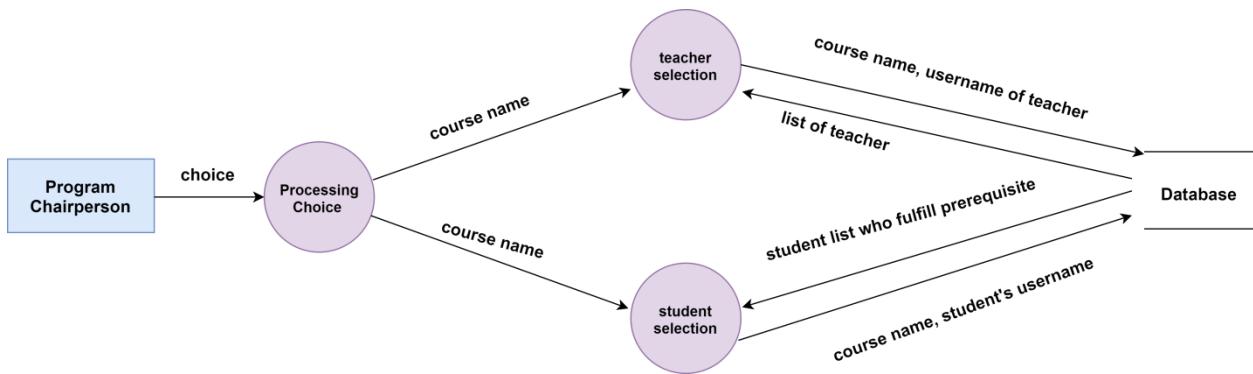


Figure 21 Program Chairperson's Dataflow Diagram of Course Management System

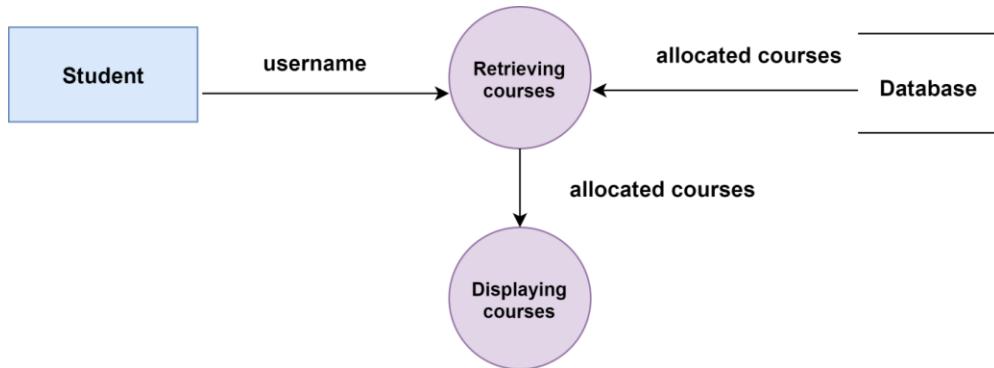


Figure 22 Student's Dataflow Diagram of Course Management System

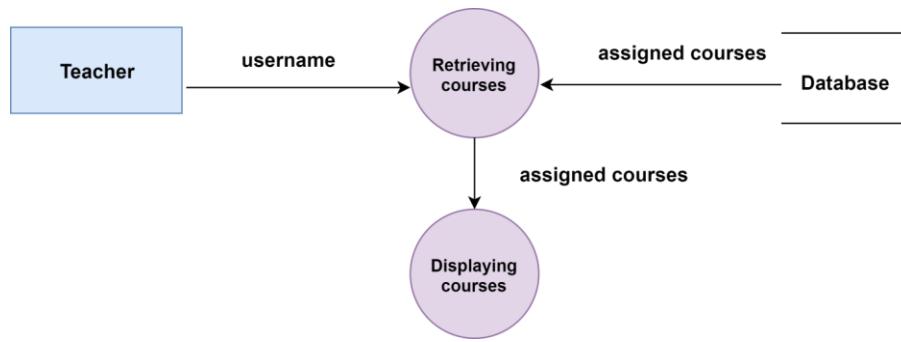


Figure 23 Teacher's Dataflow Diagram of Course Management System

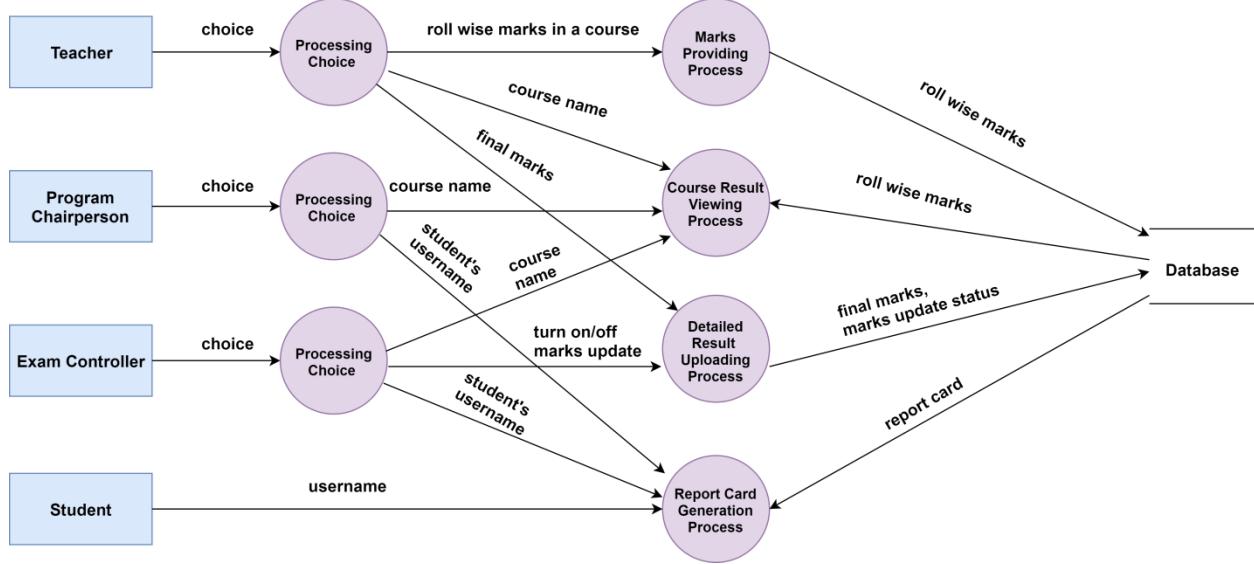


Figure 24 Dataflow Diagram of Result Generation System

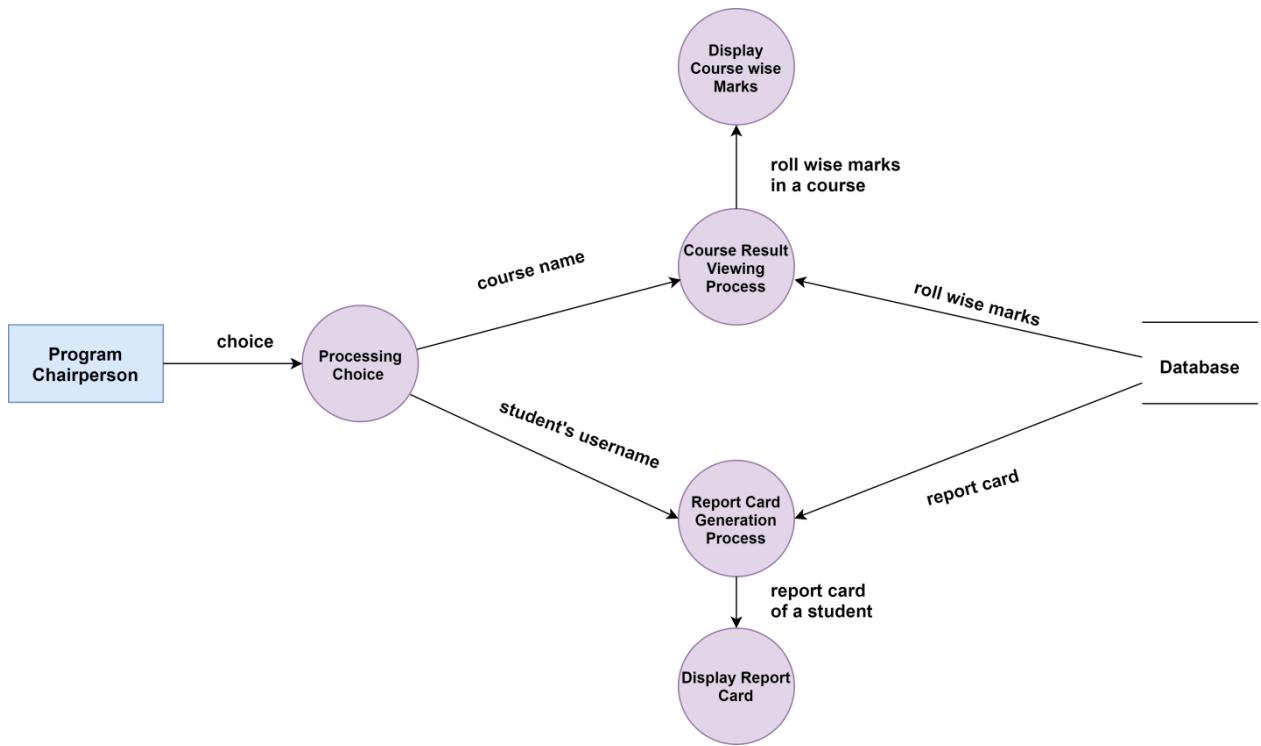


Figure 25 Program Chairperson's Dataflow Diagram of Result Generation System

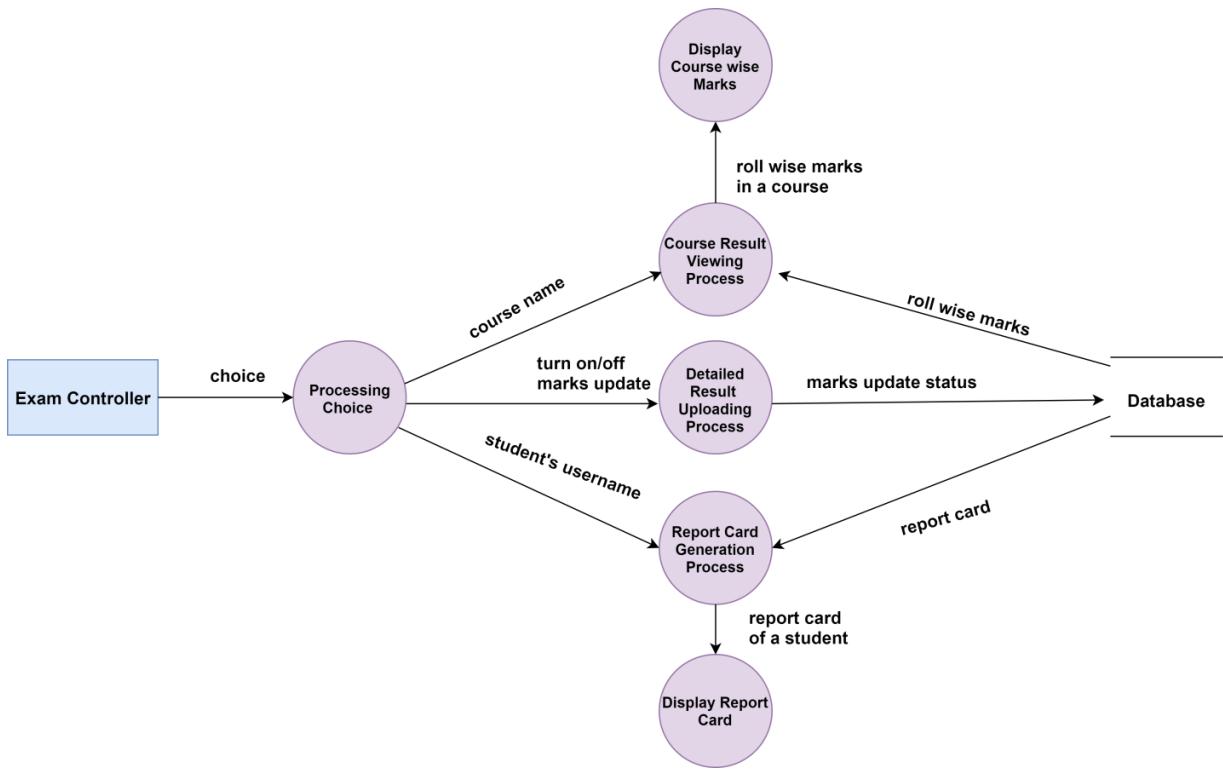


Figure 26 Exam Controller's Dataflow Diagram of Result Generation System

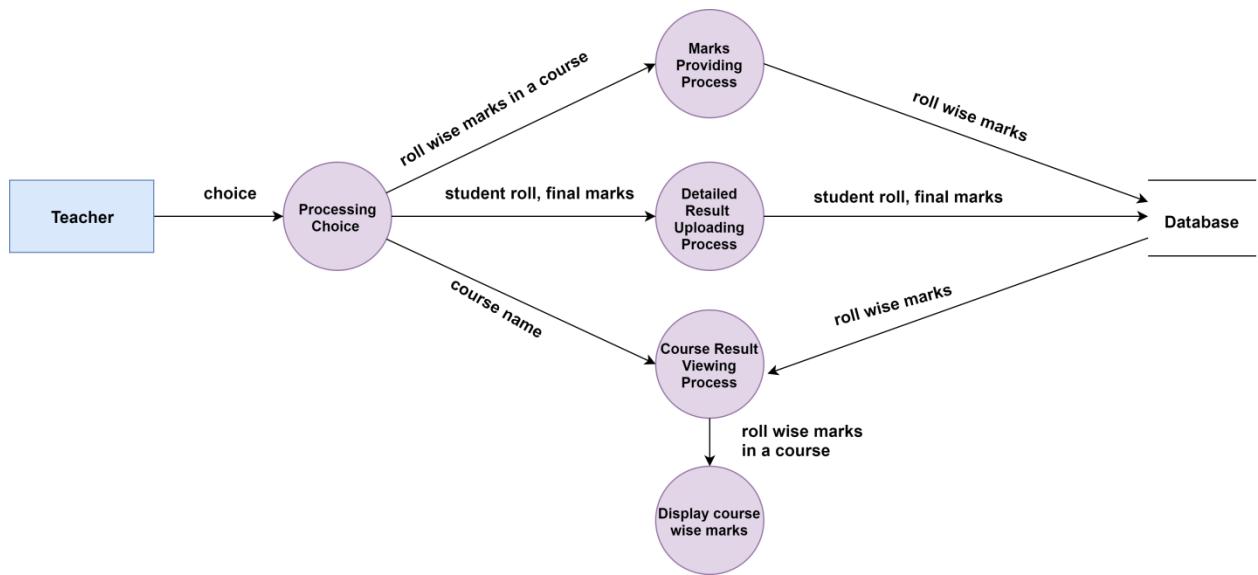


Figure 27 Teacher's Dataflow Diagram of Result Generation System

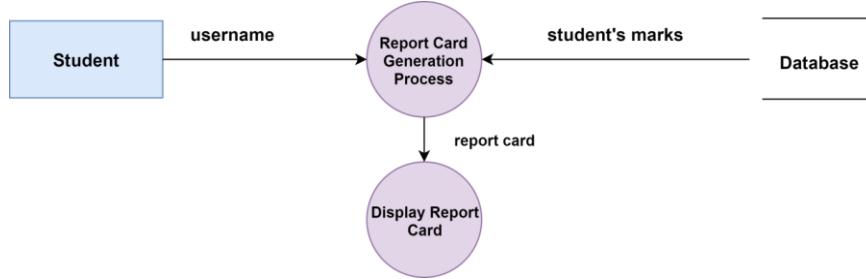


Figure 28 Student's Dataflow Diagram of Result Generation System

Figure 29 shows DFD mapping of Teacher's Dataflow Diagram of Registration.

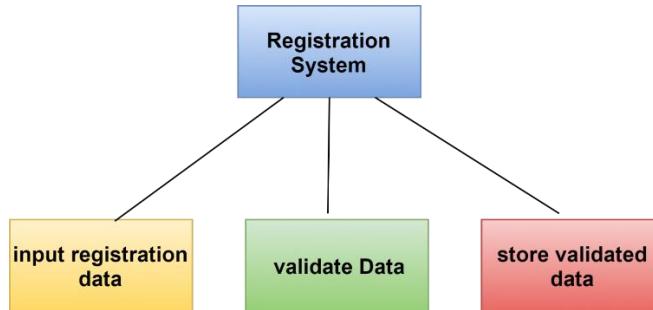


Figure 29 DFD mapping of Teacher's Dataflow Diagram of Registration

Figure 30 shows DFD mapping of Program chairperson's Dataflow Diagram of Registration.

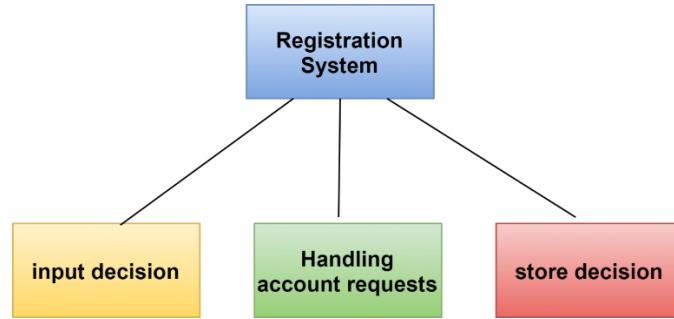


Figure 30 DFD mapping of Program chairperson's Dataflow Diagram of Registration

Figure 31 shows DFD mapping of User's Dataflow Diagram of Authentication.

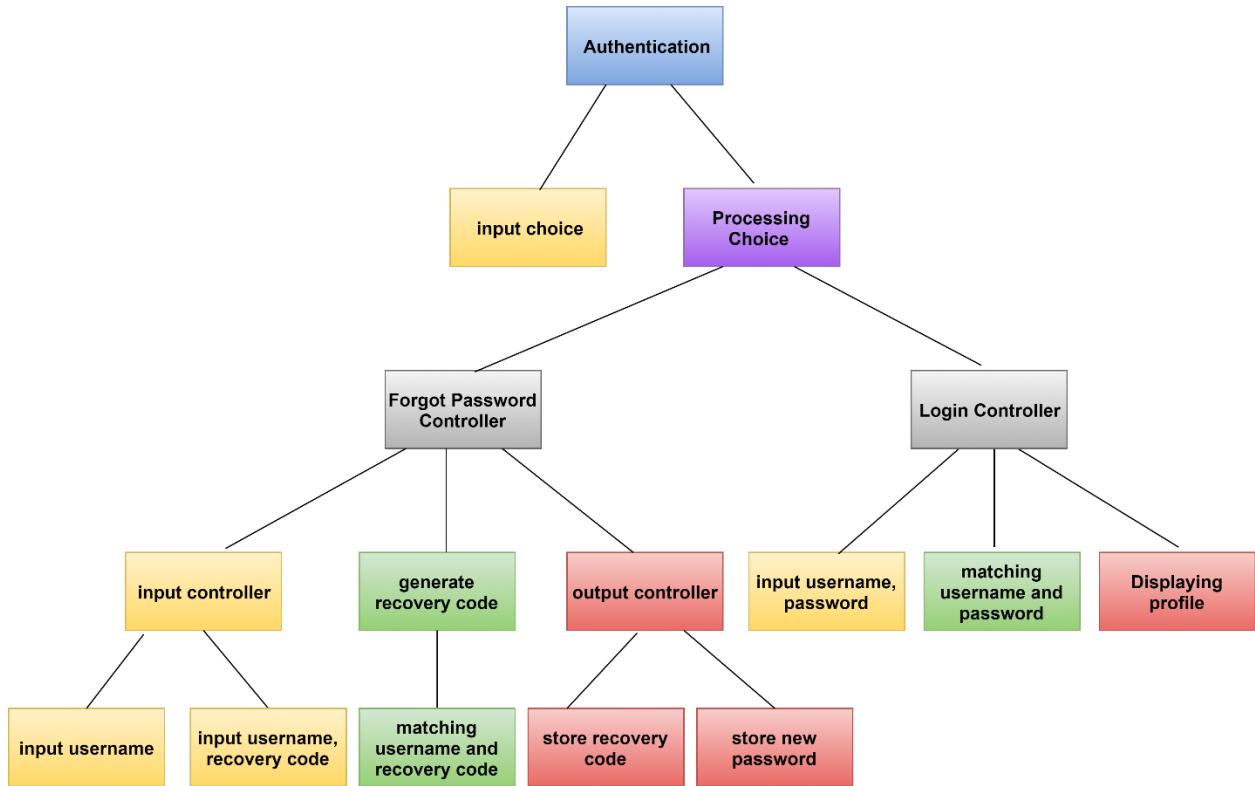


Figure 31 DFD mapping of User's Dataflow Diagram of Authentication

Figure 32 shows DFD mapping of Applicant's Dataflow Diagram of Admission System.

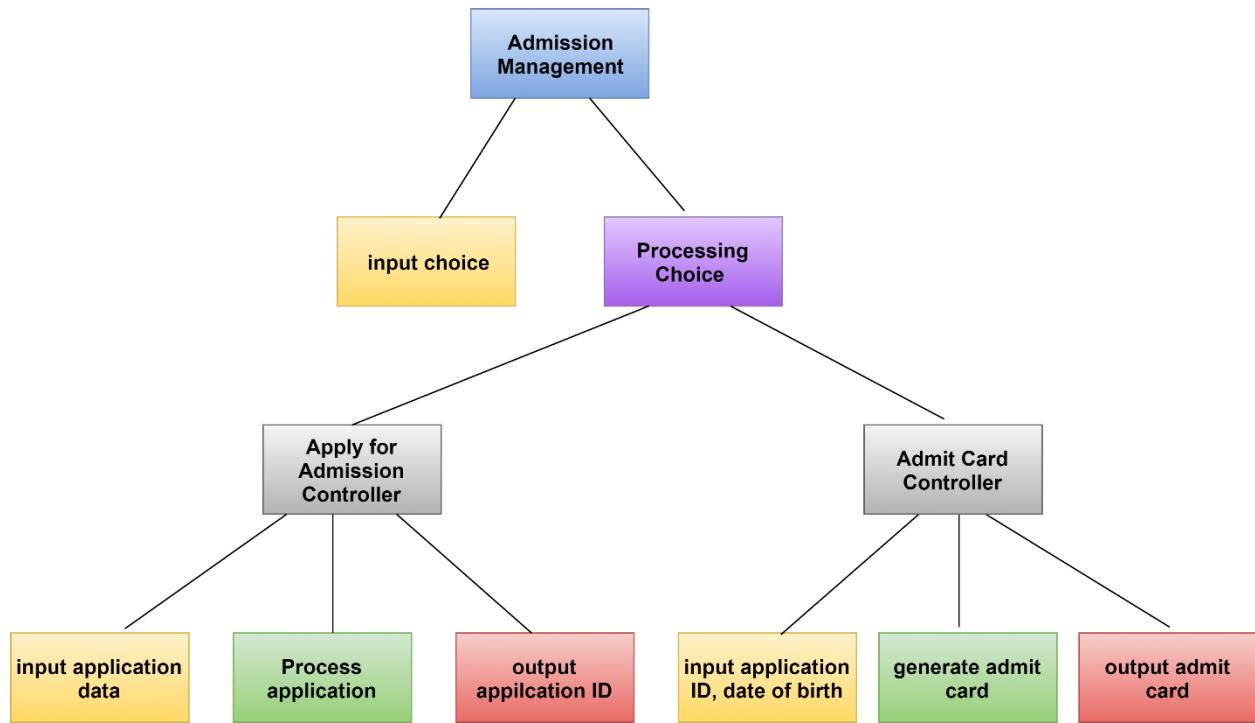


Figure 32 DFD mapping of Applicant's Dataflow Diagram of Admission System

Figure 33 shows DFD mapping of Exam Controller's Dataflow Diagram of Admission System.

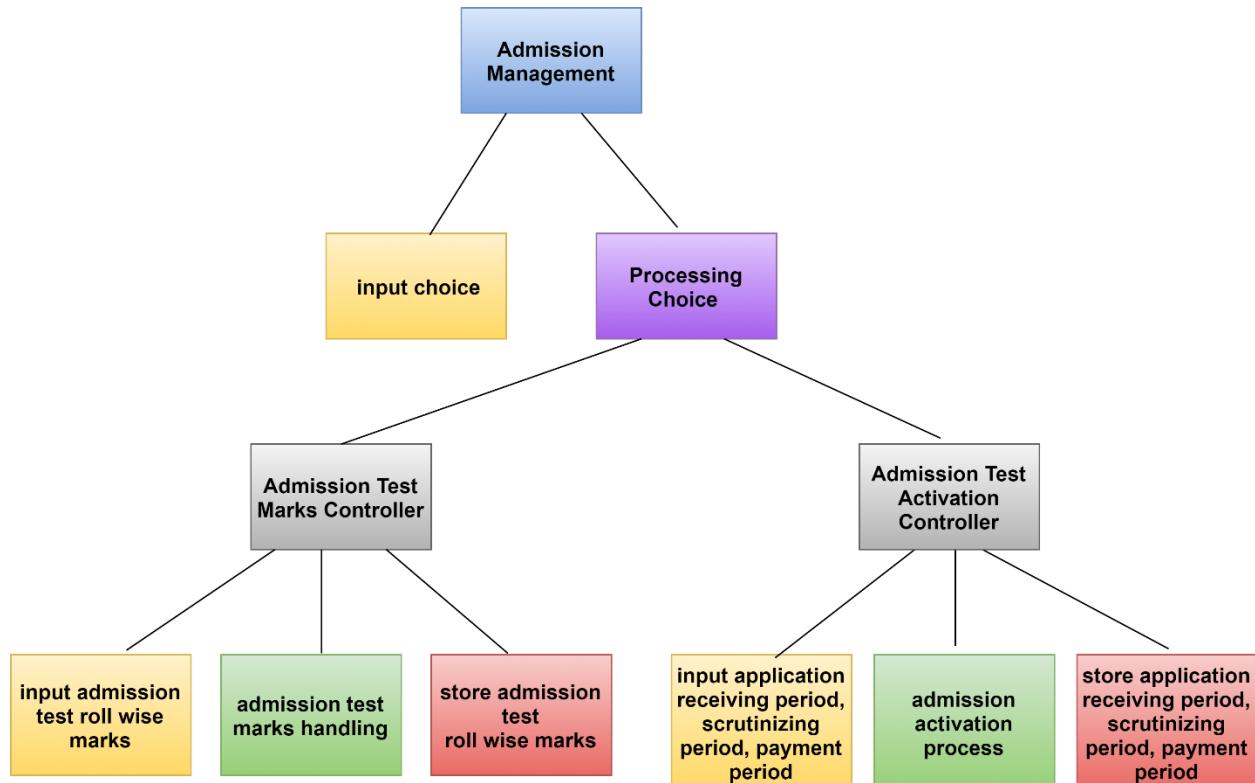


Figure 33 DFD mapping of Exam Controller's Dataflow Diagram of Admission System

Figure 34 shows DFD mapping of Receptionist's Dataflow Diagram of Admission System.

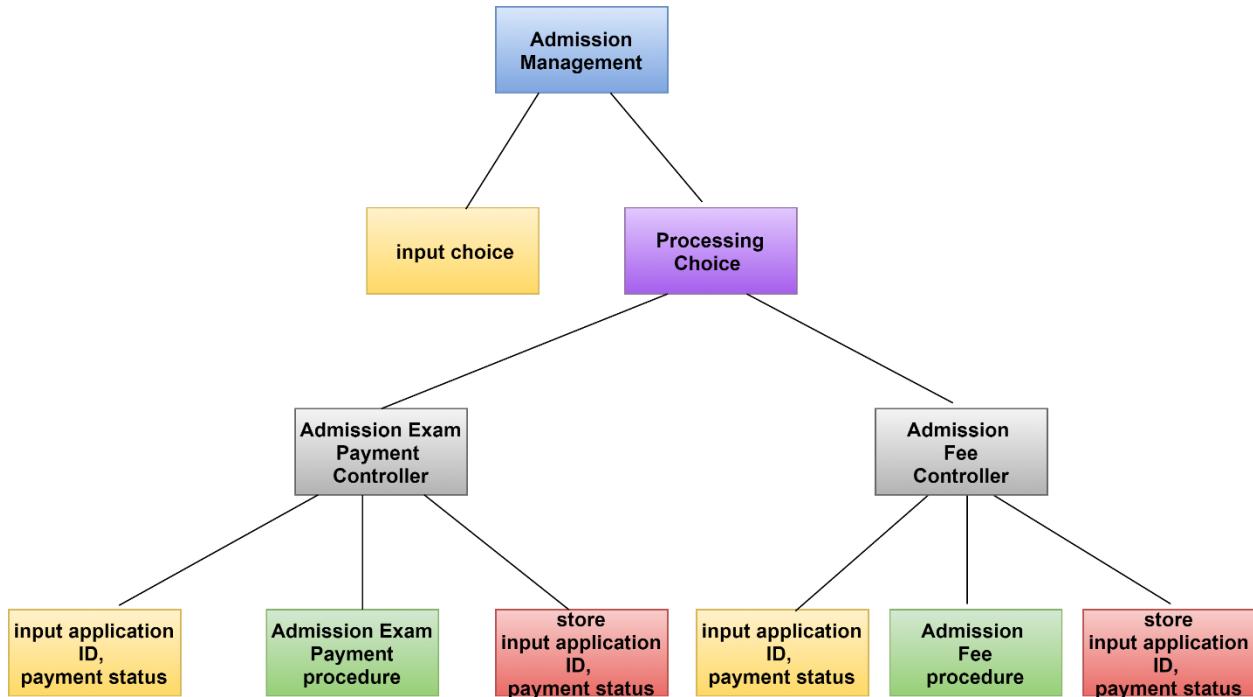


Figure 34 DFD mapping of Receptionist's Dataflow Diagram of Admission System

Figure 35 shows DFD mapping of Scrutinizer's Dataflow Diagram of Admission System.

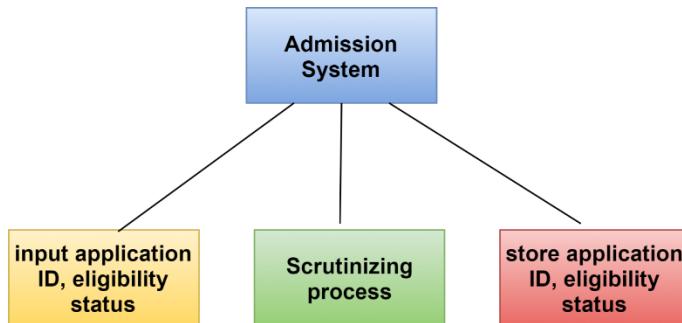


Figure 35 DFD mapping of Scrutinizer's Dataflow Diagram of Admission System

Figure 36 shows DFD mapping of Teacher's Dataflow Diagram of Admission System.

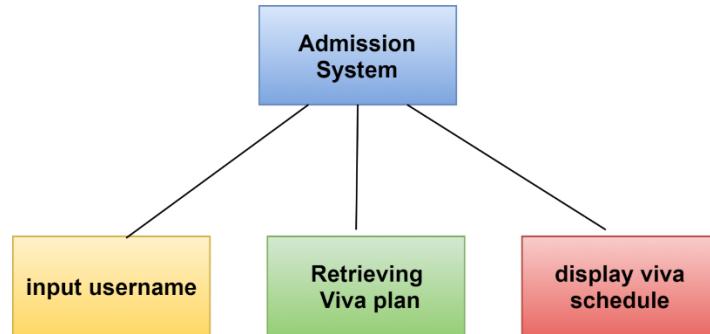


Figure 36 DFD mapping of Teacher's Dataflow Diagram of Admission System

Figure 37 shows DFD mapping of Program Chairperson's Dataflow Diagram of Course Management System.

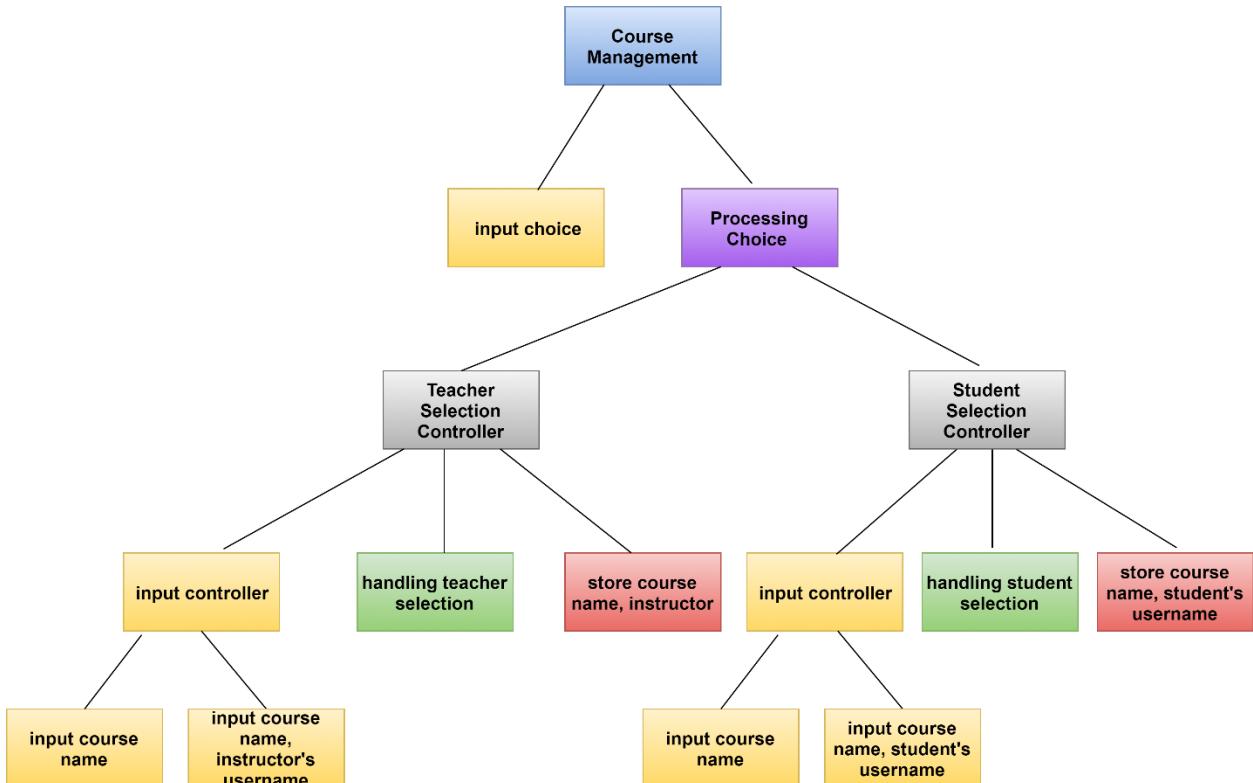


Figure 37 DFD mapping of Program Chairperson's Dataflow Diagram of Course Management System.

Figure 38 shows DFD mapping of Teacher's Dataflow Diagram of Course Management System.

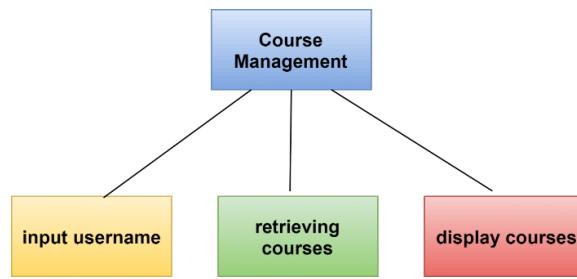


Figure 38 Teacher's Dataflow Diagram of Course Management System

Figure 39 shows DFD mapping of Student's Dataflow Diagram of Course Management System.

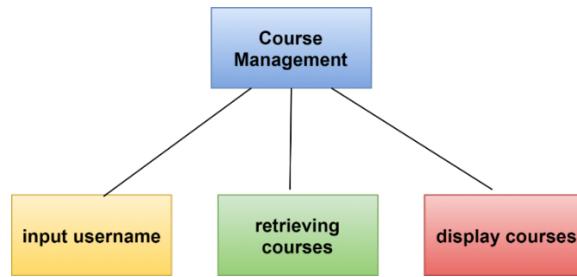


Figure 39 DFD mapping of Student's Dataflow Diagram of Course Management System

Figure 40 shows DFD mapping of Program Chairperson's Dataflow Diagram of Result Generation System.

