



# **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 .....	36
3.3.2 Identify Appropriate Interfaces for Each Component .....	54
3.3.3 Elaborate Attributes and Define Data Types and Data Structures required to Implement Them.....	61
3.3.4 Describe Processing Flow within Each Operation in Detail .....	63
3.4 Describe Persistent Data Sources and Identify the Classes required to Manage Them ..	104
3.5 Develop and Elaborate Behavioral Representations for a Class or Component .....	105
3.6 Elaborate Deployment Diagrams to Provide Additional Implementation Detail .....	112
Chapter 4. User Interface Design .....	113
4.1 Interface Analysis .....	113
4.1.1 User Analysis .....	113
4.1.2 Task Analysis .....	116
4.2 Interface Design Steps .....	121
4.2.1 Define Interface Objects and Actions .....	122
4.2.2 Define Events that will Cause the State of the User Interface to Change .....	151
4.2.3 Depict Each Interface State as It Look to End User .....	157

## 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 DFD mapping of 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 Teacher's Dataflow Diagram of Course Management System .....	21
Figure 39 DFD mapping of Student's Dataflow Diagram of Course Management System .....	21
Figure 40 DFD mapping of Program Chairperson's Dataflow Diagram of Result Generation System.....	21
Figure 41 DFD mapping of Exam Controllers's Dataflow Diagram of Result Generation System	22
Figure 42 DFD mapping of Teacher's Dataflow Diagram of Result Generation System.....	23
Figure 43 DFD mapping of Student's Dataflow Diagram of Result Generation System .....	23
Figure 44 Steps of Component-Level Design.....	24
Figure 45 Analysis classes .....	25
Figure 46 Design classes .....	26
Figure 47 Elaborated Applicant class.....	27
Figure 48 Elaborated User class.....	28
Figure 49 Elaborated ProgramChairperson class.....	29
Figure 50 Elaborated Scrutinizer class .....	30
Figure 51 Elaborated Receptionist class .....	31
Figure 52 Elaborated ExamController class .....	33
Figure 53 Elaborated Teacher class .....	35
Figure 54 Elaborated Student class.....	36
Figure 55 Method applyForAdmission() of Applicant Class.....	37
Figure 56 Method downloadAdmitCard() of Applicant Class .....	37
Figure 57 Method checkFulfillmentOfPrerequisites() of ProgramChairperson Class .....	38
Figure 58 Method addNewStudentsIntoCourse() of ProgramChairperson Class.....	38
Figure 59 Method selectInstructor() of ProgramChairperson Class.....	39
Figure 60 Method viewAccountRequest() of ProgramChairperson Class .....	39
Figure 61 Method approveAccountRequest() of ProgramChairperson Class .....	40
Figure 62 Method viewResult() of ProgramChairperson Class.....	40
Figure 63 Method viewFilteredApplications() of Scrutinizer Class.....	41
Figure 64 Method updateStatusOfEligibility() of Scrutinizer Class.....	41
Figure 65 Method viewApplicantInformation() of Receptionist Class .....	42
Figure 66 Method updateStatusOfPayment() of Receptionist Class .....	42
Figure 67 Method addApplicationTimeRange() of ExamController Class .....	43
Figure 68 Method addPaymentTimeRange() of ExamController Class .....	43
Figure 69 Method addScrutinizationTimeRange() of ExamController Class .....	44
Figure 70 Method generateSeatPlan() of ExamController Class .....	44
Figure 71 Method updateVivaSchedule() of ExamController Class.....	45
Figure 72 Method updateWrittenMarks() of ExamController Class .....	45

Figure 73 Method updateVIVAMarks() of ExamController Class .....	46
Figure 74 Method getVivaNotification() of Teacher Class.....	46
Figure 75 Method getCourseAssignmentNotification() of Teacher Class .....	47
Figure 76 Method viewListOfCourses() of Teacher Class .....	47
Figure 77 Method viewCourseInformation() of Teacher Class.....	48
Figure 78 Method updateCourseInformation() of Teacher Class.....	48
Figure 79 Method setSupplementaryMarks() of Teacher Class .....	49
Figure 80 Method setCourseMark() of Teacher Class .....	49
Figure 81 Method generateResultPdf() of Teacher Class .....	50
Figure 82 Method checkGradeForSupplementary() of Teacher Class.....	50
Figure 83 Method generateSupplementaryList() of Teacher Class .....	51
Figure 84 Method generateSupplementaryResultPdf() of Teacher Class .....	51
Figure 85 Method notifyForSupplementaryExam() of Teacher Class.....	52
Figure 86 Method uploadFinalResultPdf() of Teacher Class.....	52
Figure 87 Method viewCourseList() of Student Class.....	53
Figure 88 Method viewCourseDetails() of Student Class .....	53
Figure 89 Method getMarks() of Student Class .....	54
Figure 90 Method getNotification() of Student Class.....	54
Figure 91 Authentication class.....	55
Figure 92 RollGenerator class .....	56
Figure 93 ReportCard class .....	56
Figure 94 RoomHandler Class .....	58
Figure 95 HeaderDisplayer class .....	59
Figure 96 PDFHandler class.....	60
Figure 97 verifyUsernameAndPassword() of Authentication.....	63
Figure 98 forgotPassword() of Authentication .....	64
Figure 99 verifyUsernameAndRecoveryCode() of Authentication .....	65
Figure 100 setNewPassword() of Authentication .....	65
Figure 101 applyForAdmission() of Applicant.....	66
Figure 102 checkAvailabilityOfAdmitCard() of Applicant .....	67
Figure 103 generateAdmitCard() of Applicant.....	67
Figure 104 generateQuery() of QueryGenerator.....	68
Figure 105 updateAddress() of User .....	69
Figure 106 updateMobileNo() of User.....	69
Figure 107 updatePicture() of User .....	70
Figure 108 getBlockingNotification() of User .....	70
Figure 109 viewName() of HeaderDisplayer.....	71
Figure 110 viewImage() of HeaderDisplayer .....	71

Figure 111 viewListOfUsers() of ProgramChairperson .....	72
Figure 112 updateFullName() of ProgramChairperson .....	72
Figure 113 updateUserEmail () of ProgramChairperson .....	73
Figure 114 addNewStudentsIntoCourse() of ProgramChairperson .....	73
Figure 115 approveAccountRequest() of ProgramChairperson .....	74
Figure 116 checkFulfillmentOfPayment() of ProgramChairperson .....	74
Figure 117 selectInstructor() of ProgramChairperson .....	75
Figure 118 createStudentAccount() of ProgramChairperson .....	75
Figure 119 viewResult() of ProgramChairperson.....	76
Figure 120 viewInformationOfOneUser() of ProgramChairperson .....	77
Figure 121 viewListOfCourses () of ProgramChairperson.....	77
Figure 122 checkGradeForSupplementary() of Teacher.....	78
Figure 123 generateResultPdf() of Teacher .....	78
Figure 124 generateGrade() of Teacher .....	79
Figure 125 generateSupplementaryList() of Teacher .....	80
Figure 126 generateSupplementaryResultPdf() of Teacher .....	81
Figure 127 getResultSheet() of Teacher .....	81
Figure 128 notifyForSupplementaryExam() of Teacher .....	82
Figure 129 setCourseMark() of Teacher .....	82
Figure 130 setSupplementaryMark() of Teacher.....	83
Figure 131 updateCourseInformation() of Teacher.....	83
Figure 132 viewCourseInformation() of Teacher.....	84
Figure 133 viewListOfCourse() of Teacher.....	84
Figure 134 viewSupplementaryList() of Teacher .....	85
Figure 135 generateResultPdf() of PDFHandler.....	85
Figure 136 generateSupplementaryResultPdf() of PDFHandler .....	86
Figure 137 updatePdf() of PDFHandler.....	86
Figure 138 uploadFinalResultPdf() of PDFHandler .....	87
Figure 139 getNotification() of Student.....	88
Figure 140 viewCourseDetails() of Student .....	89
Figure 141 viewCourseList() of Student .....	90
Figure 142 generatePdf() of ReportCard .....	91
Figure 143 getMarks() of ReportCard .....	91
Figure 144 addApplicationTimeRange() of Exam Controller .....	92
Figure 145 addPaymentTimeRange() of Exam Controller .....	92
Figure 146 addScrutinizationTimeRange() of Exam Controller .....	93
Figure 147 endAdmissionSession() of ExamController.....	93
Figure 148 lockVivaMarks() of ExamController .....	94

Figure 149 lockWrittenMarks() of ExamController .....	94
Figure 150 updateAvailabilityOfAdmitCard() of ExamController .....	95
Figure 151 updateEnrollmentState() of ExamController.....	95
Figure 152 updateVivaSchedule() of ExamController.....	96
Figure 153 updateWrittenMarks() of ExamController .....	96
Figure 154 viewListOfApplicants() of ExamController.....	97
Figure 155 viewListOfEligibleCandidateAfterFullResult() of ExamController.....	97
Figure 156 viewListOfEnrolledCandidates() of ExamController .....	98
Figure 157 addNewRoom() of RoomHandler .....	98
Figure 158 generateSeatPlan() of RoomHandler.....	99
Figure 159 showListOfRoom() of RoomHandler.....	99
Figure 160 updateRoomCapacity() of RoomHandler .....	100
Figure 161 viewFilteredApplications() of Scrutinizer.....	100
Figure 162 searchByApplicationID () of Scrutinizer .....	101
Figure 163 updateStatusOfEligibility() of Scrutinizer.....	101
Figure 164 assignRollToEligibleApplicants() of RollGenerator .....	102
Figure 165 generateRoll() of RollGenerator .....	103
Figure 166 updateStatusOfPayment() of Receptionist.....	103
Figure 167 viewApplicantInformation() of Receptionist .....	104
Figure 168 DatabaseHandler class.....	104
Figure 169 State Diagram of the Applicant class .....	105
Figure 170 State Diagram of the Authentication class .....	105
Figure 171 state diagram of the Authentication class.....	106
Figure 172 State Diagram of the HeaderDisplayer class .....	106
Figure 173 State Diagram of the ProgramChairperson class.....	107
Figure 174 State Diagram of the ExamController class .....	108
Figure 175 State Diagram of the RoomHandler class .....	109
Figure 176 State Diagram of the Receptionist class .....	109
Figure 177 State Diagram of the Scrutinizer class .....	109
Figure 178 State Diagram of the RollGenerator class.....	109
Figure 179 State Diagram of the Teacher class .....	110
Figure 180 State Diagram of the PDFHandler class .....	110
Figure 181 State Diagram of the Student class.....	111
Figure 182 State Diagram of the ReportCard class.....	111
Figure 183 Deployment Diagram of PGDIT Automation System.....	112
Figure 184 Interface Design Steps .....	121
Figure 185 Home.....	122
Figure 186 About us .....	123

Figure 187 Contact us .....	123
Figure 188 Registration .....	124
Figure 189 Forgot Password .....	125
Figure 190 Reset Password .....	126
Figure 191 New Password.....	126
Figure 192 Admission form.....	127
Figure 193 Download Admit Card.....	128
Figure 194 Profile of Program Chairperson .....	129
Figure 195 Update Profile .....	129
Figure 196 Course Management .....	130
Figure 197 Select Teacher .....	130
Figure 198 View Students of a Course .....	131
Figure 199 Allocate Students .....	132
Figure 200 Account Requests .....	132
Figure 201 View Users .....	133
Figure 202 View User Details .....	133
Figure 203 Edit User Information .....	134
Figure 204 Course wise Result.....	134
Figure 205 Student wise Result .....	135
Figure 206 Teacher's Profile.....	136
Figure 207 View Courses.....	137
Figure 208 View Students .....	137
Figure 209 Manage Result .....	138
Figure 210 Provide Marks .....	139
Figure 211 Notifications.....	139
Figure 212 Student Profile .....	140
Figure 213 Download Report Card .....	141
Figure 214 View Courses.....	141
Figure 215 Exam Controller's Profile .....	142
Figure 216 Manage Admission Initiation .....	143
Figure 217 Setting New Admission .....	143
Figure 218 Manage Admission Exam .....	144
Figure 219 Provide Written Marks.....	144
Figure 220 Room Management .....	145
Figure 221 Term Result .....	146
Figure 222 Update Signature .....	146
Figure 223 Receptionist Profile.....	148
Figure 224 View Applicants.....	148

Figure 225 Update Payment Status .....	149
Figure 226 View Applicants (Scrutinizer) .....	150
Figure 227 Update Eligibility Status .....	150
Figure 228 Home Page .....	157
Figure 229 Registration Page .....	158
Figure 230 About us Page .....	159
Figure 231 Contact us Page.....	160
Figure 232 Program Chairperson:Profile .....	161
Figure 233Program Chairperson: Course Management.....	162
Figure 234 Program Chairperson: Account Requests .....	163
Figure 235 Program Chairperson: View Users .....	164
Figure 236 Teacher: Profile .....	165
Figure 237 Teacher: View Course .....	166
Figure 238 Teacher: Manage Results.....	167
Figure 239 Teacher: Notifications.....	167
Figure 240 Exam Controller: Profile.....	169
Figure 241 Exam Controller: Introduce New Admission.....	170
Figure 242Exam Controller: Update Admission Information .....	170
Figure 243 Exam Controller: Additional Admission Initiation Activities.....	171
Figure 244 Exam Controller: Upload Marks written/VIVA .....	172
Figure 245 Exam Controller: Generate Seat Plan .....	173
Figure 246 Exam Controller: Update Enrollment State .....	173
Figure 247 Exam Controller: Manage Rooms .....	174
Figure 248 Exam Controller: Term Result Management .....	175
Figure 249 Exam Controller: Update Director's Signature .....	175
Figure 250 Exam Controller: Notifications.....	176
Figure 251 Student: Profile .....	177
Figure 252 Student: View Running Courses.....	178
Figure 253 Student: View Courses Not Yet Taken .....	180
Figure 254 Student: View Courses Already Taken .....	181
Figure 255 Student: Notifications .....	182
Figure 256 Scrutinizer: View Applicants .....	182
Figure 257 Receptionist: Profile.....	183
Figure 258 Receptionist: View Applicants .....	184

## List of Tables

Table 1 Elaborated Attributes of Design classes.....	61
Table 2 Analyzing Teacher .....	113
Table 3 Analyzing Student.....	114
Table 4 Analyzing Program Chairperson.....	114
Table 5 Analyzing Applicant .....	114
Table 6 Analyzing Exam Controller .....	115
Table 7 Analyzing Scrutinizer .....	115
Table 8 Analyzing Receptionist .....	116

## 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.

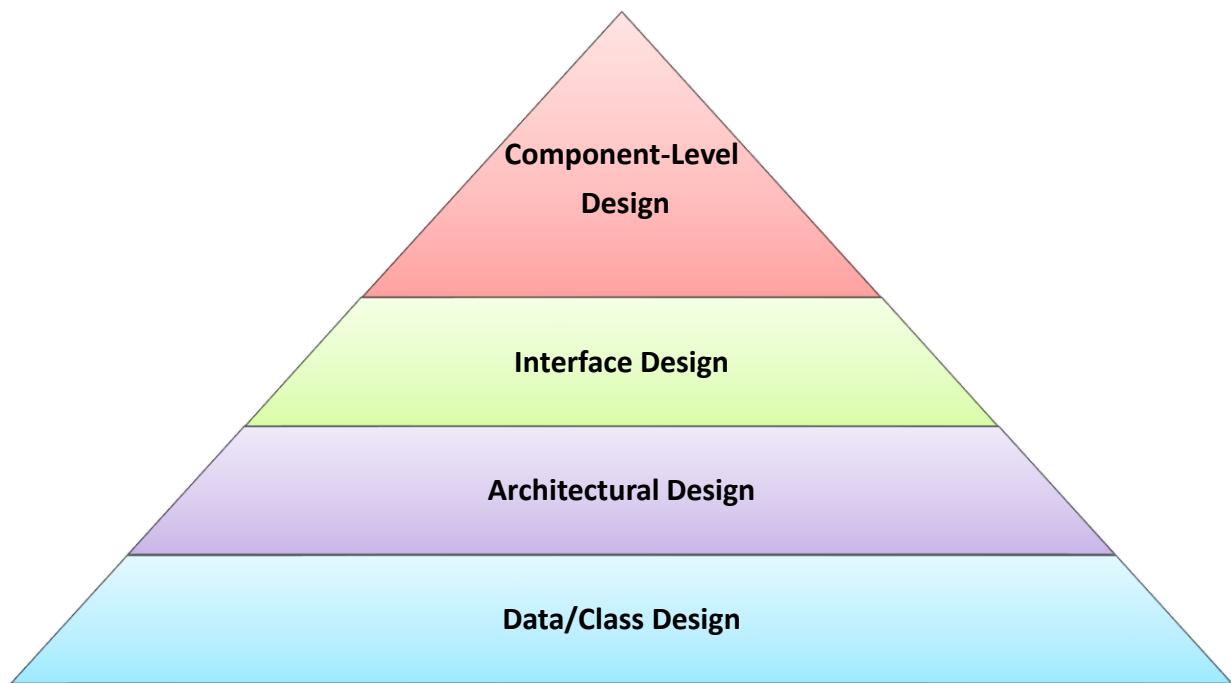


Figure 1 Design Model

## 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.

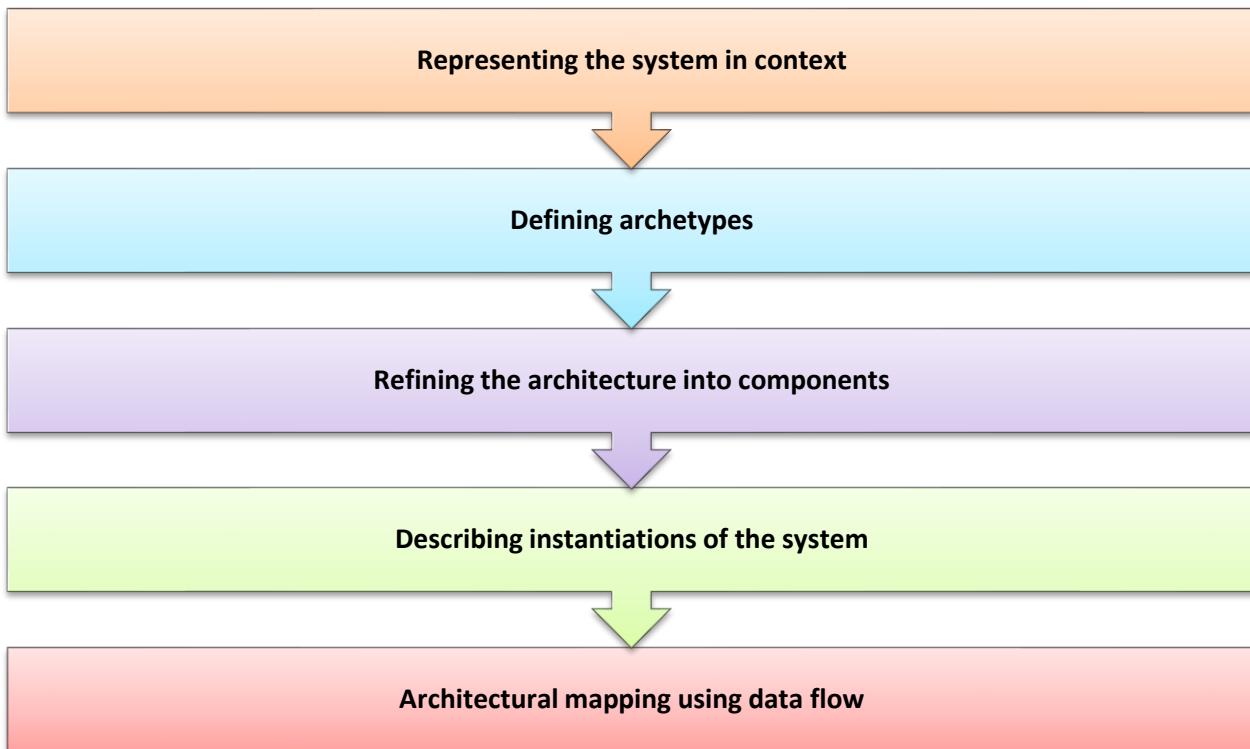


Figure 2 Steps of Architectural Design

### 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 interoperates 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.

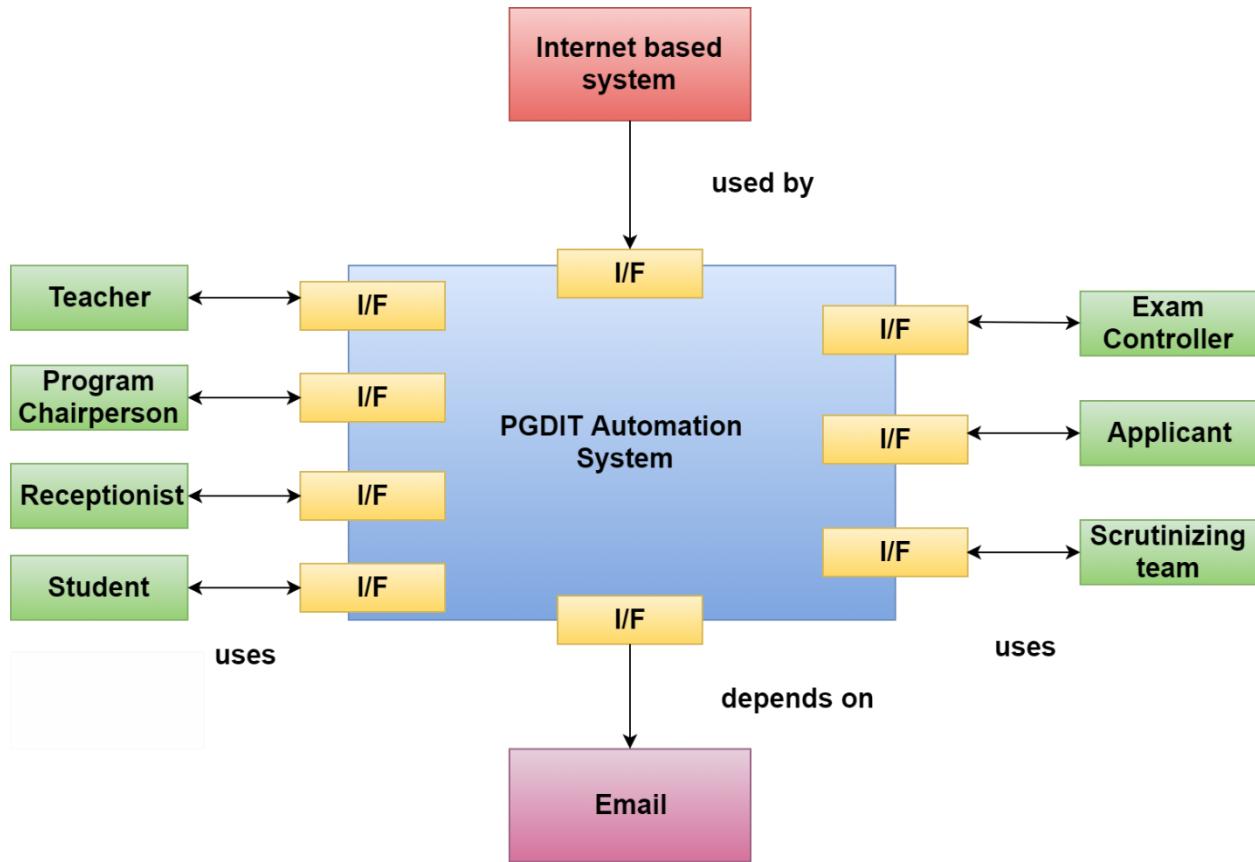


Figure 3 Architectural Context Diagram of PGDIT Automation System

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.