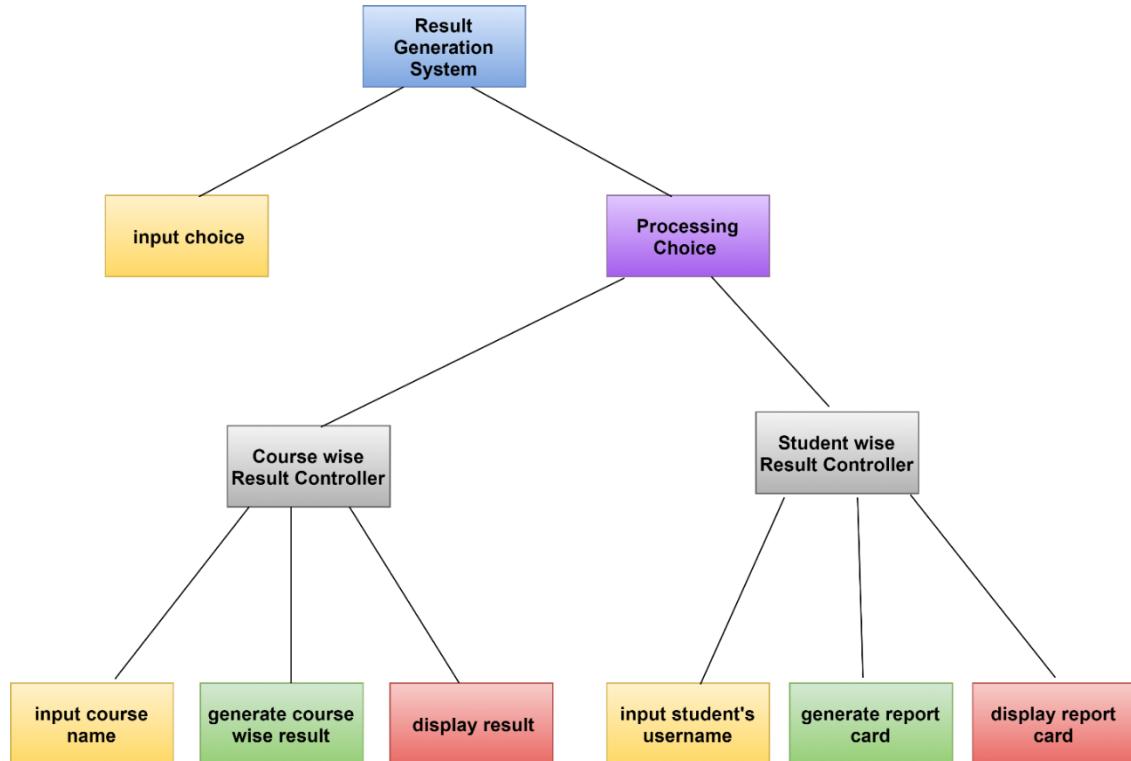


Figure 40 DFD mapping of Program Chairperson's Dataflow Diagram of Result Generation System

Figure 41 shows DFD mapping of Exam Controllers's Dataflow Diagram of Result Generation System.

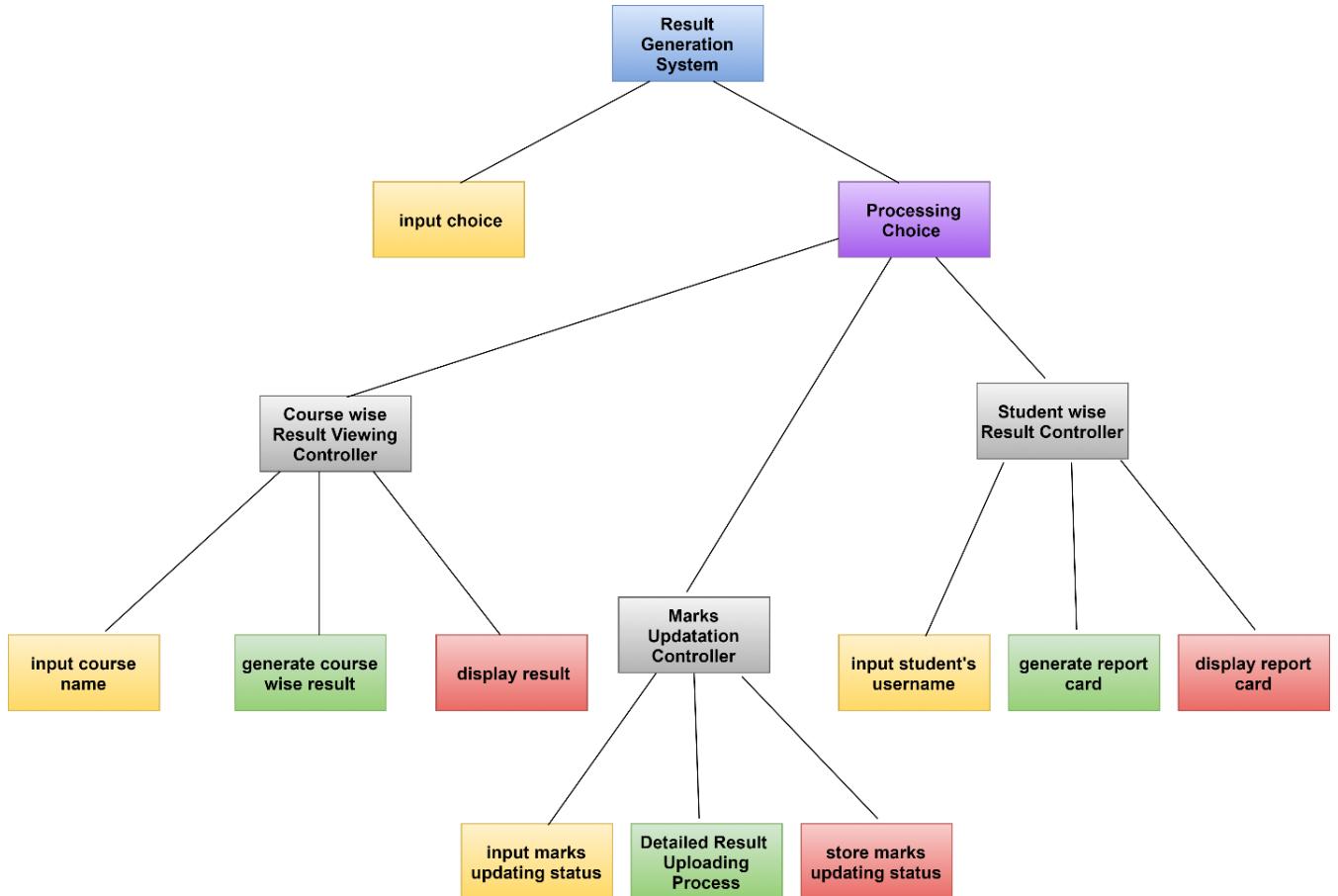


Figure 41 DFD mapping of Exam Controllers's Dataflow Diagram of Result Generation System

Figure 42 shows DFD mapping of Teacher's Dataflow Diagram of Result Generation System.

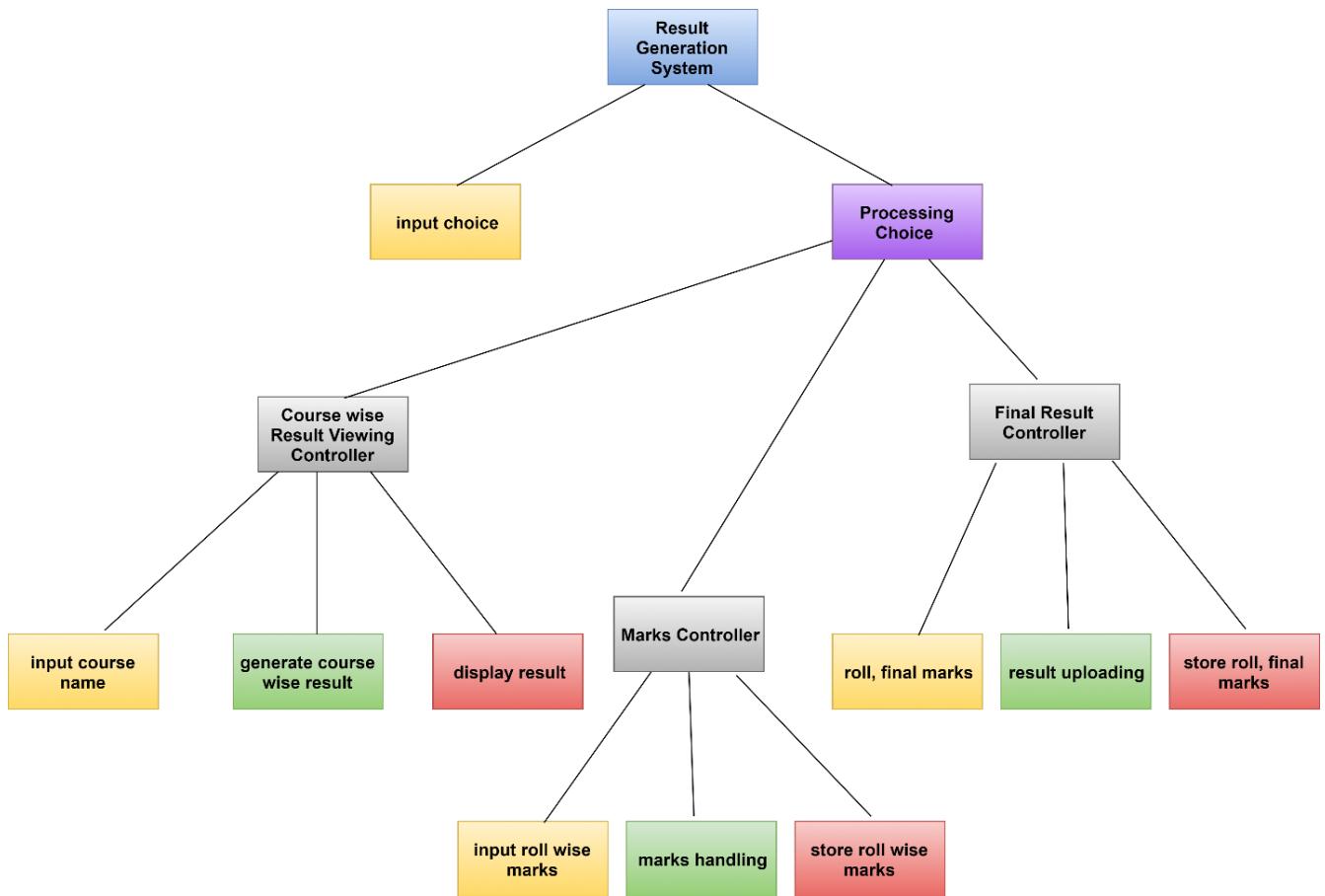


Figure 42 DFD mapping of Teacher's Dataflow Diagram of Result Generation System

Figure 43 shows DFD mapping of Student's Dataflow Diagram of Result Generation System.

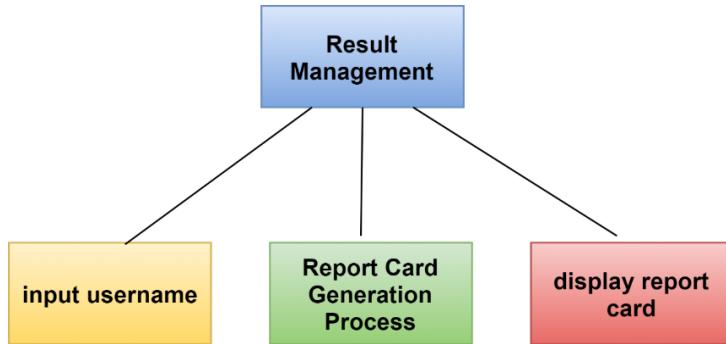


Figure 43 DFD mapping of Student's Dataflow Diagram of Result Generation System

## Chapter 3. Component-Level Design of PGDIT Automation System

A component is a modular building block for computer software. According to OMG Unified Modeling Language Specification [OMG03a] component is a modular, deployable, and replaceable part of a system that encapsulates implementation and exposes a set of interfaces. Component-level design defines the data structures, algorithms, interface characteristics, and communication mechanisms allocated to each software component. Component-level design helps to determine whether the software will work before building it. The component-level design represents the software in a way that allows to review the details of the design for correctness and consistency with other design representations. It provides a means for assessing whether data structures, interfaces, and algorithms will work.

Component-level design comprises of the following steps:

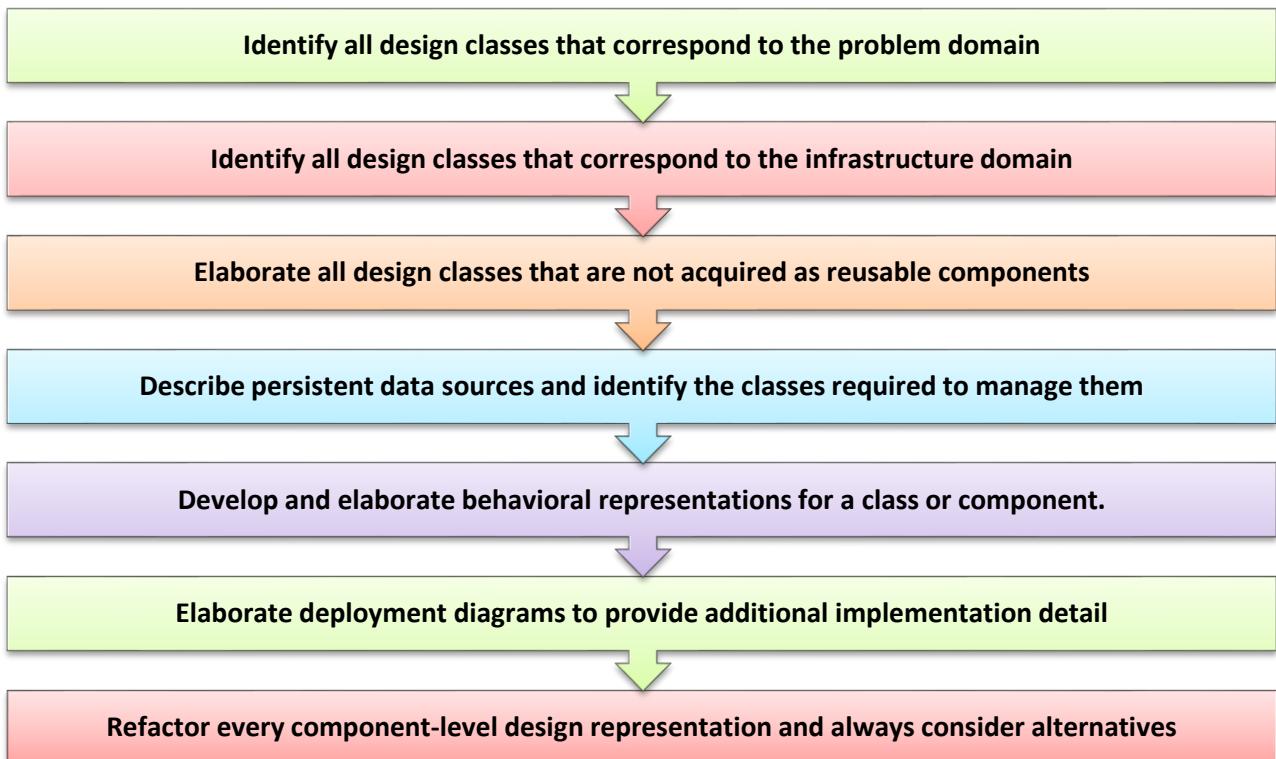


Figure 44 Steps of Component-Level Design

### 3.1 Identify All Design Classes that Correspond to the Problem Domain

PGDIT Automation System has the following analysis class.

- ⊕ Teacher
- ⊕ Student

-  ProgramChairperson
-  ExamController
-  Scrutinizer
-  Applicant
-  Receptionist

All analysis class and their attributes and methods are shown in figure 45.

<b>ProgramChairperson</b>	<b>ExamController</b>	<b>Teacher</b>
viewListOfCourses () viewCourseWiseResult () selectInstructor () addNewStudentsIntoCourse () viewListOfUser () viewInfoOfOneUser () updateUserInformation ()	viewListOfRoom () updateRoomCapacity () addNewRoom () viewVivaSchedule () addVivaSchedule () viewListOfApplicants () introduceNewAdmission () viewAdmissionPanel () viewListOfEligibleCandidatesOnScrutinize () viewListOfEligibleCandidatesAfterScrutinize () updateAvailabilityOfAdmitCard () viewListOfEligibleCandidatesAfterWrittenExam () updateWrittenMarks () viewListOfEligibleCandidatesAfterVIVAExam () updateVivaMarks () lockVivaMarks () viewListOfEligibleCandidatesAfterFullResult () viewListOfEnrolledCandidates () updateEnrollmentState () lockEnrollmentState () viewListOfEnrolledApplicants () endAdmissionSession ()	viewListOfCourses () viewResultSheet () updateStudentMarks () viewSupplementaryList () viewVivaSchedule () viewCourseInformation () notifyForSupplementaryExam () updateCourseInformation () updatePdf ()
<b>Student</b>		<b>User</b>
rollNumber  viewResultSheet () viewRunningCourses () viewCourseInfo ()		username fullName  viewHeader() updatePassword() updateEmail() updateMobileNo() updateAddress() updatePicture() verifyPassword()
<b>Scrutinizer</b>		<b>Applicant</b>
listOfApplicants  updateStatusOfEligibility () searchByApplicationID () searchByIndex ()		applicationID  applyForAdmission() downloadAdmitCard()
<b>Receptionist</b>		
viewPaymentBoard () updateStatusOfPayment ()		

Figure 45 Analysis classes

### 3.2 Identify All Design Classes that Correspond to the Infrastructure Domain

There are some classes that are not described in the requirements model and are often missing from the architecture model. These classes have been described here. We have found 2 infrastructure domain classes.

- DatabaseHandler: for handling database operations( insert, update, delete, retrieve)
- QueryGenerator: for generating sql query using javascript object.

Figure 46 shows infrastructure domain classes, their attributes and methods.

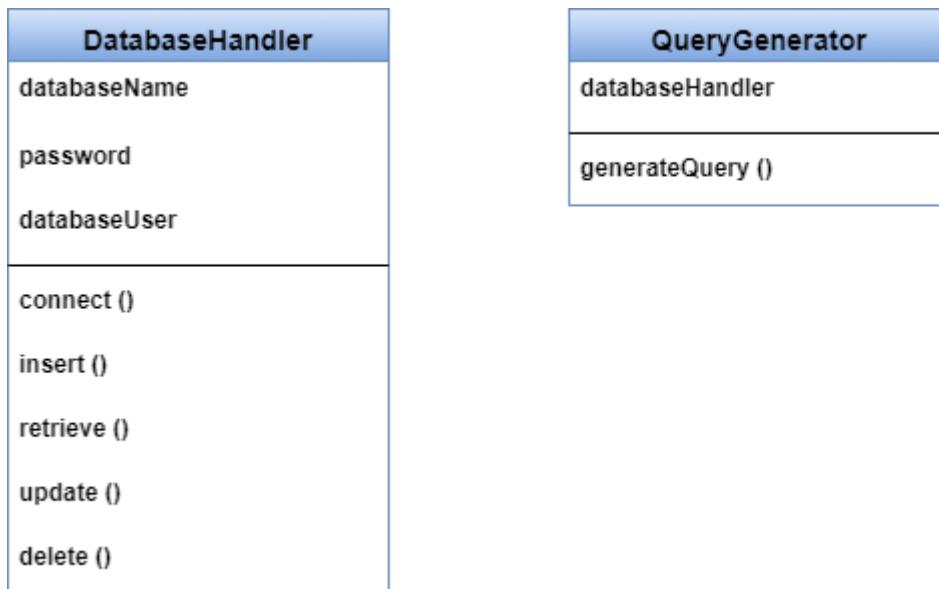


Figure 46 Design classes

### 3.3 Elaborate All Design Classes that are not Acquired as Reusable Components

Elaboration requires that all interfaces, attributes, and operations necessary to implement the class be described in detail.

Figure 47-54 show elaborated design classes. The attributes and operations defined during requirement analysis are noted at the top of the figures. The interfaces of these design classes are represented using the “lollipop” symbols shown to the left of the component boxes.

Figure 47 shows elaborated Applicant class. It has two interfaces.

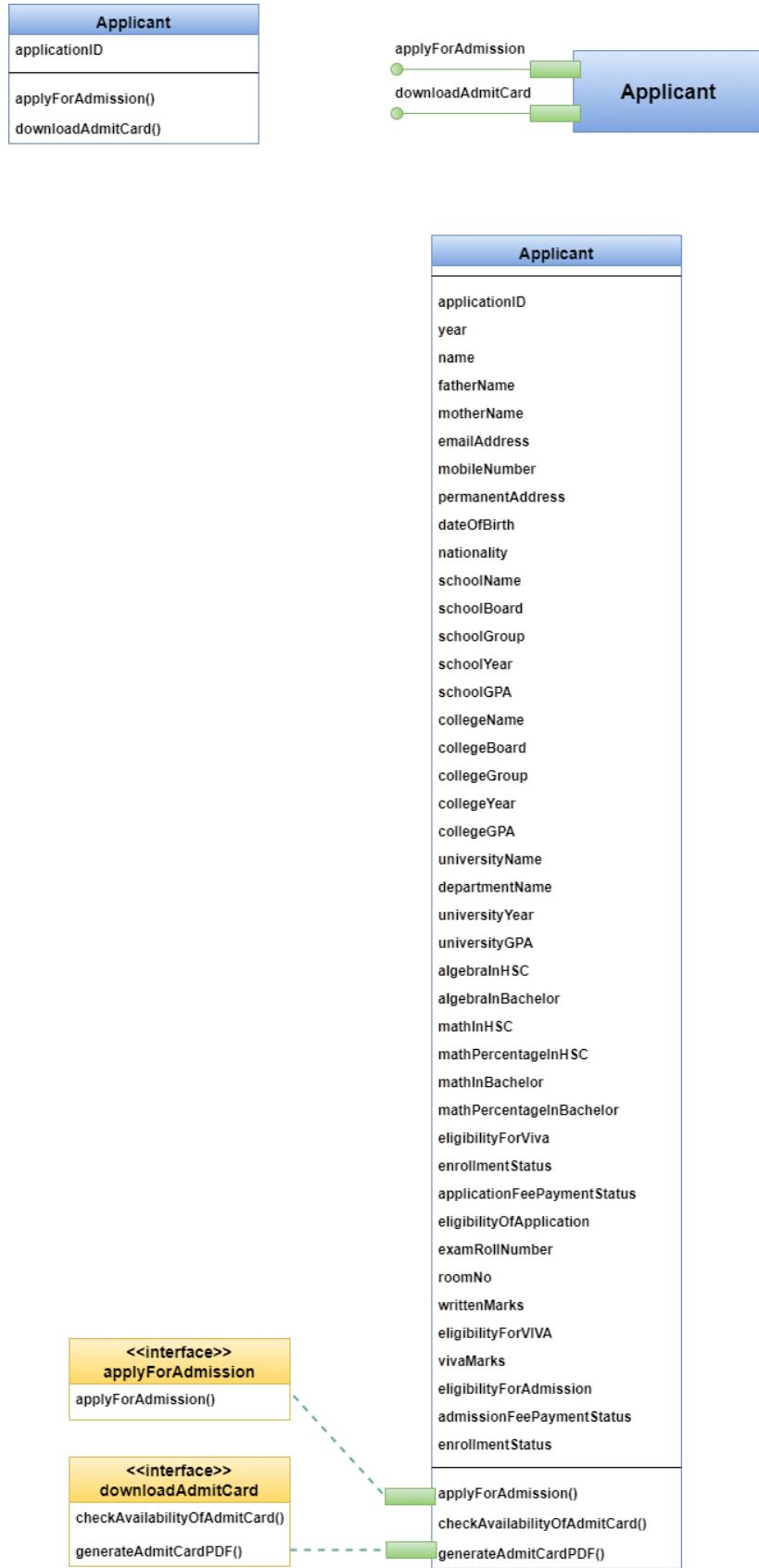


Figure 47 Elaborated Applicant class

Figure 48 shows elaborated User class. It has ten interfaces:

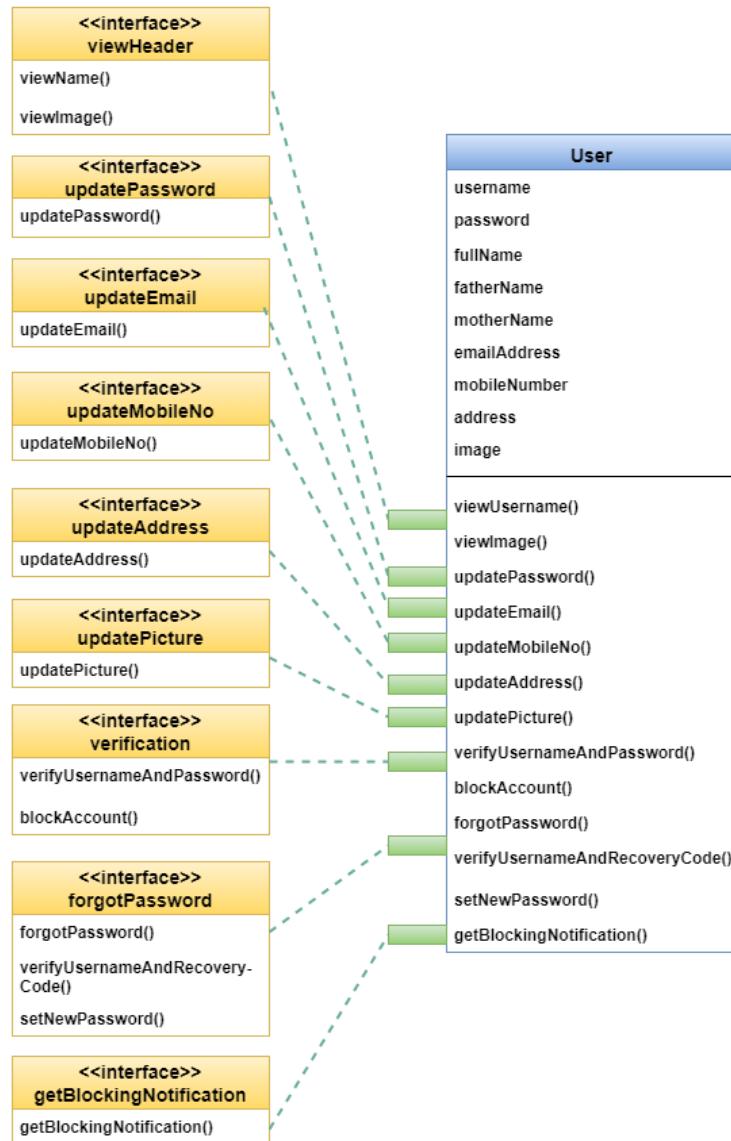
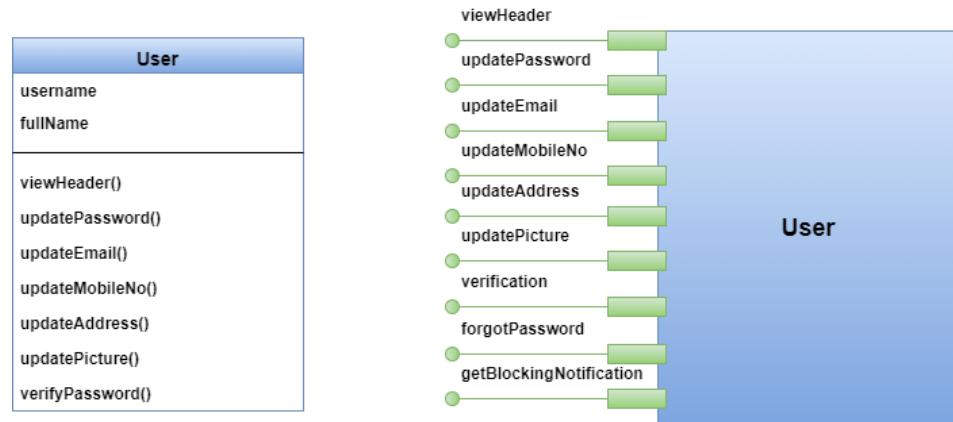


Figure 48 Elaborated User class

Figure 49 shows elaborated ProgramChairperson class. It has eleven interfaces.

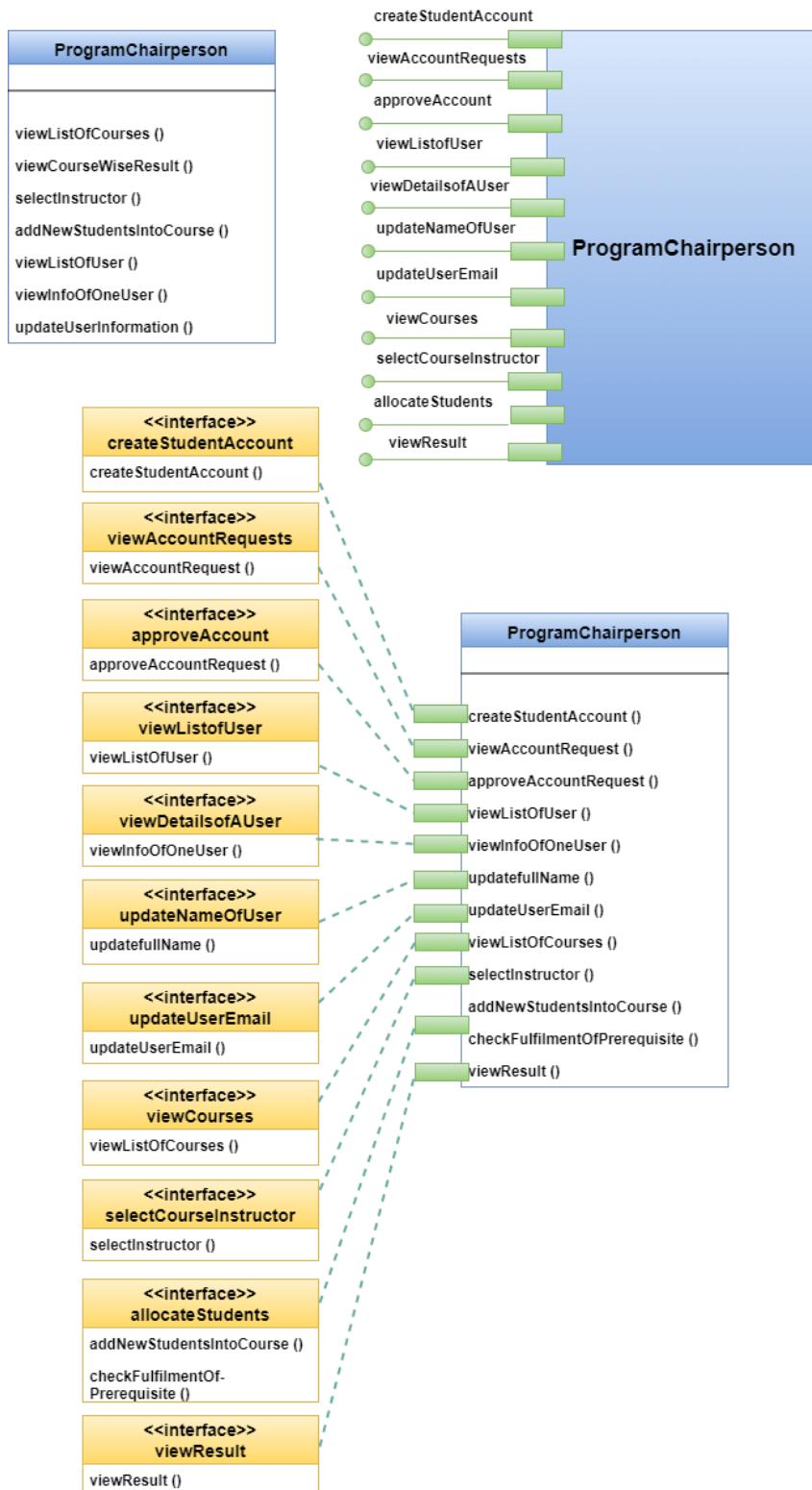


Figure 49 Elaborated ProgramChairperson class

Figure 50 shows elaborated Scrutinizer class. It has four interfaces.

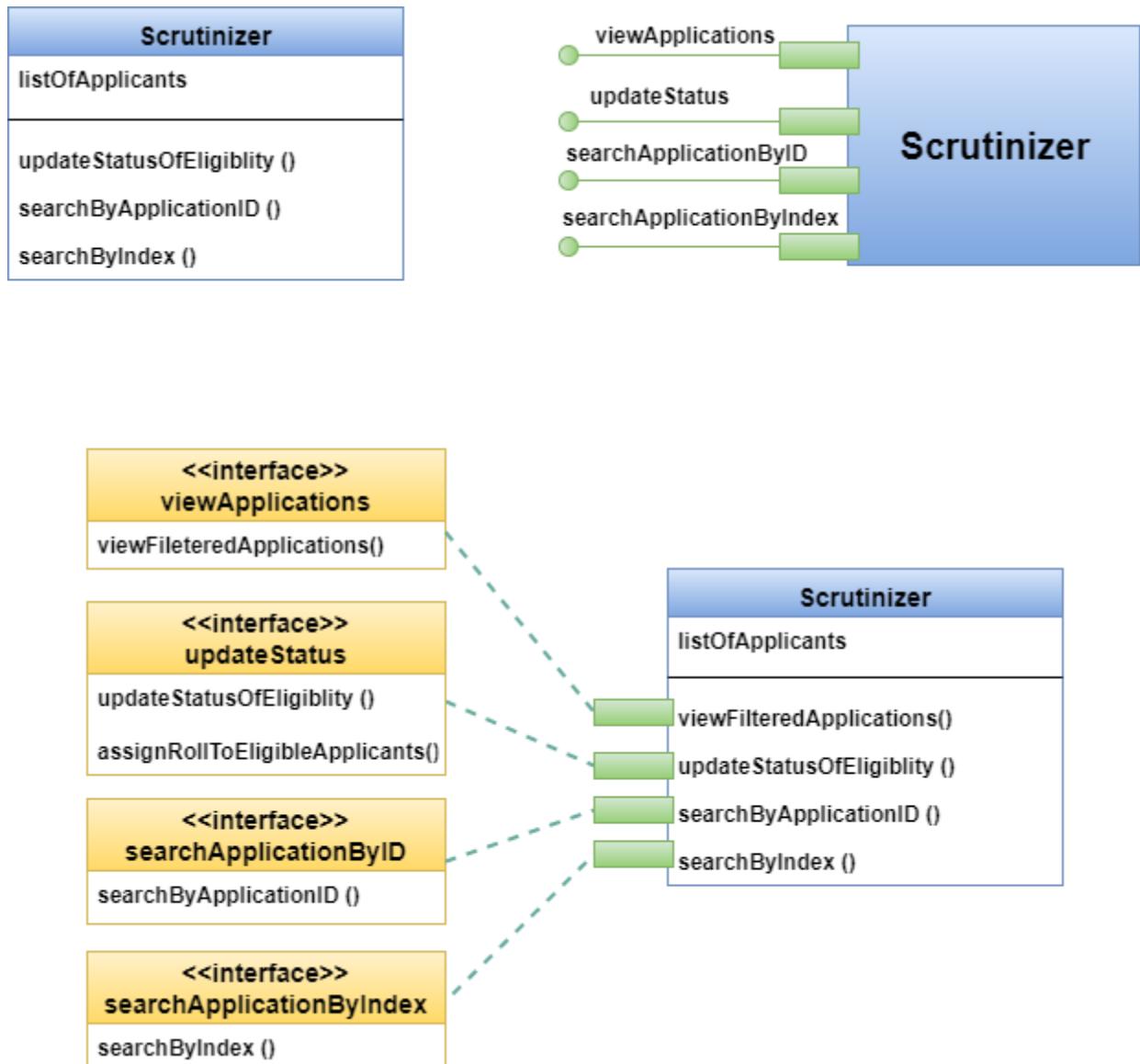


Figure 50 Elaborated Scrutinizer class

Figure 51 shows elaborated Receptionist class. It has two interfaces.

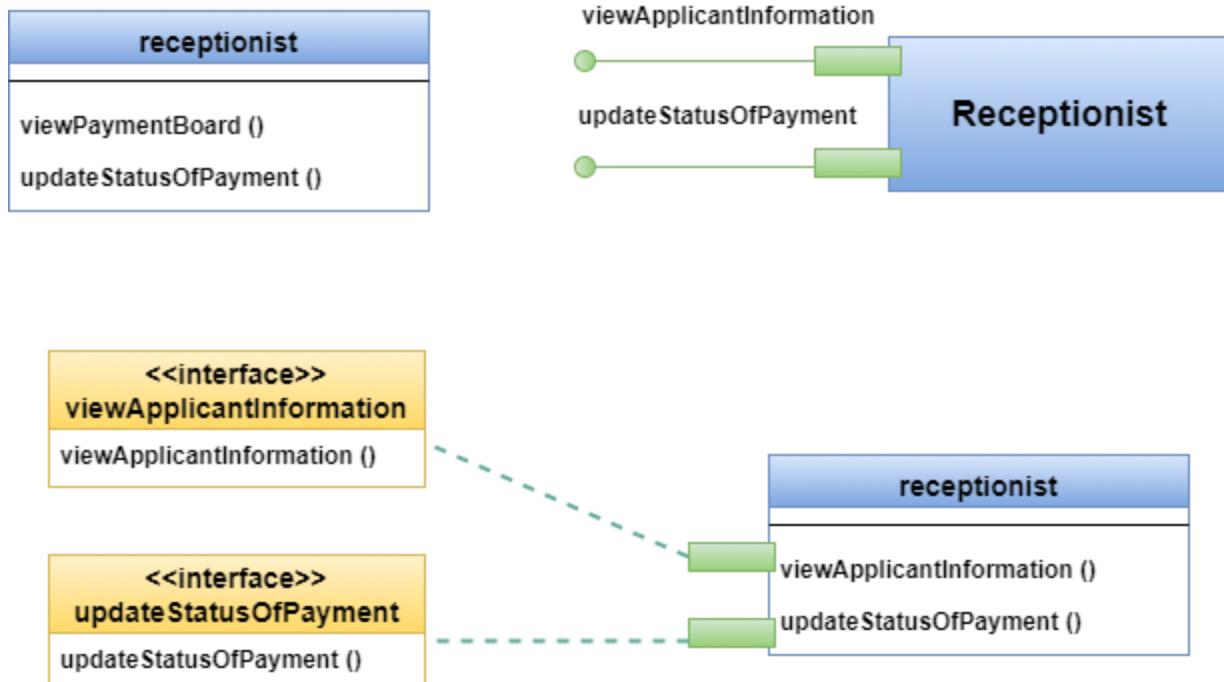


Figure 51 Elaborated Receptionist class

Figure 52 shows elaborated ExamController class. It has twenty four interfaces.



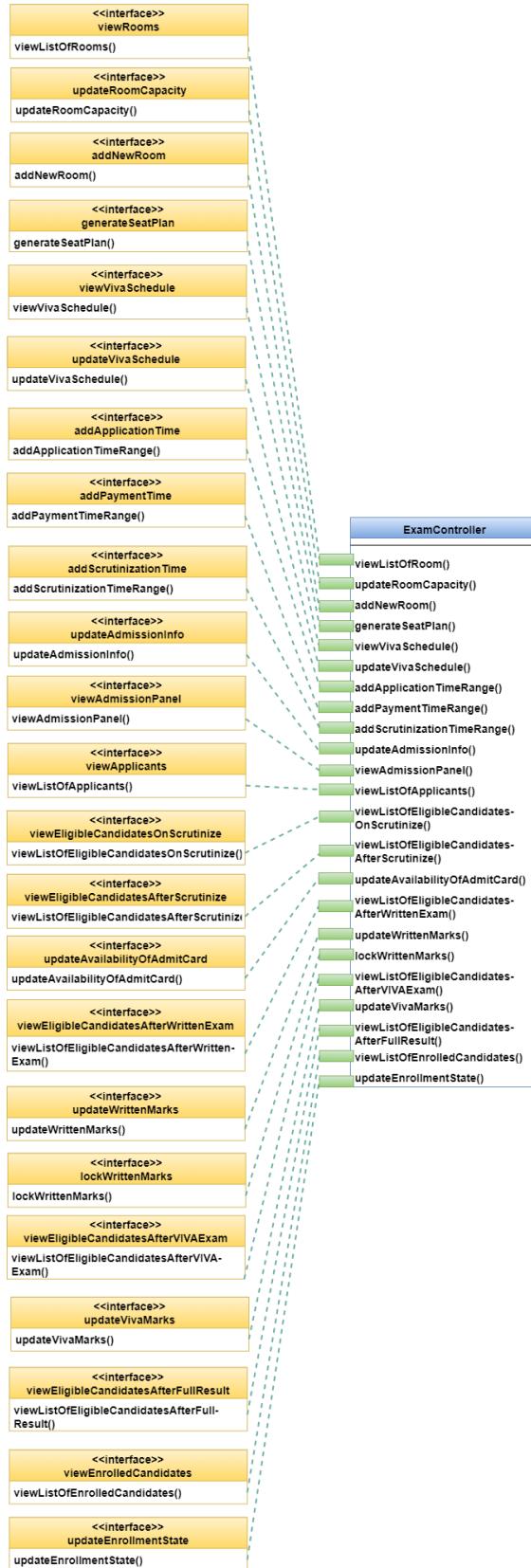
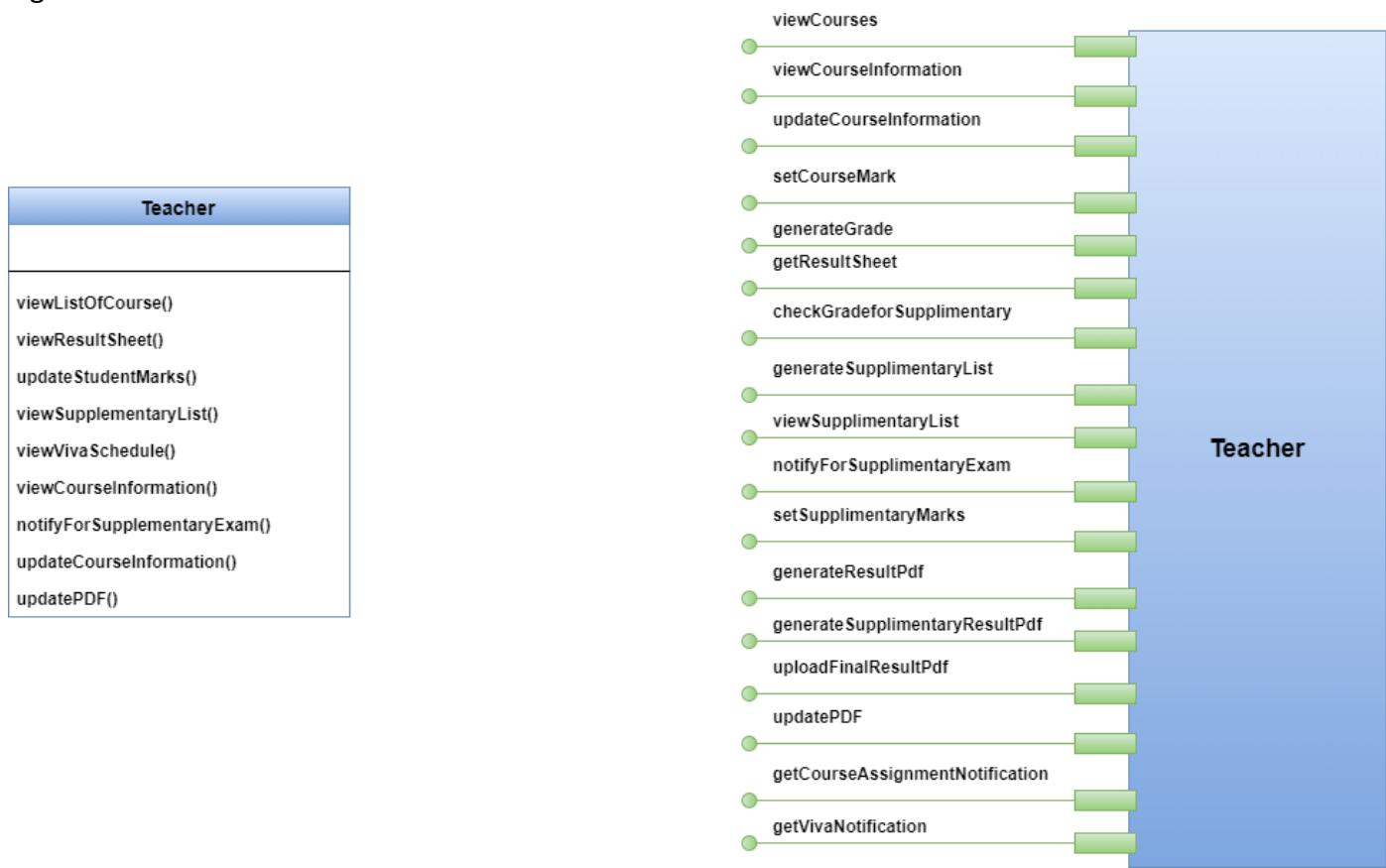


Figure 52 Elaborated ExamController class

Figure 53 shows elaborated Teacher class. It has seventeen interfaces.



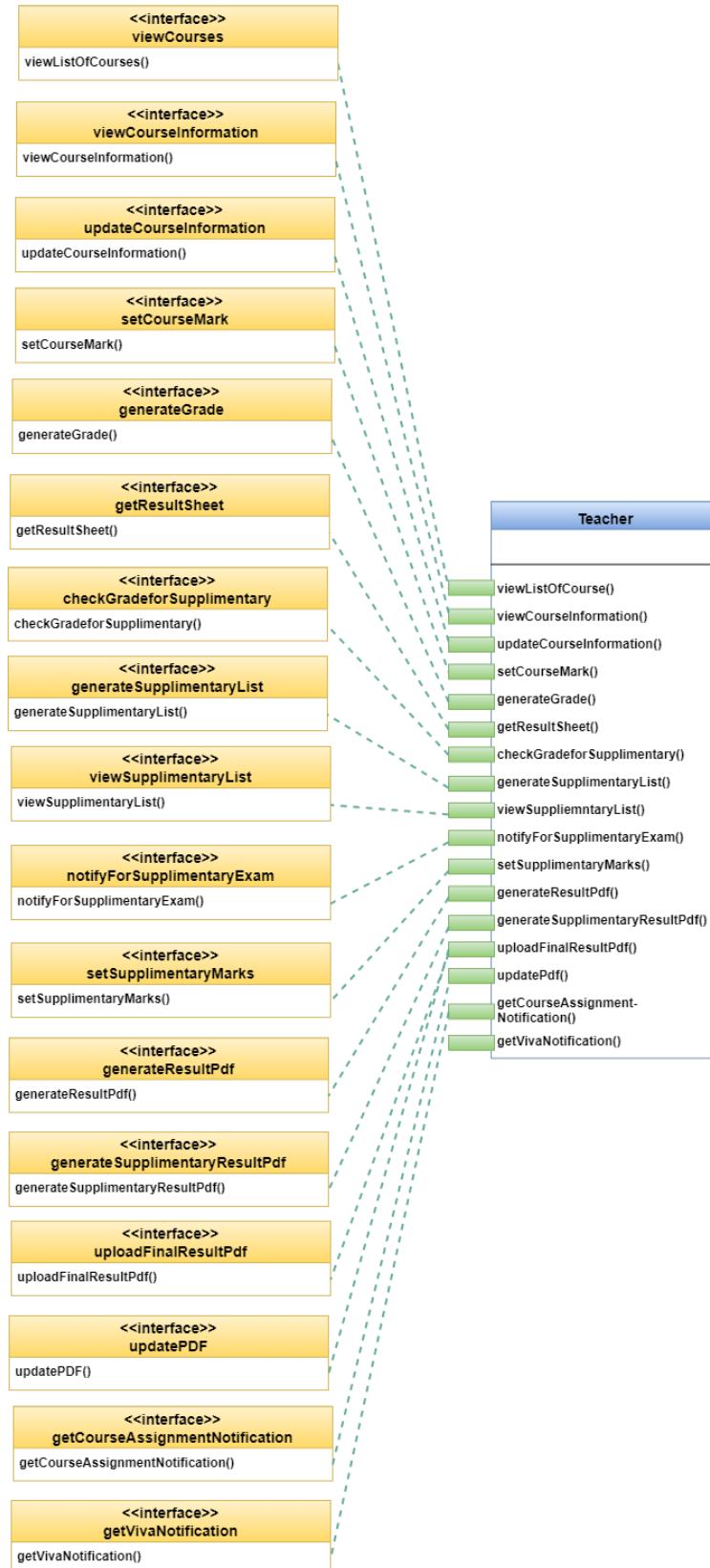


Figure 53 Elaborated Teacher class

Figure 54 shows elaborated Student class. It has four interfaces.

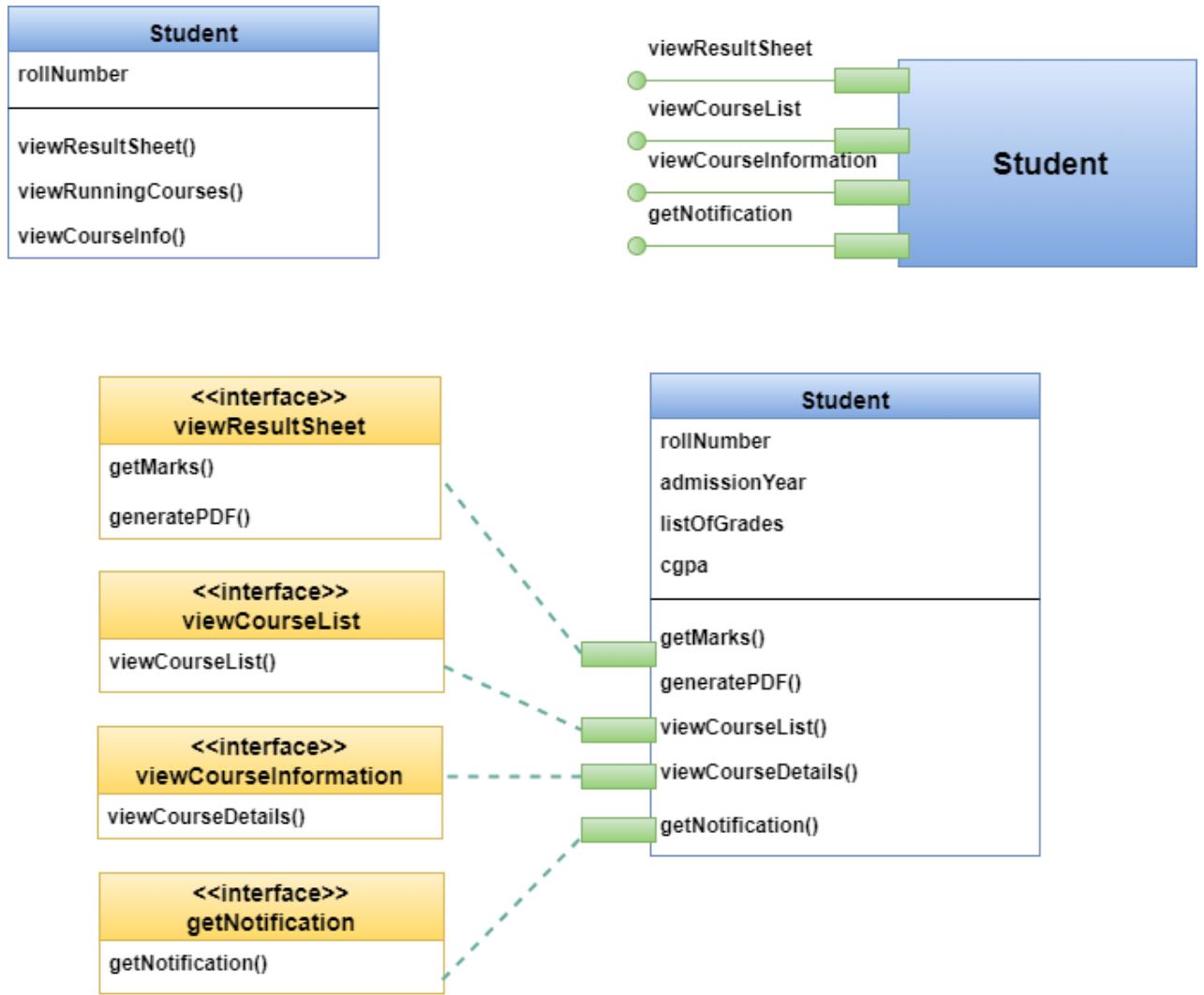


Figure 54 Elaborated Student class

### 3.3.1 Specify Message Details when Classes or Components Collaborate

The requirements model makes use of a collaboration diagram to show how analysis classes collaborate with one another. As component-level design proceeds, it is sometimes useful to show the details of these collaborations by specifying the structure of messages that are passed between objects within a system. This design activity shows how components within the system communicate and collaborate.

Figure \*-\* shows collaboration diagrams for \*\*\*\*\* class.

Figure \*-\* shows collaboration diagrams for \*\*\*\*\* class.

Figure \*-\* shows collaboration diagrams for \*\*\*\*\* class.

### 3.3.2 Identify Appropriate Interfaces for Each Component

Within the context of component-level design, a UML interface is a group of externally visible operations.

User class has some interfaces named verification, forgot password which deal with authentication related activities. But it is not the responsibility of User class to handle authentication. So we have defined a new class named Authentication which will be responsible for handling authentication. Figure 64 shows Authentication class.

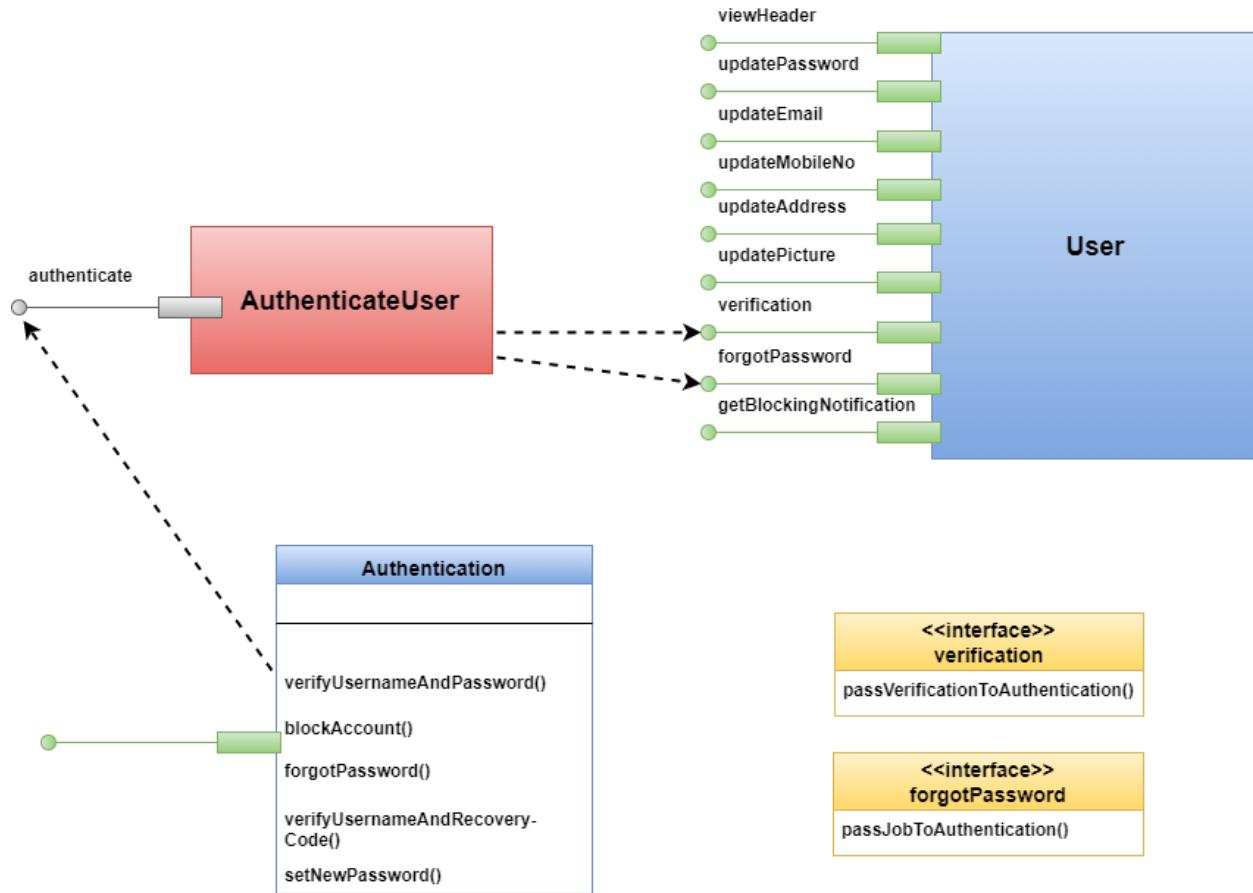


Figure 55 Authentication class

It is not the responsibility of Scrutinizer class to generate roll number for applicants. So a new class RollGenerator has been defined for handling this activity. Figure 65 shows RollGenerator class.

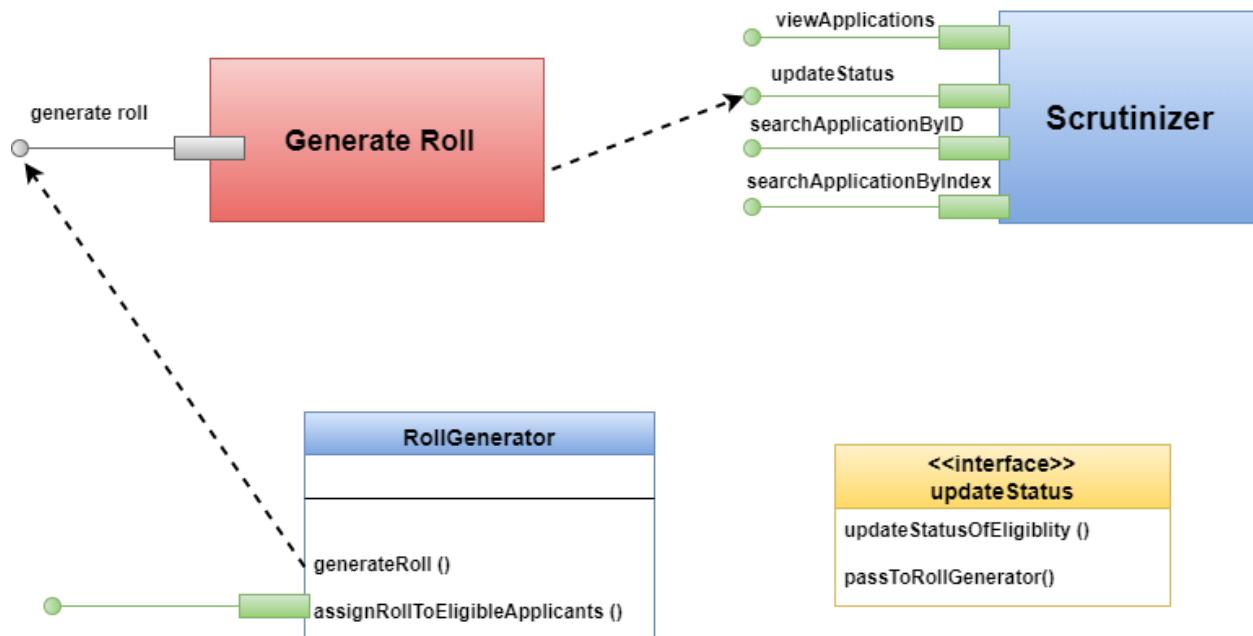


Figure 56 RollGenerator class

Student class has an interface named `viewResultSheet` which deals with result generation process. But it is not the responsibility of Student class. So we have defined a new class named `ReportCard` which will be responsible for generation report card. Figure 66 shows `ReportCard` class.

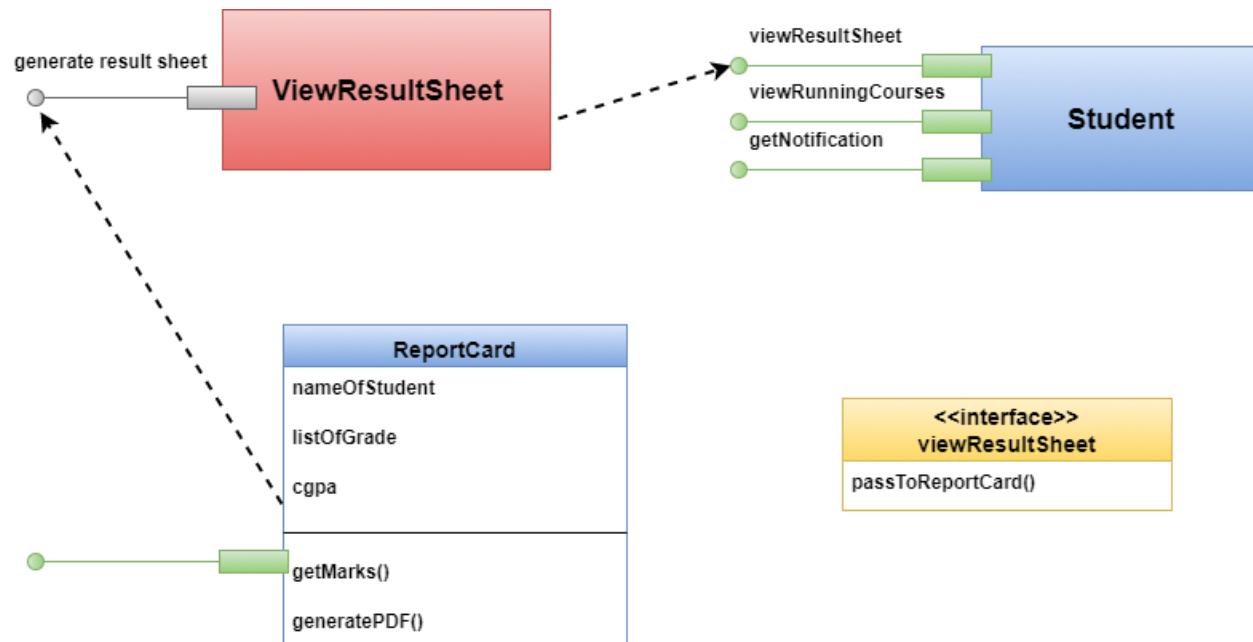


Figure 57 ReportCard class

It is not the responsibility of ExamController class to generate seat plan for applicants. So a new class RoomHandler has been introduced for handling this activity. Figure 67 shows RoomHandler class.

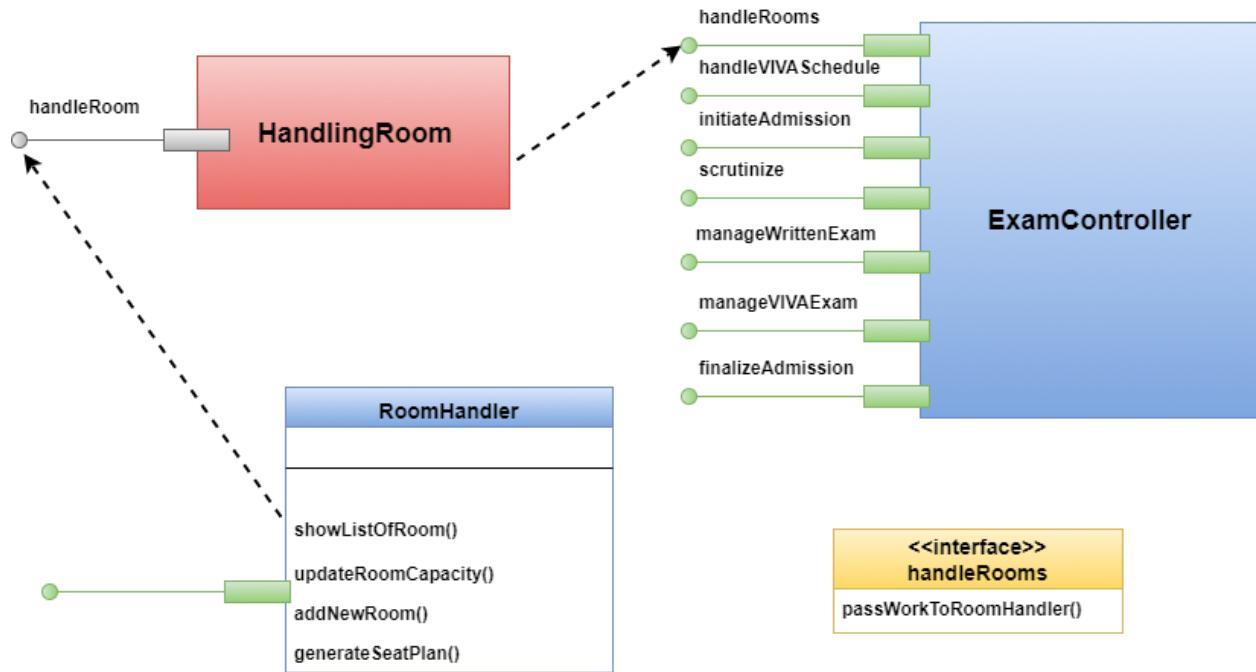


Figure 58 RoomHandler class

It is not the responsibility of User class to retrieve and display name and image. So a new class DisplayHeader has been introduced for handling this activity. Figure 68 shows DisplayHeader class.

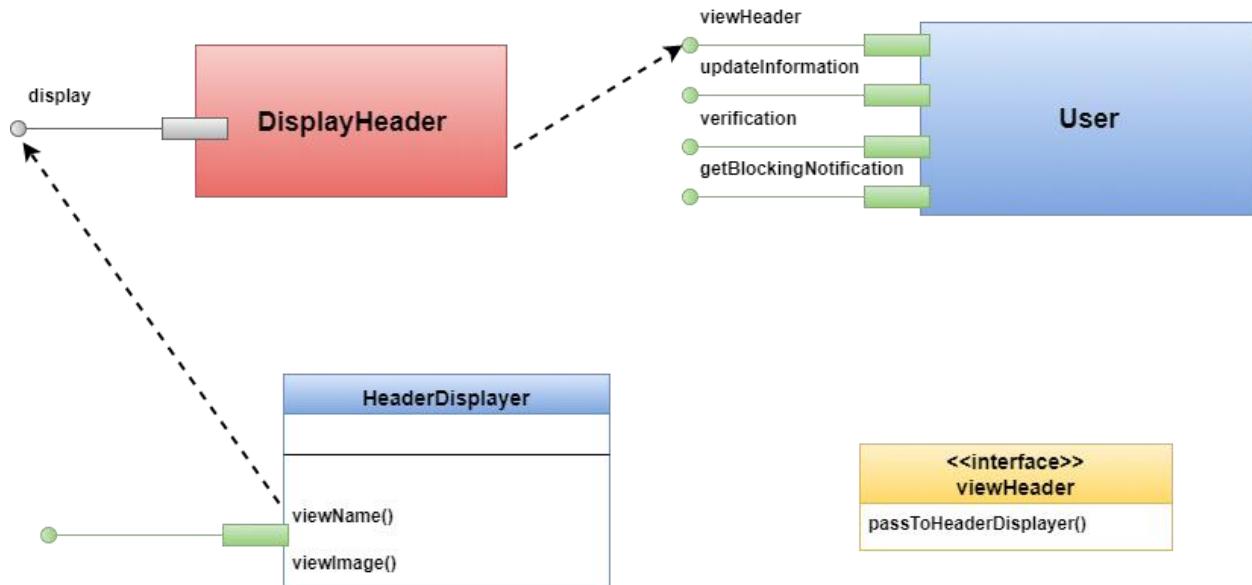


Figure 59 HeaderDisplayer class

It is not the task of Teacher class to generate PDF. So PDFHandler class has been introduced which is shown in figure 69.

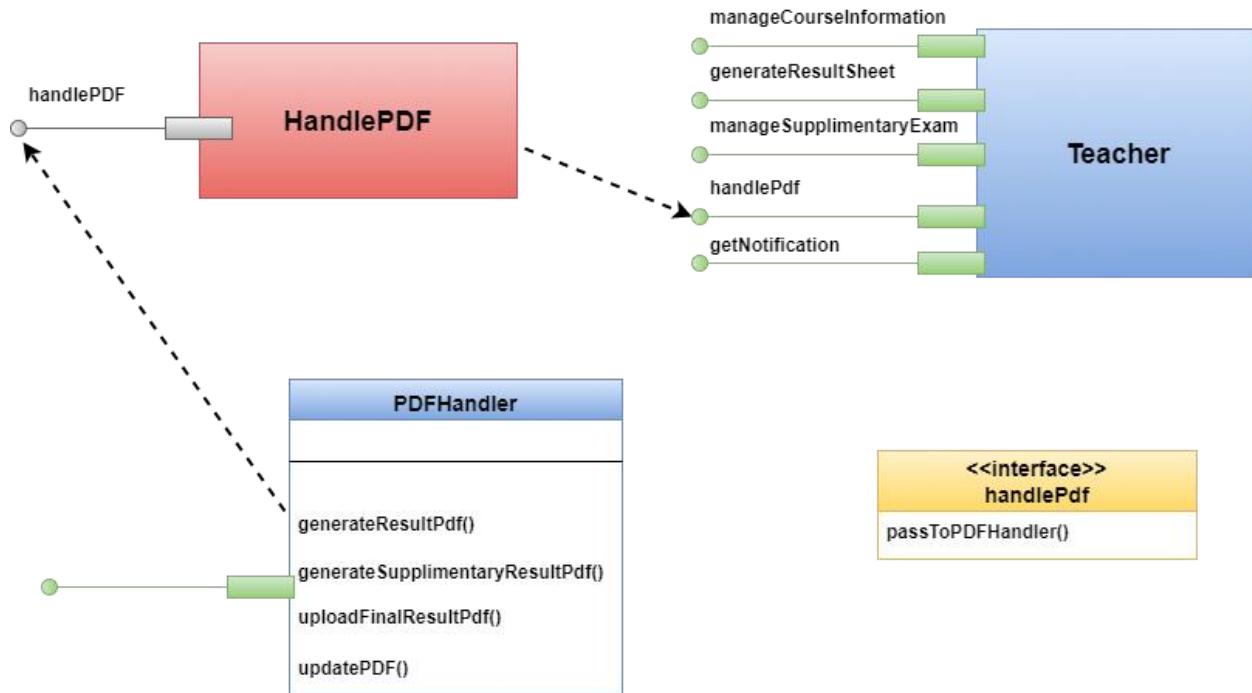


Figure 60 PDFHandler class

### 3.3.3 Elaborate Attributes and Define Data Types and Data Structures required to Implement Them

Generally data structures and types used to define attributes are defined within the context of the programming language that is to be used for implementation.

Table 1 elaborates all attributes of design classes.