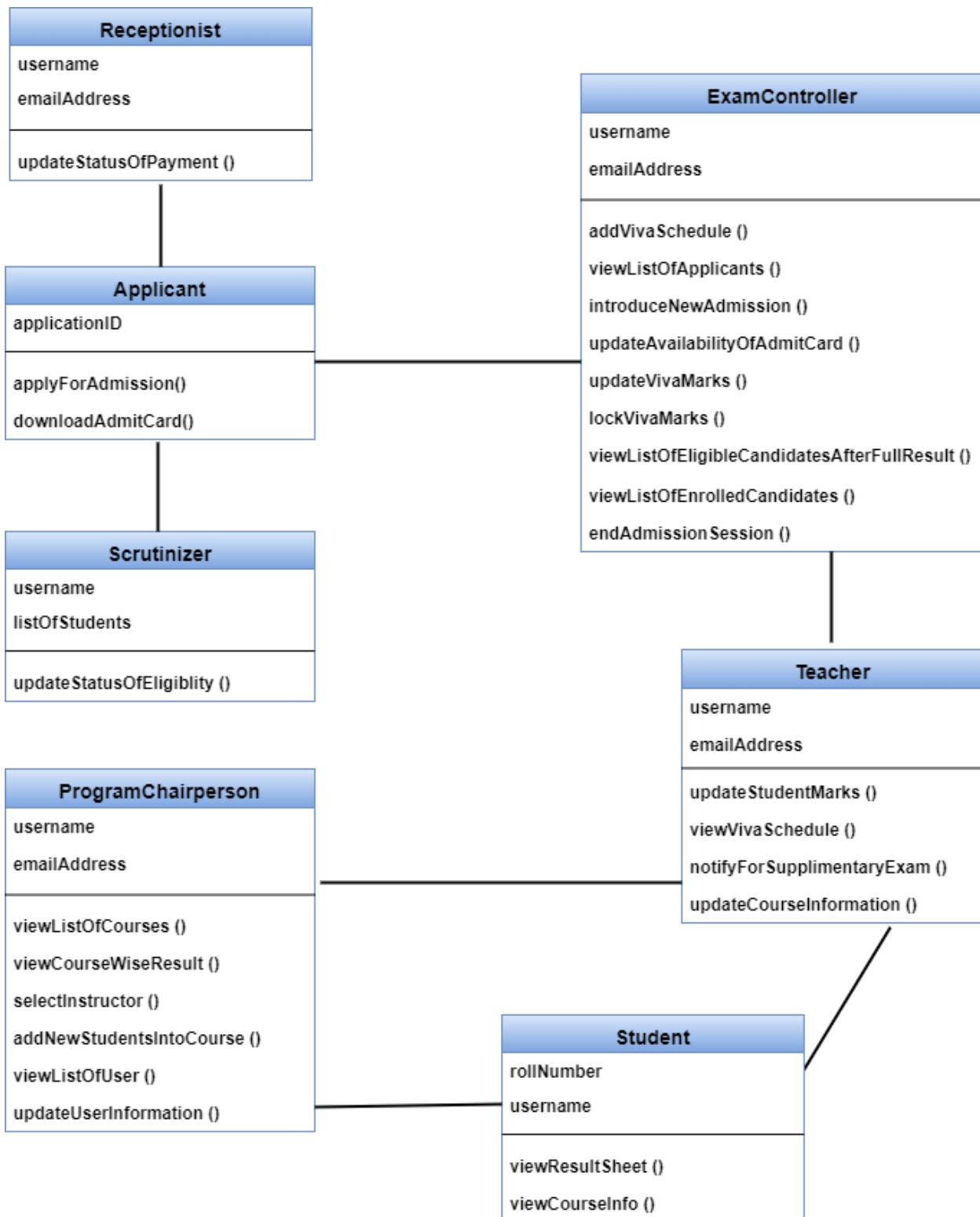


Figure 4 Archetypes

## 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.

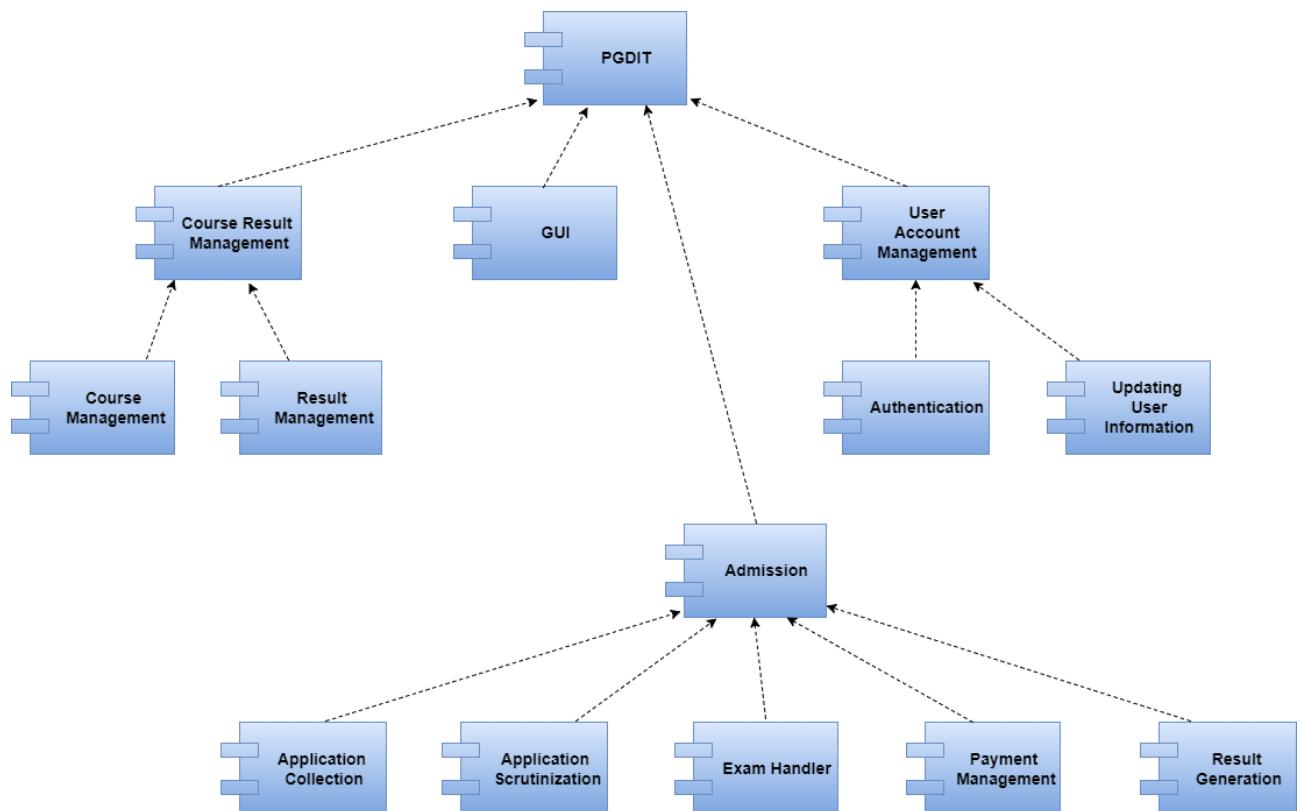


Figure 5 Overall Architectural Structure with Top-Level Components

## 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.

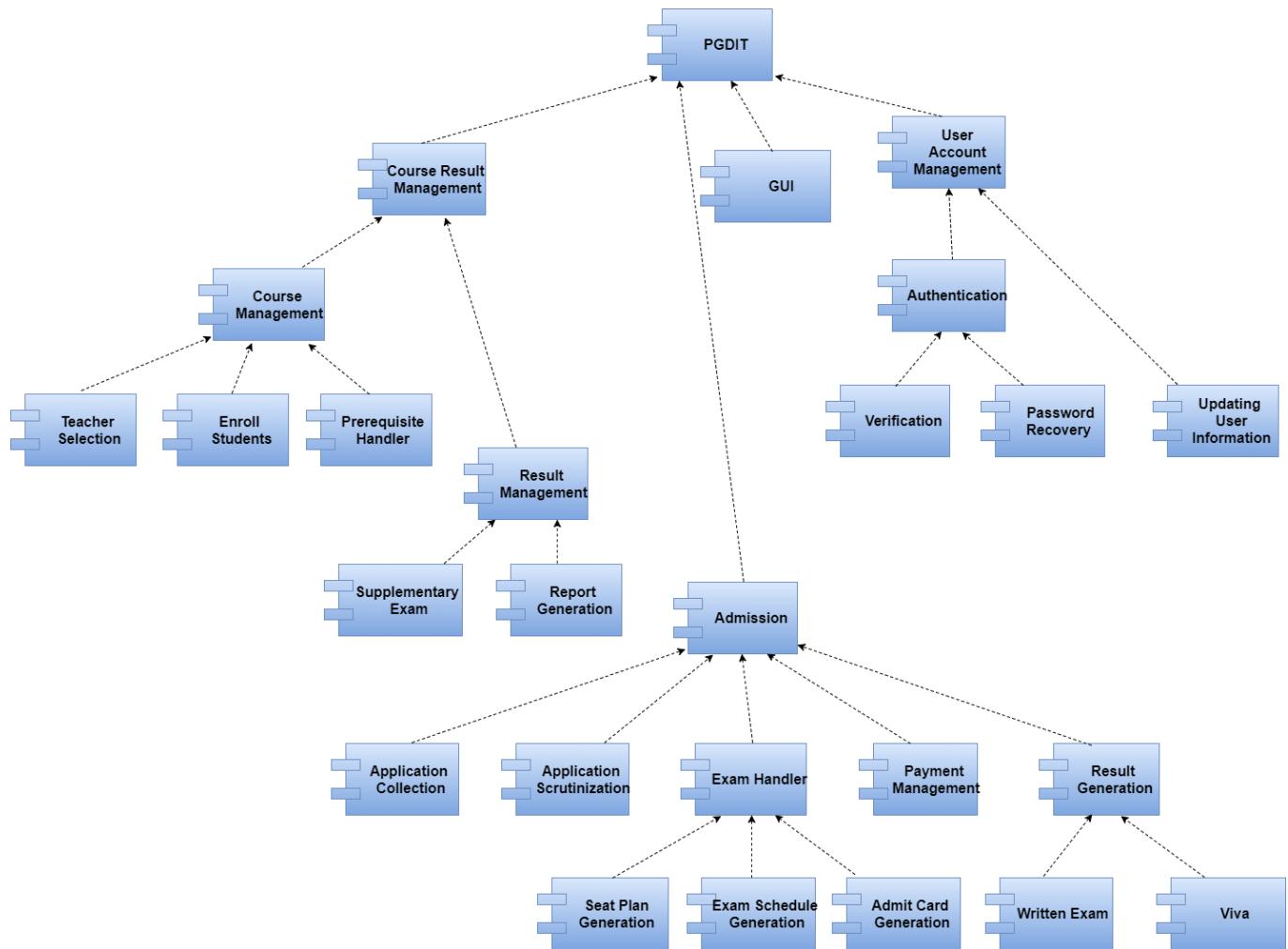


Figure 6 An Instantiation of the PGDIT Automation System Architecture

## 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:

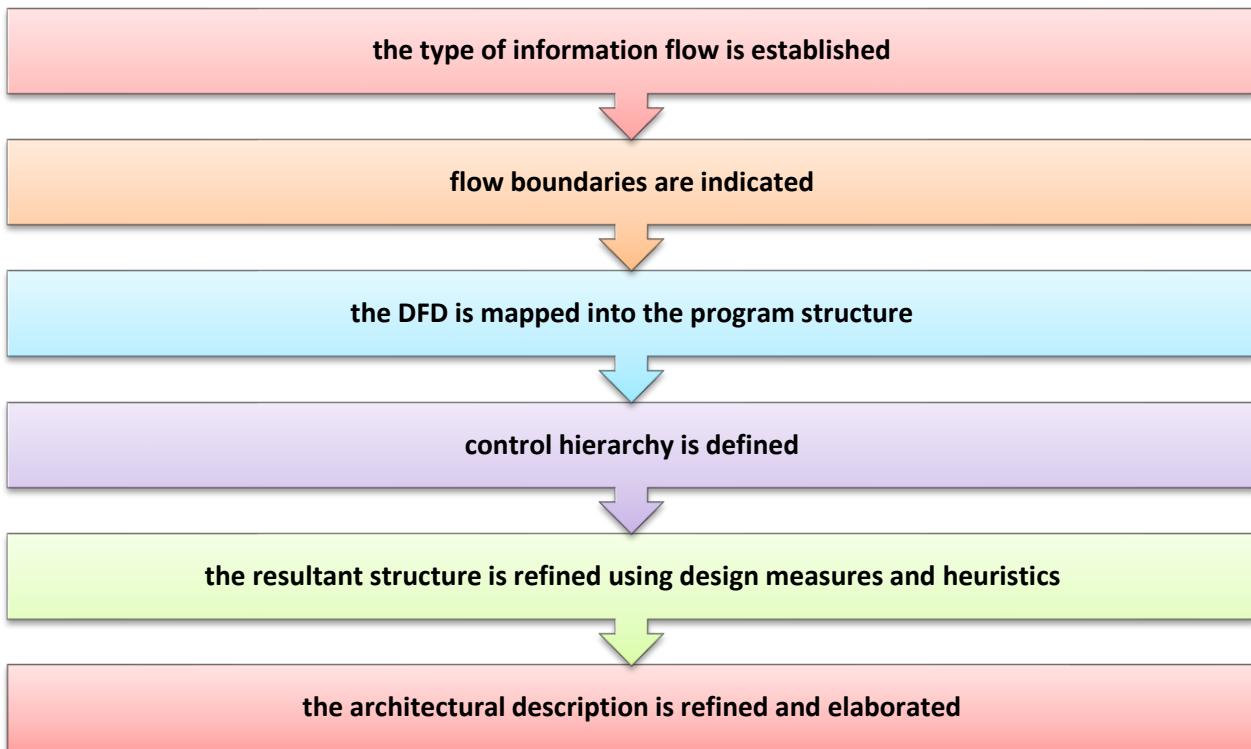


Figure 7 Architectural Mapping

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

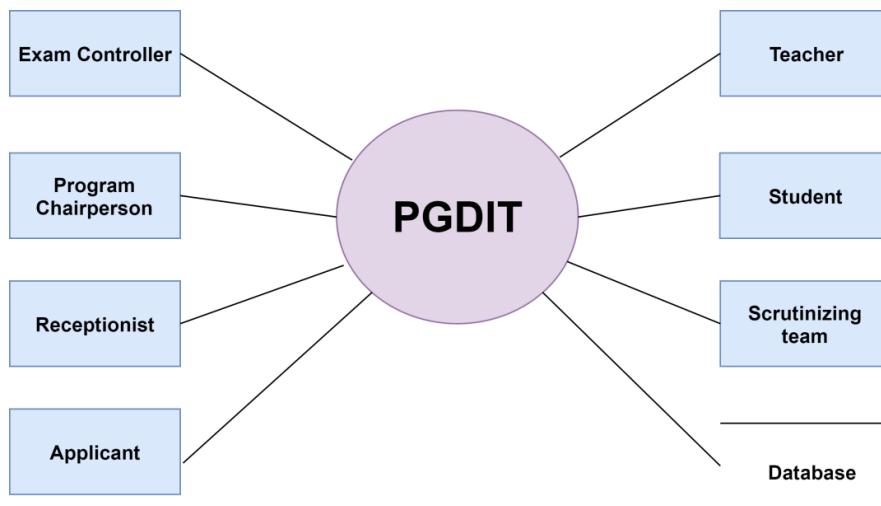


Figure 8 Level 0 Dataflow Diagram

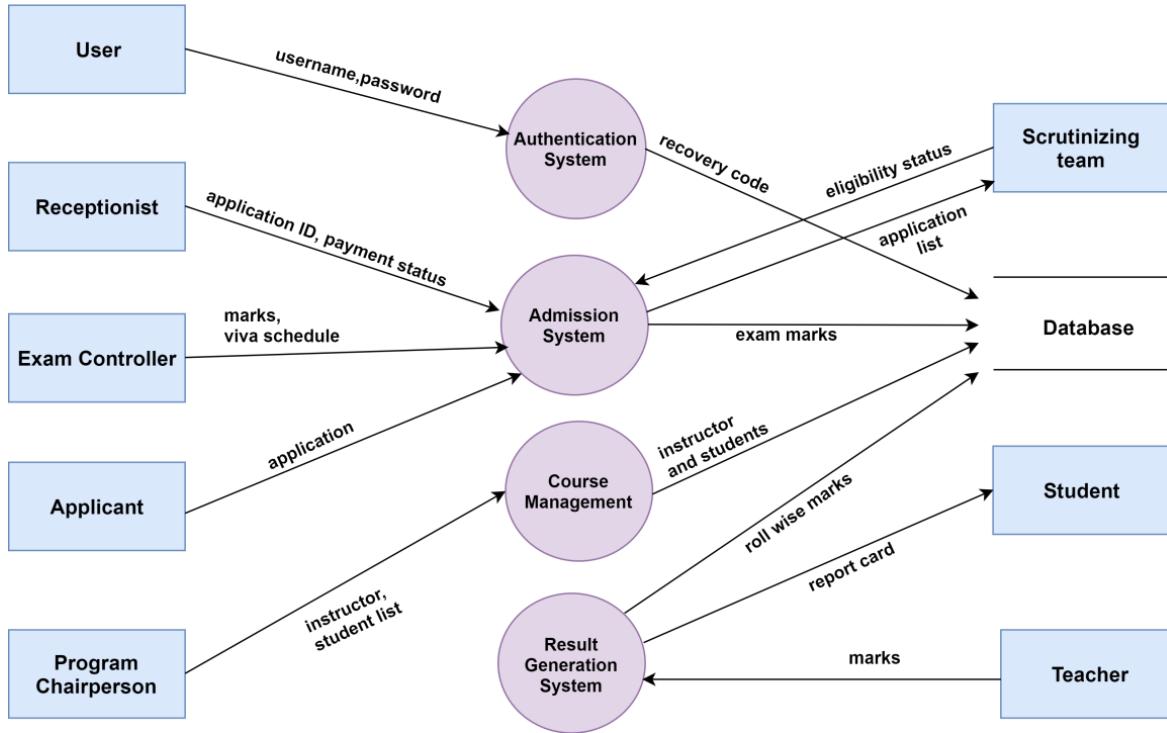


Figure 9 Level 1 Dataflow Diagram

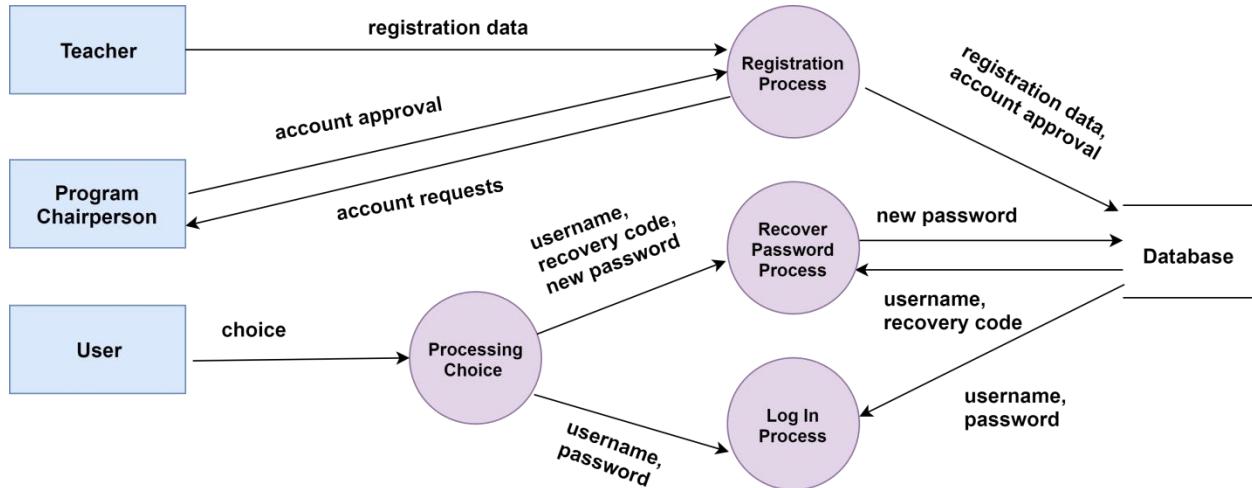


Figure 10 Dataflow Diagram of Authentication and Registration

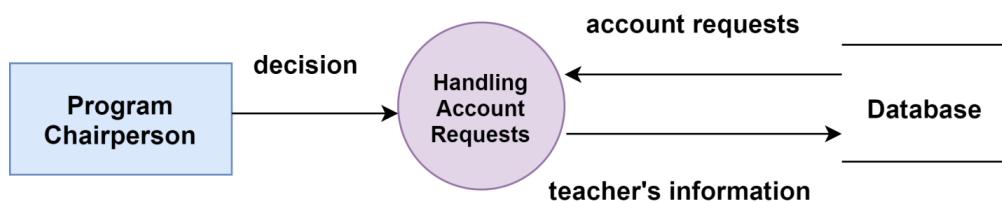


Figure 11 Program Chairperson's Dataflow Diagram of Registration

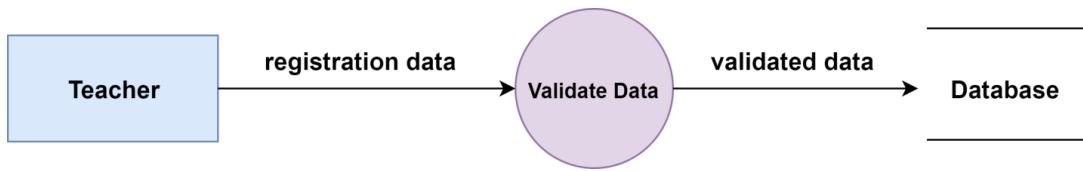


Figure 12 Teacher's Dataflow Diagram of Registration

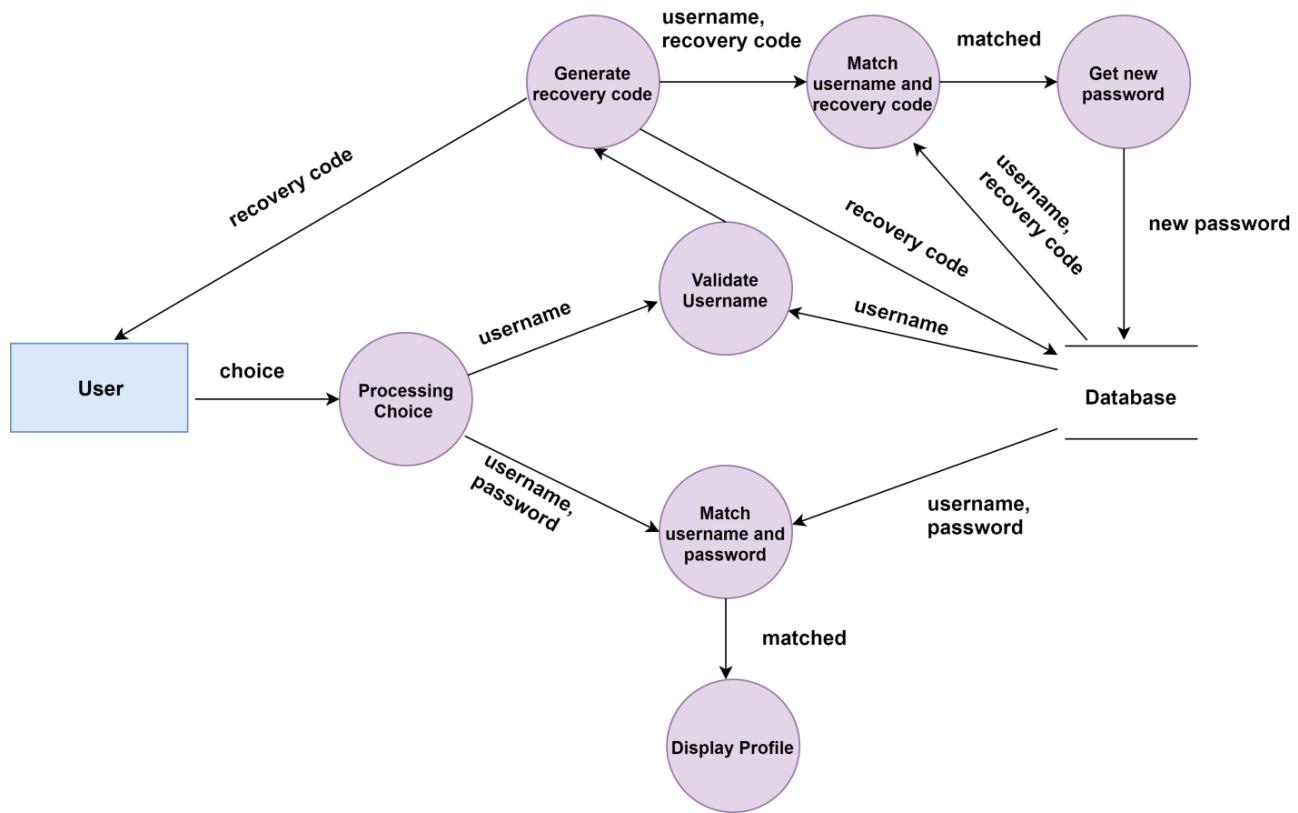


Figure 13 User's Dataflow Diagram of Authentication

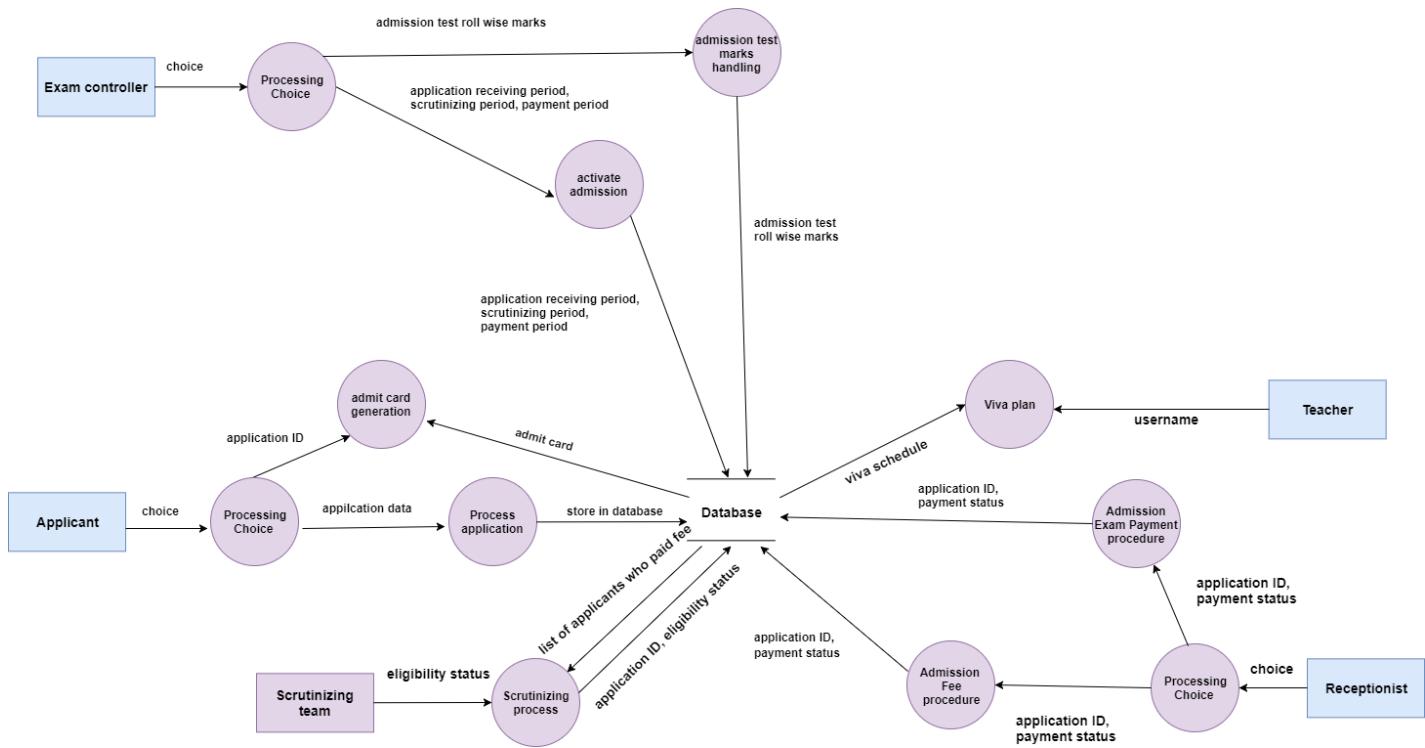


Figure 14 Dataflow Diagram of Admission System

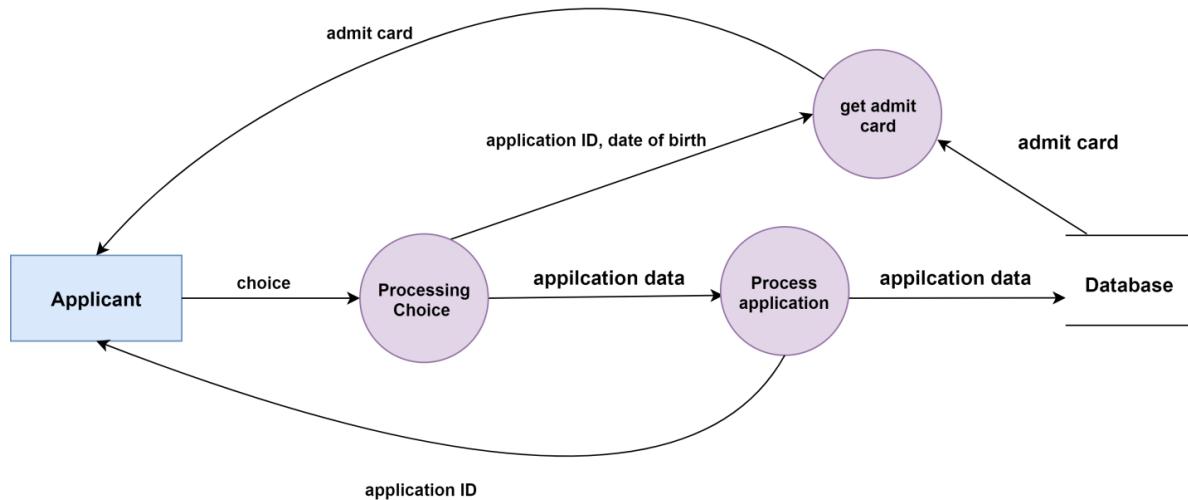


Figure 15 Applicant's Dataflow Diagram of Admission System

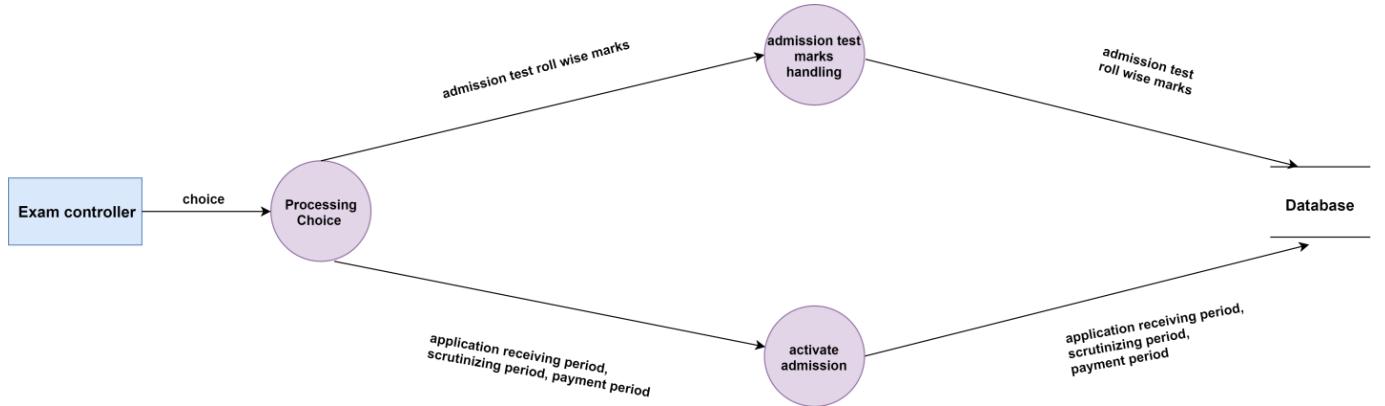


Figure 16 Exam Controller's Dataflow Diagram of Admission System

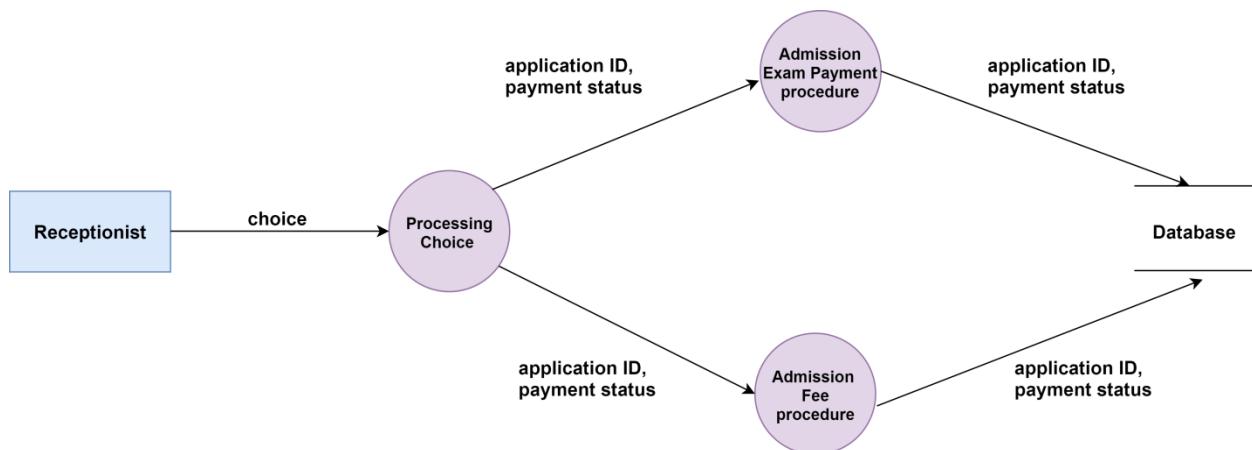


Figure 17 Receptionist's Dataflow Diagram of Admission System

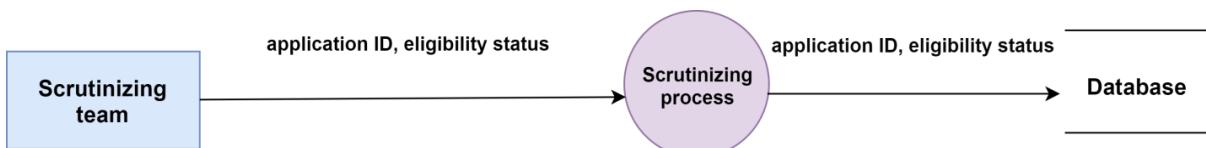


Figure 18 Scrutinizing team's Dataflow Diagram of Admission System

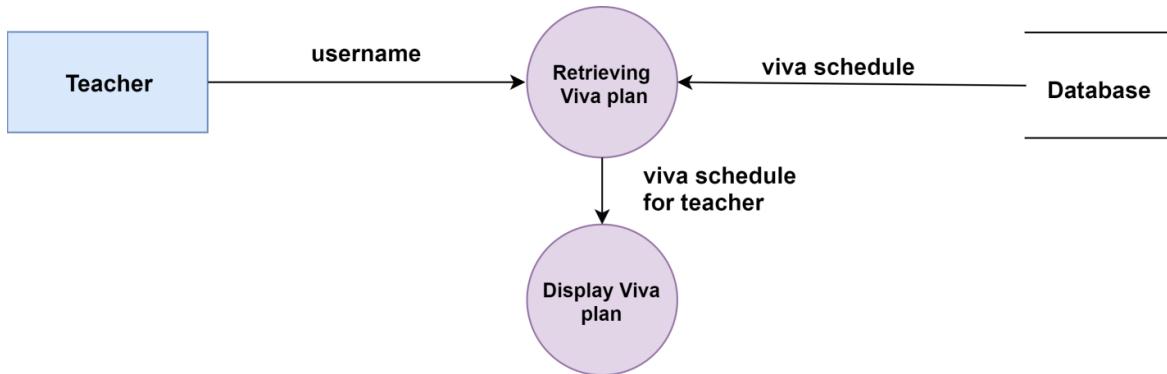


Figure 19 Teacher's Dataflow Diagram of Admission System

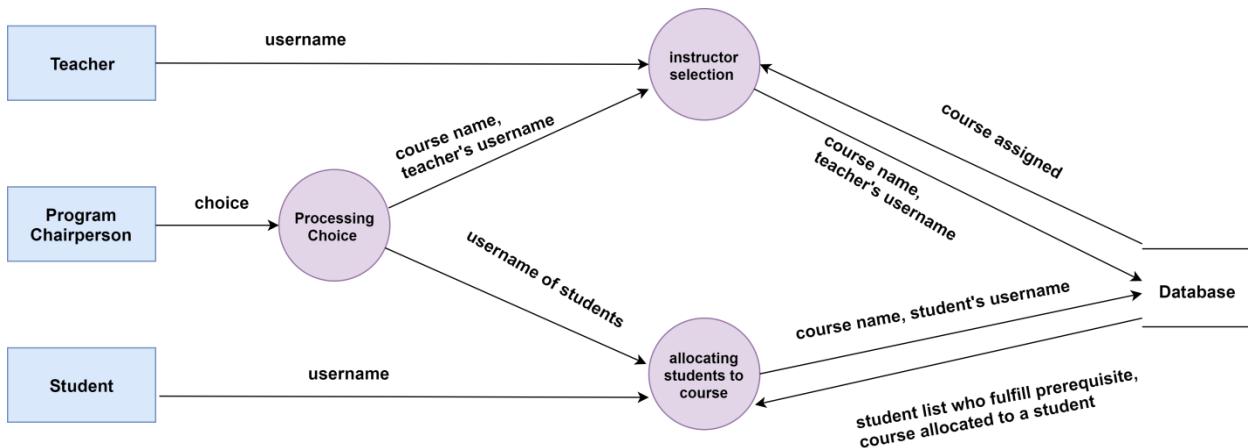


Figure 20 Dataflow Diagram of Course Management System

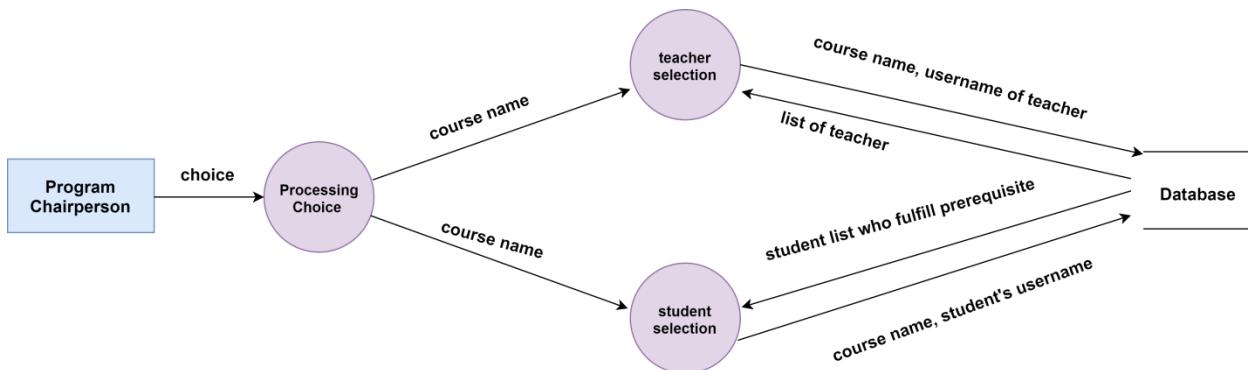


Figure 21 Program Chairperson's Dataflow Diagram of Course Management System

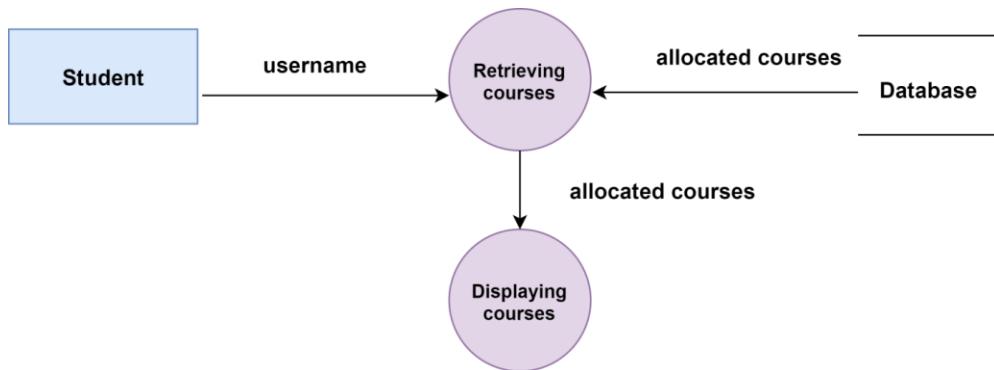


Figure 22 Student's Dataflow Diagram of Course Management System

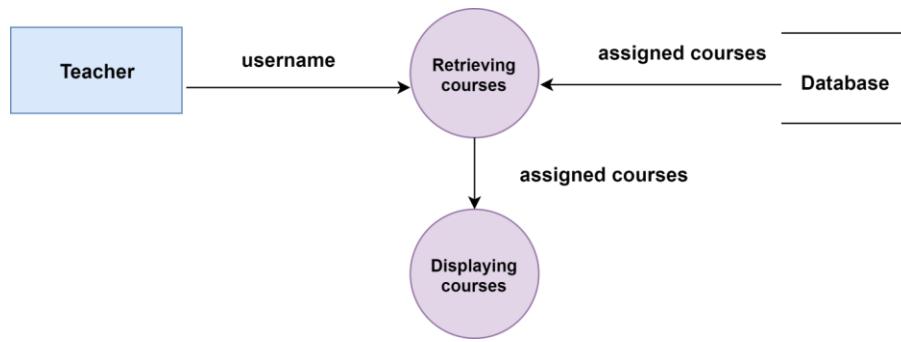


Figure 23 Teacher's Dataflow Diagram of Course Management System

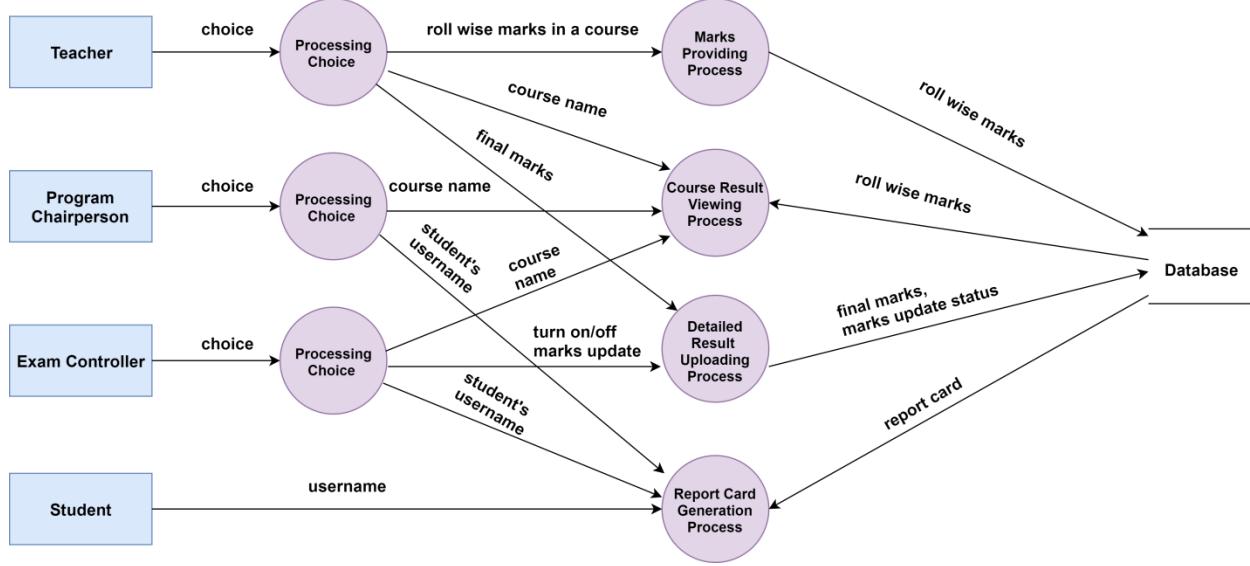


Figure 24 Dataflow Diagram of Result Generation System

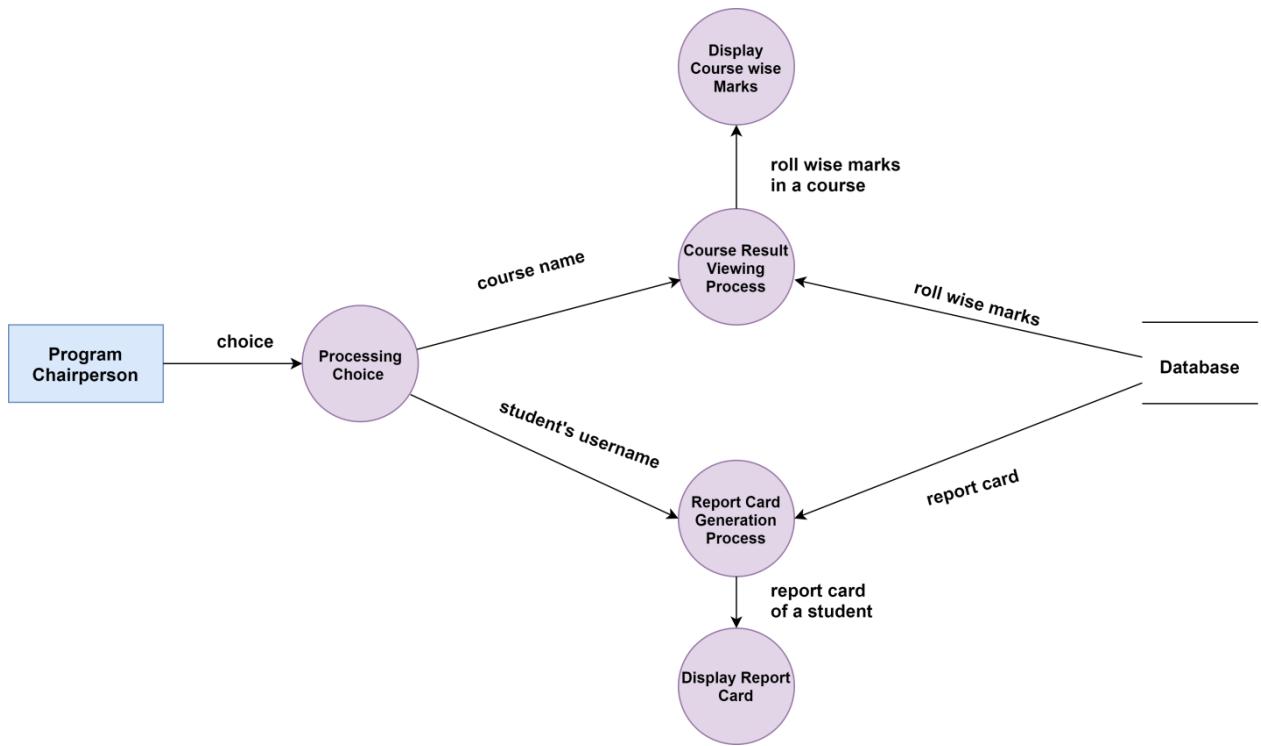


Figure 25 Program Chairperson's Dataflow Diagram of Result Generation System

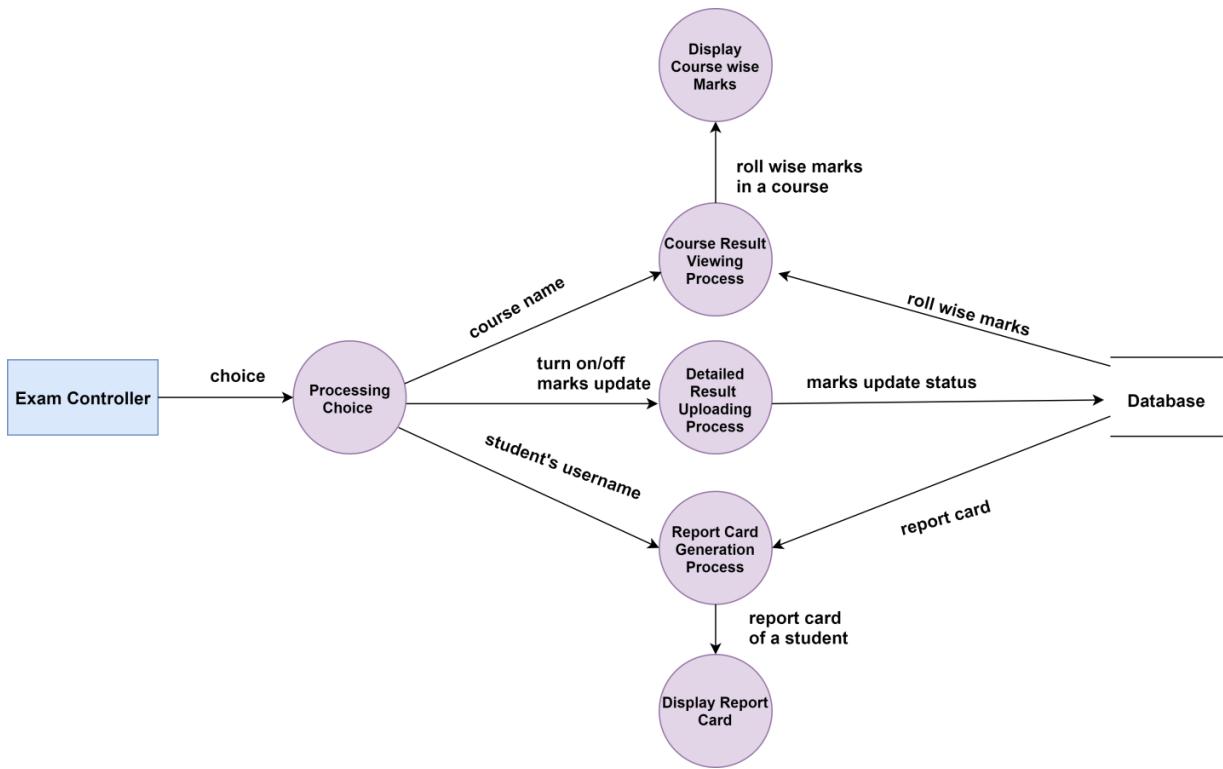


Figure 26 Exam Controller's Dataflow Diagram of Result Generation System

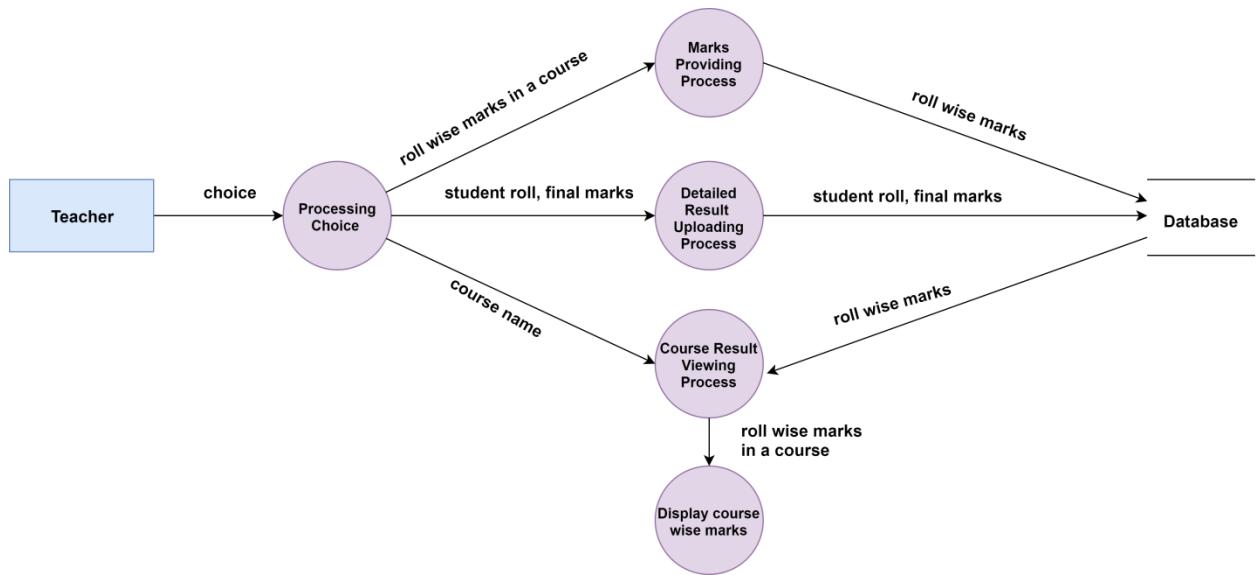


Figure 27 Teacher's Dataflow Diagram of Result Generation System

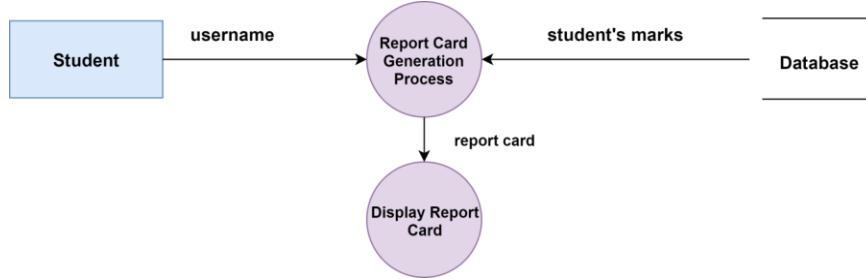


Figure 28 Student's Dataflow Diagram of Result Generation System

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

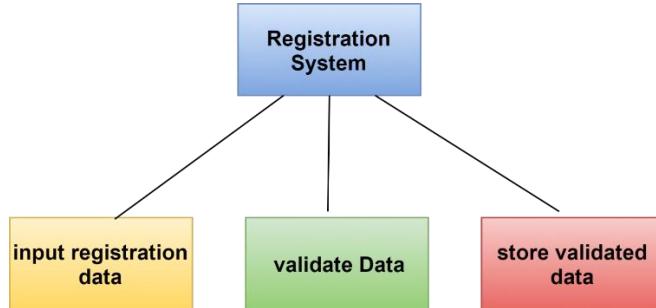