Table 1 Elaborated Attributes of Design classes

No	Attribute Name	Class	Data Type	Initial Value	Property String
1	username	User	String	null	[a-zA-Z]{1,20}
2	password	User	String	null	^(?=.*[A-Za-z])(?=.*\d)[A-Za-z\d]{8,30}\$
3	fullName	User, Applicant, ReportCard	String	null	[a-zA-Z]{1,50}
4	fatherName	User, Applicant	String	null	[a-zA-Z]{1,50}
5	motherName	User, Applicant	String	null	[a-zA-Z]{1,50}
6	mobileNumber	User, Applicant	String	null	[0-9]{11}
7	emailAddress	User, Applicant	String	null	Maximum length = 50 characters
8	address	User, Applicant	String	null	Maximum length = 250 characters
9	rollNumber	Student	int	0	[0-9]{6}
10	year	Student, Applicant	int	2017	[0-9]{4}
11	listOfGrade	ReportCard	Associative array	0.00	0.00 ≤ grade of a course ≤ 4.00
12	cgpa	ReportCard	float	0.00	0.00 ≤ cgpa ≤ 4.00
13	applicationID	Applicant	String	null	[0-9]{6}
14	dateOfBirth	Applicant	Date	Current date	Minimum date 01-01-1990
15	nationality	Applicant	String	null	[a-zA-Z]{1,20}
16	schoolName	Applicant	String	null	[a-zA-Z]{1,50}
17	schoolBoard	Applicant	String	null	[a-zA-Z]{1,50}
18	schoolGroup	Applicant	String	null	[a-zA-Z]{1,50}
19	schoolYear	Applicant	int	2010	[0-9]{4}
20	schoolGPA	Applicant	float	0.00	0.00 ≤ value ≤ 5.00

21	collegeName	Applicant	String	null	[a-zA-Z ]{1,50}
22	collegeBoard	Applicant	String	null	[a-zA-Z ]{1,50}
23	collegeGroup	Applicant	String	null	[a-zA-Z ]{1,50}
24	collegeYear	Applicant	int	2012	[0-9]{4}
25	collegeGPA	Applicant	float	0.00	0.00 ≤ value ≤ 5.00
26	universityName	Applicant	String	null	[a-zA-Z ]{1,50}
27	departmentName	Applicant	String	null	[a-zA-Z ]{1,50}
28	universityYear	Applicant	int	2016	[0-9]{4}
29	universityGPA	Applicant	float	0.00	0.00 ≤ value ≤ 4.00
30	algebraInHSC	Applicant	boolean	false	Either true or false
31	algebraIn Bachelor	Applicant	boolean	false	Either true or false
32	mathInHSC	Applicant	boolean	false	Either true or false
33	mathPercentage InHSC	Applicant	float	0.00	0.00 ≤ value ≤ 100.00
34	mathInBachelor	Applicant	boolean	false	Either true or false
35	mathPercentage InBachelor	Applicant	float	0.00	0.00 ≤ value ≤ 100.00
36	eligibilityForViva	Applicant	boolean	false	Either true or false
37	enrollmentStatus	Applicant	boolean	false	Either true or false
38	applicationFee PaymentStatus	Applicant	boolean	false	Either true or false
39	eligibilityOf Application	Applicant	boolean	false	Either true or false
40	roomNo	Applicant	int	0	[0-9]{3}
41	writtenMarks	Applicant	float	0.00	0.00 ≤ value ≤ 100.00
42	eligibilityForVIVA	Applicant	boolean	false	Either true or false
43	VIVAmarks	Applicant	float	0.00	0.00 ≤ value ≤ 100.00
44	eligibilityFor	Applicant	boolean	false	Either true or false

	Admission				
45	admissionFee PaymentStatus	Applicant	boolean	false	Either true or false
46	enrollmentStatus	Applicant	boolean	false	Either true or false

### 3.3.4 Describe Processing Flow within Each Operation in Detail

This may be accomplished using UML activity diagram.

Figure 68-71 show UML activity diagrams for Authentication class.

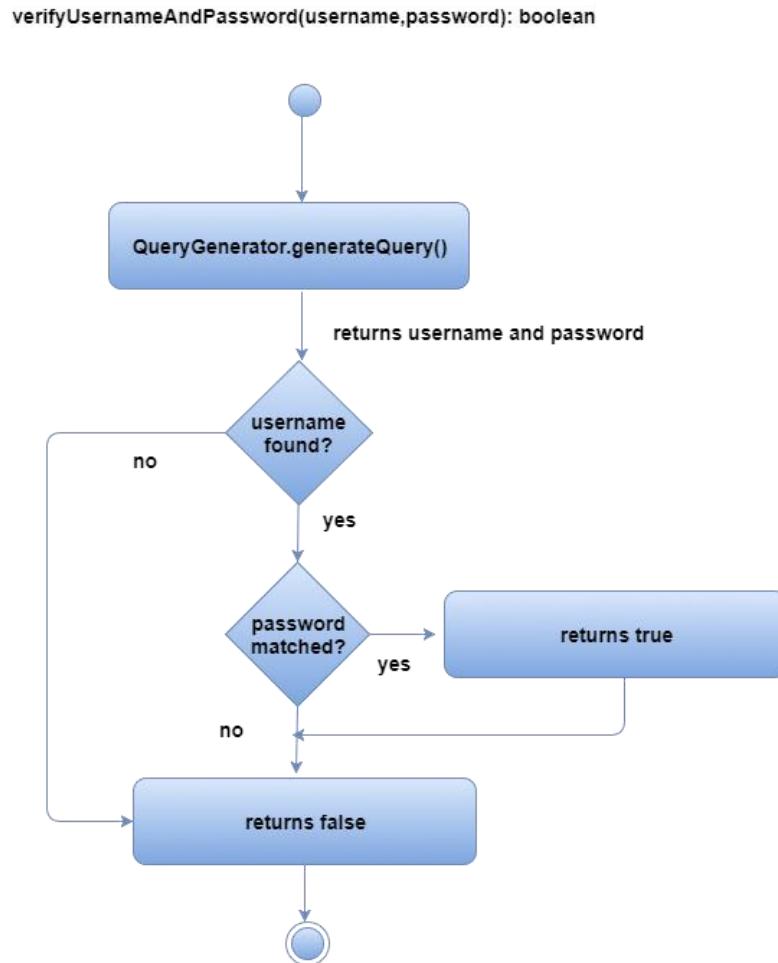


Figure 61 verifyUsernameAndPassword() of Authentication

```
forgotPassword(username): void
```

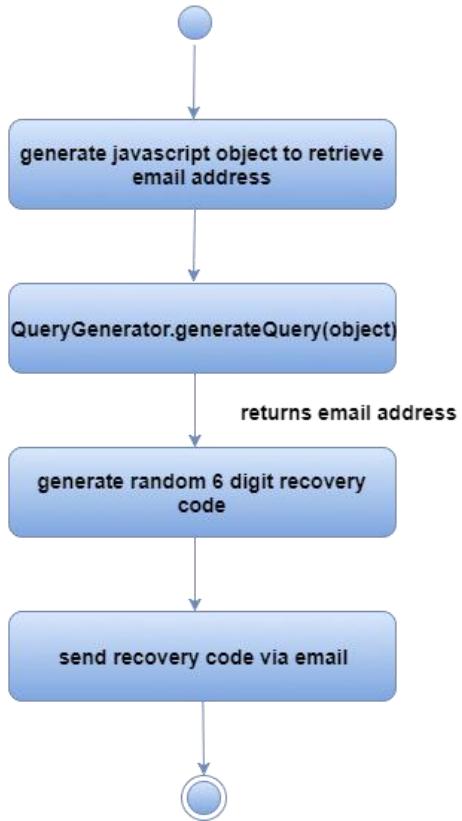


Figure 62 forgotPassword() of Authentication

```
verifyUsernameAndRecoveryCode(username, recovery code): boolean
```

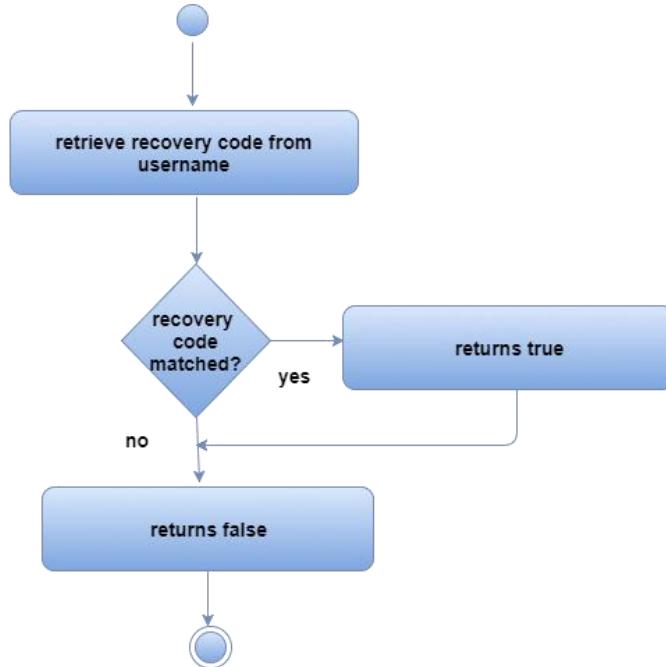


Figure 63 `verifyUsernameAndRecoveryCode()` of Authentication

```
setNewPassword(username, password): void
```

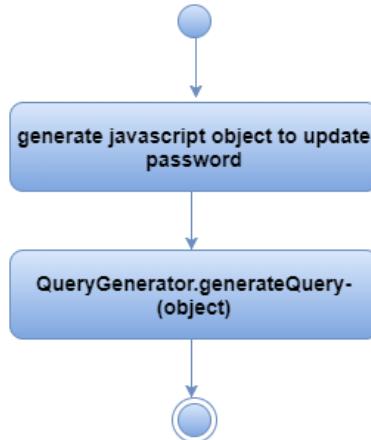


Figure 64 `setNewPassword()` of Authentication

Figure 72-74 show UML activity diagrams for Applicant class.

`applyForAdmission(application data): String`

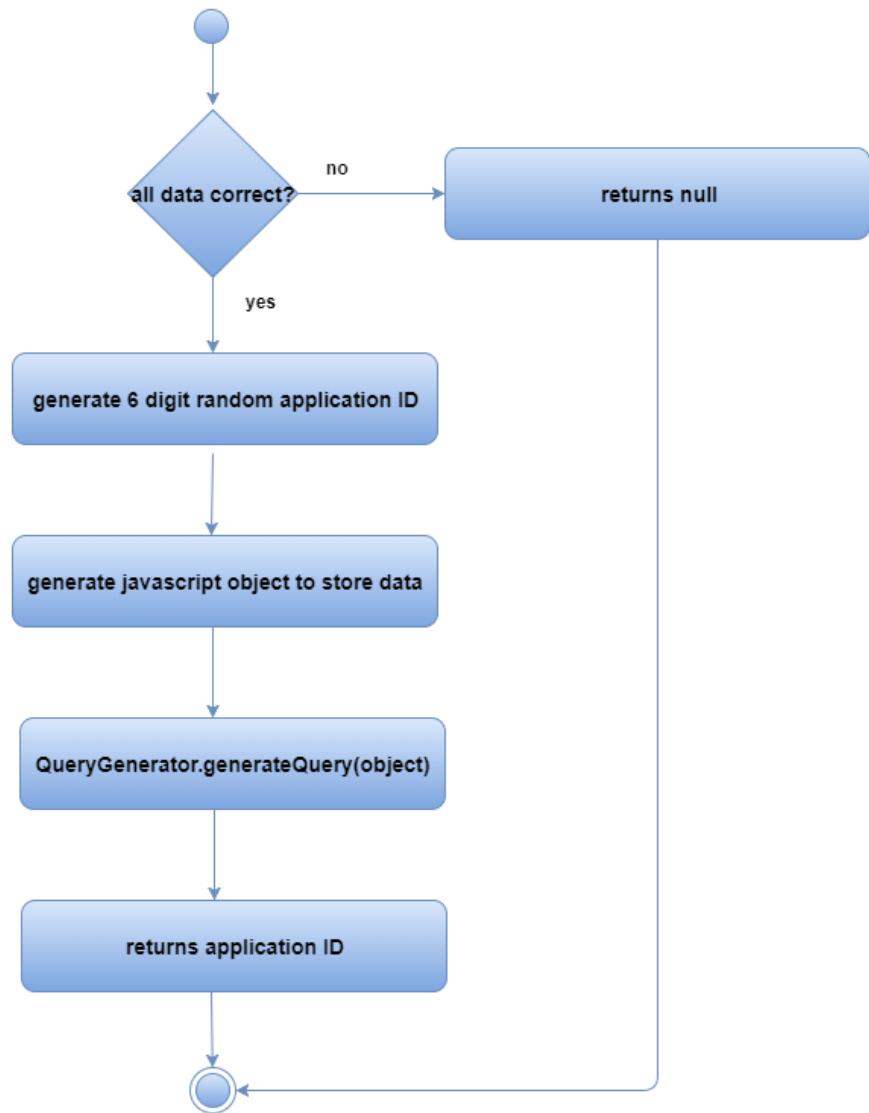


Figure 65 `applyForAdmission()` of Applicant

`checkAvailabilityOfAdmitCard(ApplicationID): boolean`

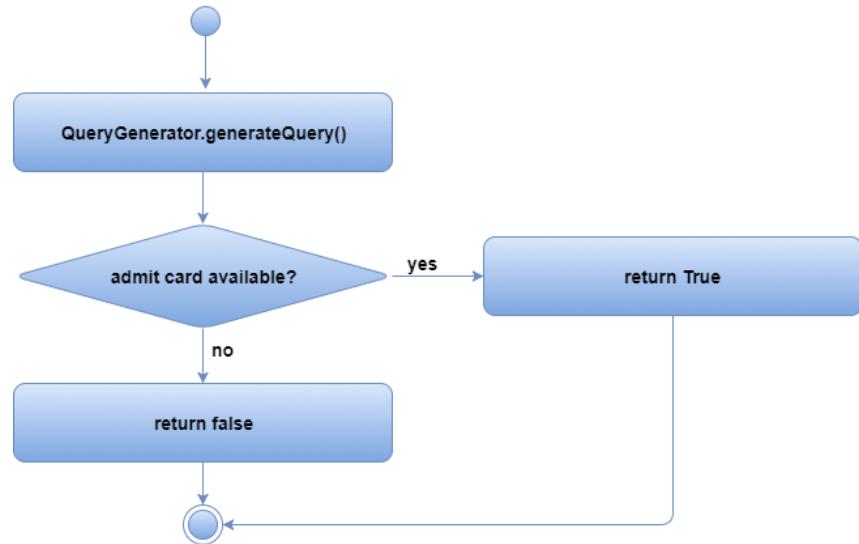


Figure 66 `checkAvailabilityOfAdmitCard()` of Applicant

`generateAdmitCard(applicationID): void`

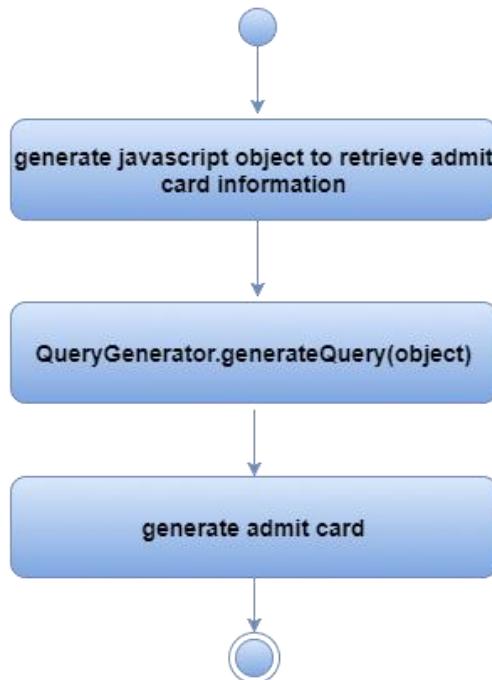


Figure 67 `generateAdmitCard()` of Applicant

Figure 75 shows UML activity diagram for `QueryGenerator` class.

```
generateQuery(javascript object): resultset
```

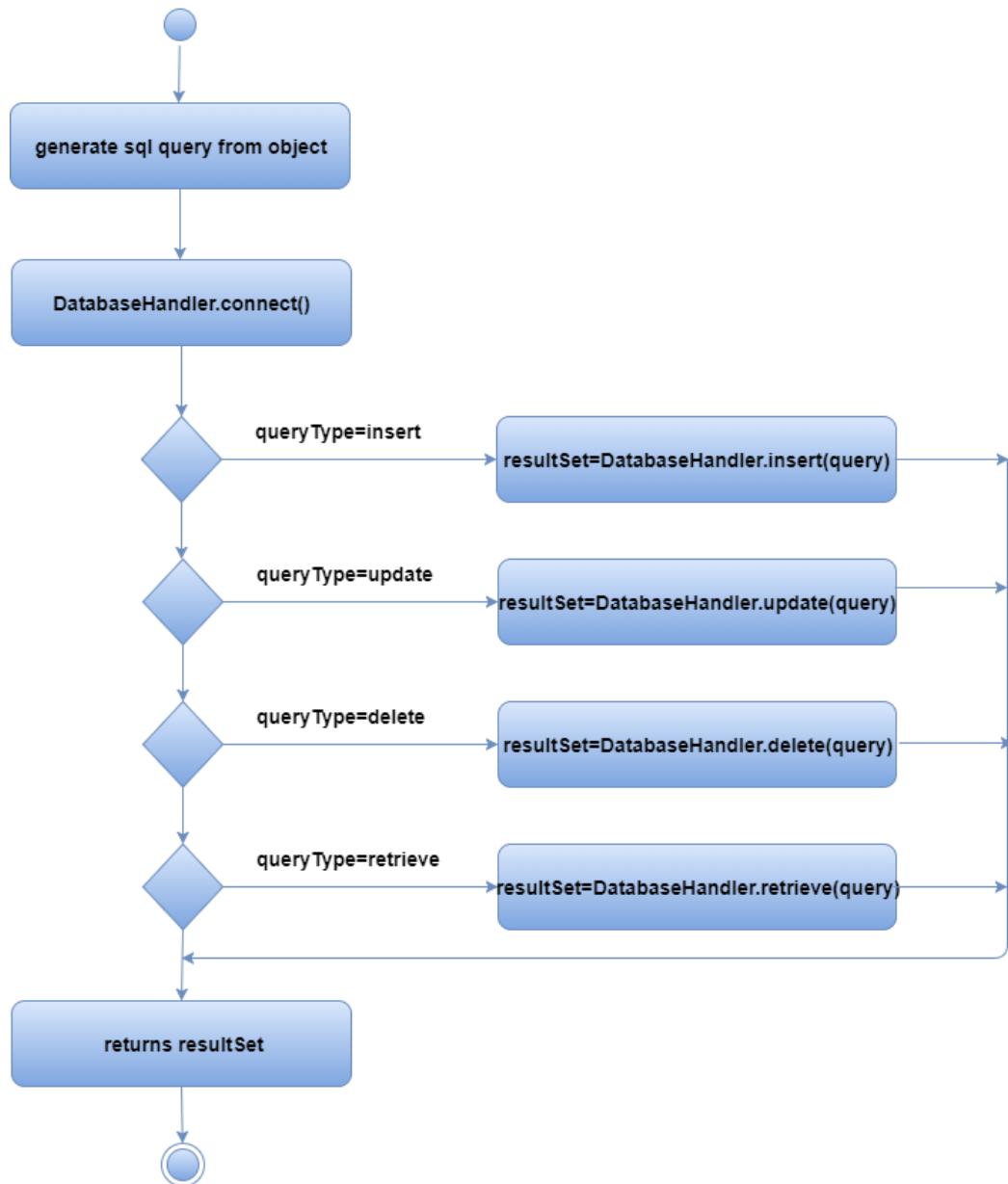


Figure 68 `generateQuery()` of `QueryGenerator`

Figure \*-\* show UML activity diagrams for User class.

`updateAddress(username): void`

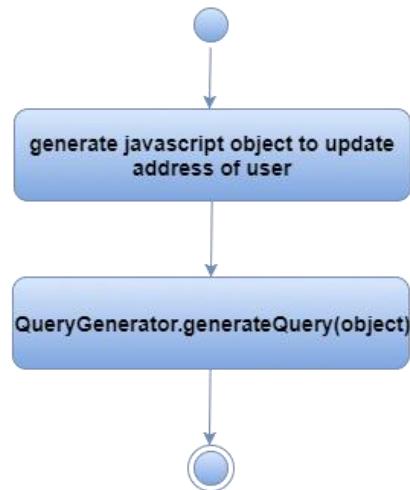


Figure 69 `updateAddress()` of User

`updateMobileNo(username): void`

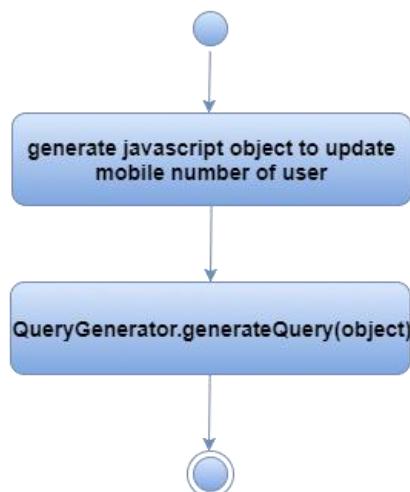


Figure 70 `updateMobileNo()` of User

```
updatePicture(username): void
```

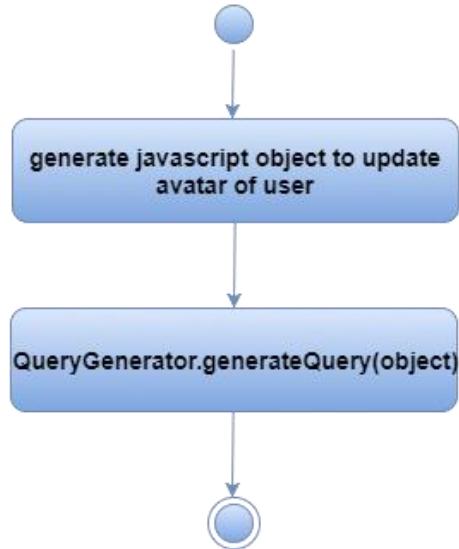


Figure 71 updatePicture() of User

```
getBlockingNotification(): notificationMessage
```

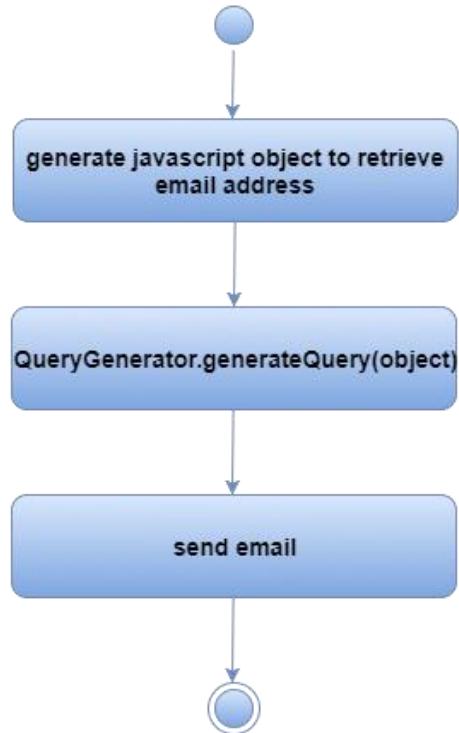


Figure 72 getBlockingNotification() of User

Figure \*-\* show UML activity diagrams for HeaderDisplayer class.

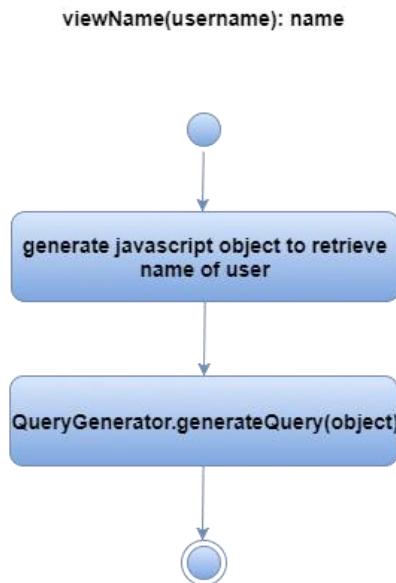


Figure 73 `viewName()` of HeaderDisplayer

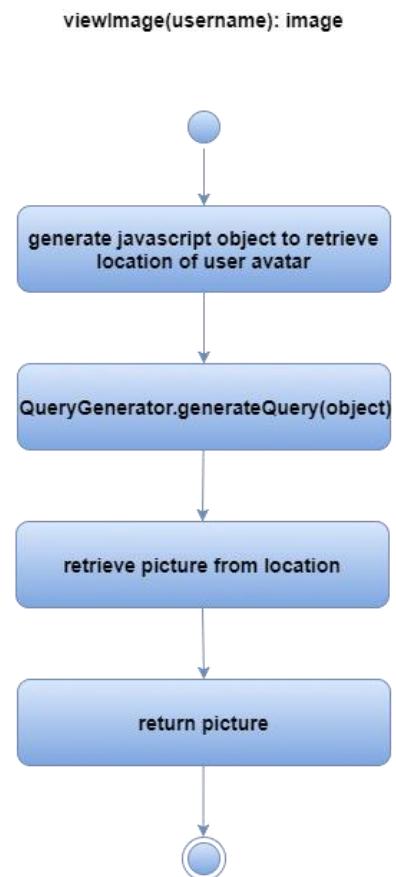


Figure 74 `viewImage()` of HeaderDisplayer

Figure \*-\* show UML activity diagrams for ProgramChairperson class.

`viewListOfUsers(): void`

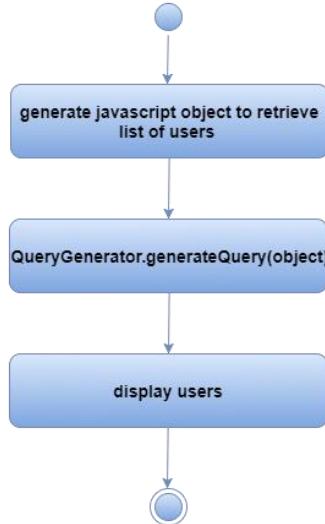


Figure 75 `viewListOfUsers()` of ProgramChairperson

`updatefullName(username): void`

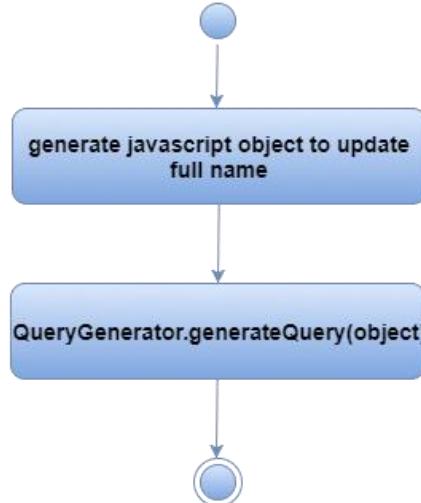


Figure 76 `updateFullName()` of ProgramChairperson

```
updateUserEmail(username): void
```

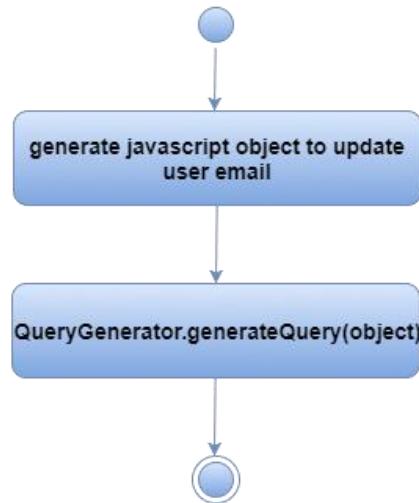


Figure 77 updateUserEmail () of ProgramChairperson

```
addNewStudentsIntoCourse(course, students): void
```

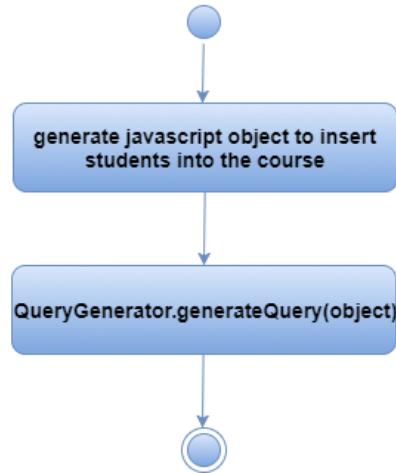


Figure 78 addNewStudentsIntoCourse() of ProgramChairperson

```
approveAccountRequest(registration number, decision): void
```

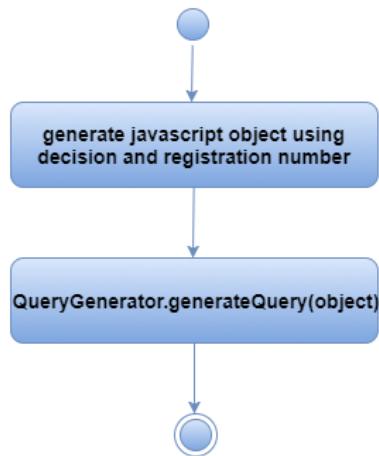


Figure 79 approveAccountRequest() of ProgramChairperson

```
checkFulfilmentOfPrerequisite(courseID, studentUsername ): boolean
```

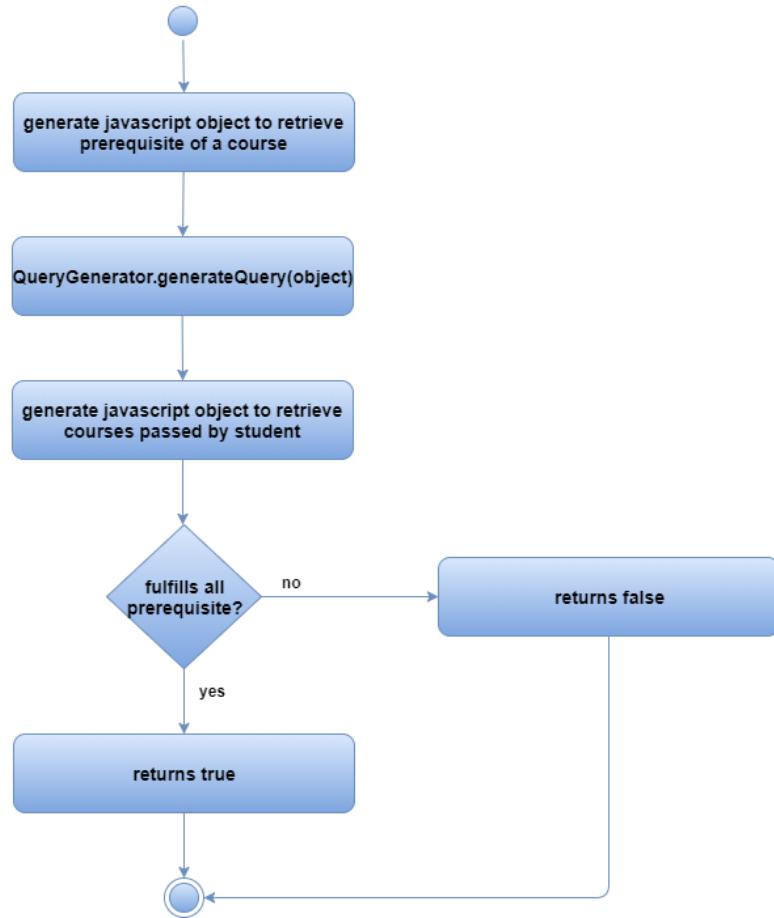


Figure 80 checkFulfillmentOfPayment() of ProgramChairperson

```
selectInstructor(course, teacher): void
```

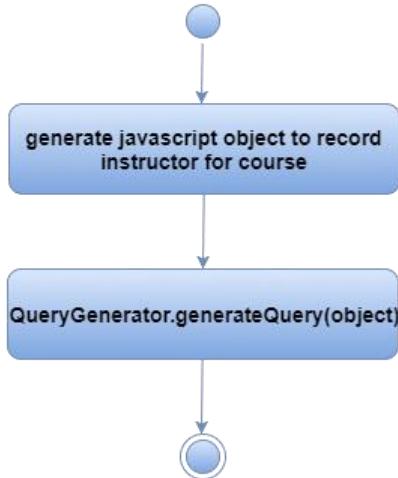


Figure 81 `selectInstructor()` of ProgramChairperson

```
createStudentAccount(): void
```

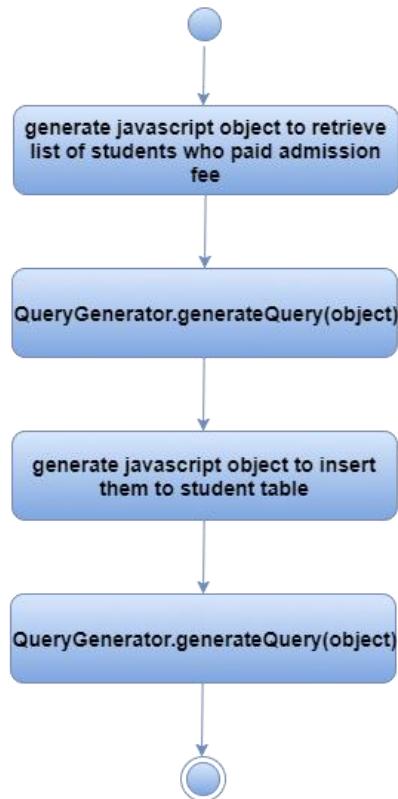


Figure 82 `createStudentAccount()` of ProgramChairperson

`viewResult(choice,value): void`

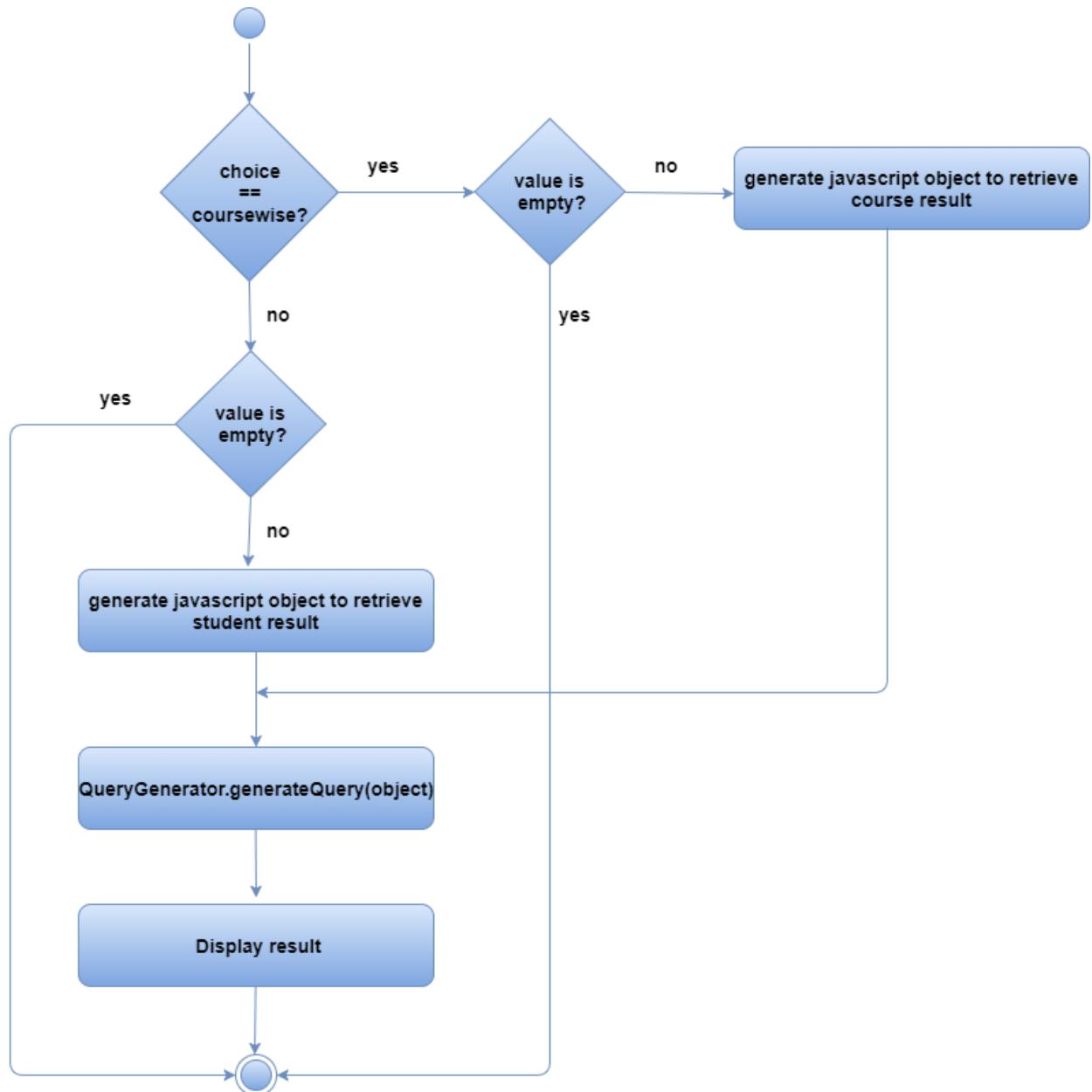


Figure 83 `viewResult()` of ProgramChairperson

`viewInformationOfOneUser(username): void`

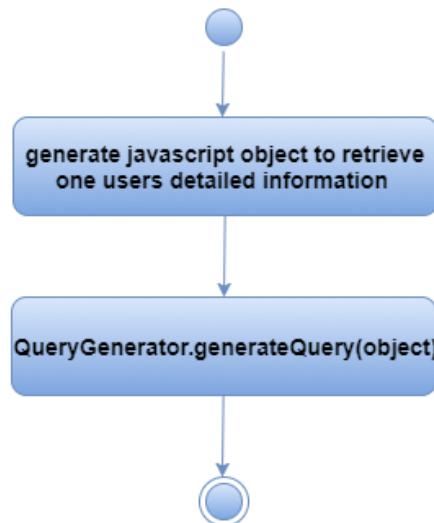


Figure 84 `viewInformationOfOneUser()` of ProgramChairperson

`viewListOfCourses(): void`

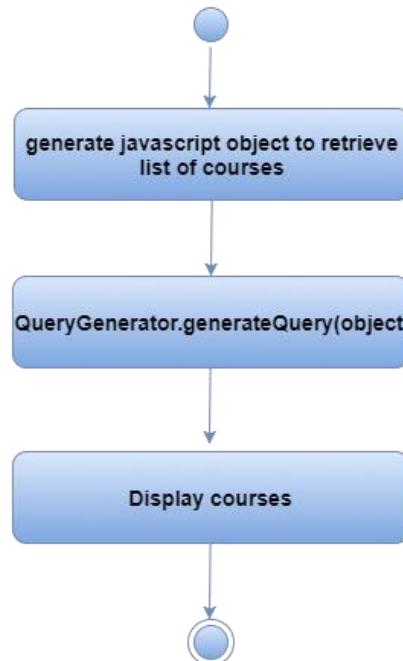


Figure 85 `viewListOfCourses ()` of ProgramChairperson

Figure \*-\* show UML activity diagrams for Teacher class.

```
checkGradeForSupplementary(courseCode,StudentID): boolean
```