Figure 29 DFD mapping of Teacher's Dataflow Diagram of Registration

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

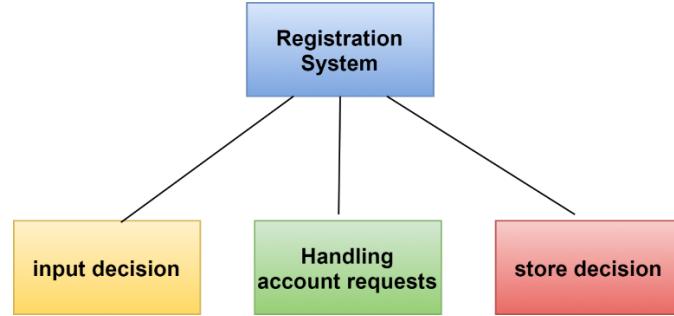


Figure 30 DFD mapping of Program chairperson's Dataflow Diagram of Registration

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

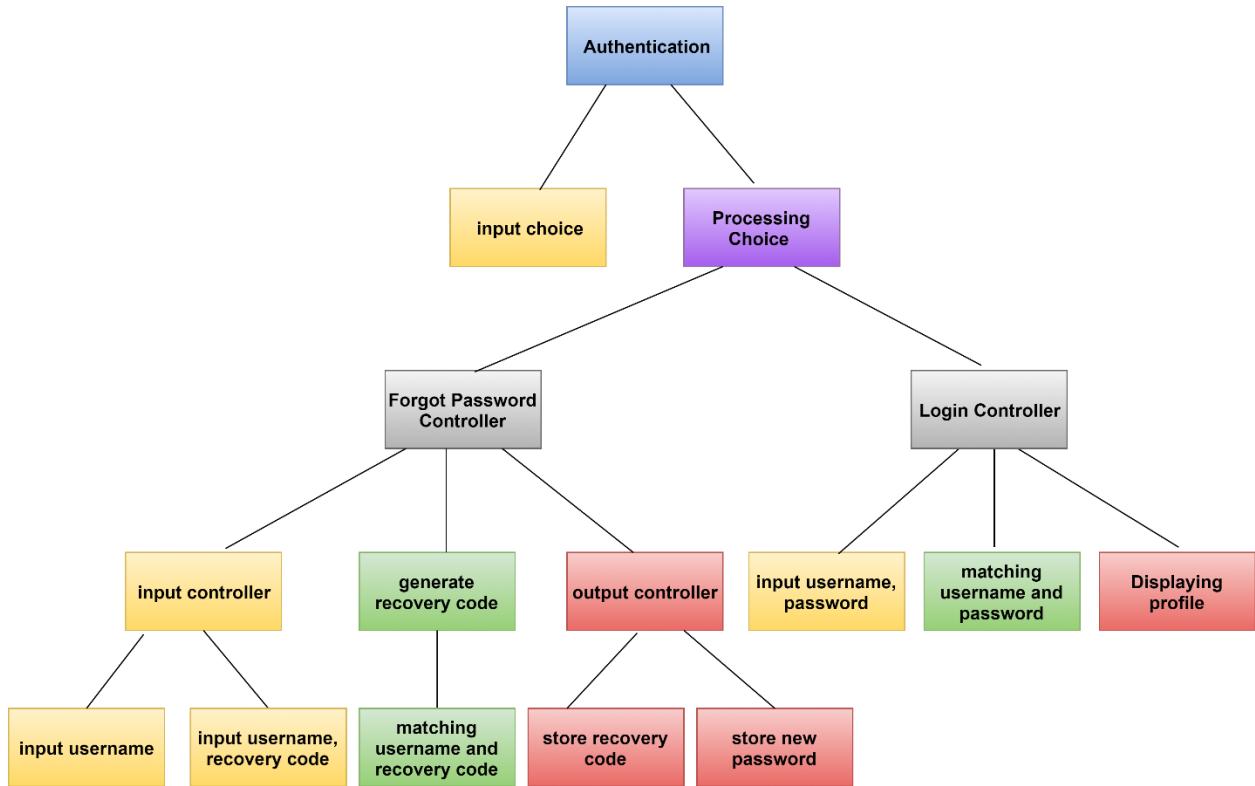


Figure 31 DFD mapping of User's Dataflow Diagram of Authentication

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

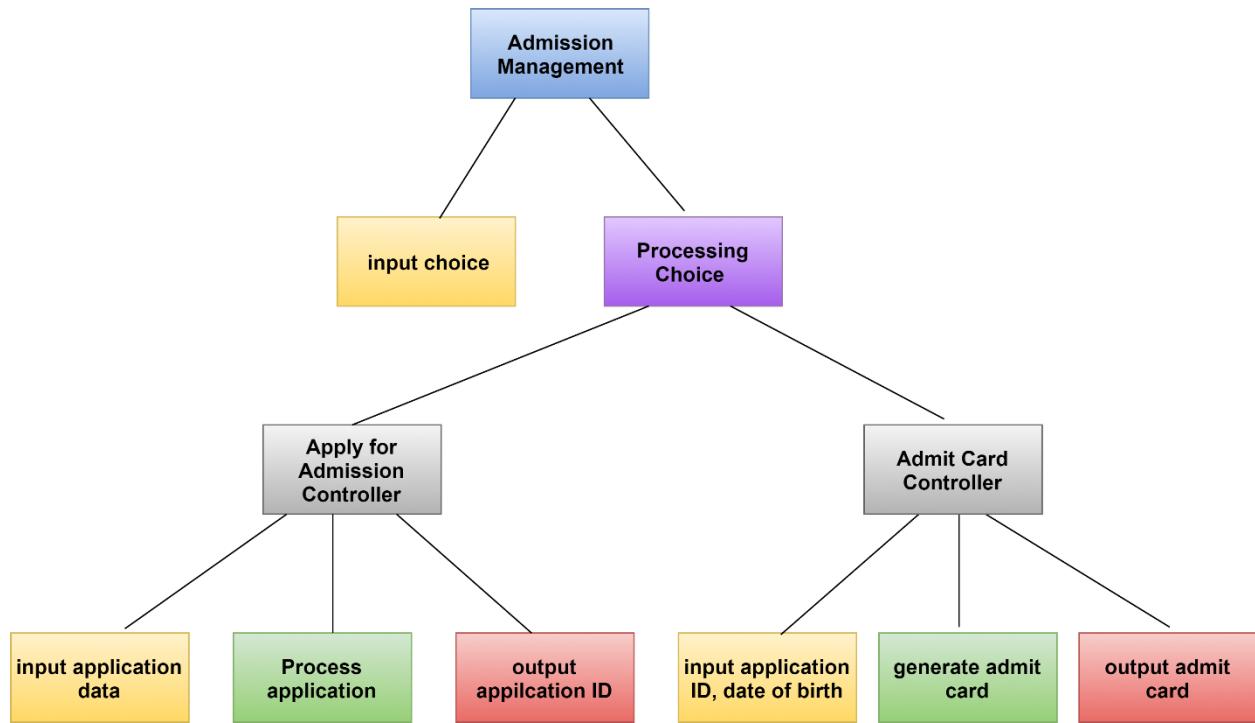


Figure 32 DFD mapping of Applicant's Dataflow Diagram of Admission System

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

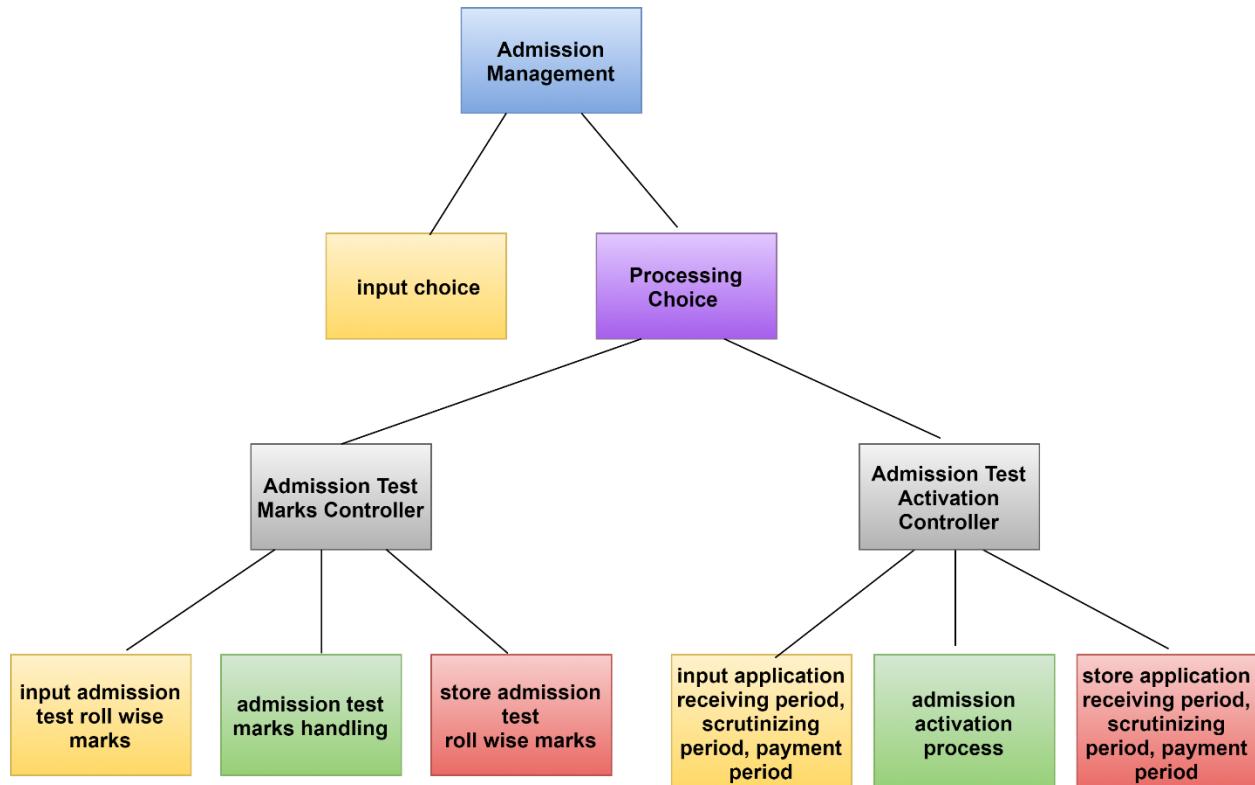


Figure 33 DFD mapping of Exam Controller's Dataflow Diagram of Admission System

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

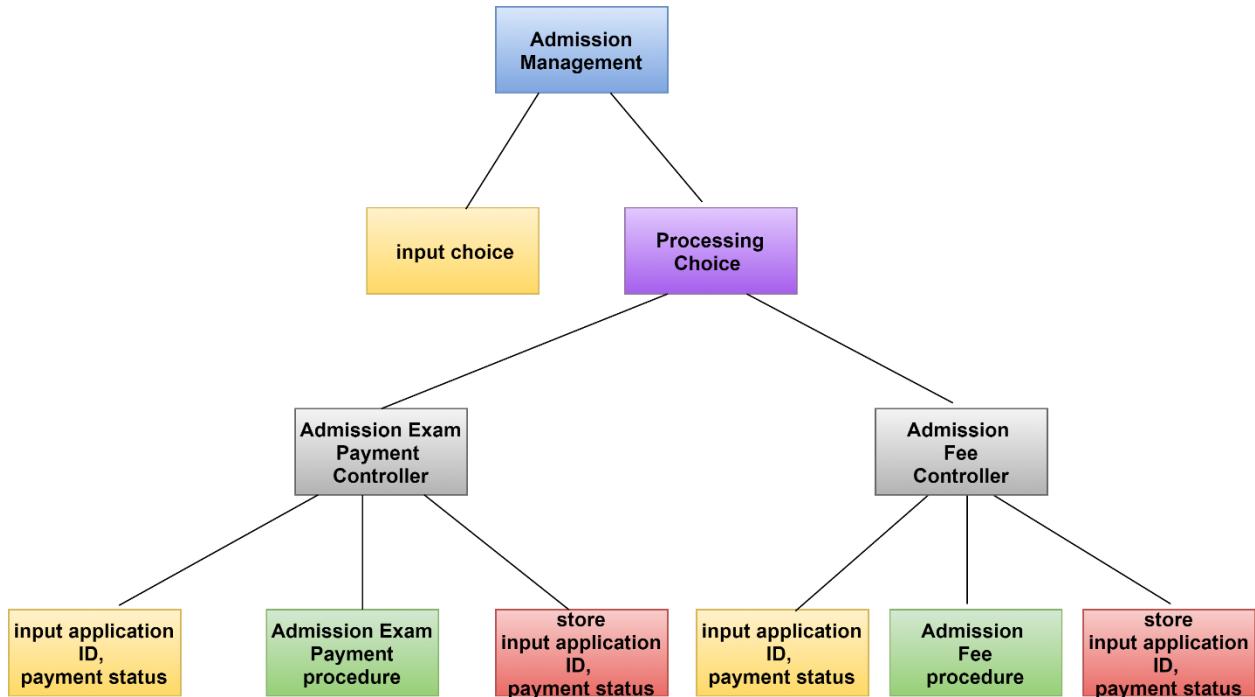


Figure 34 DFD mapping of Receptionist's Dataflow Diagram of Admission System

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

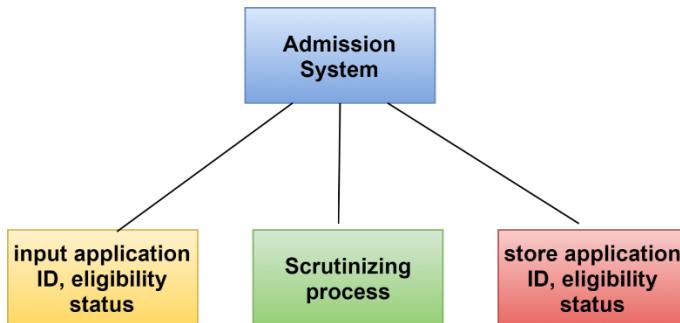


Figure 35 DFD mapping of Scrutinizer's Dataflow Diagram of Admission System

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

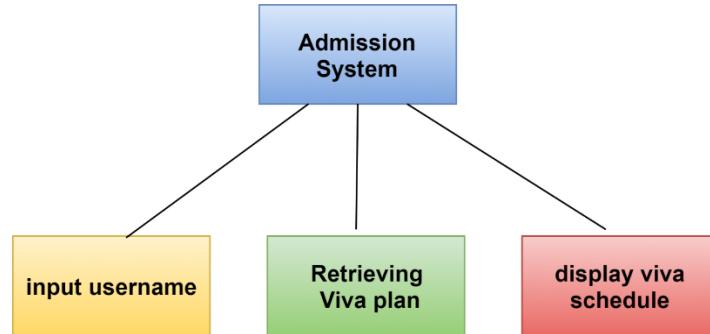


Figure 36 DFD mapping of Teacher's Dataflow Diagram of Admission System

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

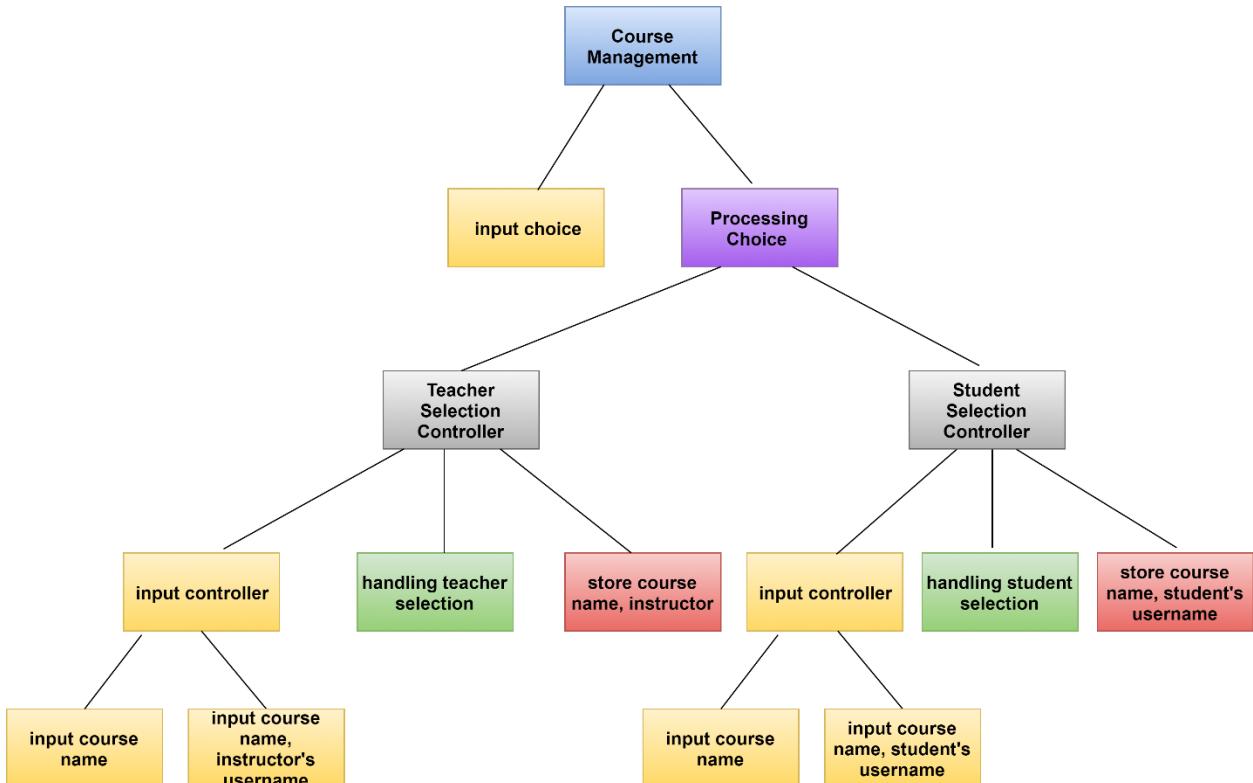


Figure 37 DFD mapping of Program Chairperson's Dataflow Diagram of Course Management System.

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

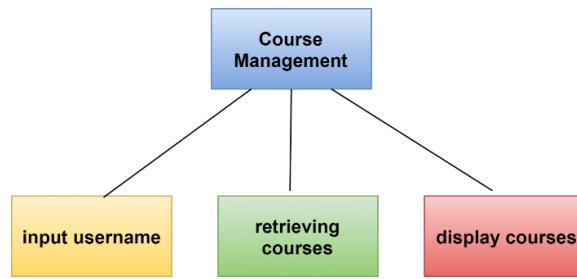


Figure 38 Teacher's Dataflow Diagram of Course Management System

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

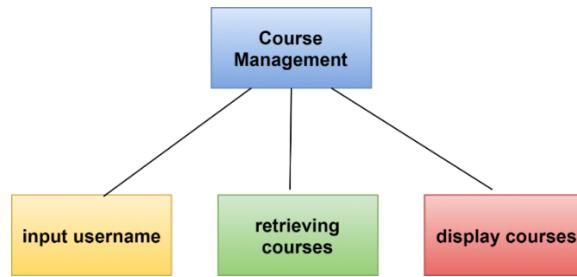


Figure 39 DFD mapping of Student's Dataflow Diagram of Course Management System

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

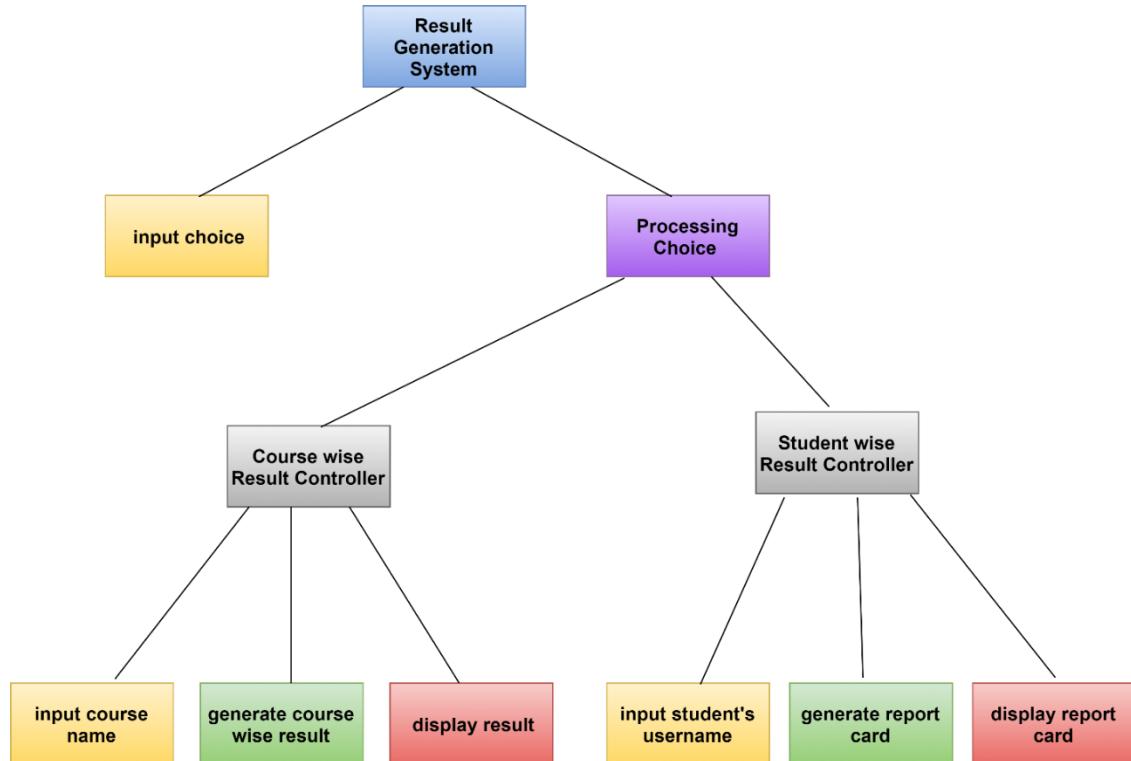


Figure 40 DFD mapping of Program Chairperson's Dataflow Diagram of Result Generation System

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

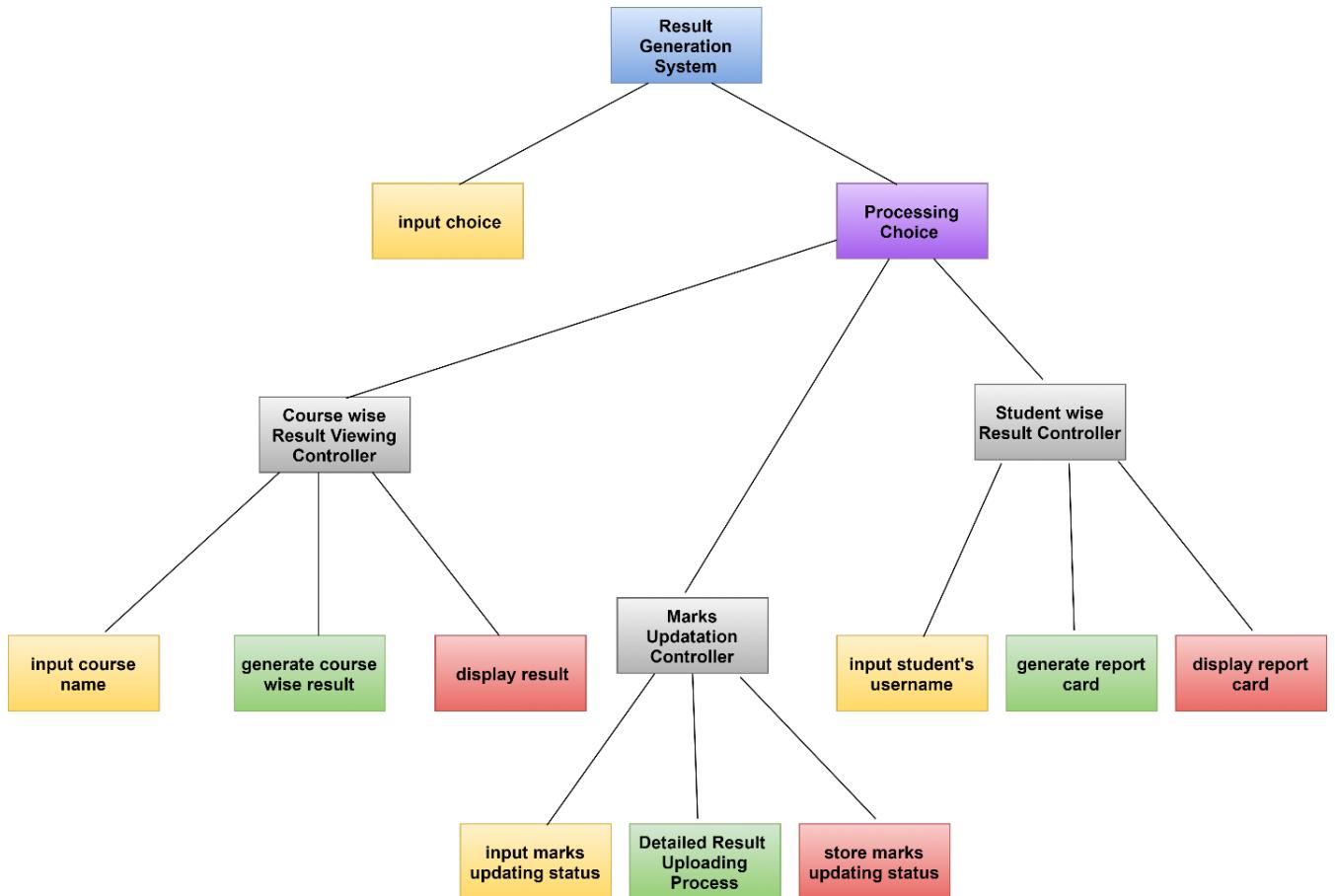


Figure 41 DFD mapping of Exam Controllers's Dataflow Diagram of Result Generation System