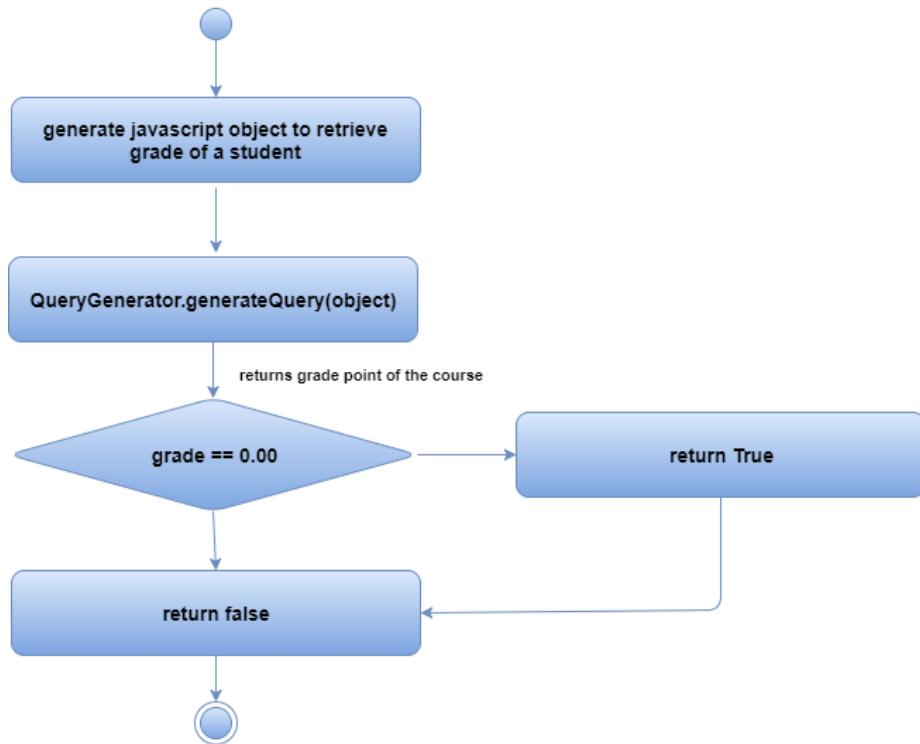


Figure 86 `checkGradeForSupplementary()` of Teacher

```
generateResultPdf(): void
```

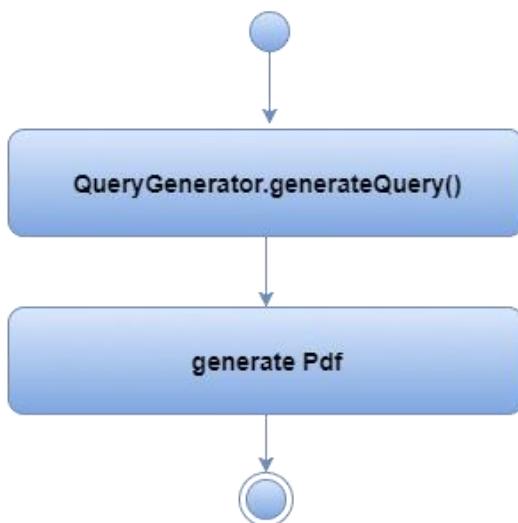


Figure 87 `generateResultPdf()` of Teacher

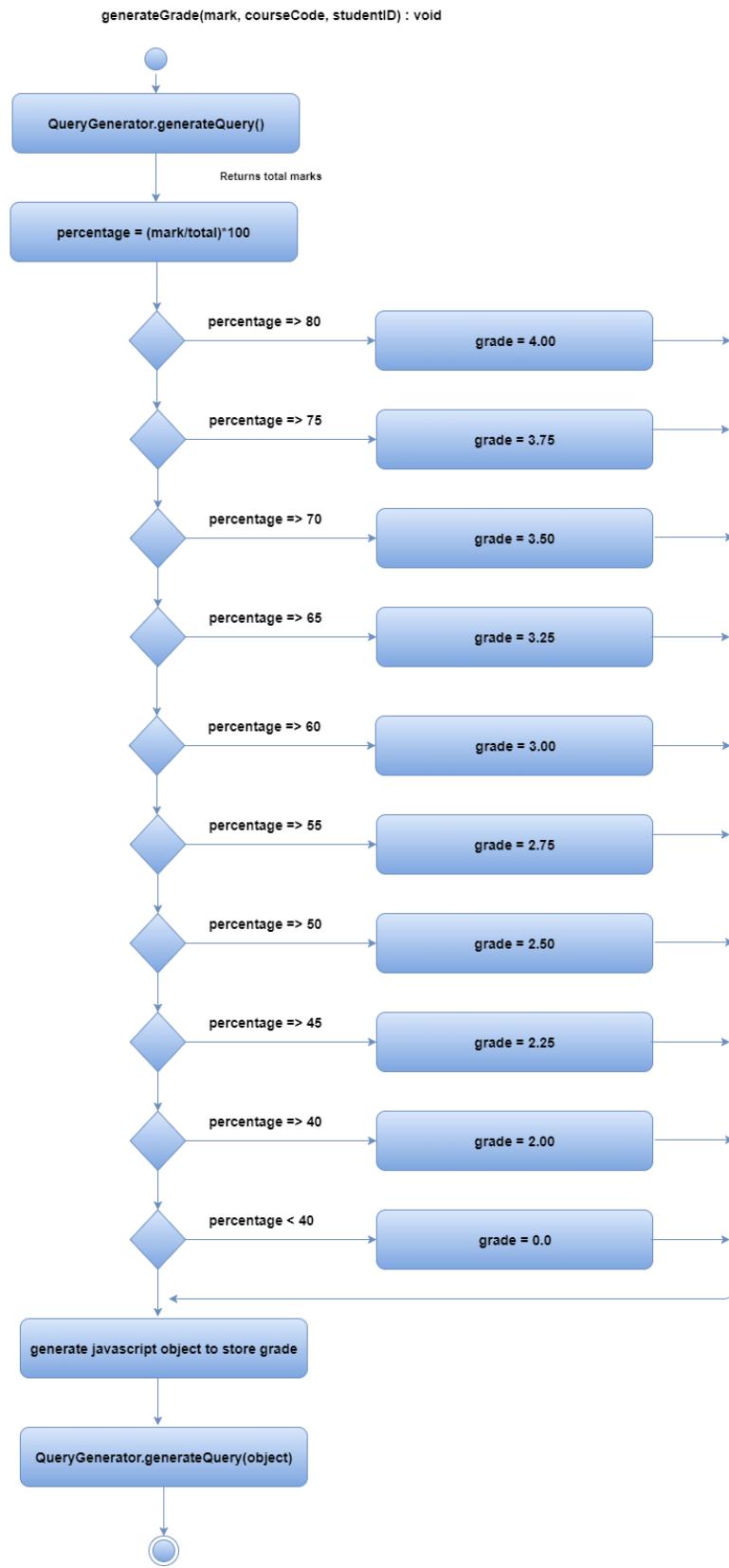


Figure 88 generateGrade() of Teacher

`generateSupplementaryList(courseID): supplementaryList`

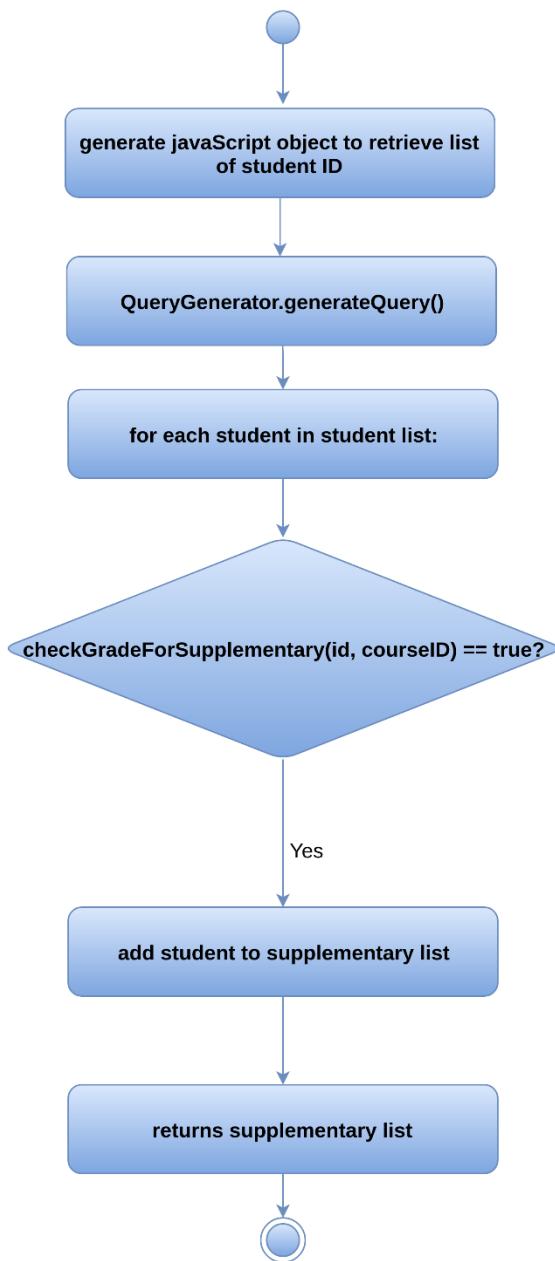


Figure 89 `generateSupplementaryList()` of Teacher

`generateSupplementaryResultPdf(): void`

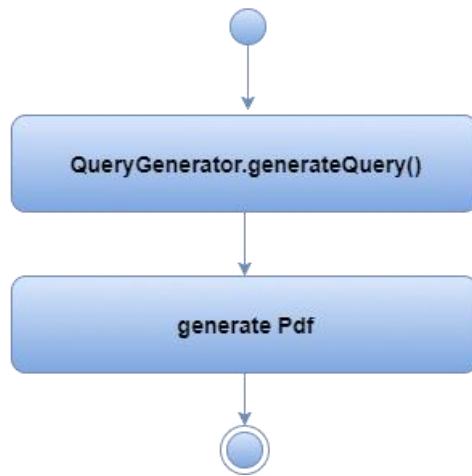


Figure 90 `generateSupplementaryResultPdf()` of Teacher

`getResultSet(): result sheet`

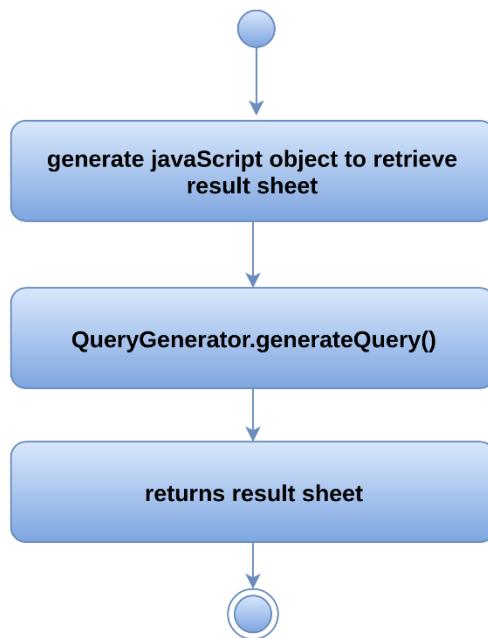


Figure 91 `getResultSheet()` of Teacher

`notifyForSupplementaryExam(): void`

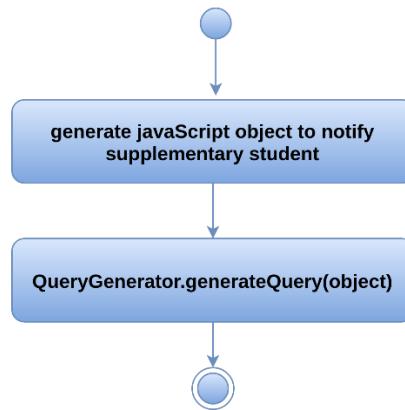


Figure 92 `notifyForSupplementaryExam()` of Teacher

`setCourseMark(mark) : void`

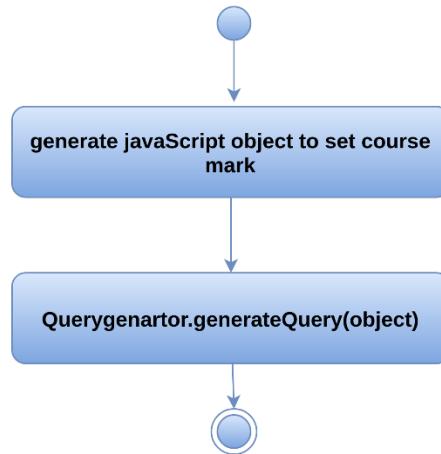


Figure 93 `setCourseMark()` of Teacher

`setSupplementaryMark(mark): void`

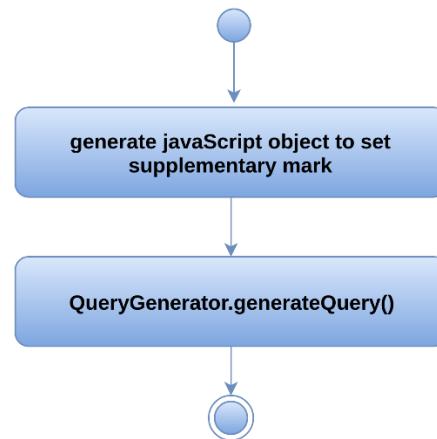


Figure 94 `setSupplementaryMark()` of Teacher

`updateCourseInformation(course ID, information):void`

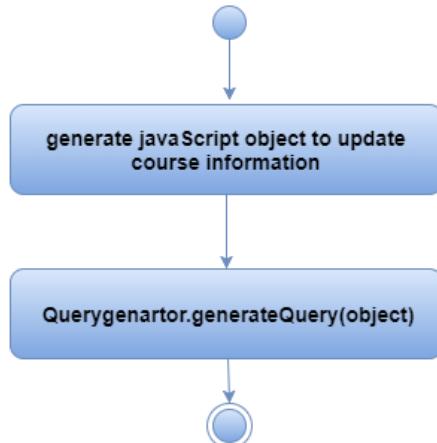


Figure 95 `updateCourseInformation()` of Teacher

**viewCourseInformation(courseCode): courseInformation**

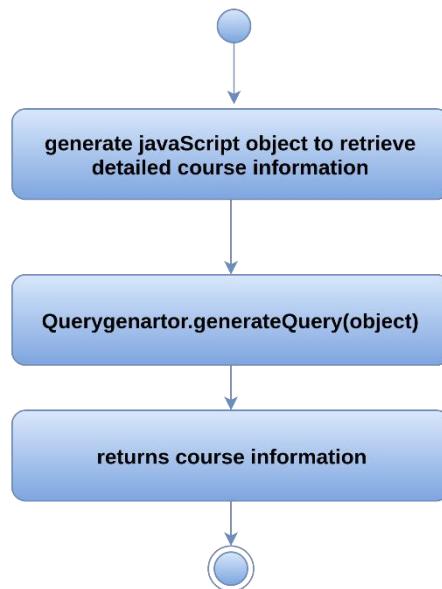


Figure 96 `viewCourseInformation()` of Teacher

**viewListOfCourse(): courseList**

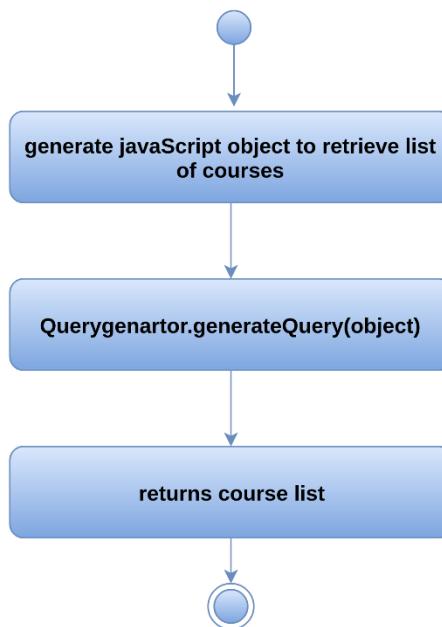


Figure 97 `viewListOfCourse()` of Teacher

`viewSupplementaryList():void`

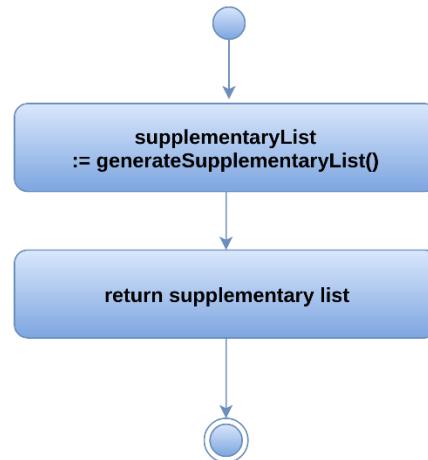


Figure 98 `viewSupplementaryList()` of Teacher

Figure \*-\* show UML activity diagrams for PDFHandler class.

`generateResultPdf(result): resultPDF`

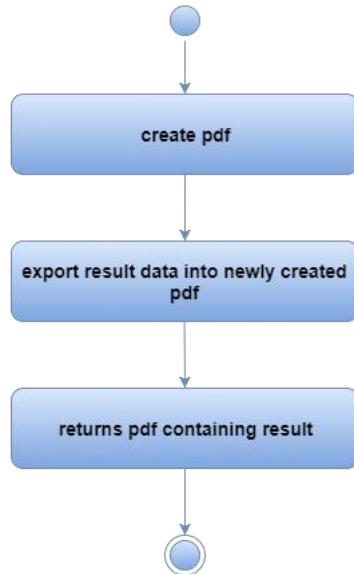


Figure 99 `generateResultPdf()` of PDFHandler

```
generateSupplementaryResultPdf(supplementaryResult): resultPDF
```

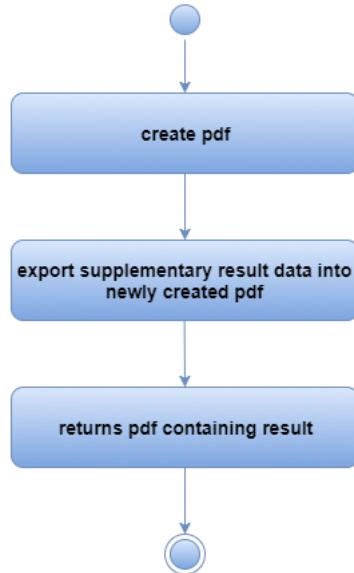


Figure 100 generateSupplementaryResultPdf() of PDFHandler

```
updatePdf(username, courseId, year): void
```

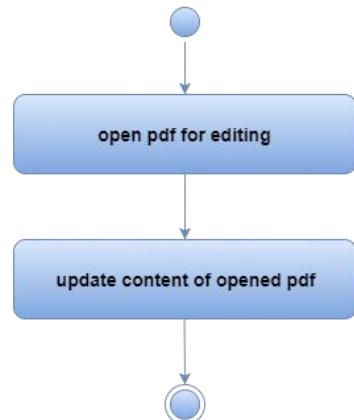


Figure 101 updatePdf() of PDFHandler

`uploadFinalResultPdf(finalResult): resultPDF`

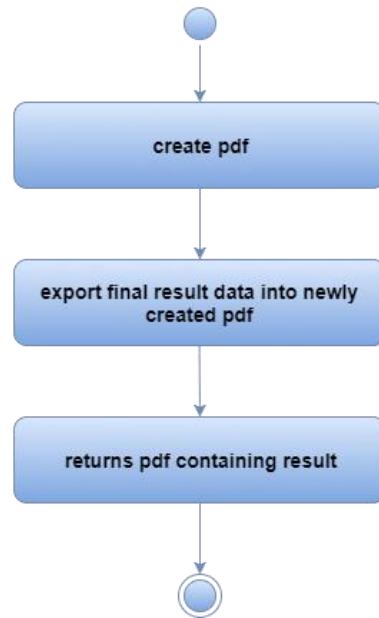


Figure 102 `uploadFinalResultPdf()` of PDFHandler

Figure \*-\* show UML activity diagrams for Student class.

`getNotification(): void`

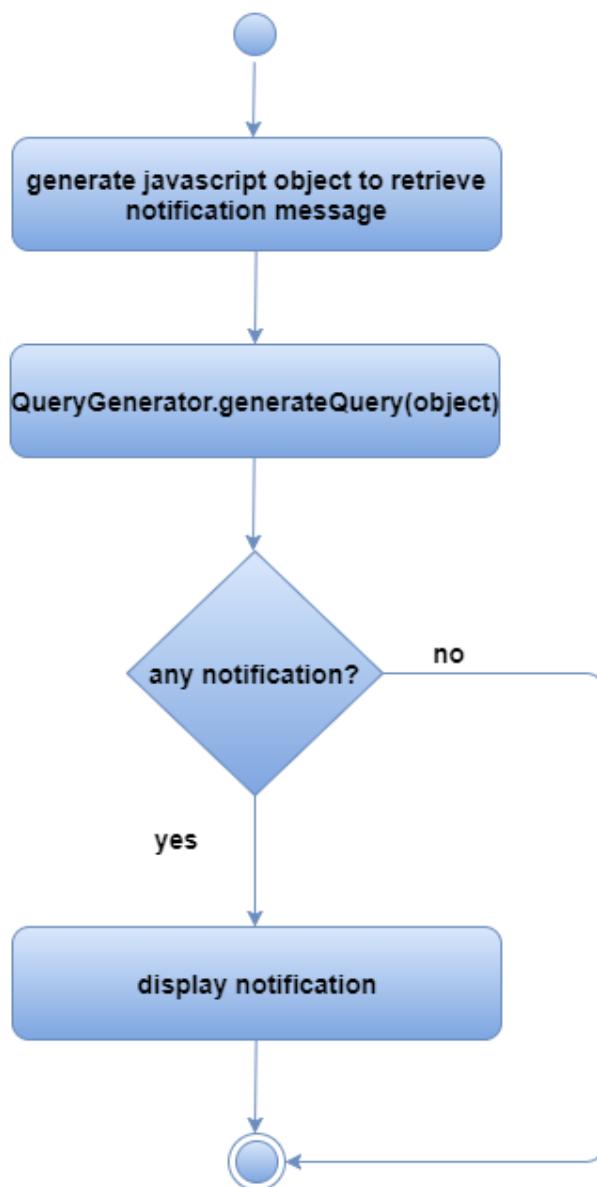


Figure 103 `getNotification()` of Student

`viewCourseDetails(courseID): courseInformation`

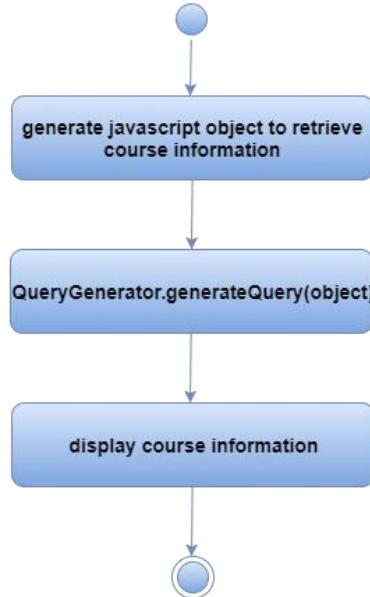


Figure 104 `viewCourseDetails()` of Student

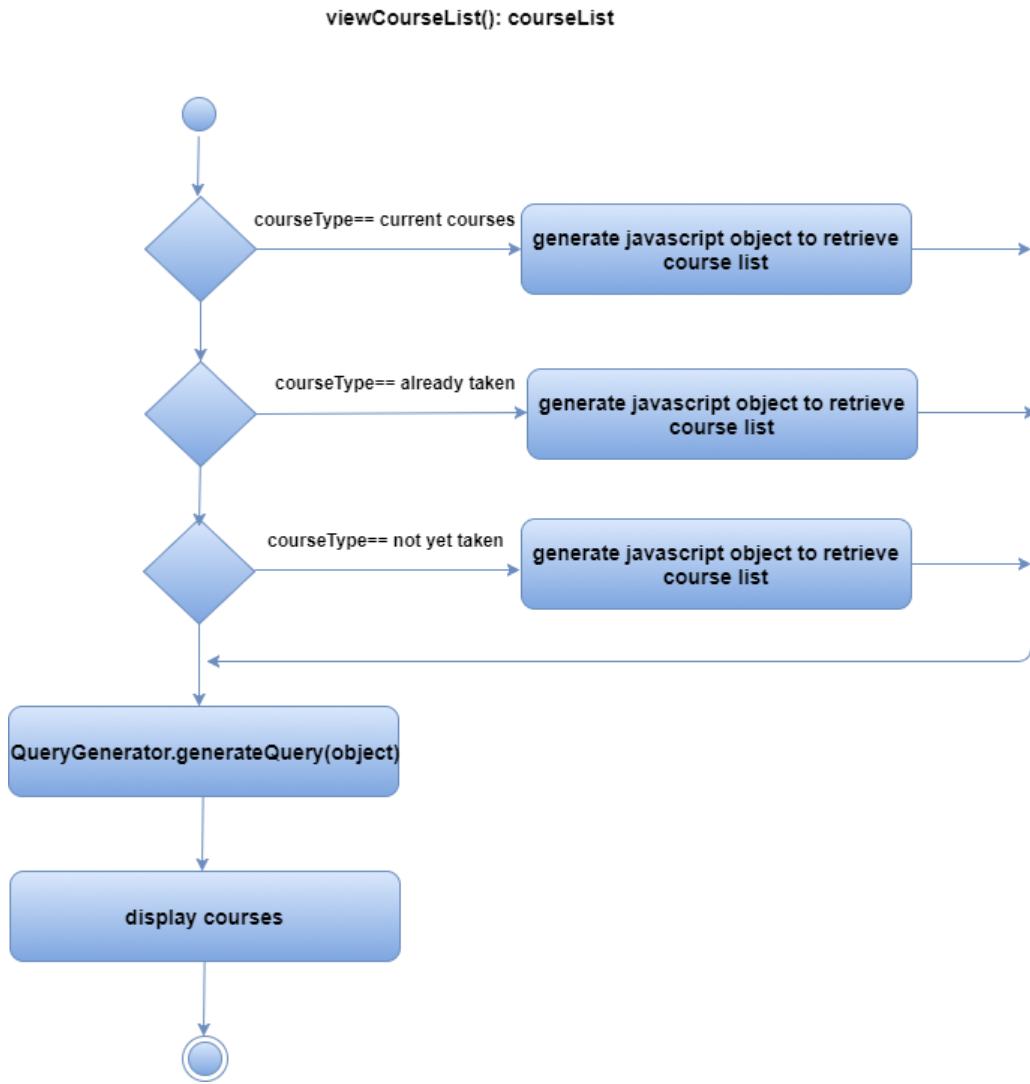


Figure 105 viewCourseList() of Student

Figure \*-\* show UML activity diagrams for ReportCard class.

`generatePdf(): progressReport`

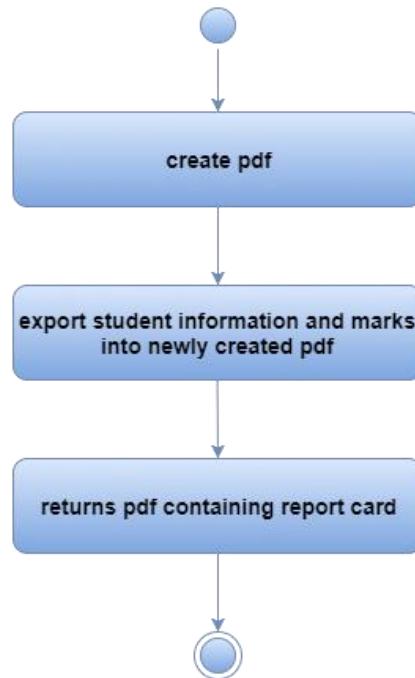


Figure 106 `generatePdf()` of ReportCard

`getMarks(): marks`

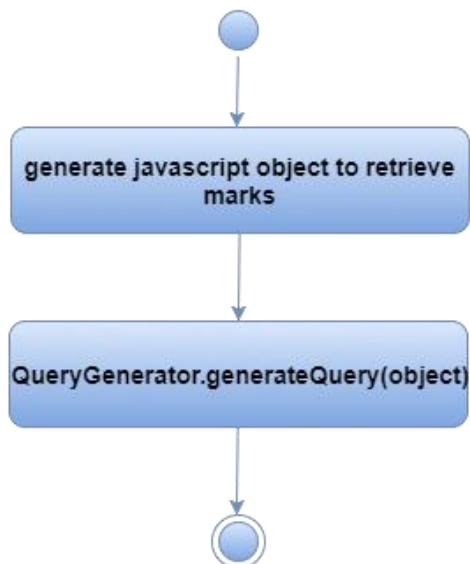


Figure 107 `getMarks()` of ReportCard

Figure \*-\* show UML activity diagrams for ExamController class.

```
addApplicationTimeRange(timeRange): void
```

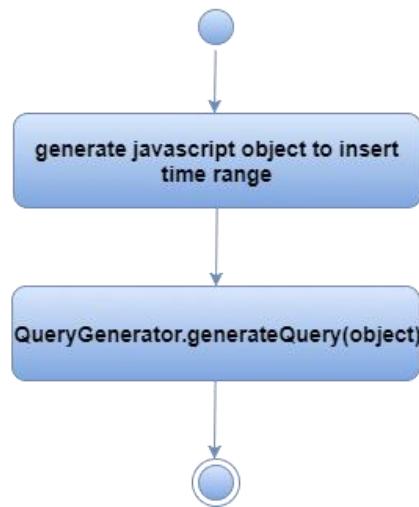


Figure 108 `addApplicationTimeRange()` of Exam Controller

```
addPaymentTimeRange(timeRange): void
```

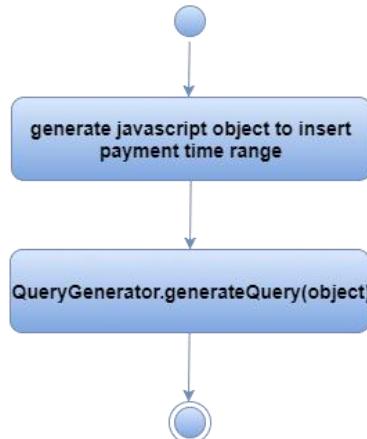


Figure 109 `addPaymentTimeRange()` of Exam Controller

```
addScrutinizationTimeRange(timeRange): void
```

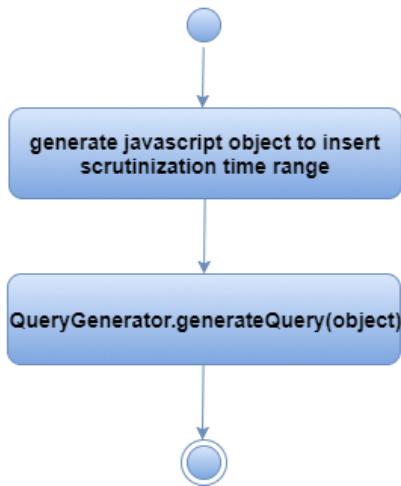


Figure 110 addScrutinizationTimeRange() of Exam Controller

```
endAdmissionSession(): void
```

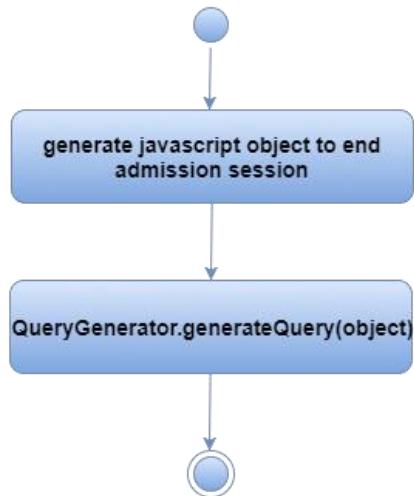


Figure 111 endAdmissionSession() of ExamController

`lockVivaMarks(): void`

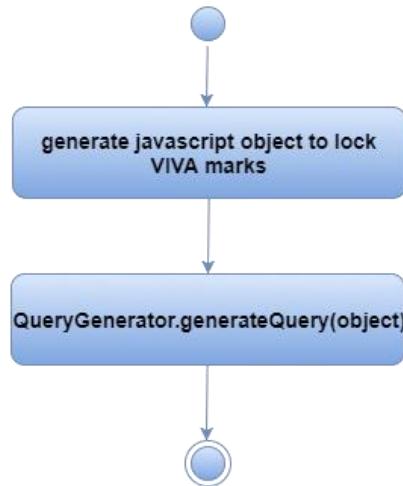


Figure 112 lockVivaMarks() of ExamController

`lockWrittenMarks(): void`

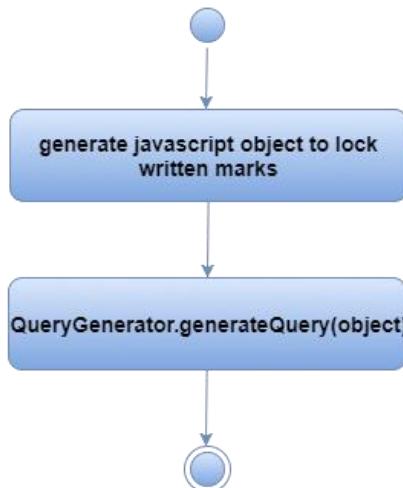


Figure 113 lockWrittenMarks() of ExamController

**updateAvailabilityOfAdmitCard(): void**

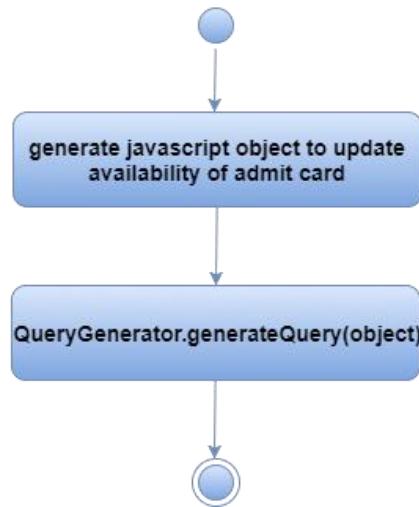


Figure 114 `updateAvailabilityOfAdmitCard()` of ExamController

**updateEnrollmentState(): void**

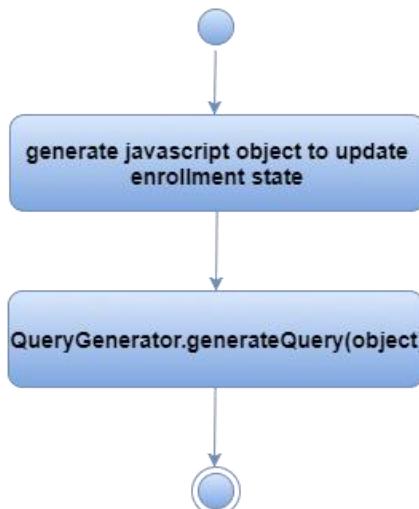


Figure 115 `updateEnrollmentState()` of ExamController

`updateVivaSchedule(): void`

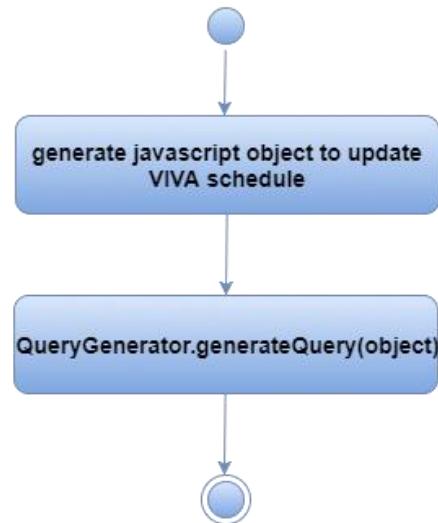


Figure 116 updateVivaSchedule() of ExamController

`updateWrittenMarks(): void`

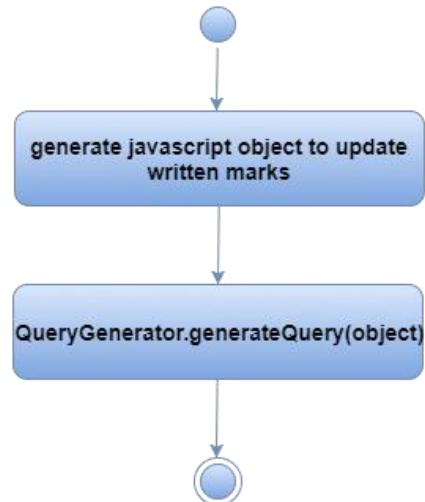


Figure 117 updateWrittenMarks() of ExamController

```
viewListOfApplicants(): void
```

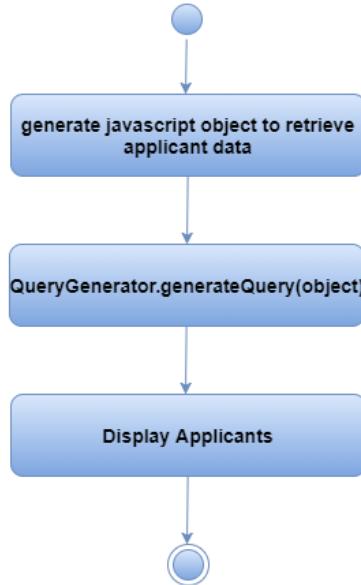


Figure 118 `viewListOfApplicants()` of ExamController

```
viewListOfEligibleCandidatesAfterFullResult(): List<Applicants>
```

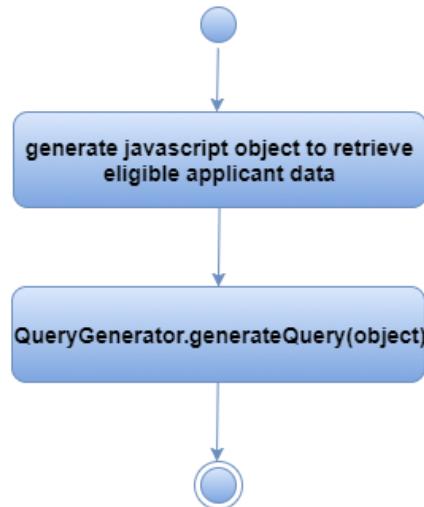


Figure 119 `viewListOfEligibleCandidateAfterFullResult()` of ExamController

`viewListOfEnrolledCandidates(): List<Applicants>`

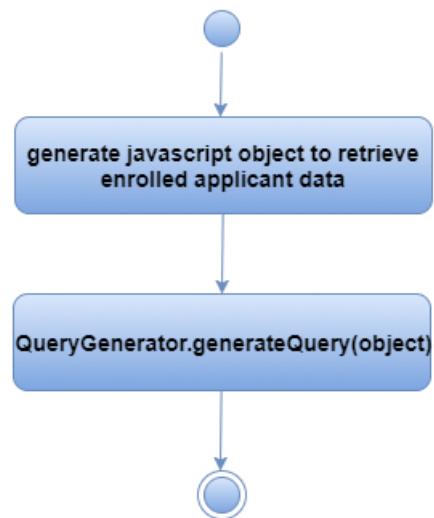


Figure 120 `viewListOfEnrolledCandidates()` of ExamController

Figure \*-\* show UML activity diagrams for RoomHandler class.