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

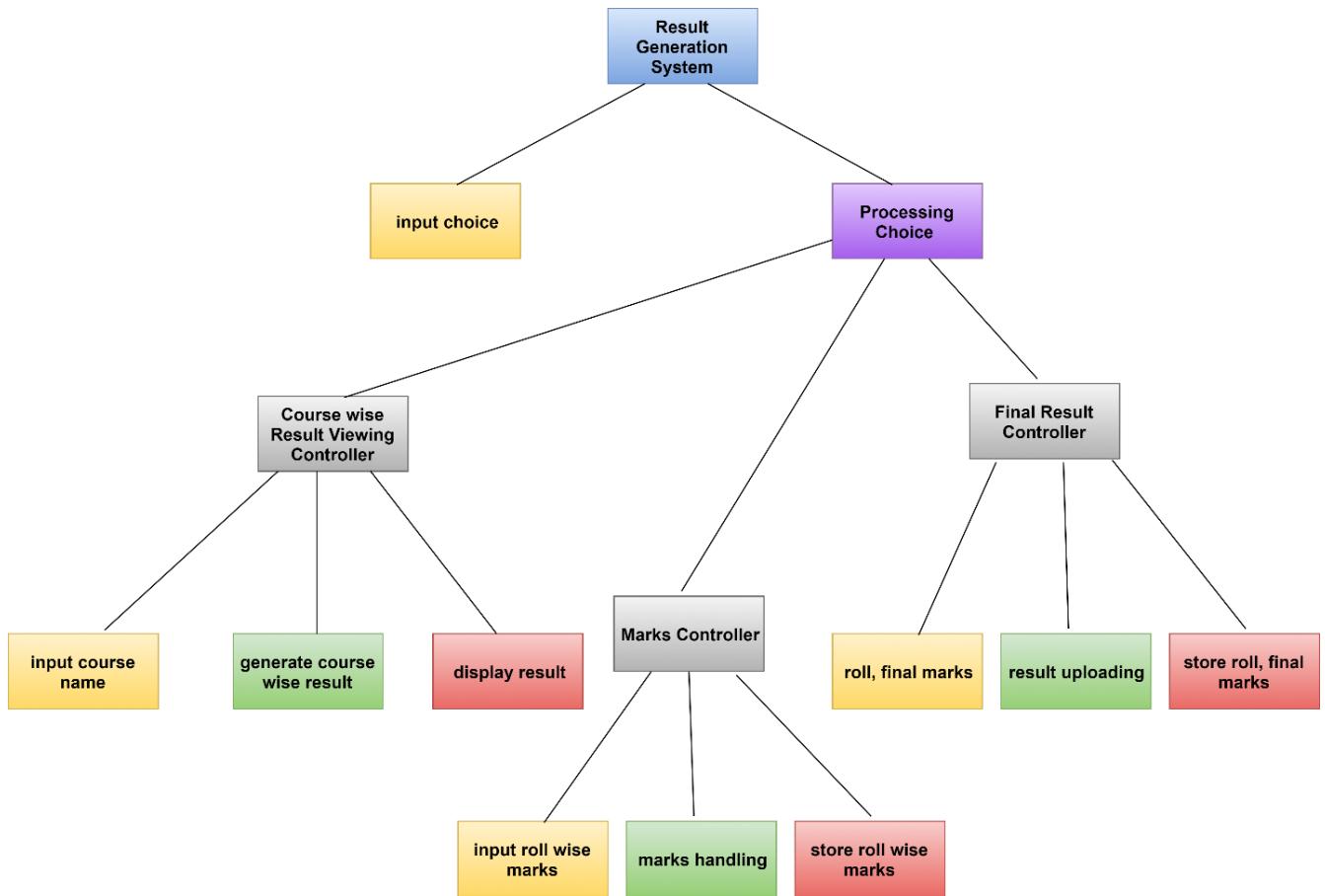


Figure 42 DFD mapping of Teacher's Dataflow Diagram of Result Generation System

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

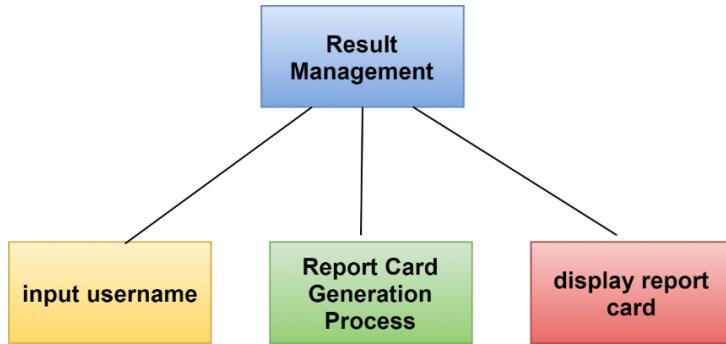


Figure 43 DFD mapping of Student's Dataflow Diagram of Result Generation System

## 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:

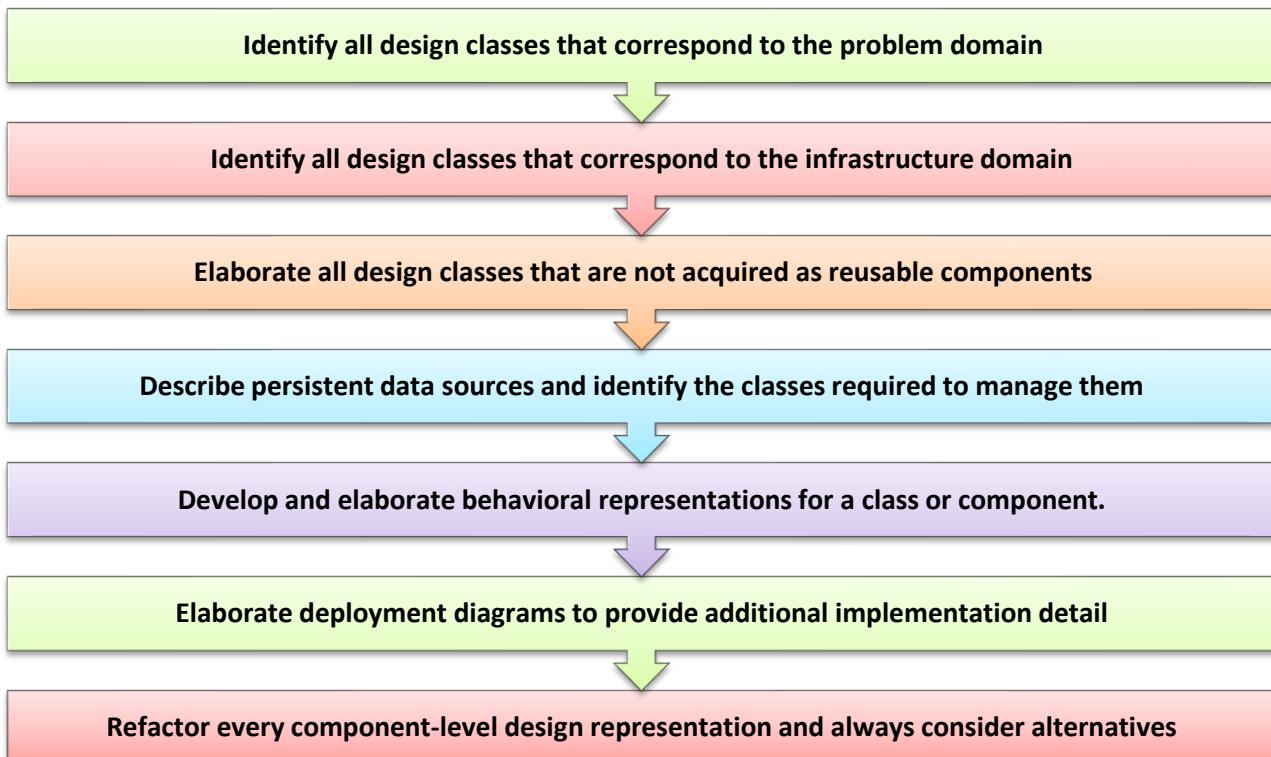


Figure 44 Steps of Component-Level Design

### 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.

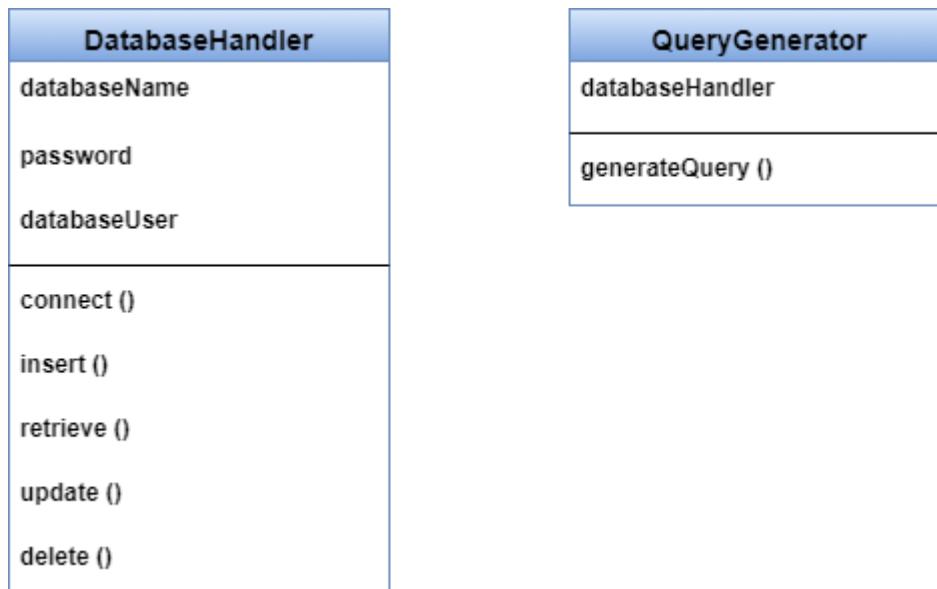


Figure 46 Design classes

### 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.

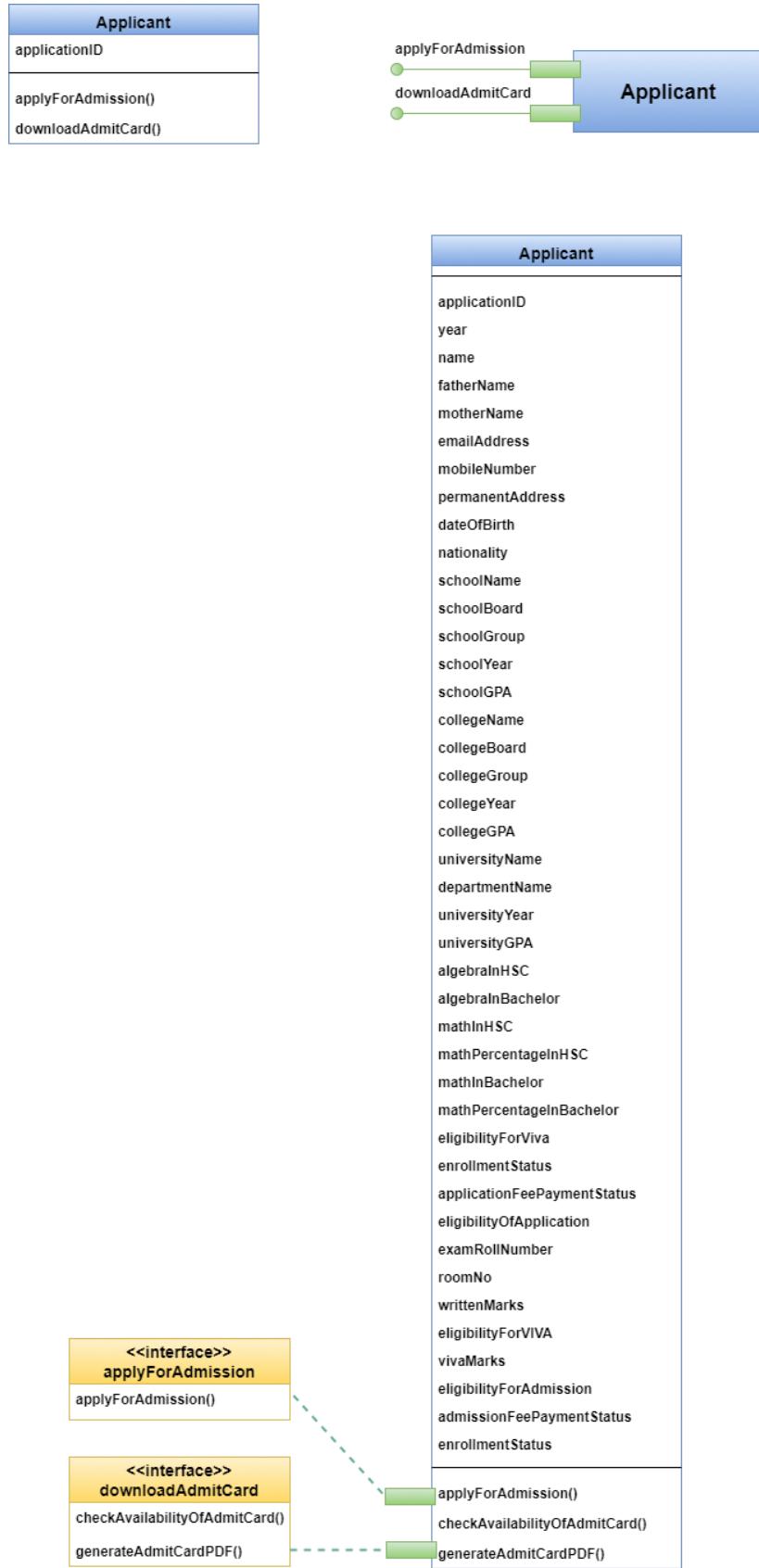


Figure 47 Elaborated Applicant class

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

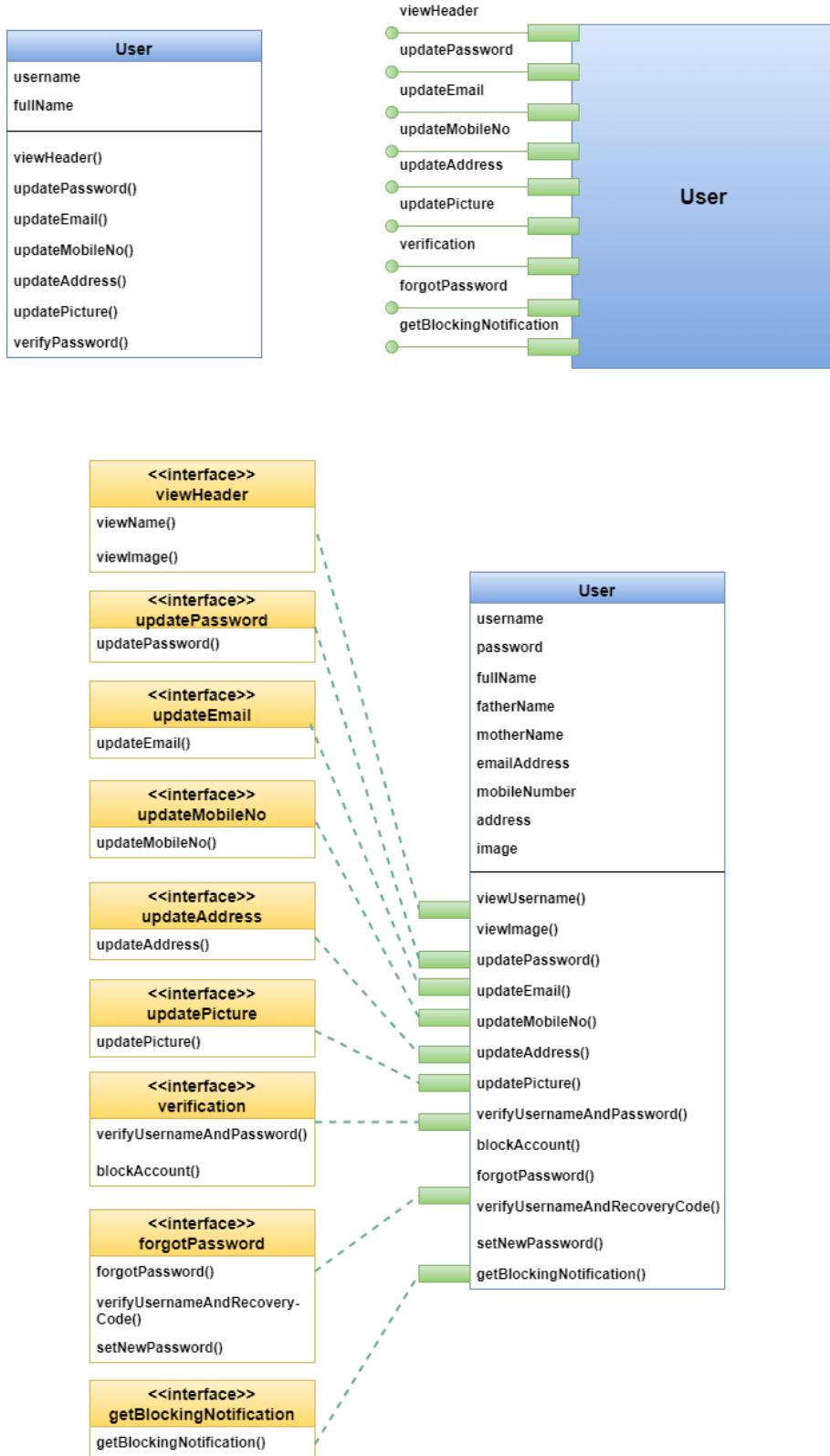


Figure 48 Elaborated User class

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

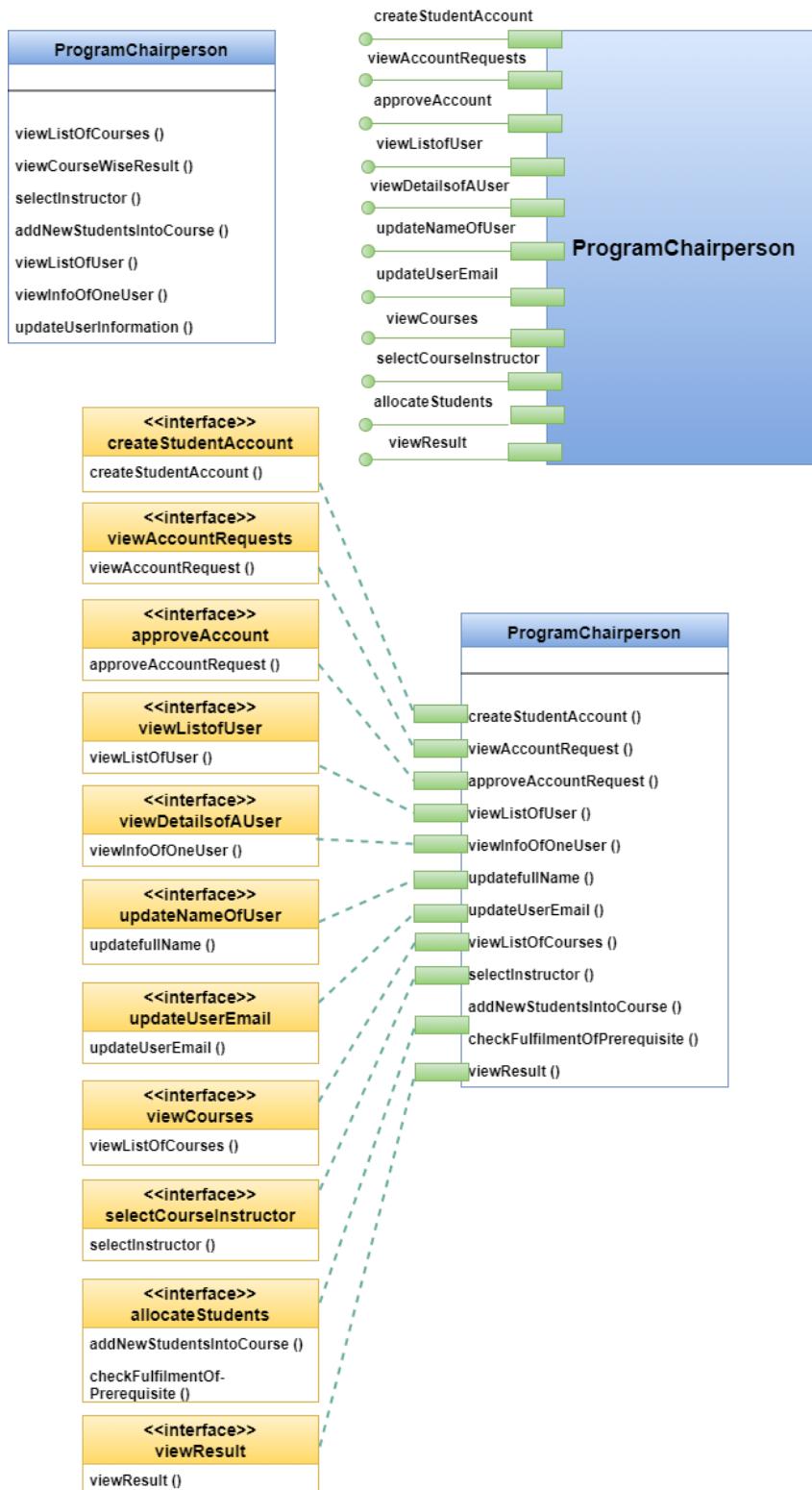


Figure 49 Elaborated ProgramChairperson class

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

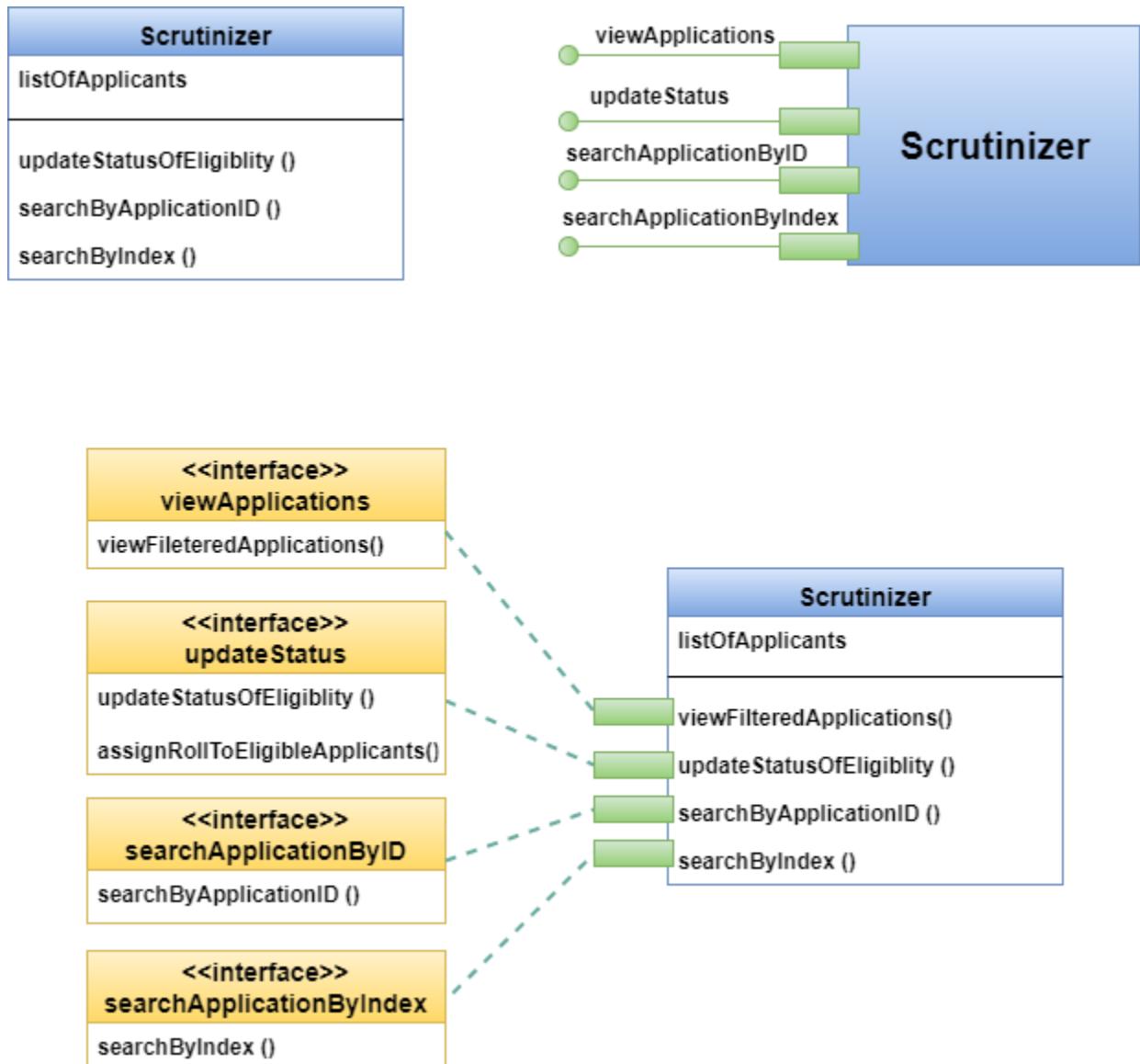


Figure 50 Elaborated Scrutinizer class

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

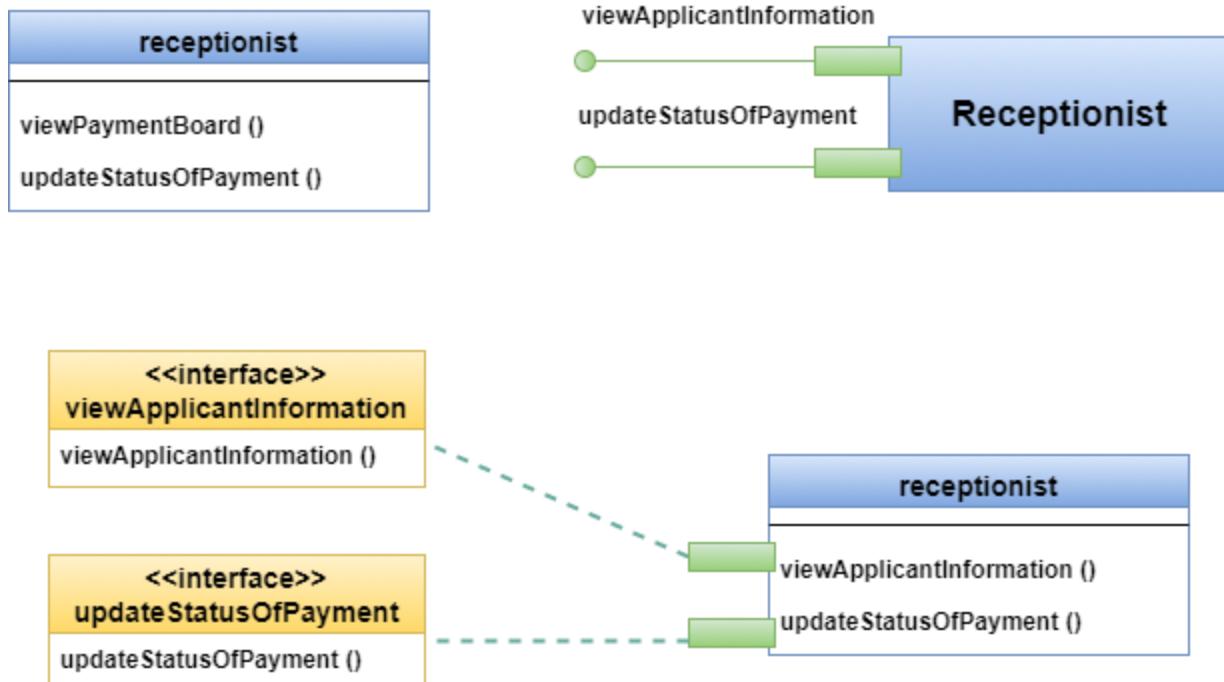


Figure 51 Elaborated Receptionist class

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



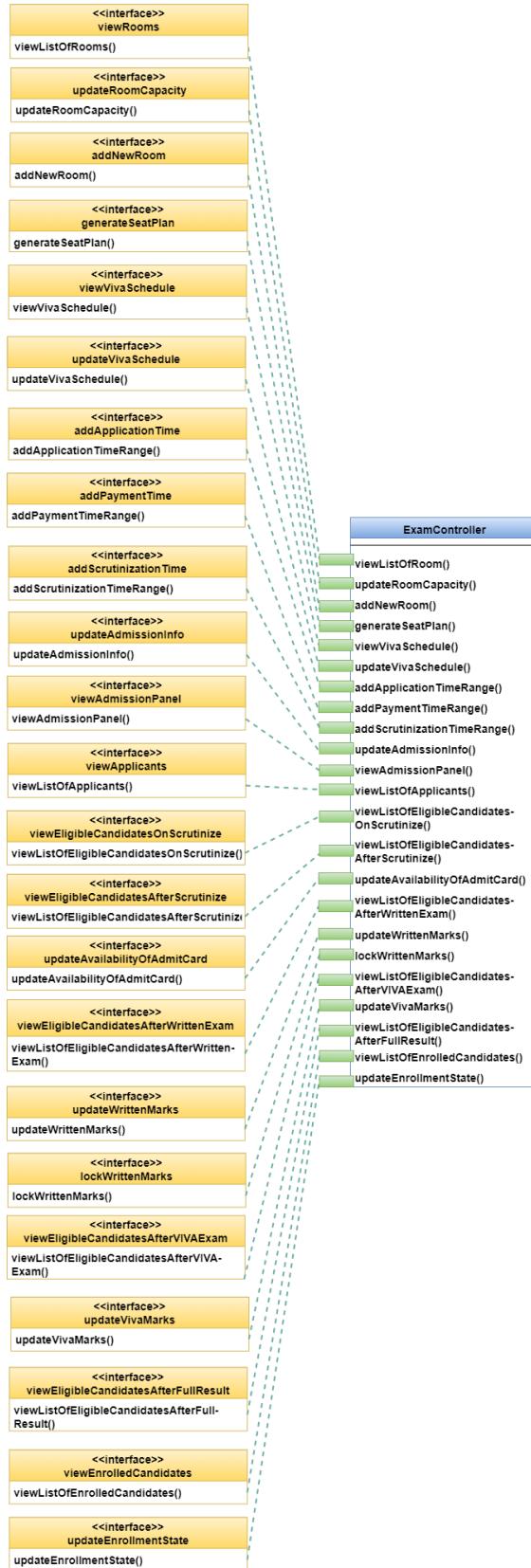
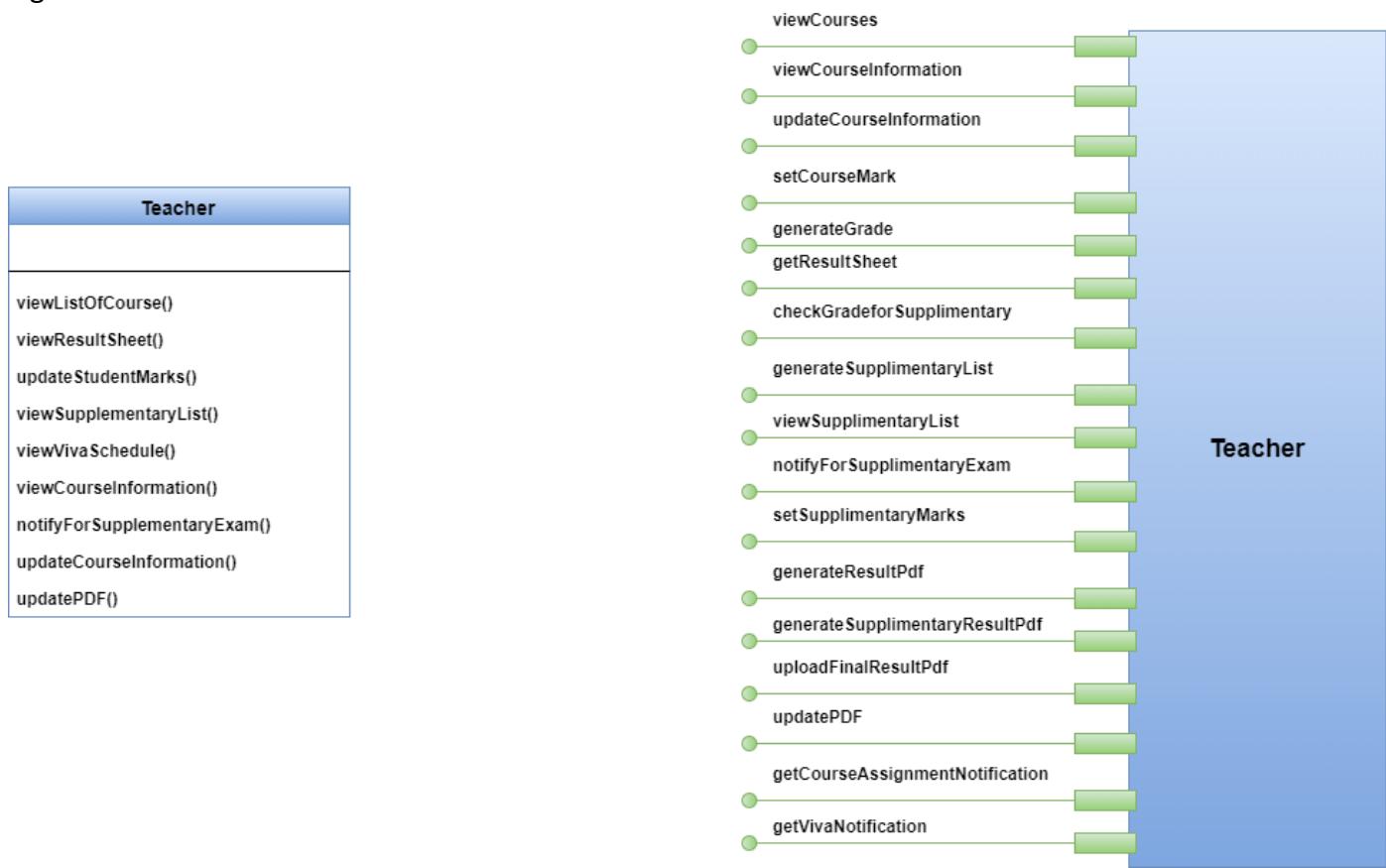