`addNewRoom(roomNumber, capacity): void`

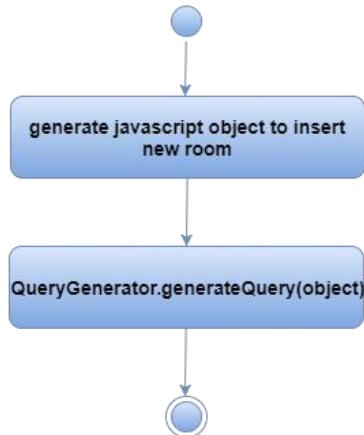


Figure 121 `addNewRoom()` of RoomHandler

**generateSeatPlan(): seatPlan**

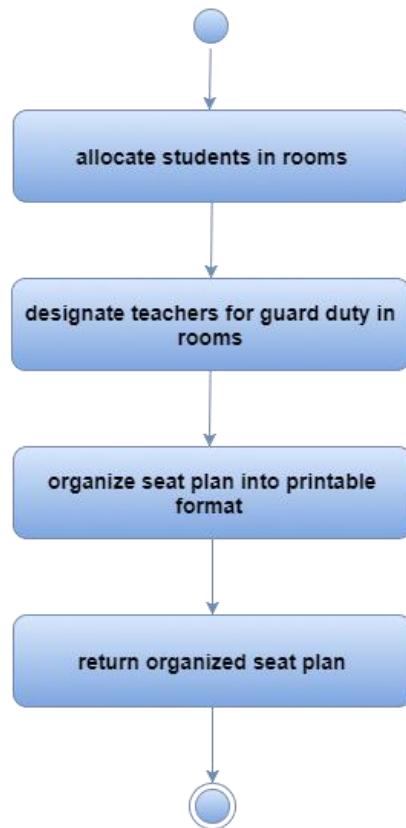


Figure 122 generateSeatPlan() of RoomHandler

**showListOfRoom(): roomList**

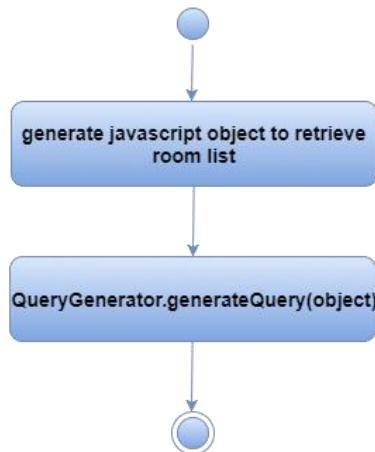


Figure 123 showListOfRoom() of RoomHandler

`updateRoomCapacity(roomNumber, capacity): void`

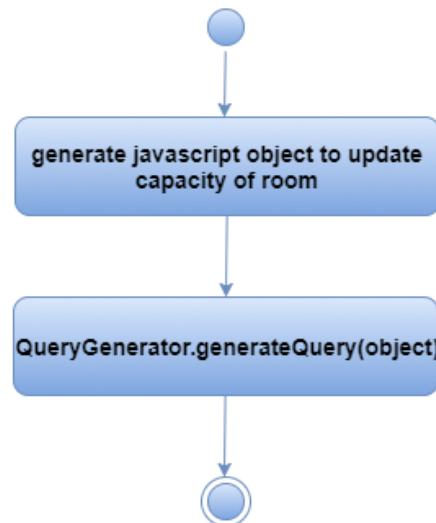


Figure 124 `updateRoomCapacity()` of RoomHandler

Figure \*-\* show UML activity diagrams for Scrutinizer class.

`viewFilteredApplications(listOfStudents):void`

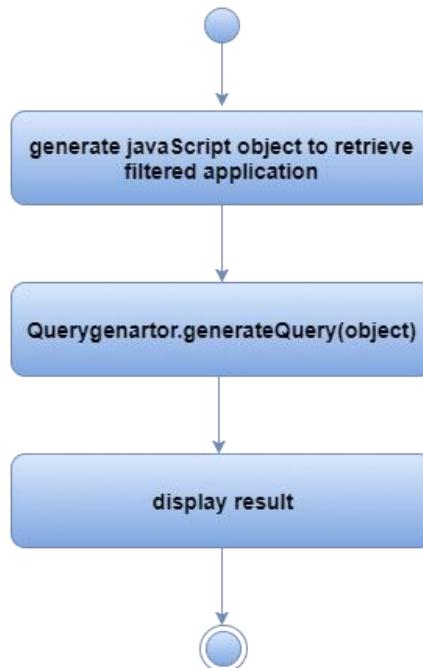


Figure 125 `viewFilteredApplications()` of Scrutinizer

```
searchByApplicationID(ApplicationID): void
```

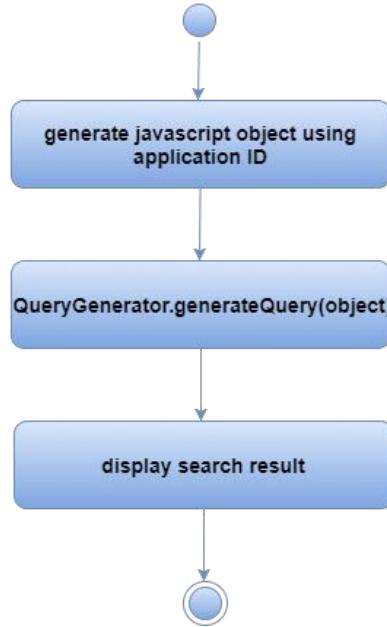


Figure 126 searchByApplicationID () of Scrutinizer

```
updateStatusOfEligibility(ApplicationID):void
```

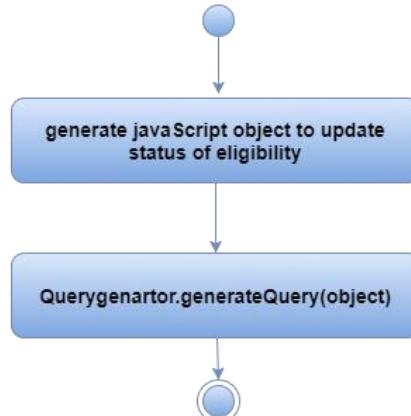


Figure 127 updateStatusOfEligibility() of Scrutinizer

Figure \*-\* show UML activity diagrams for RollGenerator class.

`assignRollToEligibleApplicants (): void`

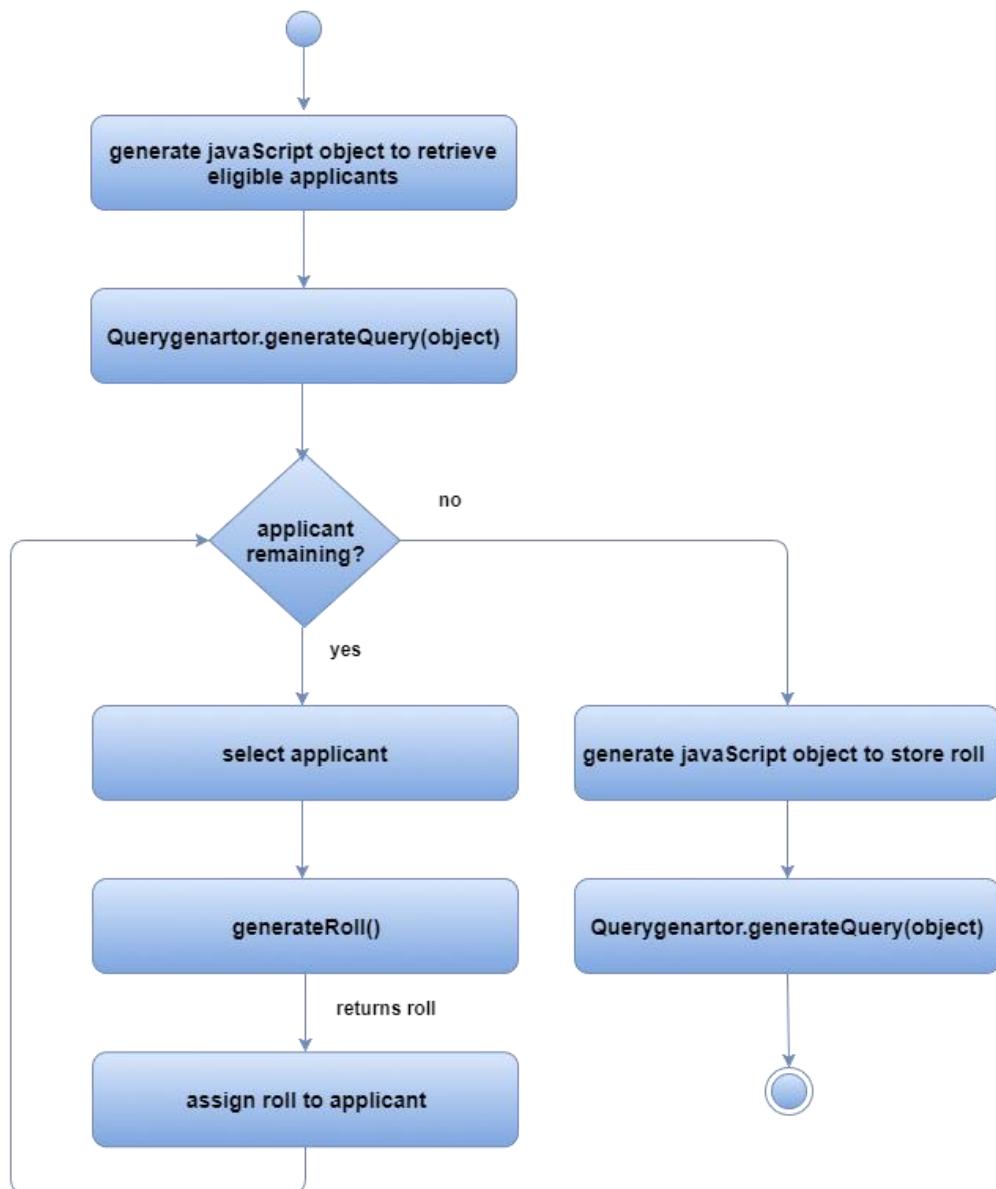


Figure 128 `assignRollToEligibleApplicants()` of `RollGenerator`

`generateRoll():string`

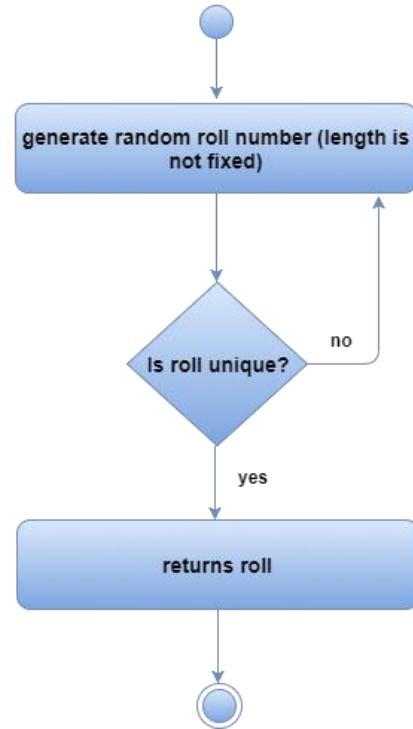


Figure 129 `generateRoll()` of RollGenerator

Figure \*-\* show UML activity diagrams for Receptionist class.

`updateStatusOfPayment():void`

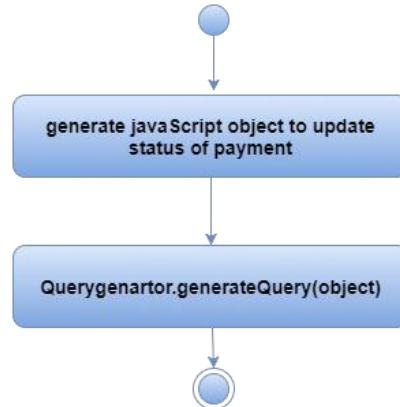


Figure 130 `updateStatusOfPayment()` of Receptionist

```
viewApplicantInformation(ApplicationID):void
```

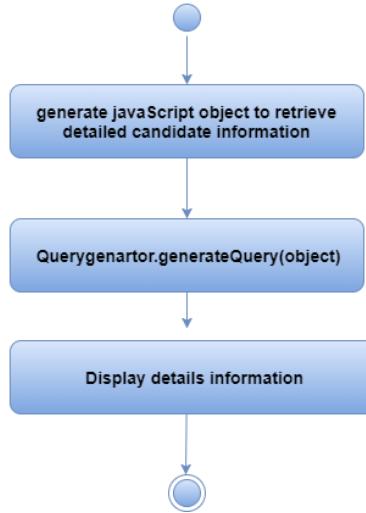


Figure 131 viewApplicantInformation() of Receptionist

### 3.4 Describe Persistent Data Sources and Identify the Classes required to Manage Them

For PGDIT Automation System we will use MySQL Database as data source and DatabaseHandler class is required to manage the database (shown in figure \*).



Figure 132 DatabaseHandler class

### 3.5 Develop and Elaborate Behavioral Representations for a Class or Component

During component-level design, it is sometimes necessary to model the behavior of a design class. For this purpose state diagram can be used. A state diagram is an illustration of the states an object can attain as well as the transitions between those states

Figure \* shows state diagram of the Applicant class

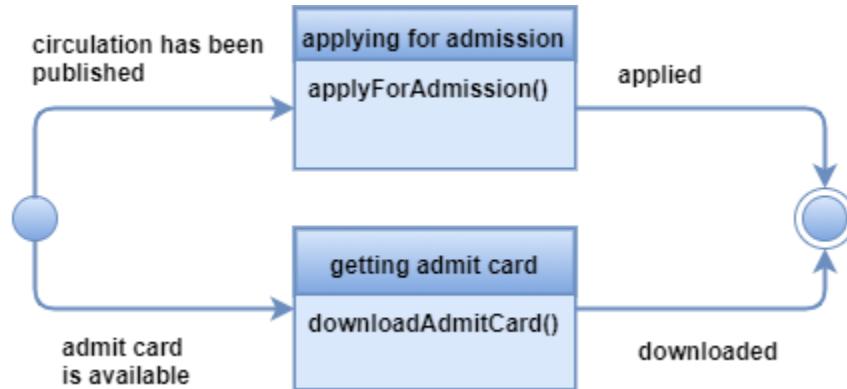


Figure 133 State Diagram of the Applicant class

Figure \* shows state diagram of the Authentication class

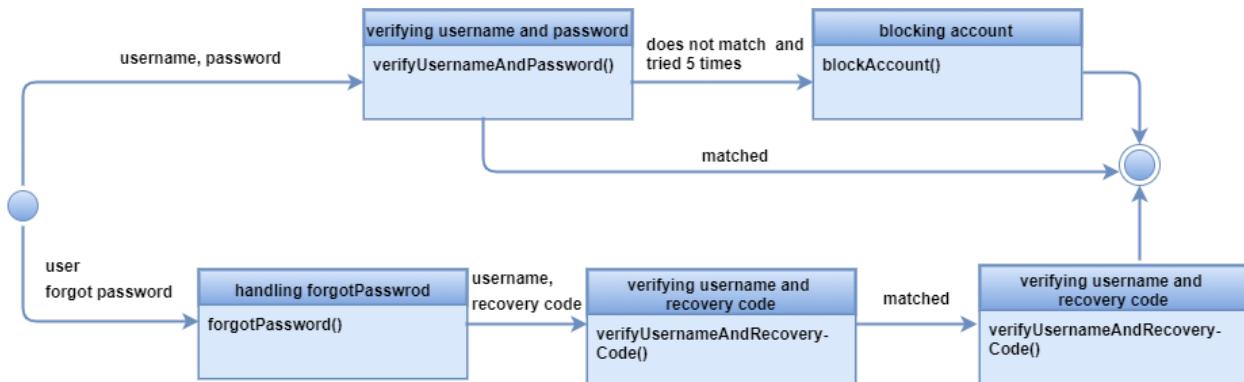


Figure 134 State Diagram of the Authentication class

Figure \* shows state diagram of the User class

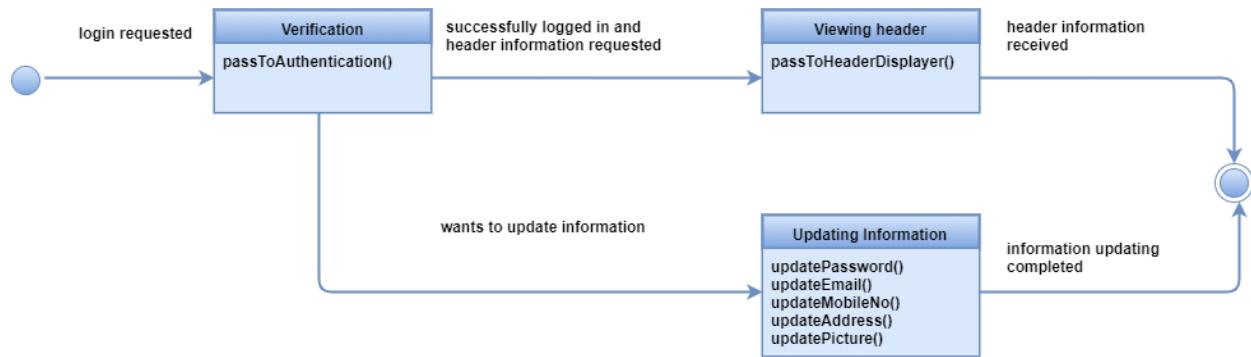


Figure 135 state diagram of the Authentication class

Figure \* shows state diagram of the HeaderDisplayer class

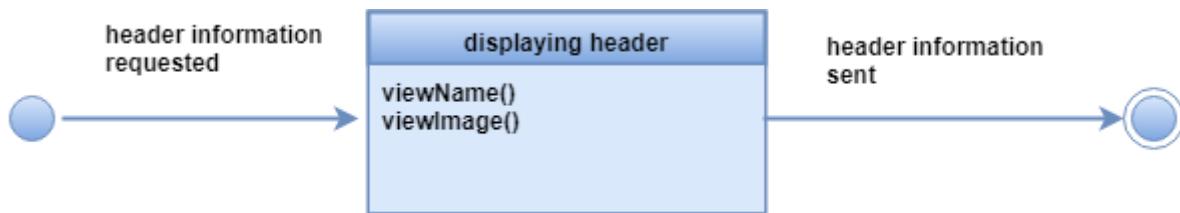


Figure 136 State Diagram of the HeaderDisplayer class

Figure \* shows state diagram of the ProgramChairperson class

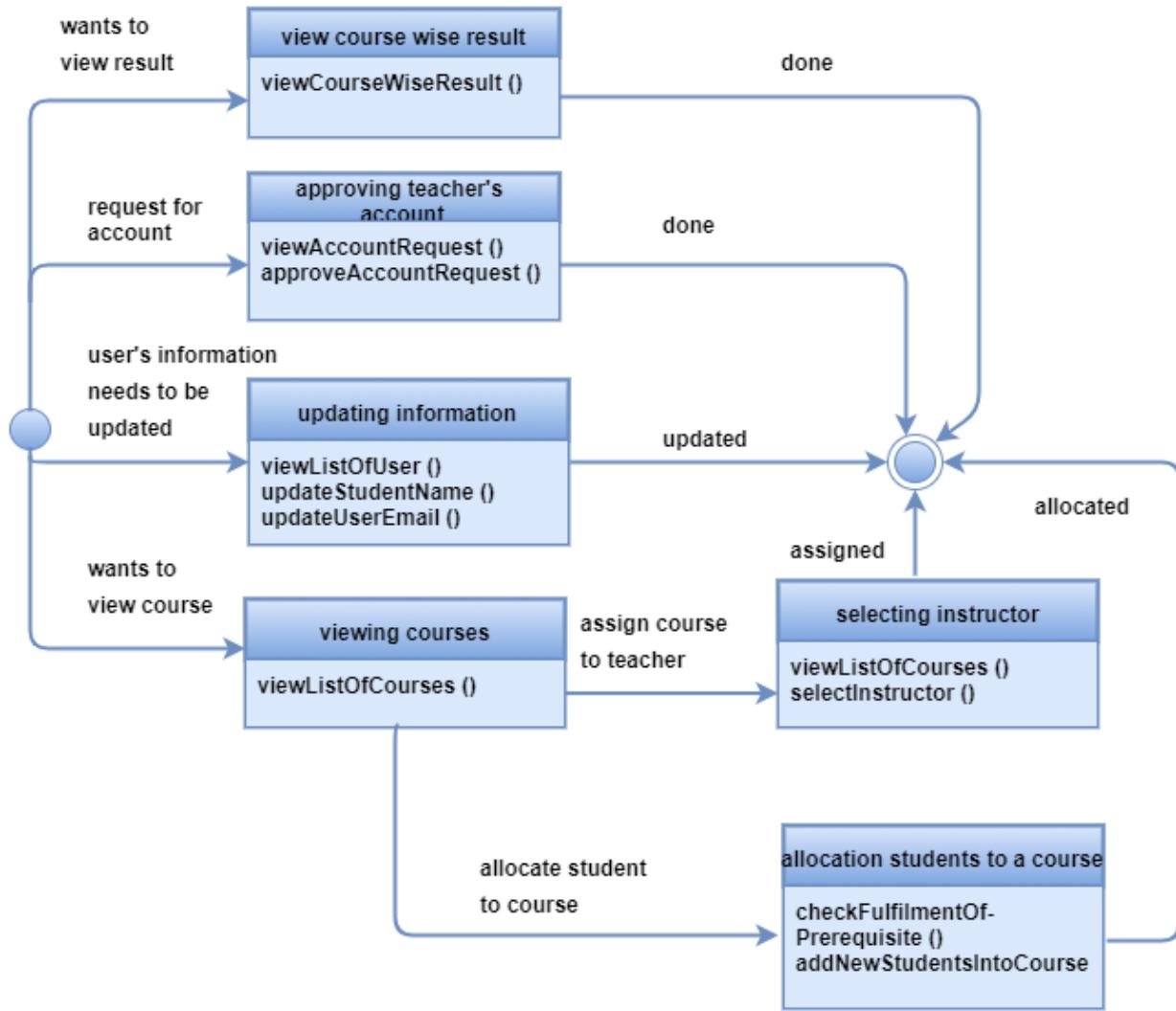


Figure 137 State Diagram of the ProgramChairperson class

Figure \* shows state diagram of the ExamController class

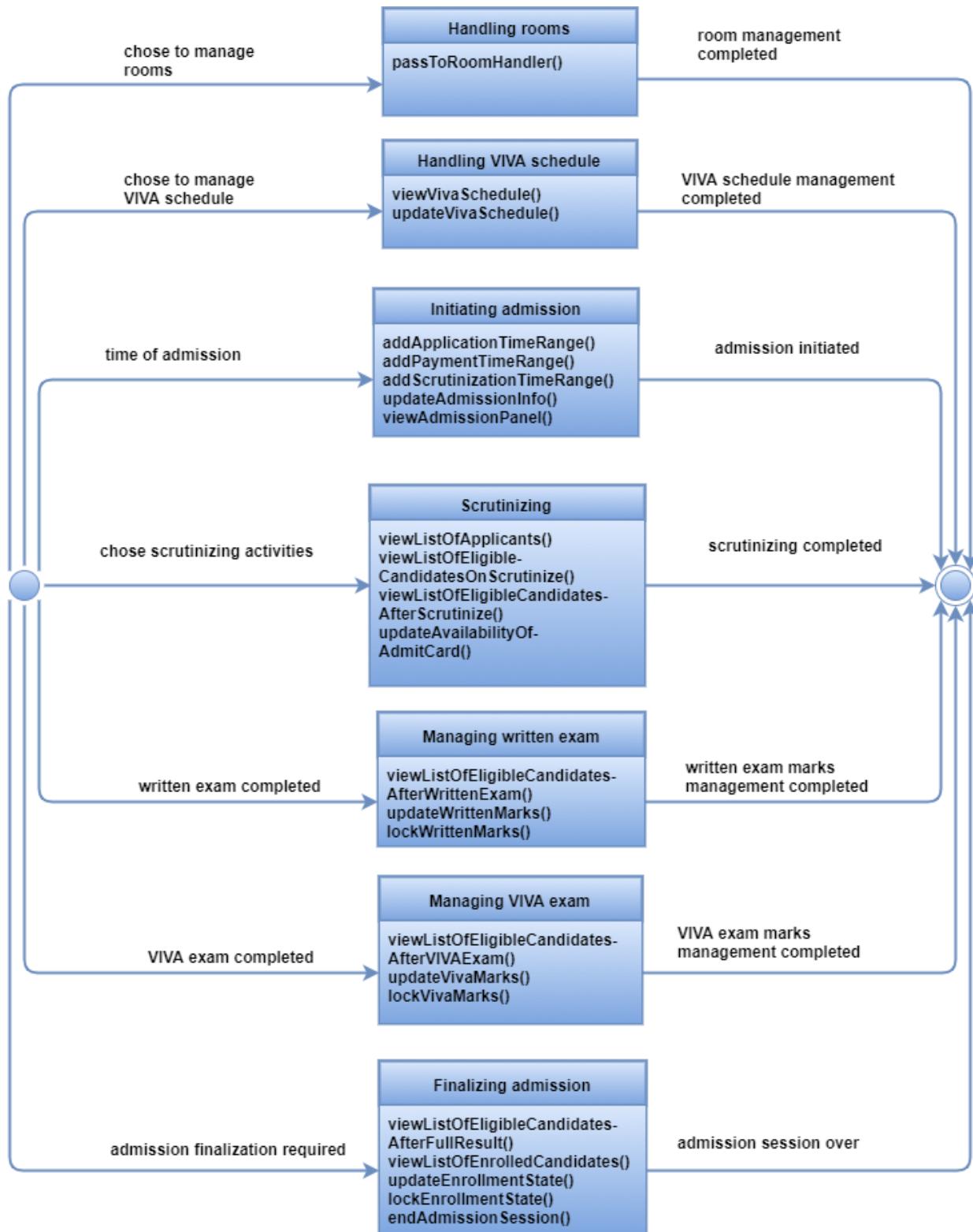


Figure 138 State Diagram of the ExamController class

Figure \* shows state diagram of the RoomHandler class

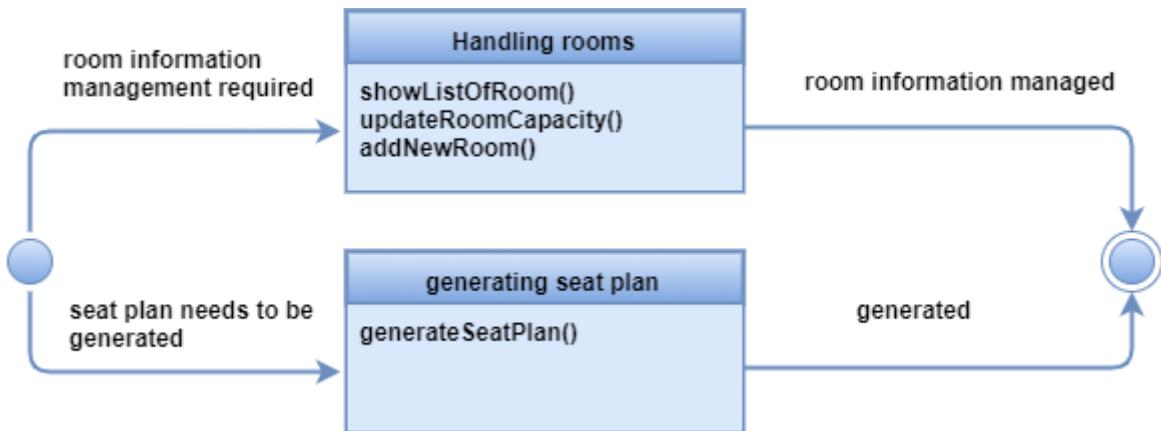


Figure 139 State Diagram of the RoomHandler class

Figure \* shows state diagram of the Receptionist class

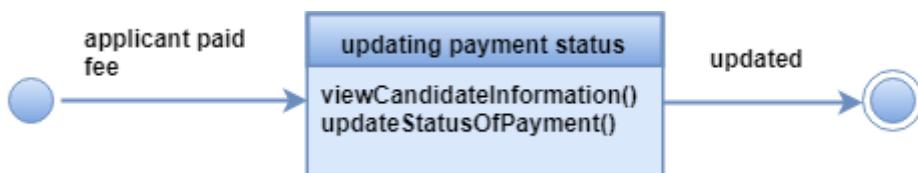


Figure 140 State Diagram of the Receptionist class

Figure \* shows state diagram of the Scrutinizer class

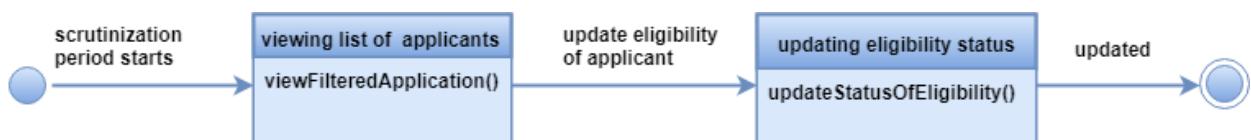


Figure 141 State Diagram of the Scrutinizer class

Figure \* shows state diagram of the RollGenerator class

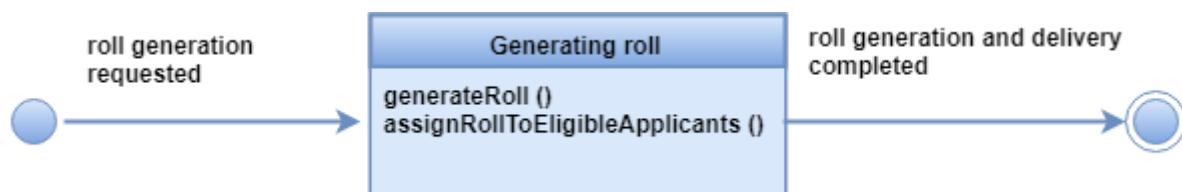


Figure 142 State Diagram of the RollGenerator class

Figure \* shows state diagram of the Teacher class

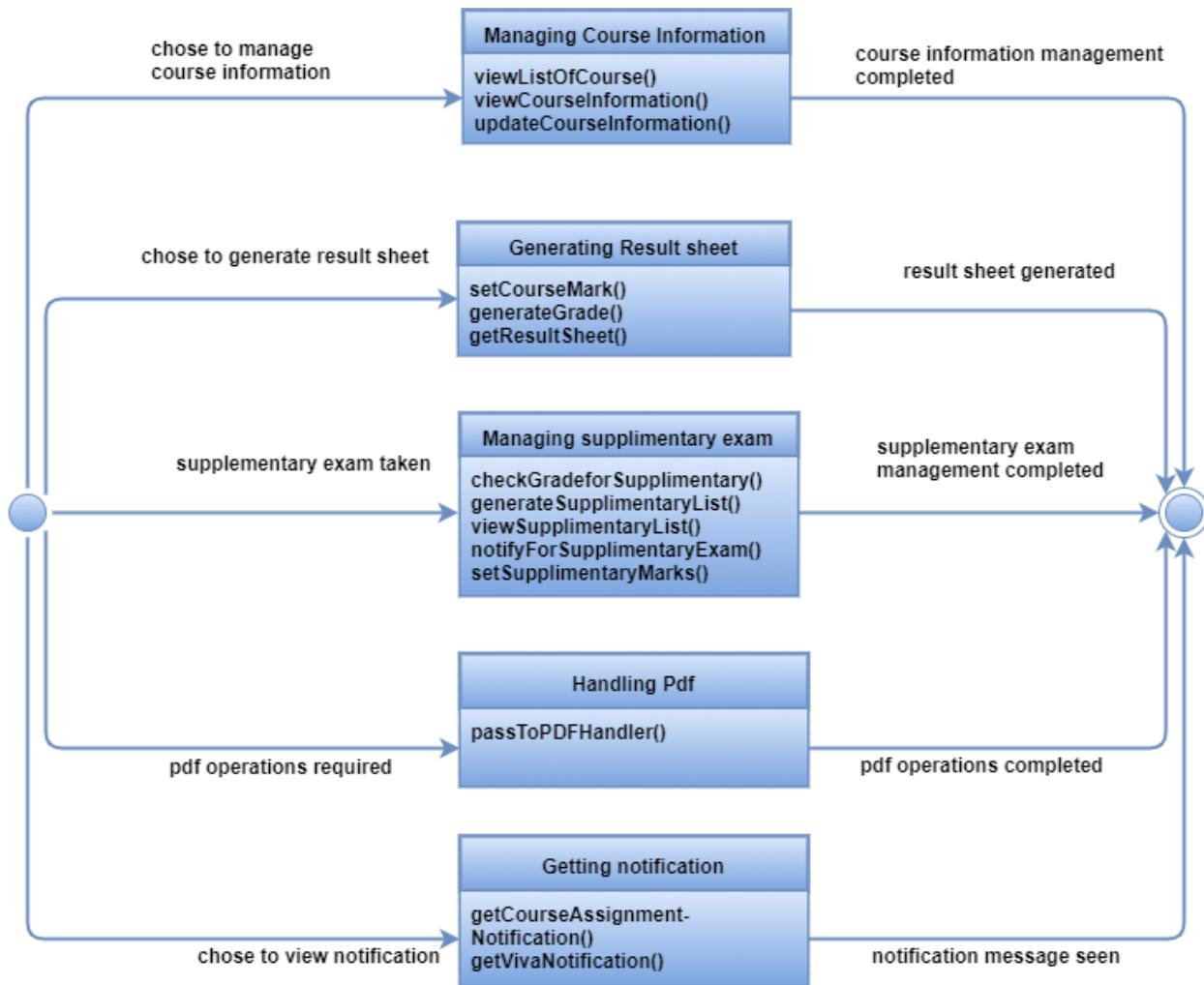


Figure 143 State Diagram of the Teacher class

Figure \* shows state diagram of the PDFHandler class

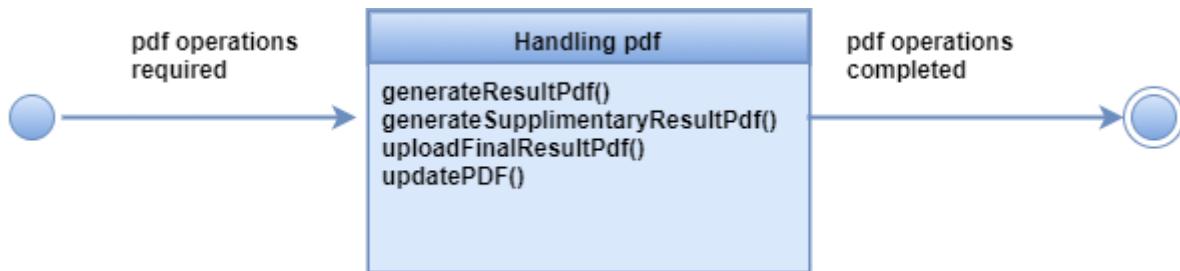


Figure 144 State Diagram of the PDFHandler class

Figure \* shows state diagram of the Student class

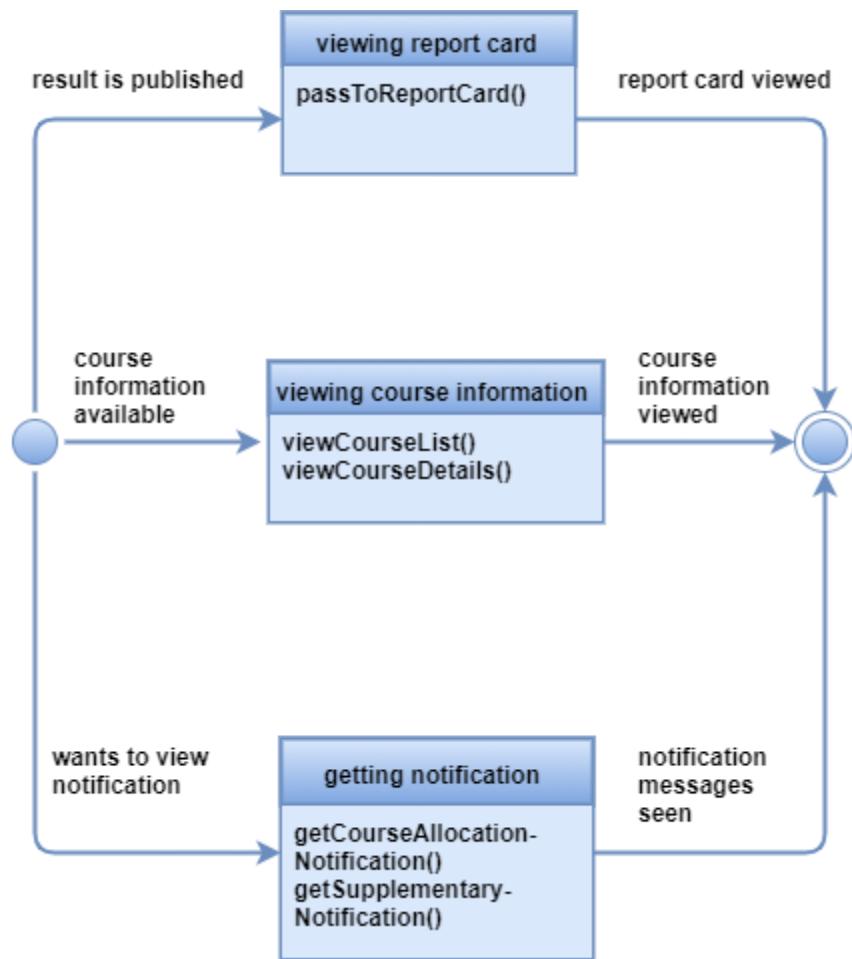


Figure 145 State Diagram of the Student class

Figure \* shows state diagram of the ReportCard class

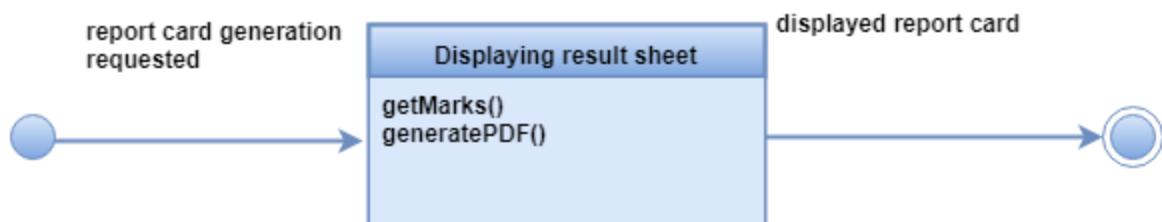


Figure 146 State Diagram of the ReportCard class

### 3.6 Elaborate Deployment Diagrams to Provide Additional Implementation Detail

Deployment diagrams are represented in descriptor form. In this form, major system functions are represented within the context of the computing environment that will house them.

Figure \* shows deployment diagram of PGDIT Automation System.

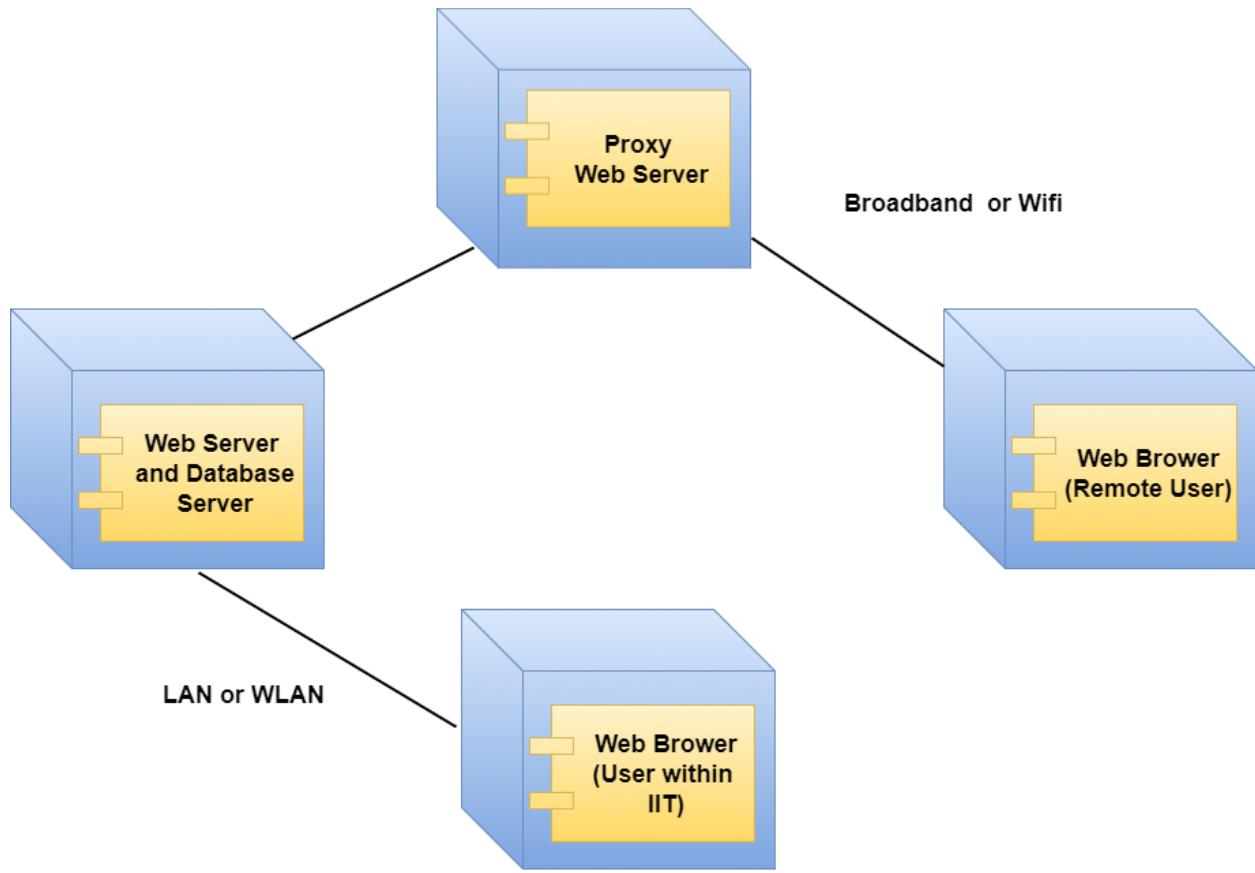


Figure 147 Deployment Diagram of PGDIT Automation System

## Chapter 4. User Interface Design

User interface design creates an effective communication medium between a human and a computer. User Interface (UI) Design focuses on anticipating what users might need to do and ensuring that the interface has elements that are easy to access, understand, and use to facilitate those actions.

### 4.1 Interface Analysis

We divide interface analysis into following parts:

1. User Analysis
2. Task Analysis

#### 4.1.1 User Analysis

User analysis helps us to know who the end users are, what is likely to motivate and please them, how they can be grouped into different user classes or profiles, what their mental models of the system are, and how the user interface must be characterized to meet their

needs. From the requirements specification, we have found the following users for PGDIT Automation System:

- Teacher
- Student
- Program Chairperson
- Applicant
- Exam Controller
- Scrutinizer
- Receptionist

Table 2 Analyzing Teacher

Teacher	
Work type	Trained Professionals
Educational Level	Masters
Learning capability	Training
Skills	Above Average
Age	25-65
Domain expert	Yes
Application expert	No
Office hour	Normal
Frequency of use	Occasionally
Consequence of a mistake	Low
General computer experience	Yes
Want to know about technology that sits behind the interface	No

Table 3 Analyzing Student

Student	
Educational Level	Honors
Learning capability	Training
Skills	Average
Age	22-40
Application expert	No
Frequency of use	Occasionally
Consequence of a mistake	Low
General computer experience	Yes
Want to know about technology that sits behind the interface	No

Table 4 Analyzing Program Chairperson

Program Chairperson	
Work type	Trained Professionals
Educational Level	Masters
Learning capability	Training
Skills	Above Average
Age	25-65
Domain expert	Yes
Application expert	No
Office hour	Normal
Frequency of use	Occasionally
Consequence of a mistake	Low
General computer experience	Yes
Want to know about technology that sits behind the interface	No

Table 5 Analyzing Applicant

Applicant	
Educational Level	Honors
Learning capability	Training
Skills	Average
Age	22-40
Application expert	No
Frequency of use	Occasionally
Consequence of a mistake	Low
General computer experience	Yes
Want to know about technology that sits behind the interface	No

Table 6 Analyzing Exam Controller

Exam Controller	
Work type	Trained Professionals
Educational Level	Masters
Learning capability	Training
Skills	Above Average
Age	25-65
Domain expert	Yes
Application expert	No
Office hour	Normal

Frequency of use	Occasionally
Consequence of a mistake	High
General computer experience	Yes
Want to know about technology that sits behind the interface	No

Table 7 Analyzing Scrutinizer

Scrutinizer	
Work type	Trained Professionals
Educational Level	Masters
Learning capability	Training
Skills	Above Average
Age	25-65
Domain expert	Yes
Application expert	No
Office hour	Normal
Frequency of use	Occasionally
Consequence of a mistake	Medium
General computer experience	Yes
Want to know about technology that sits behind the interface	No

Table 8 Analyzing Receptionist

Receptionist	
Work type	Clerical
Educational Level	Masters
Learning capability	Training
Skills	Average
Age	25-50
Domain expert	Yes
Application expert	No
Office hour	Normal
Frequency of use	Occasionally
Consequence of a mistake	Low
General computer experience	Yes
Want to know about technology that sits behind the interface	No

#### 4.1.2 Task Analysis

In this step we identify and analyze the tasks of every users separately.

 Applicants: The tasks of applicants are given below:

1. Apply for admission

Goal: Apply for admission by filling up application form.

Precondition: Submission of application has been activated.

Sub-tasks:

- I. fill up form
- II. receive application ID

2. Download admit card

Goal: Download admit card of admission test.

Precondition:

- I. eligible for admission test
- II. application ID and date of birth are valid
- III. admit card is available

Sub-tasks:

- I. provide application ID and date of birth
- II. get admit card

 Program Chairperson: Program Chairperson has following tasks:

1. Create student account

Goal: Create account for new students.

Precondition: Must be logged in as program chairperson.

Sub-tasks:

- I. view list of students who paid admission fee
- II. create account

2. Enroll student in course

Goal: Allocate course to a student.

Precondition: Must be logged in as program chairperson.

Sub-tasks:

- I. check students who fulfill prerequisite
- II. select students
- III. notify selected students

3. Assign course to teacher

**Goal:** Select a teacher as instructor of a course.

**Precondition:** Must be logged in as program chairperson.

**Sub-tasks:**

- I. view list of teachers
- II. select teacher
- III. notify teacher

**4. Update user's email**

**Goal:** Update any user's email address.

**Precondition:** Must be logged in as program chairperson.

**Sub-tasks:**

- I. select user
- II. update email

**5. Update student's name**

**Goal:** Update any student's name.

**Precondition:** Must be logged in as program chairperson.

**Sub-tasks:**

- I. select student
- II. update name

**6. Approve teacher's account**

**Goal:** Approve teacher's account request.

**Precondition:** Must be logged in as program chairperson.

**Sub-tasks:**

- I. check account requests
- II. approve requests

**7. View result**

**Goal:** View course wise marks or result of a particular student.

**Precondition:** Must be logged in as program chairperson.

**Sub-tasks:**

- I. select category
- II. view result

 **Exam Controller:** The tasks of Exam Controller are given below:

**1. Introduce new admission**

**Goal:** Setup new admission session and publish circular.

**Precondition:** Must be logged in.

**Sub-tasks:**

- I. provide timeline for applying online

- II. provide timeline for payment
  - III. provide timeline for scrutinization
- 2. Manage room information
  - Goal: Keep information about rooms of the institution up-to-date.
  - Precondition: Must be logged in.
  - Sub-tasks:
    - I. view information of all rooms
    - II. change capacity of a room
    - III. add new room
- 3. Initiate written admission exam
  - Goal: Generate seat plan and make admit cards available for download.
  - Precondition: Must be logged in.
  - Sub-tasks:
    - I. generate seat plan for written exam
    - II. turn off scrutinizing process
    - III. make admit cards available for download
- 4. End written admission exam
  - Goal: Prepare result of written admission exam.
  - Precondition: Must be logged in.
  - Sub-tasks:
    - I. provide marks of written exam
    - II. lock marks of written exam
- 5. Initiate VIVA admission exam
  - Goal: Generate seat plan and VIVA schedule.
  - Precondition: Must be logged in.
  - Sub-tasks:
    - I. generate seat plan for VIVA exam
    - II. generate VIVA schedule
    - III. notify teachers about VIVA shift
- 6. End VIVA admission exam
  - Goal: Prepare result of VIVA admission exam.
  - Precondition: Must be logged in.
  - Sub-tasks:
    - I. provide marks of VIVA exam
    - II. lock marks of VIVA exam
- 7. Finalize admission exam
  - Goal: Finalize admission examination and complete enrollment of students.
  - Precondition: Must be logged in.

Sub-tasks:

- I. publish merit and waiting list
- II. enroll students
- III. end enrollment session
- IV. end admission session

8. Finalize term examination result

Goal: Finalize term examination results.

Precondition: Must be logged in.

Sub-tasks:

- I. lock marks providing session
- II. upload final result for Exam Controller and Program Chairperson

 Teacher: Teacher has following tasks:

1. Edit Course Information

Goal: Change existing information or introduce new information about the course.

Precondition: Must be logged in.

Sub-tasks:

- I. view course list
- II. select Course
- III. change information
- IV. lock course information

2. Generate Result Sheet

Goal: Generate result sheet of students with marks obtained in the respective teacher's course.

Precondition: Must be logged in.

Sub-tasks:

- I. input marks obtained by individual students
- II. generate Result sheet
- III. lock result sheet

3. Take Supplementary Exam

Goal: Provide supplementary Exam results

Precondition: Must be logged in. Exam result sheet must be available.

Sub-tasks:

- I. set-up a criteria for supplementary exam.
- II. view supplementary candidates' list
- III. notify candidates
- IV. input marks of supplementary exam
- V. generate supplementary result sheet
- VI. lock supplementary result sheet

**4. Upload Result Sheet PDF**

**Goal:** Upload pdf of result sheet in respective course and make it available for download.

**Precondition:**

- I. Must be logged in.
- II. Exam result sheet must be available.
- III. Supplementary result sheet are to be available if any student had been required to appear.

**Sub-tasks:**

- I. generate Result Sheet PDF
- II. generate Supplementary Result Sheet PDF
- III. upload PDF to the server
- IV. make PDF's available for download

 **Student:** Student has following tasks:

**1. View result sheet**

**Goal:** View report card of semesters completed so far.

**Precondition:** Must be logged in.

**Sub-tasks:**

- I. invoke generation of report card
- II. download report card

**2. View course information**

**Goal:** View information of all courses in the program.

**Precondition:** Must be logged in.

**Sub-tasks:**

- I. view list of courses
- II. view information of a specific course

**3. Get notifications**

**Goal:** Get course allocation and supplementary notification

**Precondition:** Must be logged in

**Sub-tasks:**

- I. get course allocation notification
- II. get supplementary notification

 **Receptionist:** The tasks of Receptionist are given below:

**1. Update payment status**

**Goal:** Update payment status of applications

**Precondition:** Must be logged in

**Sub-tasks:**

- I. retrieve applications
  - II. update payment status of applications
2. View candidate information  
Goal: View information of an application  
Precondition: Must be logged in  
Sub-tasks:
- I. retrieve application
  - II. view details of application

## 4.2 Interface Design Steps

Interface design steps are shown in figure \*.

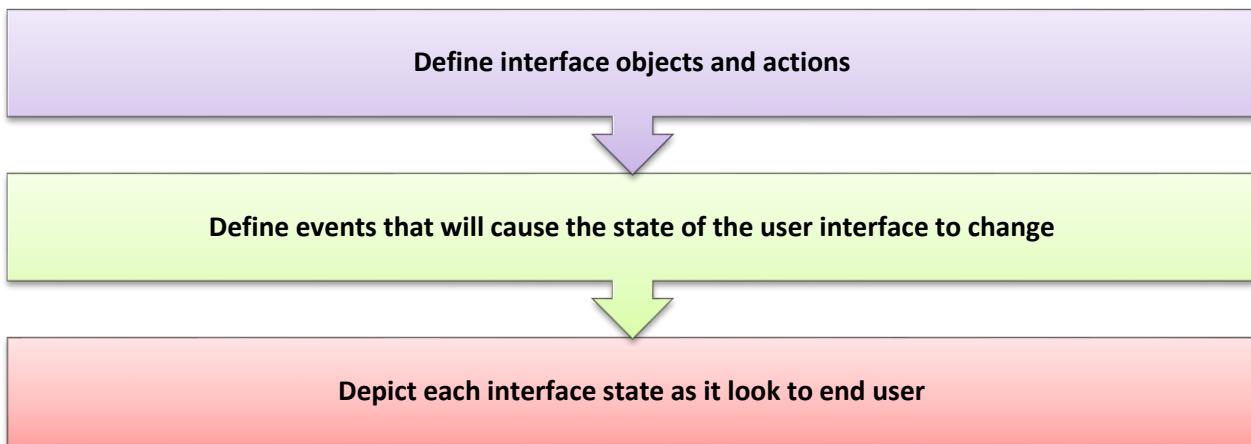
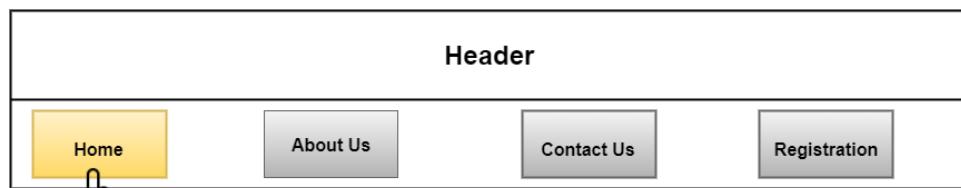


Figure 148 Interface Design Steps

### 4.2.1 Define Interface Objects and Actions

We identified following objects and actions for the user interface.

- ⊕ External
  - Home



**Sign-in**

Slider Photos

Username

Password

**Login**

[Forgot Password?](#)

Notice



o About us



**About Us**

.....  
.....  
.....

Image



o Contact us

**Header**

Home      About Us      **Contact Us**      Registration



**Contact Us**

Institute of Information Technology



Please Fill up the form below

Name	<input type="text"/>
Email	<input type="email"/>
Subject	<input type="text"/>
Message	<input type="text"/>

**Submit**

**Footer**

- Registration

**Header**

Home      About Us      Contact Us      **Registration**



**Sign Up**

Full Name   
 Username   
 Email Address   
 Teacher's Registration No.   
 Father's Name   
 Mother's Name   
 Address   
 Password   
 Confirm Password

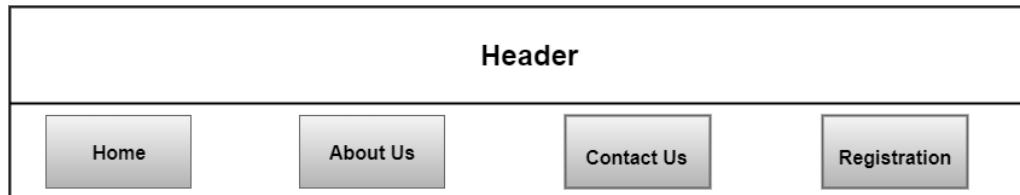
**Send**



Upload Photo

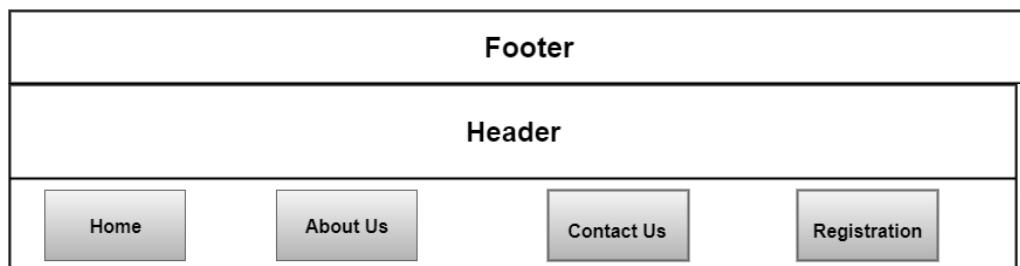
**Footer**

- Forgot Password



## Forgot Password?

**Enter Your Username**

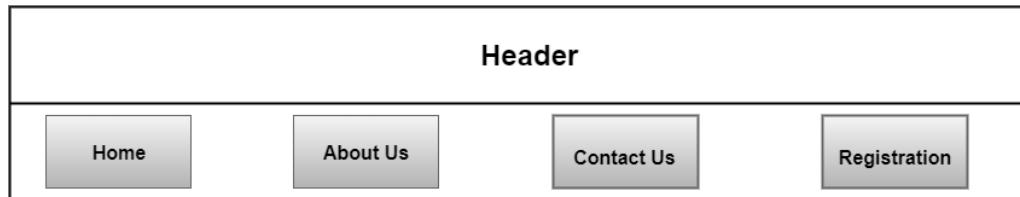


## Reset Password

Username

Recovery Code





### New Password

New Password

Confirm Password

**Submit**



- Apply For Admission

Header																
<a href="#">Home</a>	<a href="#">About Us</a>	<a href="#">Contact Us</a>	<a href="#">Registration</a>													
<b>Personal Information</b>																
Name of Applicant:	<input type="text"/>															
Father's Name:	<input type="text"/>															
Mother's Name:	<input type="text"/>															
Email Address:	<input type="email"/>															
Mobile Number:	<input type="number"/>															
Date of Birth:	<input type="date"/>															
Nationality:	<input type="text"/>															
Permanent Address:	<input type="text"/>															
<b>Academic Career</b>																
<input type="radio"/> SSC/O-Level	<table border="1"> <thead> <tr> <th>School Name</th> <th>Board</th> <th>Group</th> <th>Year</th> <th>Marks/GPA</th> </tr> </thead> <tbody> <tr> <td><input type="text"/></td> <td><input type="text"/></td> <td><input type="text"/></td> <td><input type="text"/></td> <td><input type="text"/></td> </tr> </tbody> </table>				School Name	Board	Group	Year	Marks/GPA	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>		
School Name	Board	Group	Year	Marks/GPA												
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>												
<input type="radio"/> HSC/A-Level	<table border="1"> <thead> <tr> <th>School Name</th> <th>Board</th> <th>Group</th> <th>Year</th> <th>Marks/GPA</th> </tr> </thead> <tbody> <tr> <td><input type="text"/></td> <td><input type="text"/></td> <td><input type="text"/></td> <td><input type="text"/></td> <td><input type="text"/></td> </tr> </tbody> </table>				School Name	Board	Group	Year	Marks/GPA	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>		
School Name	Board	Group	Year	Marks/GPA												
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>												
<input type="radio"/> Bachelor Level	<table border="1"> <thead> <tr> <th>University Name</th> <th>Department</th> <th>Year</th> <th>Division/CGPA</th> </tr> </thead> <tbody> <tr> <td><input type="text"/></td> <td><input type="text"/></td> <td><input type="text"/></td> <td><input type="text"/></td> </tr> </tbody> </table>				University Name	Department	Year	Division/CGPA	<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>				
University Name	Department	Year	Division/CGPA													
<input type="text"/>	<input type="text"/>	<input type="text"/>	<input type="text"/>													
<input type="radio"/> Marks In Mathematics	<table border="1"> <thead> <tr> <th>Examination</th> <th>Marks Obtained (Math)</th> <th>% of Obtained Marks (Math)</th> <th>Did it include Algebra or Statistics?</th> </tr> </thead> <tbody> <tr> <td>HSC/A-Level</td> <td><input type="text"/></td> <td><input type="text"/></td> <td><input type="radio"/> Yes <input type="radio"/> No</td> </tr> <tr> <td>Bachelor Level</td> <td><input type="text"/></td> <td><input type="text"/></td> <td><input type="radio"/> Yes <input type="radio"/> No</td> </tr> </tbody> </table>				Examination	Marks Obtained (Math)	% of Obtained Marks (Math)	Did it include Algebra or Statistics?	HSC/A-Level	<input type="text"/>	<input type="text"/>	<input type="radio"/> Yes <input type="radio"/> No	Bachelor Level	<input type="text"/>	<input type="text"/>	<input type="radio"/> Yes <input type="radio"/> No
Examination	Marks Obtained (Math)	% of Obtained Marks (Math)	Did it include Algebra or Statistics?													
HSC/A-Level	<input type="text"/>	<input type="text"/>	<input type="radio"/> Yes <input type="radio"/> No													
Bachelor Level	<input type="text"/>	<input type="text"/>	<input type="radio"/> Yes <input type="radio"/> No													
 <input type="button" value="Upload Photo"/> <span style="float: right;"><input type="button" value="Submit"/></span>																
<b>Footer</b>																

- Download Admit card

<b>Header</b>			
Home	About Us	Contact Us	Registration
<b>Download Admit Card</b>			
Application Id		<input type="text"/>	
Date of Birth		<input type="text"/>	
<input type="button" value="Submit"/>			
<b>Footer</b>			

 Internal

- Program Chairperson
  - Profile

<b>Header</b>					
 <input type="button" value="Profile"/>	<input type="button" value="Course Management"/>	<input type="button" value="Account Requests"/>	<input type="button" value="View Users"/>	<input type="button" value="View Result"/>	<input type="button" value="Logout"/>
<div style="display: flex; align-items: center;">  <div style="margin-left: 20px;"> <b>User name</b>  <input type="text"/>  <span style="font-size: small;">(Change Photo)</span> </div> <div style="margin-left: 20px;"> <span style="font-size: small;">User type</span> <input type="text"/>  <span style="font-size: small;">Email address</span> <input type="text"/>  <span style="font-size: small;">Father's name</span> <input type="text"/>  <span style="font-size: small;">Mother's name</span> <input type="text"/>  <span style="font-size: small;">Mobile no</span> <input type="number"/>  <span style="font-size: small;">Address</span> <input type="text"/> </div> </div>					
<b>Footer</b>					



User name                                  text

Full Name                                  text

Email address                              text

Father's name                              text

Mother's name                              text

Mobile no                                  number

Address                                      text

Password                                  \*\*\*\*

**Save**

- Course Management

**Header**

Profile	Course Management	Account Requests	View Users	View Result	Logout
---------	-------------------	------------------	------------	-------------	--------



Course Name

Teacher's Name

Total Students

Update Teacher

View Students

Course Name

Teacher's Name

Total Students

Update Teacher

View Students

**Footer**

**Course name (Course Code)**

Select Teacher

**Course name (Course Code)**

Search

Roll no	Student name
text	text
text	text

X

## Students who fulfilled Prerequisite

Search:

Selected	Roll no	Student name
<input type="checkbox"/>	text	text
<input type="checkbox"/>	text	text

**Allocate to Course**

- Account Requests

**Header**

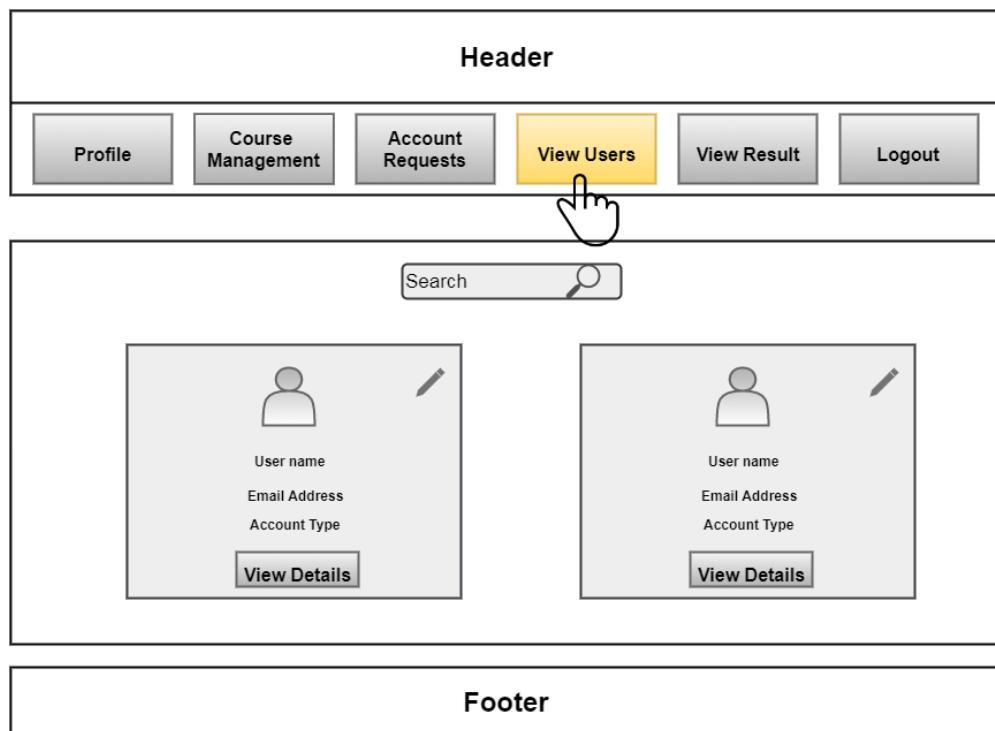
Profile	Course Management	Account Requests	View Users	View Result	Logout
---------	-------------------	------------------	------------	-------------	--------



**Footer**

Full name	Email Address	Registration No		
text	text	text	<b>approve</b>	<b>decline</b>
text	text	text	<b>approve</b>	<b>decline</b>

- View Users



**User name**

User type	text
Email address	text
Father's name	text
Mother's name	text
Mobile no	number
Address	text

**X**



User name	text
Full Name	text
Email address	text
Father's name	text
Mother's name	text
Mobile no	number
Address	text
<b>Save</b>	

- View Result
  - Course wise Result

Header																				
Profile	Course Management	Account Requests	View Users	<b>View Result</b>	Logout															
<a href="#">Course wise Result</a> <a href="#">Student wise Result</a> 																				
<table border="1"> <tr> <td>Select Course</td> <td>Course Name ▾</td> <td>Submit</td> </tr> <tr> <td colspan="3" style="text-align: center;">Search <input type="text"/> </td> </tr> <tr> <td>Roll number</td> <td>Student Name</td> <td>Obtained Marks</td> </tr> <tr> <td>text</td> <td>text</td> <td>marks</td> </tr> <tr> <td>text</td> <td>text</td> <td>marks</td> </tr> </table>						Select Course	Course Name ▾	Submit	Search <input type="text"/> 			Roll number	Student Name	Obtained Marks	text	text	marks	text	text	marks
Select Course	Course Name ▾	Submit																		
Search <input type="text"/> 																				
Roll number	Student Name	Obtained Marks																		
text	text	marks																		
text	text	marks																		
Footer																				

- Student wise Result

**Header**

Profile	Course Management	Notifications	View Users	View Result	Logout
---------	-------------------	---------------	------------	-------------	--------

Course wise Result  
Student wise Result

---

Select Student	Student username <input type="text"/>	<input type="button" value="Submit"/>
Search <input style="width: 100px; height: 20px; border: 1px solid black; border-radius: 10px;" type="text"/>		
Course Name	Obtained Marks	Obtained Grade
text	marks	grade
text	marks	grade

**Footer**

- Teacher
  - Profile

**Header**

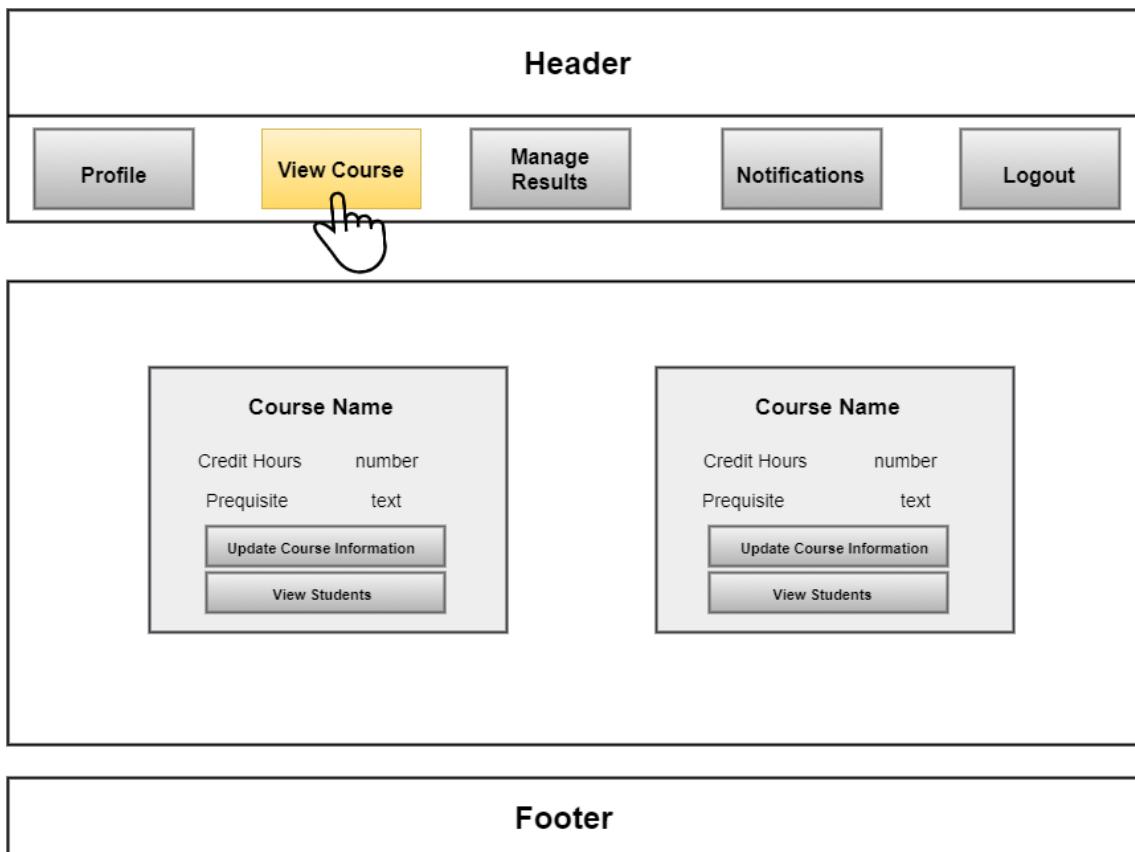
Profile	View Course	Manage Results	Notifications	Logout
---------	-------------	----------------	---------------	--------

---

 <input type="button" value="Change Photo"/>	User name  User type text Email address text Father's name text Mother's name text Mobile no number Address text
---	---

**Footer**

- View Course



A search interface for course and student information. It features a search bar with a magnifying glass icon and a large 'X' button in the top right corner. Below the search bar, there are two input fields: 'Roll no' and 'Student name', each with a corresponding 'text' label below it.