Figure 52 Elaborated ExamController class

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



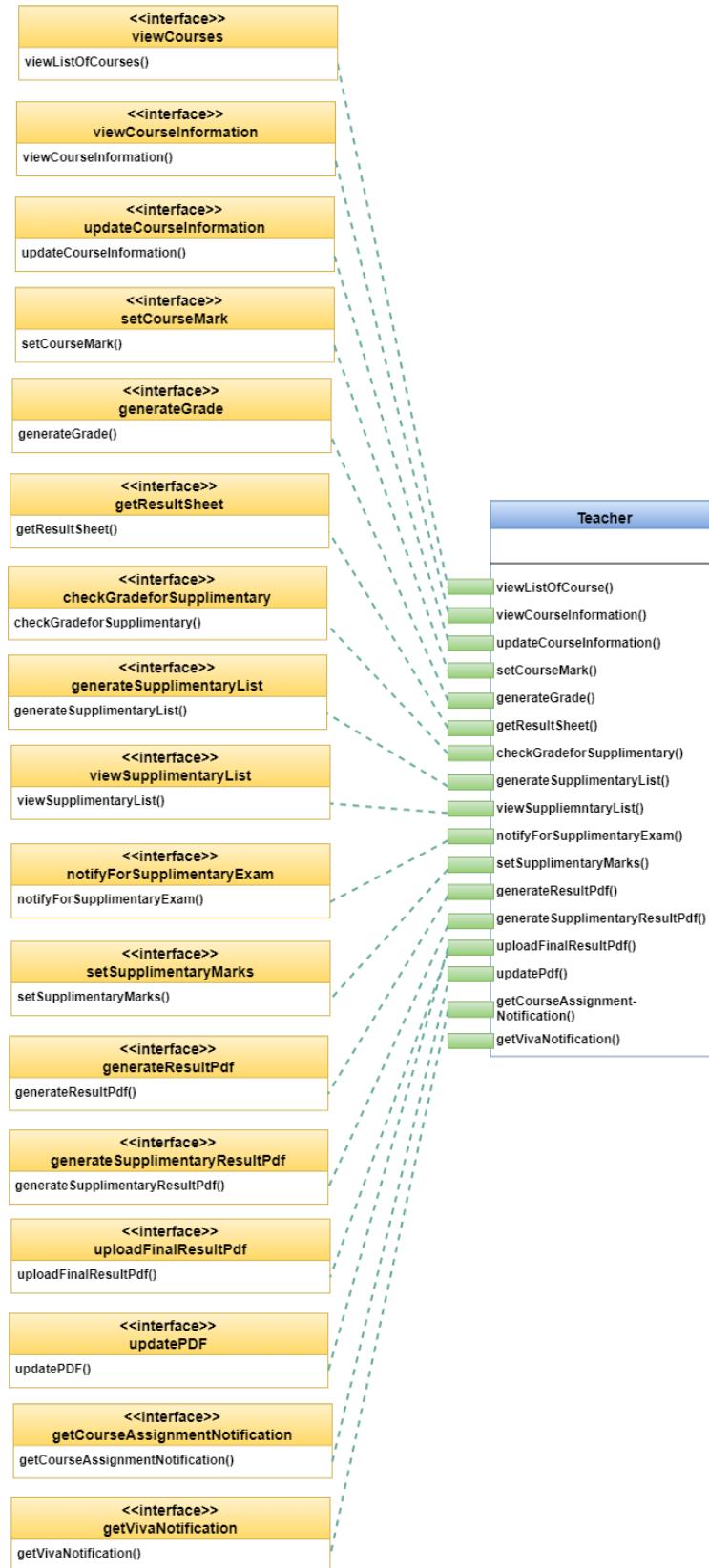


Figure 53 Elaborated Teacher class

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

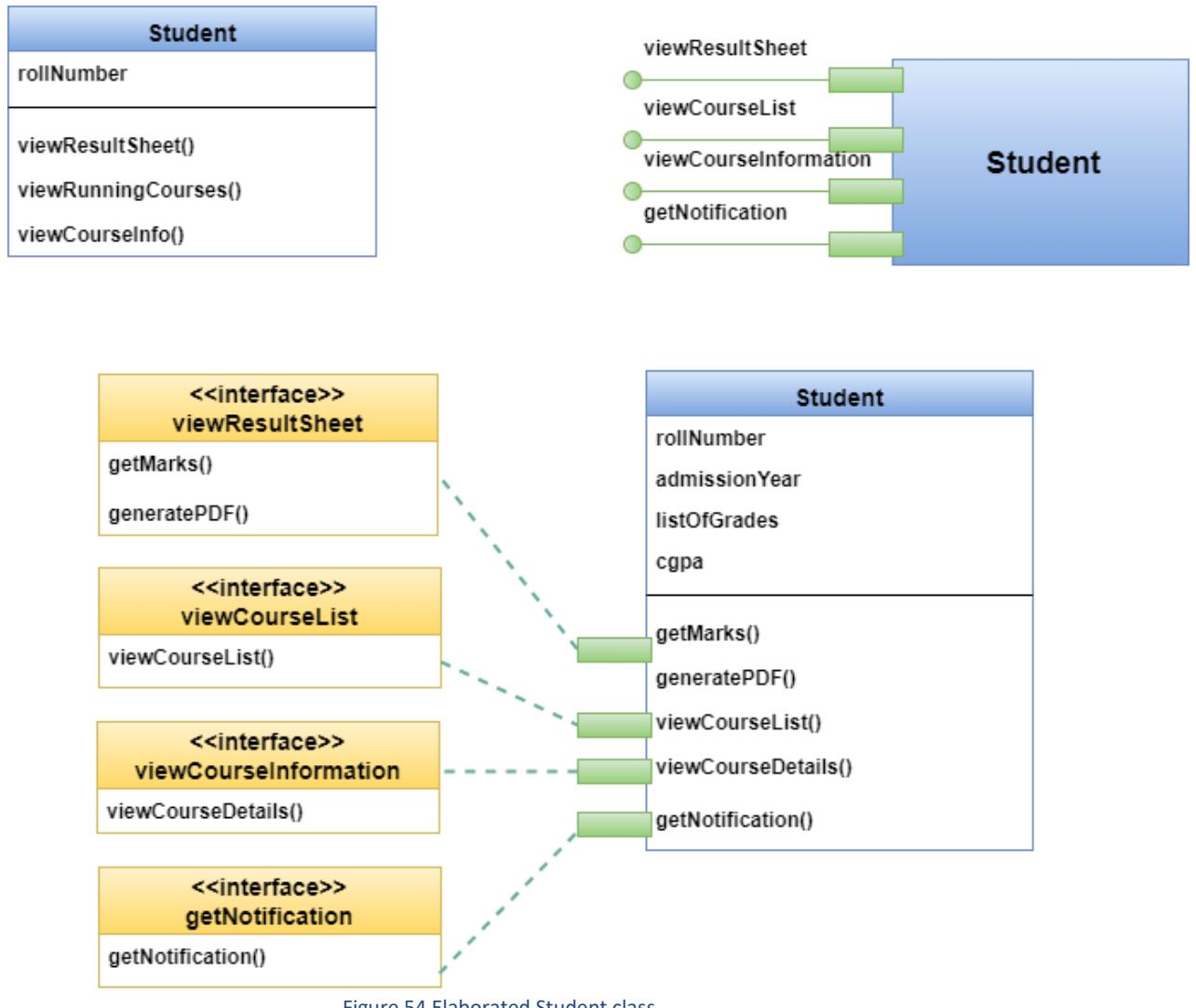


Figure 54 Elaborated Student class

### 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 55-56 shows collaboration diagrams for Applicant class.

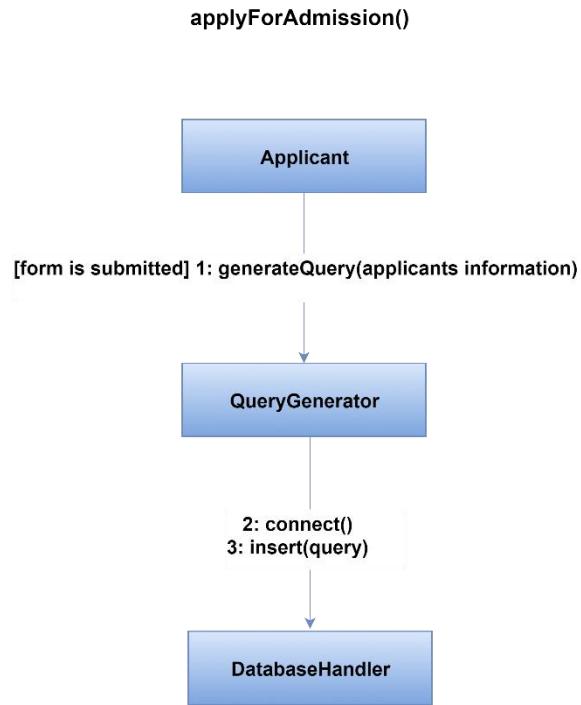


Figure 55 Method `applyForAdmission()` of Applicant Class

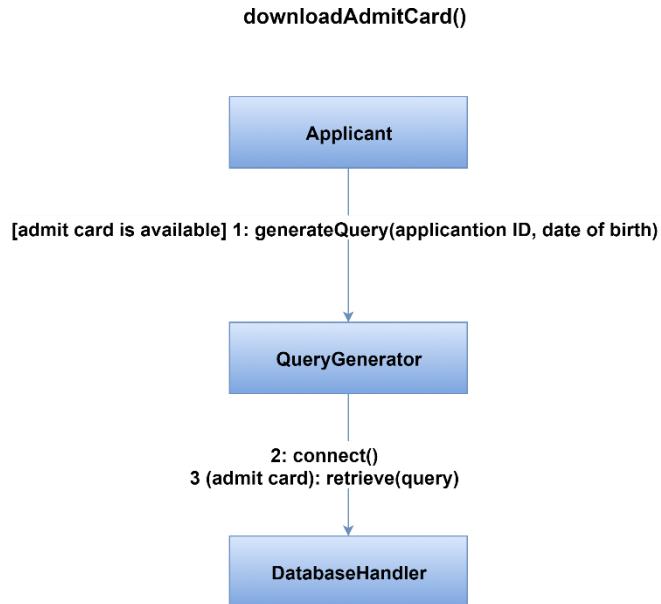


Figure 56 Method `downloadAdmitCard()` of Applicant Class

Figure 57-62 shows collaboration diagrams for ProgramChairperson class.

**checkFulfilmentOfPrerequisite ()**

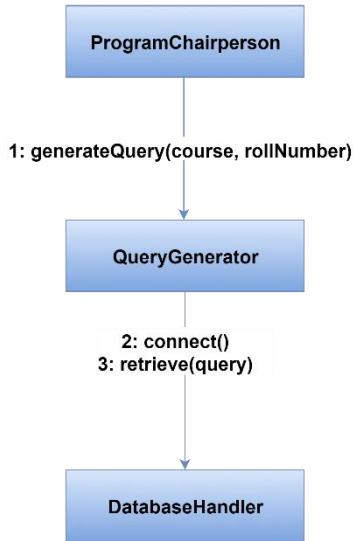


Figure 57 Method `checkFulfilmentOfPrerequisites()` of `ProgramChairperson` Class

**addNewStudentsIntoCourse ()**

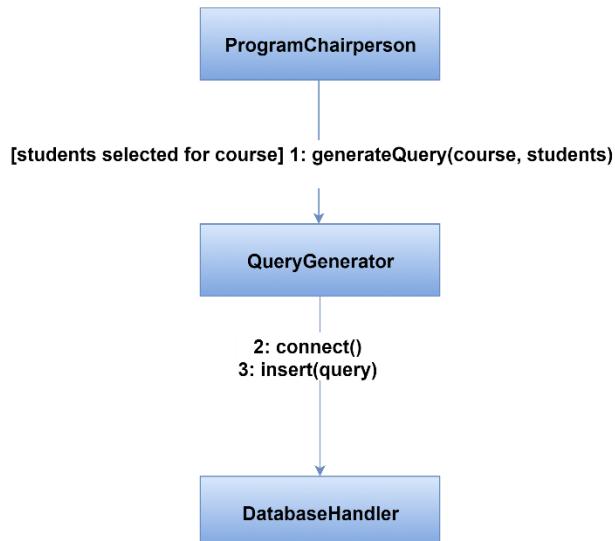


Figure 58 Method `addNewStudentsIntoCourse()` of `ProgramChairperson` Class

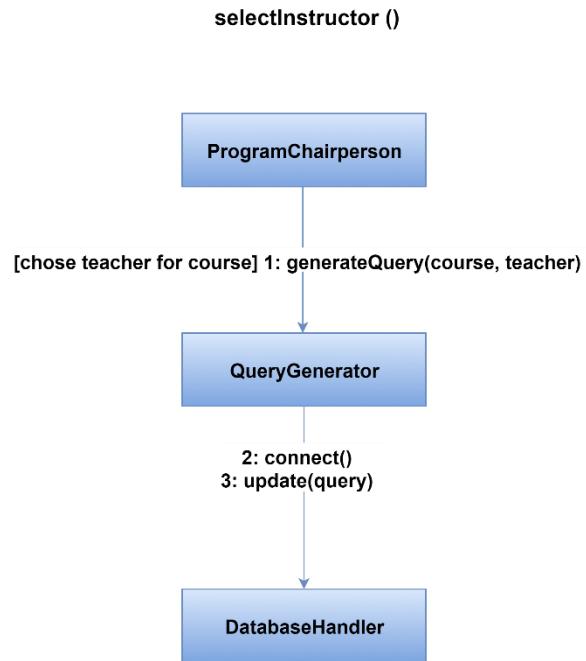


Figure 59 Method `selectInstructor()` of `ProgramChairperson` Class

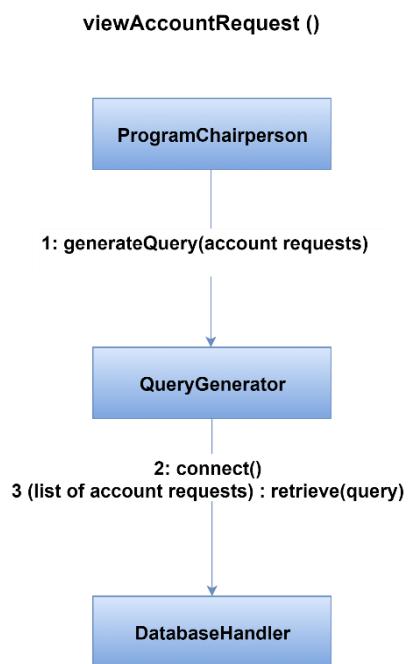


Figure 60 Method `viewAccountRequest()` of `ProgramChairperson` Class

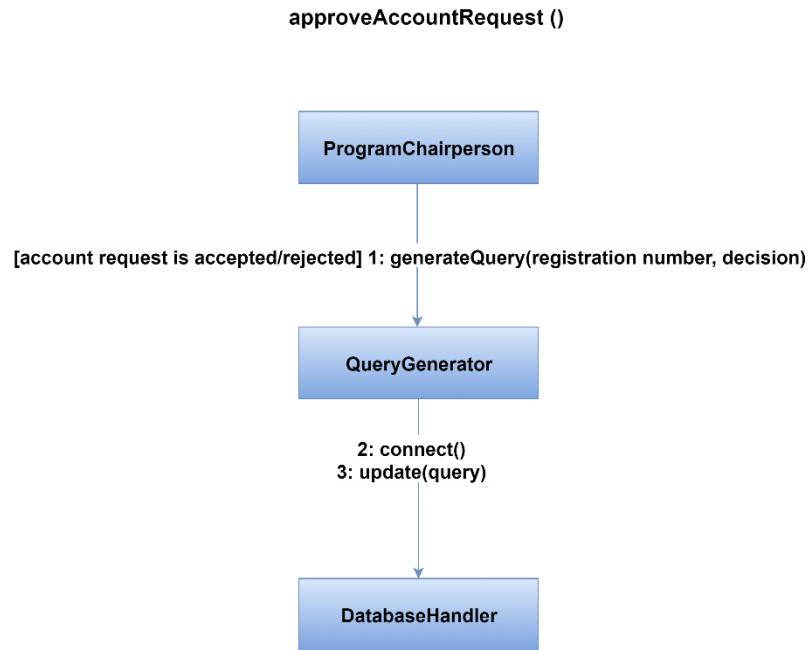


Figure 61 Method `approveAccountRequest()` of `ProgramChairperson` Class

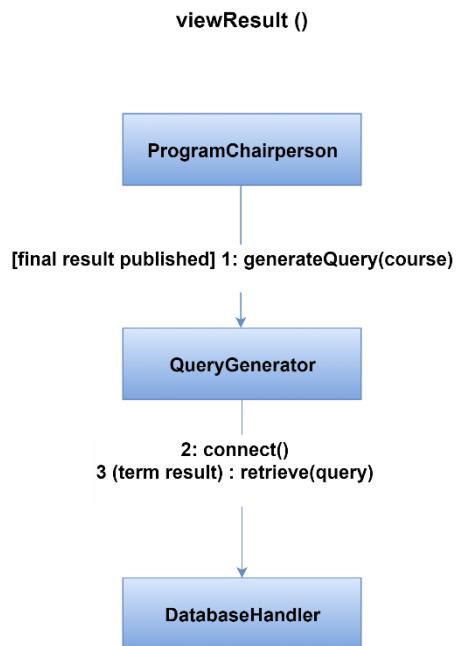


Figure 62 Method `viewResult()` of `ProgramChairperson` Class

Figure 63-64 shows collaboration diagrams for Scrutinizer class.

`viewFileteredApplications()`

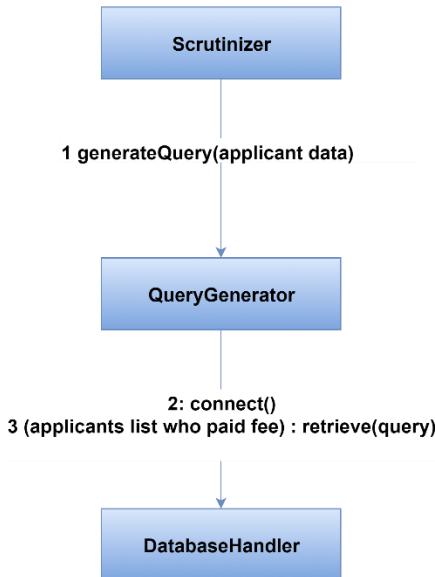


Figure 63 Method `viewFileteredApplications()` of Scrutinizer Class

`updateStatusOfEligibility ()`

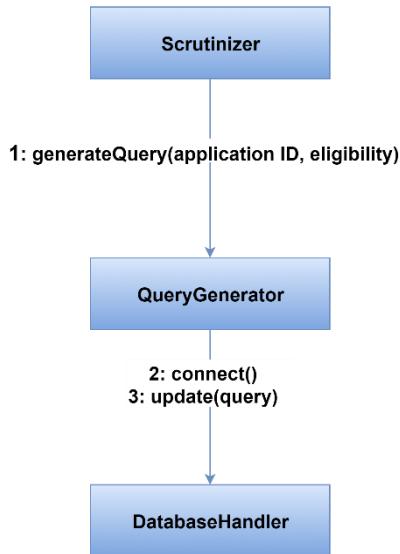


Figure 64 Method `updateStatusOfEligibility()` of Scrutinizer Class

Figure 65-66 shows collaboration diagrams for Receptionist class.

**viewApplicantInformation ()**

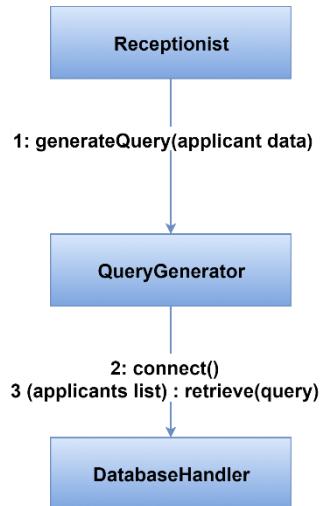


Figure 65 Method `viewApplicantInformation()` of `Receptionist` Class

**updateStatusOfPayment ()**

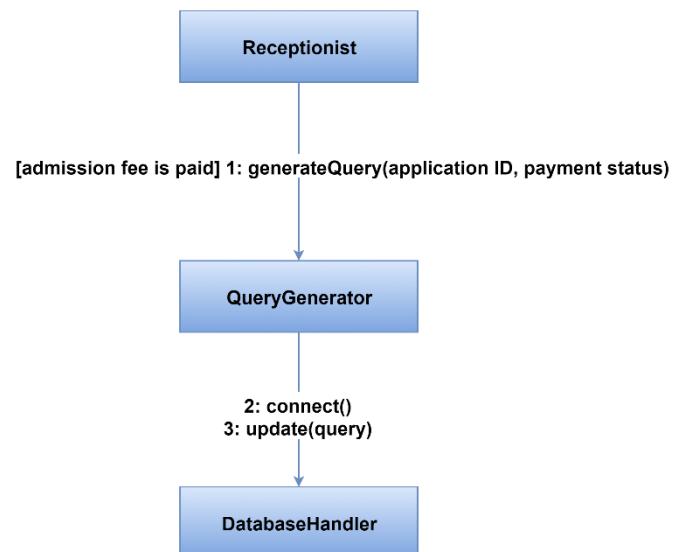


Figure 66 Method `updateStatusOfPayment()` of `Receptionist` Class

Figure 67-73 shows collaboration diagrams for `ExamController` class.

**addApplicationTimeRange()**

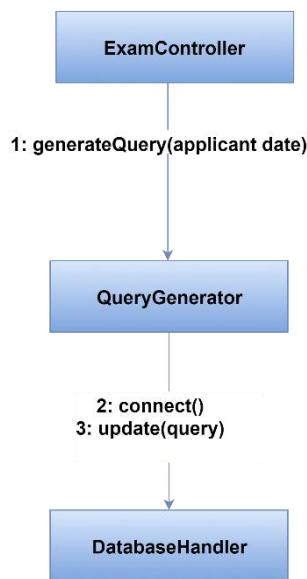


Figure 67 Method `addApplicationTimeRange()` of `ExamController` Class

**addPaymentTimeRange()**

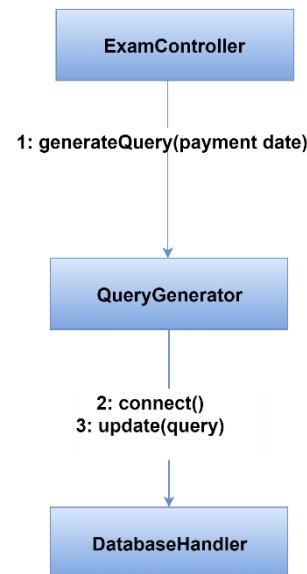


Figure 68 Method `addPaymentTimeRange()` of `ExamController` Class

**addScrutinizationTimeRange()**

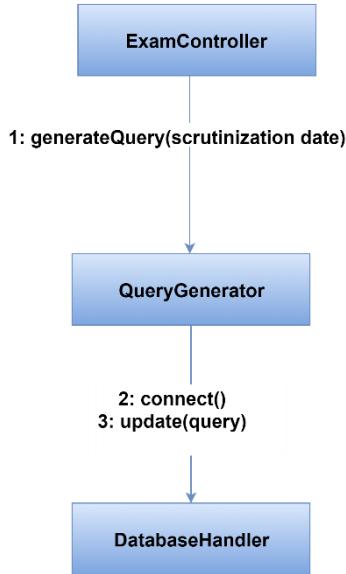


Figure 69 Method `addScrutinizationTimeRange()` of `ExamController` Class

**generateSeatPlan()**

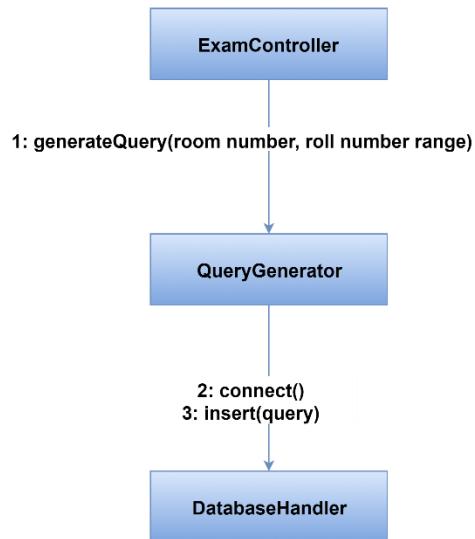


Figure 70 Method `generateSeatPlan()` of `ExamController` Class

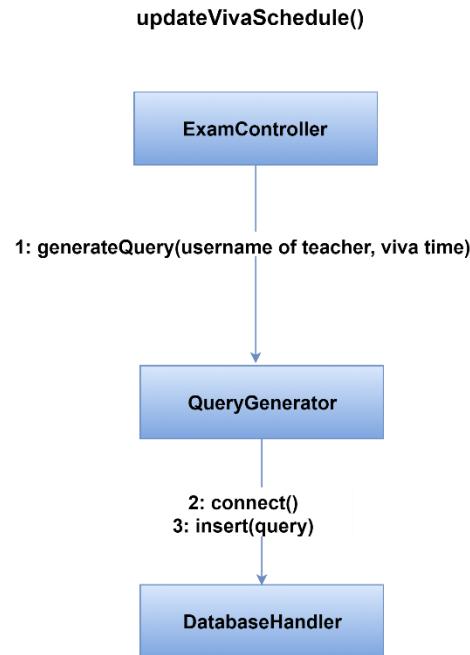


Figure 71 Method updateVivaSchedule() of ExamController Class

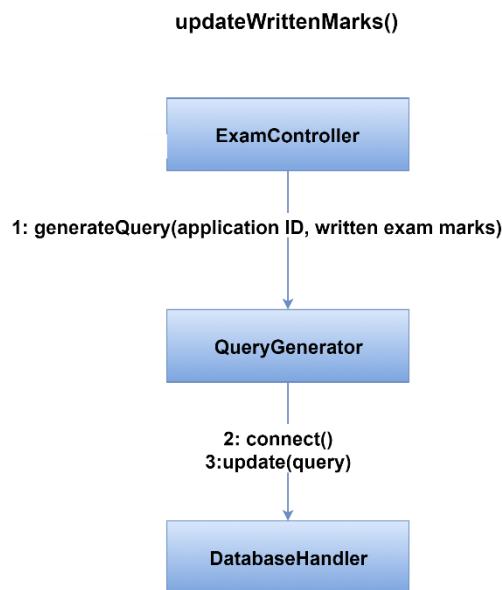


Figure 72 Method updateWrittenMarks() of ExamController Class

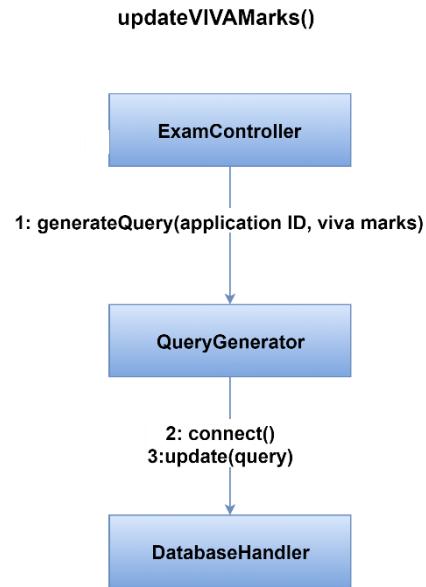


Figure 73 Method `updateVIVAMarks()` of ExamController Class

Figure 74-86 shows collaboration diagrams for Teacher class.