X

**Course name (Course Code)**

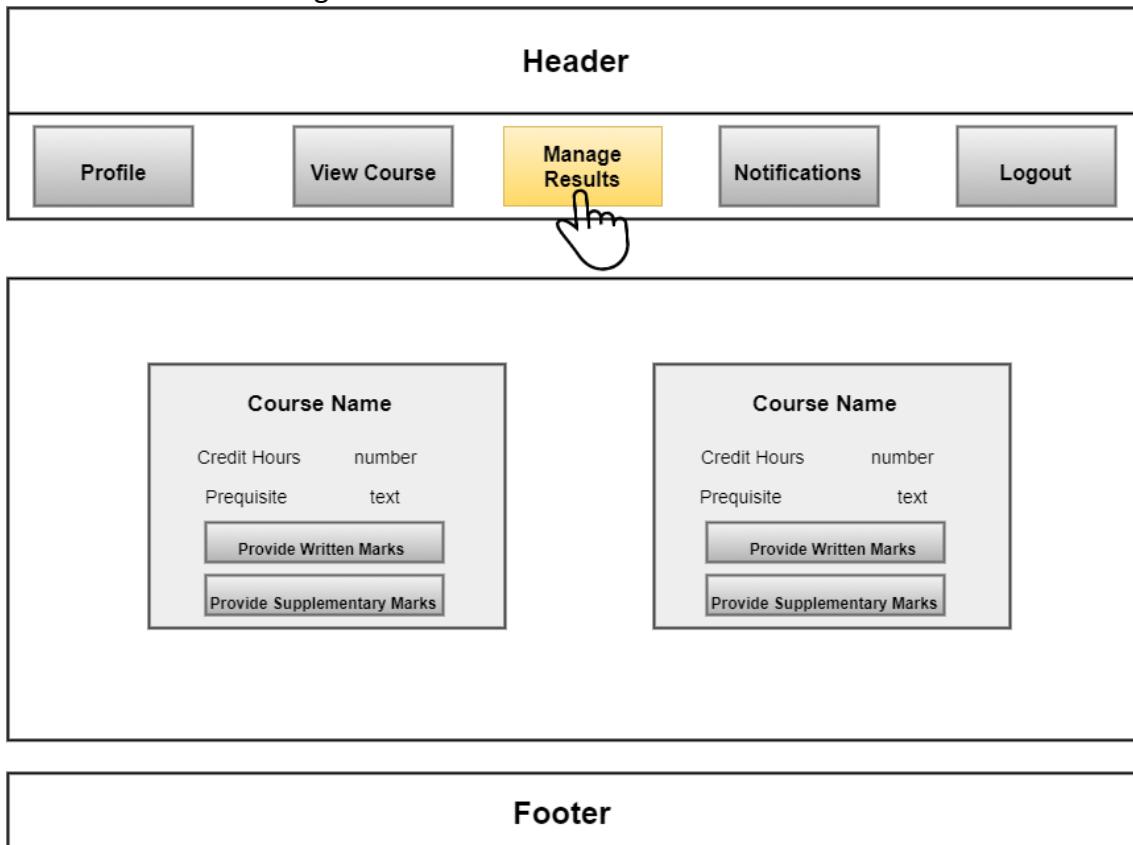
Search

Roll no                  Student name

text                  text

text                  text

- Manage Results



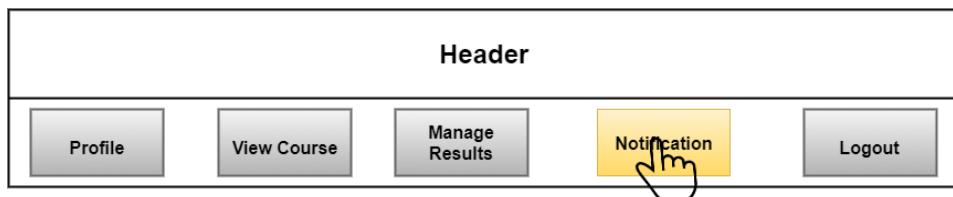
**Course (Course Code)**

Search:

Roll no	Student name	Marks
text	text	<input type="text"/>
text	text	<input type="text"/>

**Save**

- Notifications

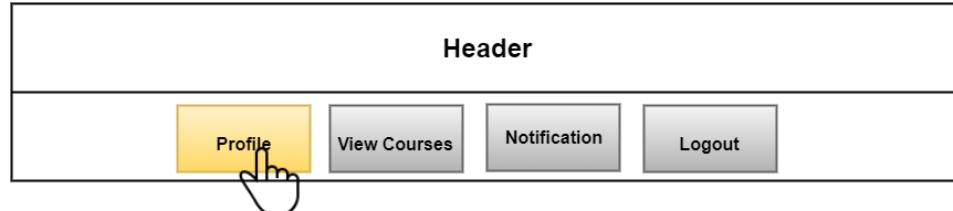


### Viva Schedule

mmm dd,yyyy    HH:MM  
**Viva Duty**  
Room no. - XXX



- Student
  - Profile



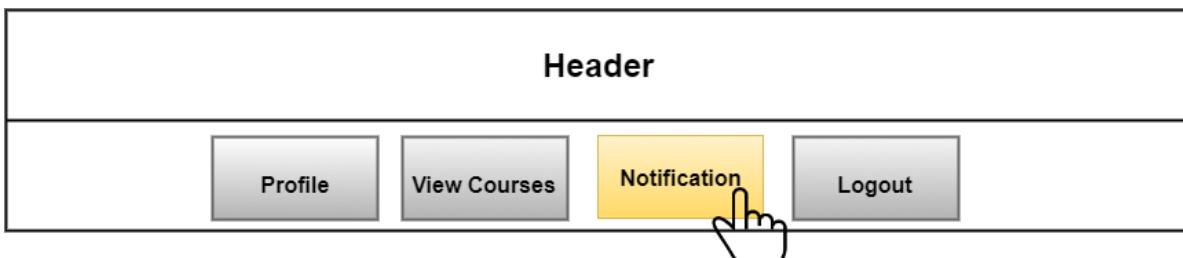


User name ✎

User type text  
Email address text  
Father's name text  
Mother's name text  
Mobile no number  
Address text



- Download Report Card



**Report Card Notification**

---

Term final result has been published. Please collect your report card by clicking the link below.

**Download Report Card**

**Footer**

- View Courses

**Header**

Profile   **View Courses**   Notification   Logout

**Operating System Concepts & UNIX OS**

---

Course Code: PGD106  
Instructor: Nawshin Nawar  
Credit Hour: 03  
Prerequisite: None

**Internet Programming**

---

Course Code: PGD102  
Instructor: Sumon Ahmed  
Credit Hour: 03  
Prerequisite: None

**Footer**

- Exam Controller
  - Profile

Header					
<b>Profile</b>	Manage Admission Initiation	Manage Admission Exam	Room Management	Term Result	More
 <b>Change Photo</b>	<b>User name</b> 				
	User type	text			
	Email address	text			
	Father's name	text			
	Mother's name	text			
	Mobile no	number			
	Address	text			

- Manage Admission Initiation

### Header

[Profile](#)[Manage Admission Initiation](#)[Manage Admission Exam](#)[Room Management](#)[Term Result](#)[More](#)



Application Submission Start: 11-10-2017 Application Submission End: 19-10-2017  
Application Payment Start: 20-10-2017 Application Payment End: 30-10-2017  
Scrutinizing Start: 1-11-2017 Scrutinizing End: 11-11-2017  
Admission Payment Start: 12-10-2017 Admission Payment End: 21-11-2017

X

Update

### Footer

X

**Introduce New Admission Session**

Application Submission Start:

Application Submission End:

Application Payment Start:

Application Payment End:

Scrutinizing Start:

Scrutinizing End:

Admission Payment Start:

Admission Payment End:

Submit

- Manage Admission Exam

Header					
Profile	Manage Admission Initiation	Manage Admission Exam	Room Management	Term Result	More
 <b>Written Examination Completed</b> <hr/> <a href="#" style="background-color: #4f81bd; color: white; padding: 5px 10px; text-decoration: none;">Upload Marks</a>					
Footer					

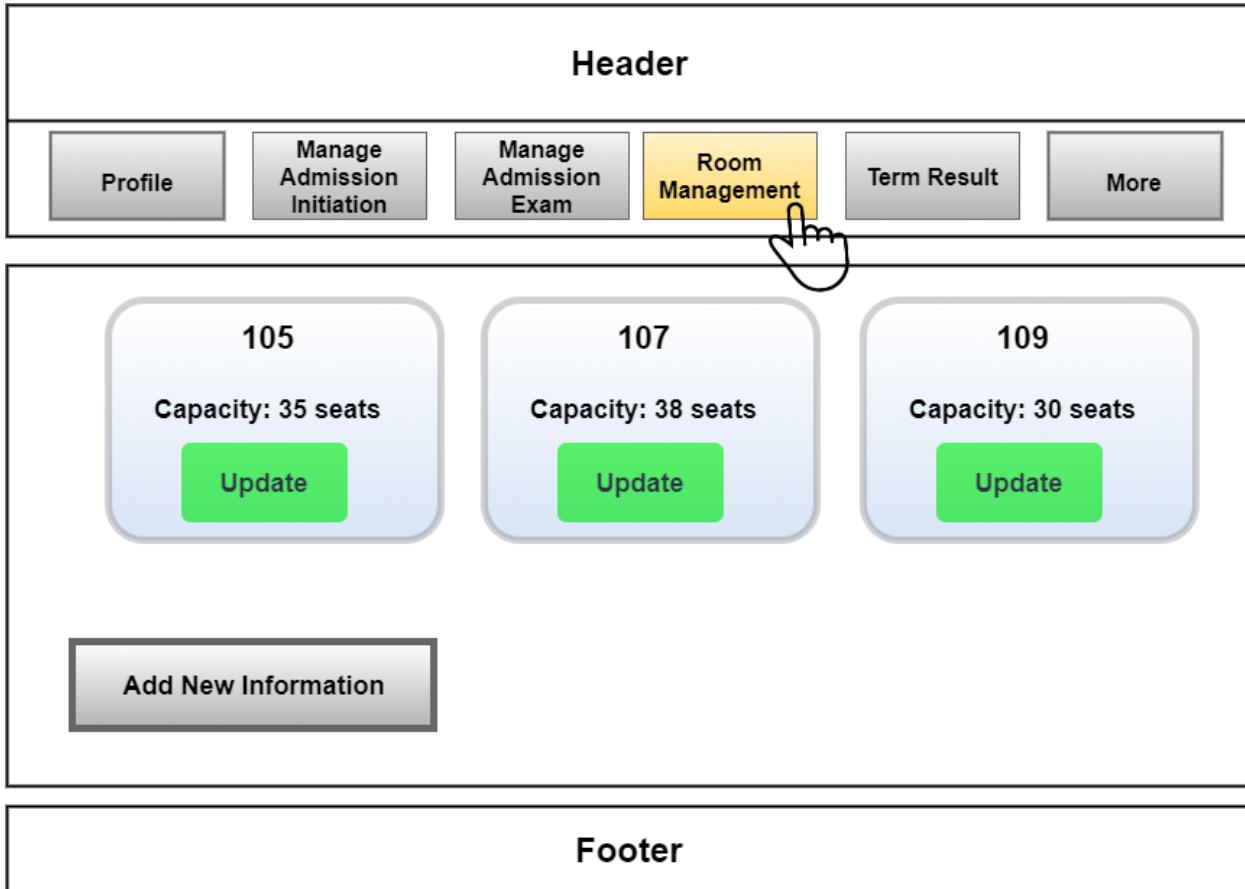
X

### Written Exam Marks

Search:

Application ID	Marks
text	<input type="text"/>
text	<input type="text"/>

- Room Management



- Term Result

**Header**

[Profile](#) [Manage Admission Initiation](#) [Manage Admission Exam](#) [Room Management](#) [Term Result](#) [More](#)



**Term Result Management**

---

Marks uploading facility is running...

Turn Off

Publish Result

**Footer**

- Update Signature

## Header

Profile

Manage  
Admission  
Initiation

Manage  
Admission  
Exam

Room  
Management

Term Result

More

Current Signature:

Satyaki Das

Update Signature

## Footer

<b>Header</b>					
<b>Profile</b>	<b>Manage Admission Initiation</b>	<b>Manage Admission Exam</b>	<b>Room Management</b>	<b>Term Result</b>	<b>More</b> 
<b>VIVA Examination</b> <hr/> <p>VIVA examination has been completed.</p> <b>Written Examination</b> <hr/> <p>Written examination has been completed.</p> <b>Admit Card</b> <hr/> <p>Admission payment completed for all applications.</p>					

**Footer**

- Receptionist
  - Profile

**Header**



Profile      View Applicants      Logout

---

User name ✎



**Change Photo**

User type	text
Email address	text
Father's name	text
Mother's name	text
Mobile no	number
Address	text

**Footer**

▪ View Applicants

**Header**



Profile      View Applicants      Logout

---

Search  🔍

Application ID number	Full Name text	Payment Status text	Options (Edit/Details)
number	text	text	<span style="border: 1px solid #ccc; border-radius: 15px; padding: 2px 10px; margin-right: 10px;">Edit</span> <span style="border: 1px solid #ccc; border-radius: 15px; padding: 2px 10px;">Details</span>

**Footer**

**Application ID (number)**

**Full Name** **Applicant Name**

**Payment Status**

**Save**

- Scrutinizer
  - View Applicants

Header			
<b>View Applicants</b>		Logout	
			
<b>Search</b> <input type="text" value="Search Site"/> 			
Application ID number	Full Name text	Eligibility Status text	Options (Edit/Details) <input type="button" value="Edit"/> <input type="button" value="Details"/>
number	text	text	<input type="button" value="Edit"/> <input type="button" value="Details"/>

Footer	
--------	--

The diagram illustrates a user interface for managing application details. It consists of a main rectangular container with a black border. In the top right corner of this container is a circular button containing a white 'X'. Inside the container, the text 'Application ID (number)' is centered at the top. Below it, there are two side-by-side input fields: 'Full Name' on the left and 'Applicant Name' on the right. Underneath these is a label 'Eligibility Status' followed by a dropdown menu with a downward-pointing arrow. At the bottom center of the container is a grey rectangular button with the word 'Save' in white capital letters.

4.2.2 Define Events that will Cause the State of the User Interface to Change  
External Object

#### Sign-in



#### Registration



#### Recover Password



#### Contact Us



#### About us



#### Scrutinizer

##### **View Applicants Scrutinizer**



##### **Edit Applicant**



#### Program Chairperson

#### Update Profile



#### Set Course Teacher



#### Allocate Students to a Course



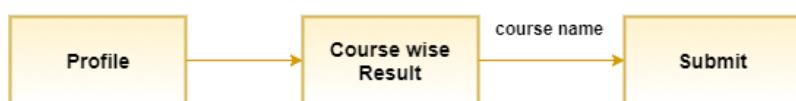
#### Approve New Account



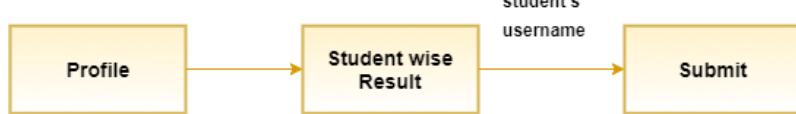
#### Update User Information



#### View Course wise Marks



#### View Student wise Marks



Receptionist

### **View Applicants**



### **Edit Applicant**

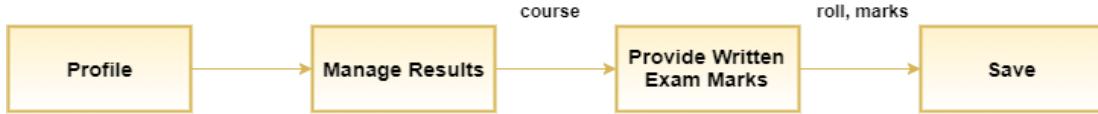


### **Teacher**

#### **View Students of a Course**



#### **Provide Written Exam Marks**



#### **Provide Supplementary Exam Marks**



### **Notification**



### **Student**

### **Update Profile**



### **View Courses**



### **View Notification**



### **Download Report Card**



## Exam Controller

### Update Profile



### Introduce New Admission



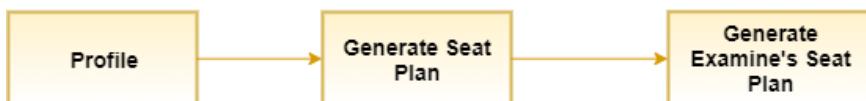
### Update Current Admission Session



### Upload Marks



### Generate Examine's Seat Plan



### Generate VIVA Schedule and Notify Teachers



### Enroll Students to Program



#### Update Room Information



#### Add New Room Information



#### Term Result Management



#### Upload Signature



#### View Notifications



#### Make Admit Card Available



#### Scrutinizing Session



4.2.3 Depict Each Interface State as It Look to End User  
Home Page

The screenshot displays the homepage of the PGDIT (Institute of Information Technology) website. At the top left is the IIT University of Dhaka logo. To the right is the PGDIT logo. A blue header bar contains the navigation links: Home (highlighted in orange), About us, Contact us, and Registration. Below the header is a photograph of the PGDIT building entrance, featuring palm trees and a blue sign that reads "তথ্য প্রযুক্তি ইনসিটিউট" and "Institute of Information Technology". To the right of the photo is a "Sign in" form with fields for "Username \*" and "Password \*", a "Send" button, and a "Forgot Password?" link. Below this is an orange "Notices" banner containing the text: "The SEAT PLAN of PGDIT Admission Test 2017-2018 has been Published" and "To Download PGDIT Admission Test SEAT PLAN [Click Here](#)".

Figure 149 Home Page

Registration Page

[Home](#)   [About us](#)   [Contact us](#)   **Registration**

## Sign-up

  
[Upload Photo](#)

Full Name

Username \*

Email Address \*

Teacher's Registration No \*

Father's Name

Mother's Name

Address

Password \*

Confirm Password \*

[Send](#)

Figure 150 Registration Page

About us Page



## About PGDIT

The Post Graduate Diploma in Information Technology (PGDIT) is a one-year program for graduates in any discipline on the principles and practice of Information Technology. The program will be conducted by the Institute of Information Technology (IIT) of the University of Dhaka (DU). The PGDIT program is offered mainly to the graduates who are willing to work in ICT domain or the graduates whose current or future career could be accelerated through advanced knowledge in ICT. The classes of the program will run four days a week from 6:00 pm to 9:30 pm.

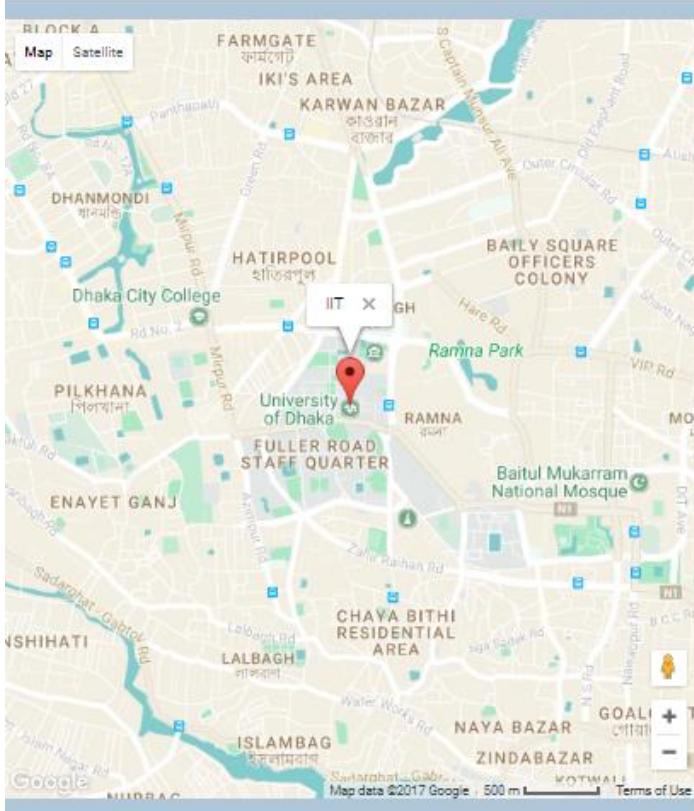
The number of seats for PGDIT is forty (40). But IIT authority will decide the number for the respective batch considering its available resource prior to the enrollment.

For more details please call at 01713 444 512



Figure 151 About us Page

[Contact us Page](#)



## Contact Us

### Institute of Information Technology

Suhrawardi Udyan Rd  
Dhaka 1200  
Bangladesh  
Mobile: 8801779482994

Please fill out the form below  
and we will contact you shortly:

Name \*

Email \*

Subject

Message

Send

Figure 152 Contact us Page

Program Chairperson: Profile

[Profile](#)[Course Management](#)[Account Requests](#)[View Users](#)[View Result](#)[Log out](#)[Change Photo](#)

## Moumita Asad



<b>User type</b>	Program Chairperson
<b>Email address</b>	moumita.asad@yahoo.com
<b>Father's name</b>	Asadul Ahad
<b>Mother's name</b>	Mahfuza Khanam
<b>Mobile no</b>	01815442207
<b>Address</b>	5/A, Fuller Road, University of Dhaka

Figure 153 Program Chairperson:Profile

Program Chairperson: Course Management



Profile

Course Management

Account Requests

View Users

View Result

Log out

Search

 Search User**Computer Fundamentals  
and Office Automation**

(PGD101)

Teacher: Nadia Nahar  
Current Students: 5[Update Teacher](#)[View Students](#)**Introduction to Software  
Engineering**

(PGD105)

Teacher: Nawshin Nahar  
Current Students: 21[Update Teacher](#)[View Students](#)**Structured  
Programming**

(PGD104)

Teacher: Zerina Begum  
Current Students: 17[Update Teacher](#)[View Students](#)

Figure 154 Program Chairperson: Course Management

Program Chairperson: Account Requests



Profile Course Management **Account Requests** View Users View Result Log out



**Alim Ul Gias**  
Email: alim@du.ac.bd      Accept  
Registration No: 140106      Decline

Figure 155 Program Chairperson: Account Requests

Program Chairperson: View Users



Profile Course Management Account Requests **View Users** View Result Log out

Q Search User

Afrina Sharmin  
Receptionist  
afrina@yahoo.com

[View Details](#)

Reshad Mollick  
Teacher  
reshad@yahoo.com

[View Details](#)

Figure 156 Program Chairperson: View Users

Teacher: Profile



Profile

View Course

Manage results

Notifications

Log out



Reshad Mollick

---

User type Teacher

Email address reshad@gmail.com

Father's name Asadul Ahad

Mother's name Mahfuza Khanam

Mobile no 01815442207

Address Nur jahan road, Mohammadpur



Figure 157 Teacher: Profile

Teacher: View Course

[Profile](#) [View Course](#) [Manage results](#) [Notifications](#) [Log out](#)**Search** Search Site**Computer Fundamentals  
and Office Automation**  
**(PGD101)**Credit Hours: 3  
Prerequisites: None[Update Course Information](#)[View Students](#)**Operating System  
Concepts & UNIX OS**  
**(PGD106)**Credit Hours: 3  
Prerequisites: PGD 101,PGD 104[Update Course Information](#)[View Students](#)**Structured  
Programming**  
**(PGD104)**Credit Hours: 3  
Prerequisites: None[Update Course Information](#)[View Students](#)

Figure 158 Teacher: View Course

Teacher: Manage Results

[Profile](#)   [View Course](#)   **Manage results**   [Notifications](#)   [Log out](#)**Search**

Q Search Course

**Computer Fundamentals  
and Office Automation**  
**(PGD101)****Credit Hours:** 3  
**Prerequisites:** None[Provide Written Marks](#)[Provide Supplementary Marks](#)**Operating System  
Concepts & UNIX OS**  
**(PGD106)****Credit Hours:** 3  
**Prerequisites:** PGD 101,PGD 104[Provide Written Marks](#)[Provide Supplementary Marks](#)**Structured  
Programming**  
**(PGD104)****Credit Hours:** 3  
**Prerequisites:** None[Provide Written Marks](#)[Provide Supplementary Marks](#)

Figure 159 Teacher: Manage Results

Teacher: Notifications

[Profile](#)[View Course](#)[Manage results](#)[Notifications](#)[Log out](#)

## Viva Schedule

Nov 20, 2017 10.30 am

**Viva Duty**

Room no: 107

Figure 160 Teacher: Notifications

Exam Controller: Profile


[Profile](#)
[Manage Admission Initiation](#)
[Manage Admission Exam](#)
[Room Management](#)
[Term Result](#)
[More](#)

[Change Photo](#)

## Dr Shariful Islam



User type	Exam Controller
Email address	shariful@yahoo.com
Father's name	Nurul Islam
Mother's name	Nasima Begum
Mobile no	01713192030
Address	5/A, Fuller Road, University of Dhaka

Figure 161 Exam Controller: Profile

### Exam Controller: Introduce New Admission


[Profile](#)
[Manage Admission Initiation](#)
[Manage Admission Exam](#)
[Room Management](#)
[Term Result](#)
[More](#)

### Introduce New Admission Session

Application Submission Start:	<input type="text"/>	Application Submission End:	<input type="text"/>
Application Payment Start:	<input type="text"/>	Application Payment End:	<input type="text"/>
Scrutinization Start:	<input type="text"/>	Scrutinization End:	<input type="text"/>
Admission Payment Start:	<input type="text"/>	Admission Payment End:	<input type="text"/>

Figure 162 Exam Controller: Introduce New Admission

## Exam Controller: Update Admission Information

The screenshot shows the 'Manage Admission Initiation' section of the Exam Controller. It displays the following dates:

Application Submission Start:	11-10-2017	Application Submission End:	02-11-2017
Application Payment Start:	04-11-2017	Application Payment End:	11-11-2017
Scrutinization Start:	12-11-2017	Scrutinization End:	19-11-2017
Admission Payment Start:	20-11-2017	Admission Payment End:	27-11-2017

A blue 'Update' button is located at the bottom right of the box.

Figure 163 Exam Controller: Update Admission Information

## Exam Controller: Additional Admission Initiation Activities

The screenshot shows the 'Scrutinization Session' section of the Exam Controller. It displays the message: "Scrutinization Session is ongoing...". Below this are two buttons: a red 'Lock' button and a green 'Unlock' button.

The 'Admit Card' section shows the message: "Admission payment completed for all applications." and a green 'Make Admit Card Available' button.

Figure 164 Exam Controller: Additional Admission Initiation Activities

Exam Controller: Upload Marks written/VIVA

The screenshot shows a web-based application for managing examinations. At the top left is the logo of IIT University of Dhaka. To its right is the text "PGDIT". Further right is a crest or seal. Below the header is a dark blue navigation bar with white text containing links: "Profile", "Manage Admission Initiation", "Manage Admission Exam", "Room Management", "Term Result", and "More". The main content area has a light blue background. It displays the message "Written Examination Completed" in large, bold, dark blue text. Below this, there is a button labeled "Upload Marks" with a dark blue background and white text. The overall layout is clean and professional.

Figure 165 Exam Controller: Upload Marks written/VIVA

Exam Controller: Generate Seat Plan

The screenshot shows the IIT University of Dhaka PGDIT Exam Controller interface. At the top, there is a logo for 'University of Dhaka' and the text 'IIT' and 'PGDIT'. A navigation bar below the logo includes links for 'Profile', 'Manage Admission Initiation', 'Manage Admission Exam', 'Room Management', 'Term Result', and 'More'. The main content area features a large blue button with two options: 'Generate Examinees' Seat Plan' and 'Generate VIVA Schedule and Notify Teachers'.

Figure 166 Exam Controller: Generate Seat Plan

#### Exam Controller: Update Enrollment State

The screenshot shows the IIT University of Dhaka PGDIT Exam Controller interface for updating enrollment state. The top navigation bar is identical to Figure 166. The main content area displays a table with four columns: 'Roll', 'Total Marks', 'Enrollment', and an 'Enroll' button. The table contains four rows of data:

Roll	Total Marks	Enrollment
123456	92	Enrolled
123480	88	<b>Enroll</b>
123471	86	Enrolled
123450	81	<b>Enroll</b>

Figure 167 Exam Controller: Update Enrollment State

### Exam Controller: Manage Rooms

The screenshot shows the 'Room Management' section of the Exam Controller. It displays three rooms with their respective numbers and capacities:

Room Number	Capacity
105	35 seats
107	40 seats
402	36 seats

Each room entry includes an 'Update' button. At the bottom left, there is a button labeled 'Add New Room Information'.

Figure 168 Exam Controller: Manage Rooms

### Exam Controller: Term Result Management

[Profile](#)   [Manage Admission Initiation](#)   [Manage Admission Exam](#)   [Room Management](#)   [Term Result](#)   [More](#)

## Term Result Management

Marks uploading facility is running...

[Turn Off](#)[Publish Result](#)

Figure 169 Exam Controller: Term Result Management

### Exam Controller: Update Director's Signature

[Profile](#)   [Manage Admission Initiation](#)   [Manage Admission Exam](#)   [Room Management](#)   [Term Result](#)   [More](#)

Current Signature:

[Upload New Signature](#)

Figure 170 Exam Controller: Update Director's Signature

Exam Controller: Notifications

**IIT**  
University of Dhaka

**PGDIT**



Profile    Manage Admission Initiation    Manage Admission Exam    Room Management    Term Result    More

---

## VIVA Examination

VIVA examination has been completed.

---

## Written Examination

Written examination has been completed.

---

## Admit Card

Admission payment completed for all applications.

Figure 171 Exam Controller: Notifications

Student: Profile

[Profile](#)[View Courses](#)[Notification](#)[Logout](#)[Change Photo](#)

## Nafis Faysal



<b>User type</b>	Student
<b>Email address</b>	aminafis@yahoo.com
<b>Father's name</b>	Sifat Mehedi
<b>Mother's name</b>	Shamima Khatun
<b>Mobile no</b>	01715442206
<b>Address</b>	3/A, Fuller Road, University of Dhaka

Figure 172 Student: Profile

Student: View Running Courses

**IIT**  
University of Dhaka

**PGDIT**



Profile    View Courses    Notification    Logout

**Operating System  
Concepts & UNIX  
OS**

**Course Code:** PGD106  
**Instructor:** Nawshin Nawar  
**Credit Hour:** 03  
**Prerequisite:** None

**Internet Programming**

**Course Code:** PGD107  
**Instructor:** Nadia Nahar  
**Credit Hour:** 03  
**Prerequisite:** None

Figure 173 Student: View Running Courses

Student: View Courses Not Yet Taken

[Profile](#)   [View Courses](#)   [Notification](#)   [Logout](#)

**Data Structure & Algorithm**

<b>Course Code:</b>	PGD201
<b>Instructor:</b>	Shah Mostafa Khaled
<b>Credit Hour:</b>	03
<b>Prerequisite:</b>	None

**Object Oriented Programming**

<b>Course Code:</b>	PGD202
<b>Instructor:</b>	Md. Mainul Islam
<b>Credit Hour:</b>	03
<b>Prerequisite:</b>	PGD 101, PGD 104

[Current Courses](#)[Courses Not Yet Taken](#)[Courses Already Taken](#)

Figure 174 Student: View Courses Not Yet Taken

Student: View Courses Already Taken



Profile   View Courses   Notification   Logout

**Structured Programming****Course Code:** PGD104**Instructor:** Nadia Nahar**Credit Hour:** 03**Prerequisite:** None**Computer Fundamentals  
and Office Automation****Course Code:** PGD101**Instructor:** Rezvi Shahriar**Credit Hour:** 03**Prerequisite:** None

Figure 175 Student: View Courses Already Taken

## Student: Notifications

Profile   View Courses   **Notification**   Logout**Report Card Notification**

Term final result has been published. Please collect your report card by clicking the link below.

**Download Report Card**

Figure 176 Student: Notifications

## Scrutinizer: View Applicants

Application ID	Full Name	Eligibility status	Options ( Edit/Details)	
121332	Aquib Azmain	-	Edit	Details
127231	Niloy Tasnim	-	Edit	Details
120291	Fazle Rabbi	-	Edit	Details
123234	Tanvir Zaman	-	Edit	Details

Figure 177 Scrutinizer: View Applicants

## Receptionist: Profile

[Profile](#)[View Applicants](#)[Logout](#)

  
[Change Photo](#)  

---

**Afrina Sharmin**

---

**User type** Receptionist  
**Email address** afrina@yahoo.com  
**Father's name** Asadul Ahad  
**Mother's name** Mahfuza Khanam  
**Mobile no** 01815442207  
**Address** 15/A, Uttora, Dhaka



Figure 178 Receptionist: Profile

Receptionist: View Applicants

Profile    **View Applicants****Search** Search

Application ID	Full Name	Payment Status	Options( Edit/ Details)	
178345	Nishat Tasnim	Done	Edit	Details
171364	Nafis Faysal	Pending	Edit	Details
175190	Fatiul Huq	Pending	Edit	Details
173238	Shuvo Saha	Pending	Edit	Details

Figure 179 Receptionist: View Applicant

## References

- Pressman, Roger S. Software Engineering: A Practitioner's Approach (7th Edition)
- User Interface Design Basics, <https://www.usability.gov/what-and-why/user-interface-design.html>, last accessed on: 10.11.2017
- What is state diagram (state machine diagram or statechart diagram),  
<http://searchsoa.techtarget.com/definition/state-diagram>, last accessed on 14.11.2017