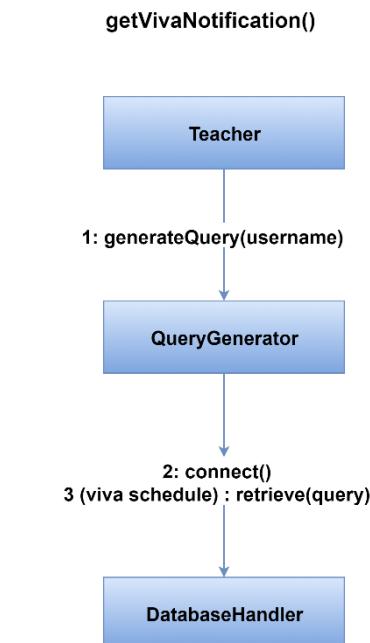


Figure 74 Method `getVivaNotification()` of Teacher Class

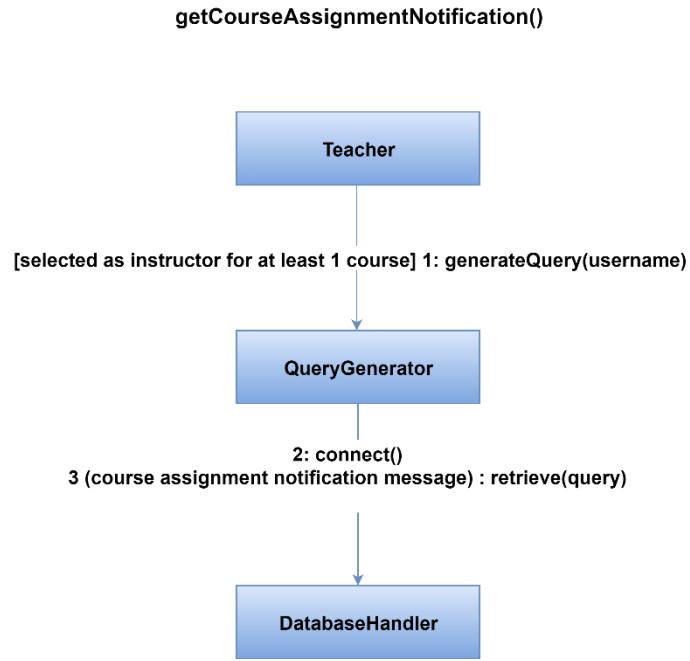


Figure 75 Method `getCourseAssignmentNotification()` of Teacher Class

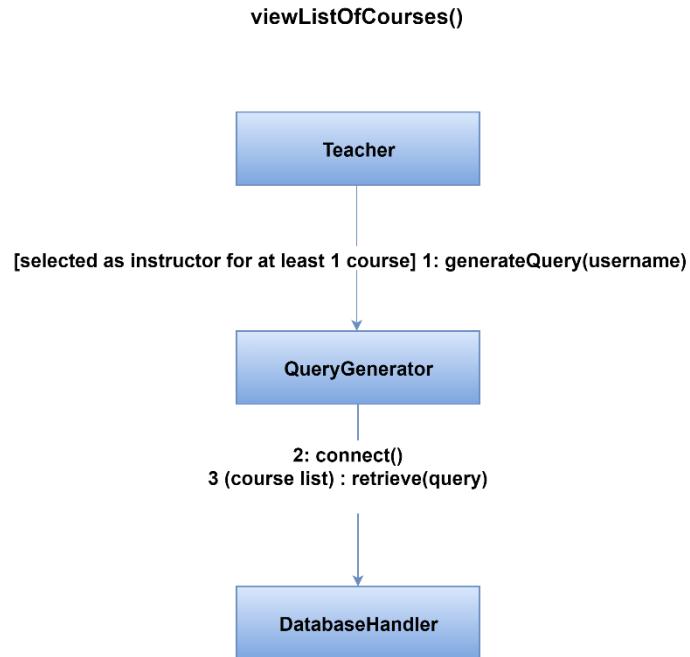


Figure 76 Method `viewListOfCourses()` of Teacher Class

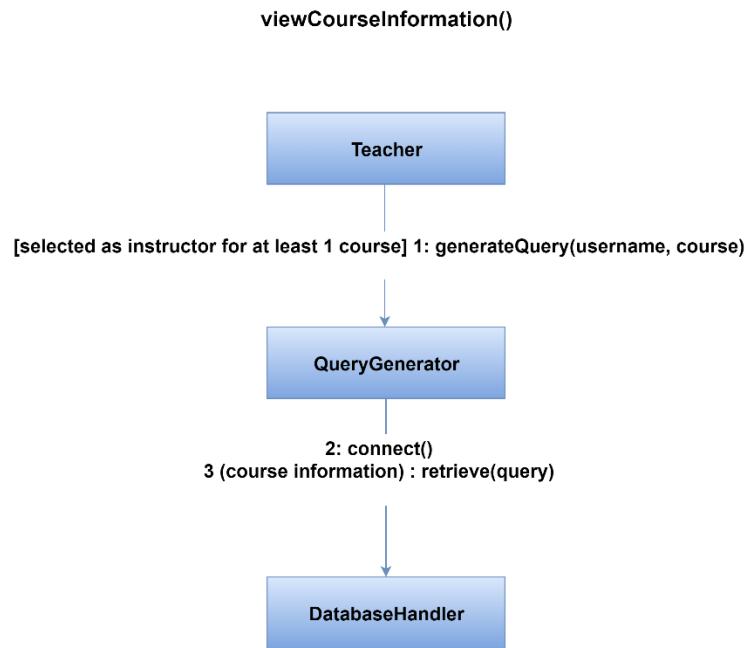


Figure 77 Method `viewCourseInformation()` of Teacher Class

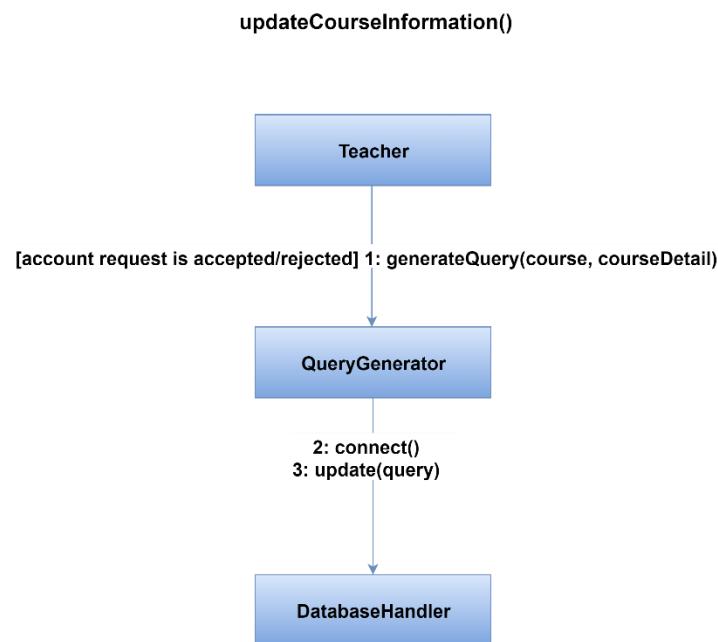


Figure 78 Method `updateCourseInformation()` of Teacher Class

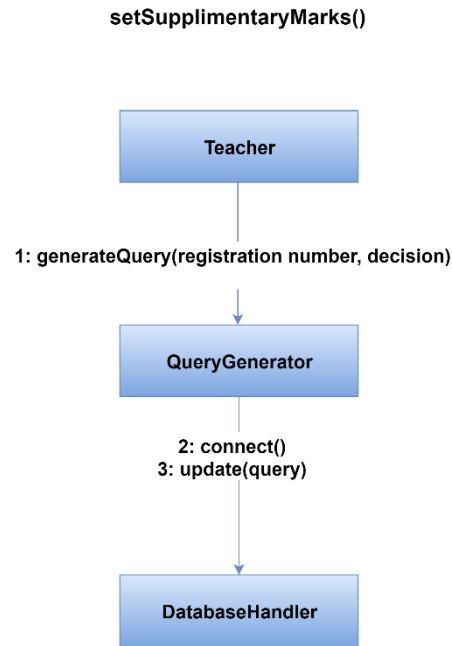


Figure 79 Method `setSupplementaryMarks()` of Teacher Class

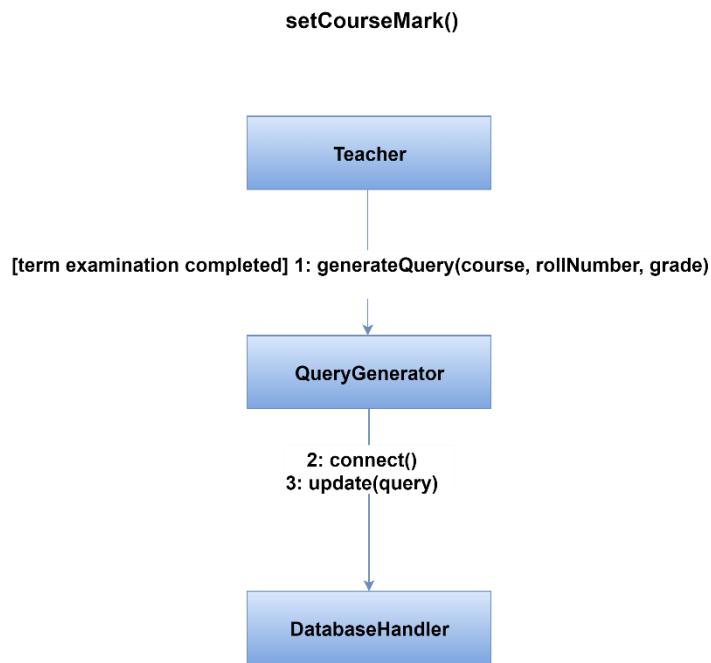


Figure 80 Method `setCourseMark()` of Teacher Class

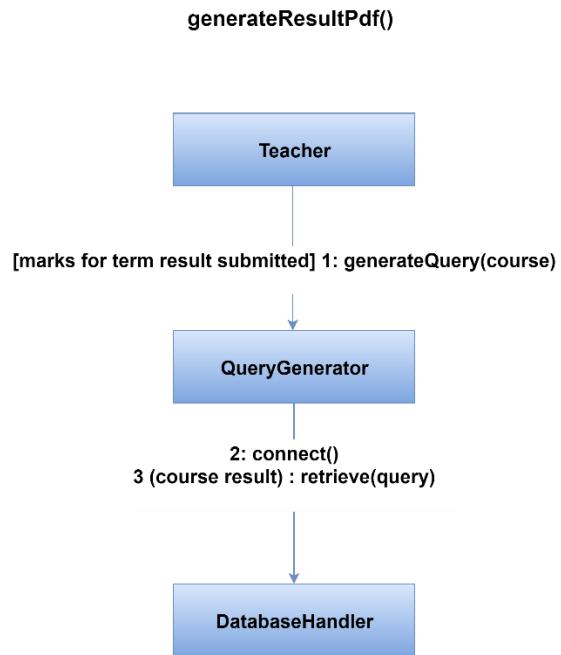


Figure 81 Method generateResultPdf() of Teacher Class

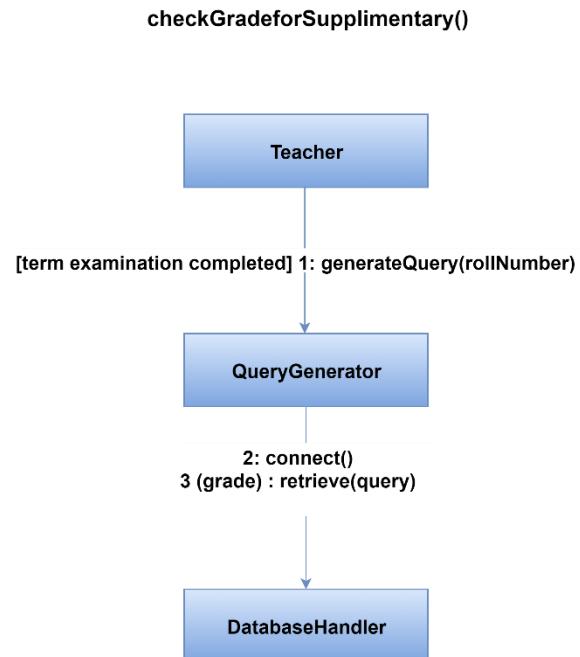


Figure 82 Method checkGradeForSupplementary() of Teacher Class

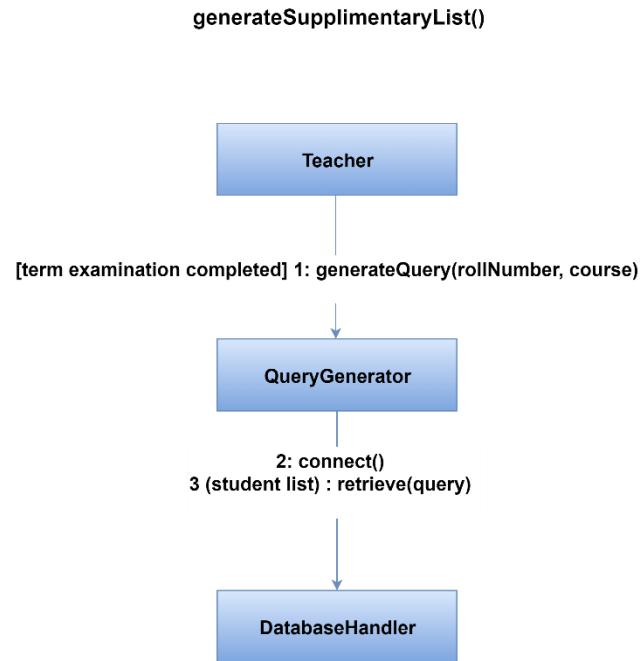


Figure 83 Method `generateSupplementaryList()` of Teacher Class

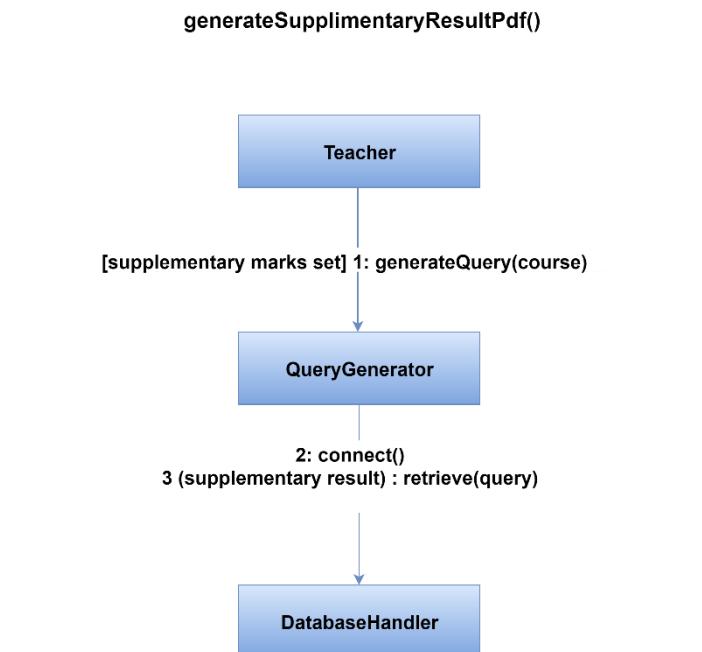


Figure 84 Method `generateSupplementaryResultPdf()` of Teacher Class

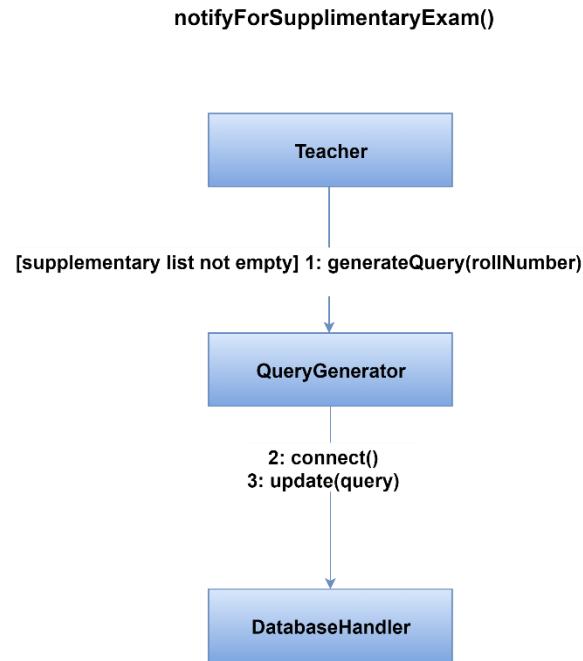


Figure 85 Method `notifyForSupplementaryExam()` of Teacher Class

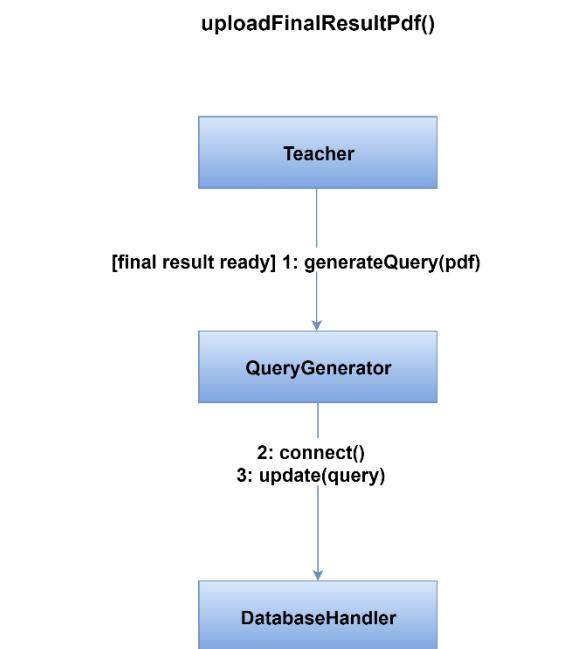


Figure 86 Method `uploadFinalResultPdf()` of Teacher Class

Figure 87-90 shows collaboration diagrams for Student class.

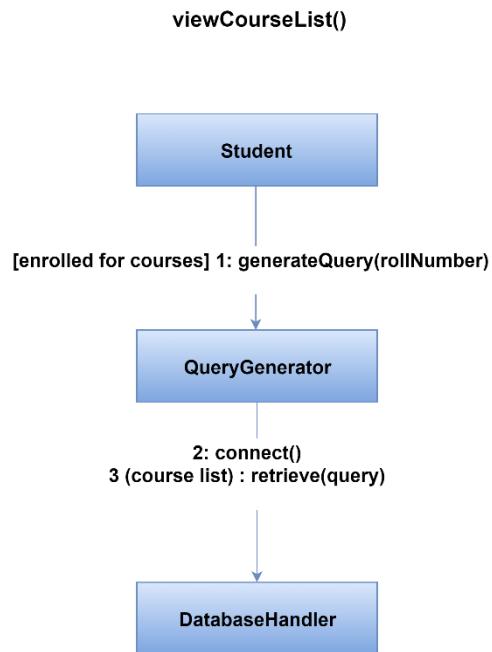


Figure 87 Method `viewCourseList()` of Student Class

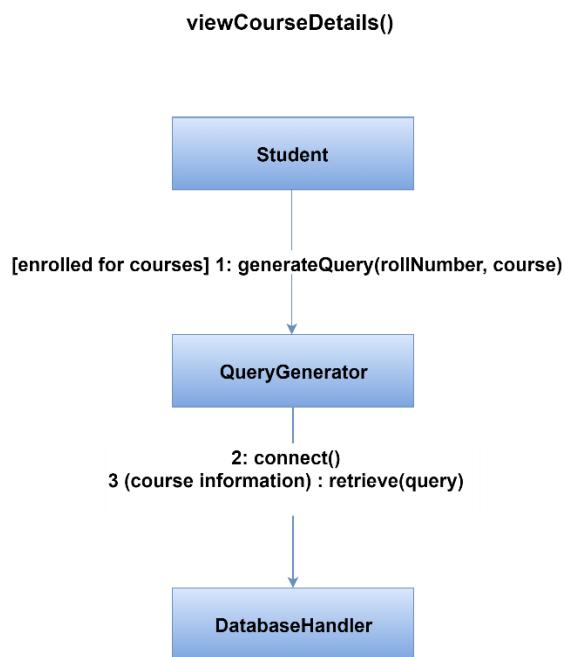


Figure 88 Method `viewCourseDetails()` of Student Class

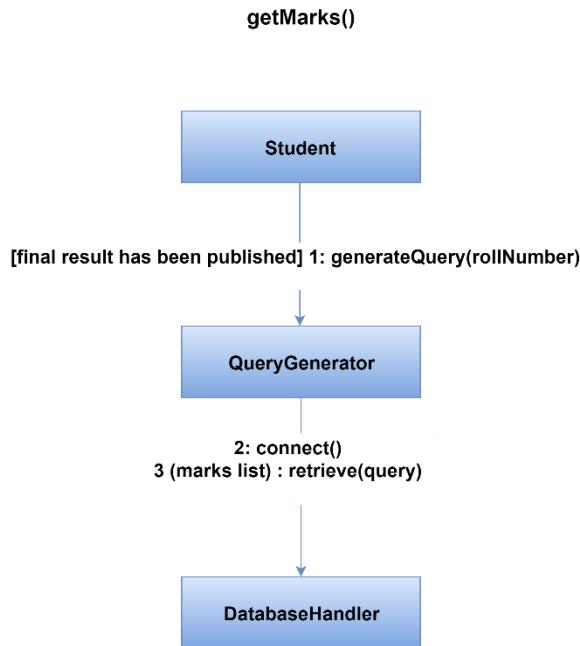


Figure 89 Method `getMarks()` of Student Class

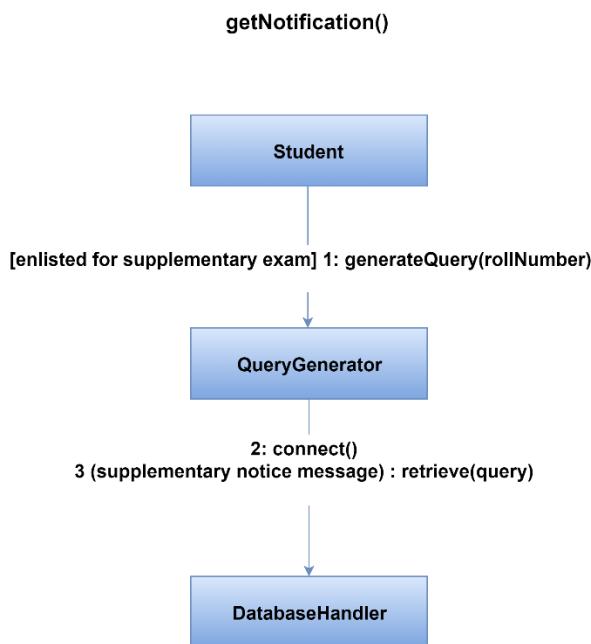


Figure 90 Method `getNotification()` of Student Class

### 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 91 shows Authentication class.

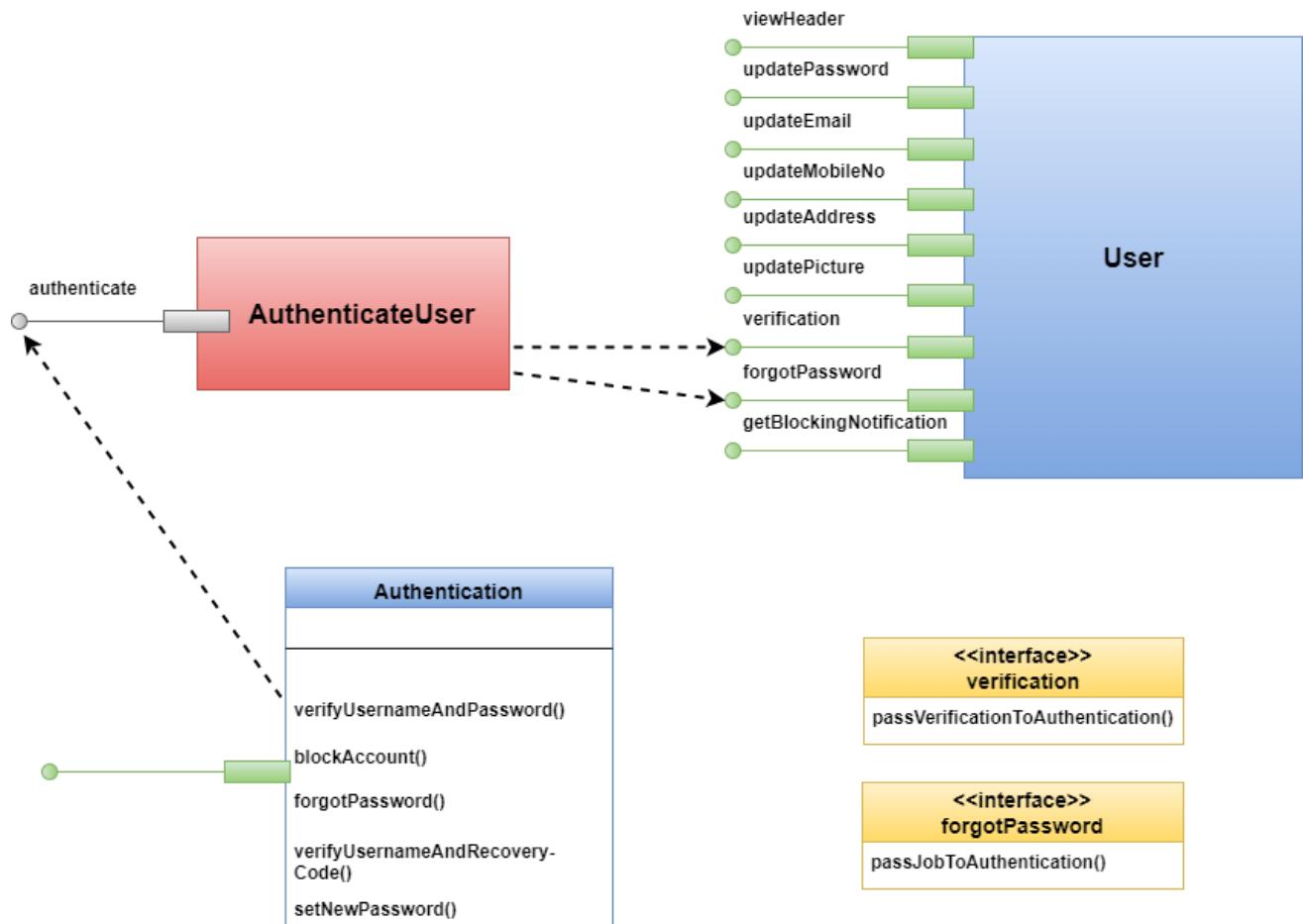


Figure 91 Authentication class

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 92 shows RollGenerator class.

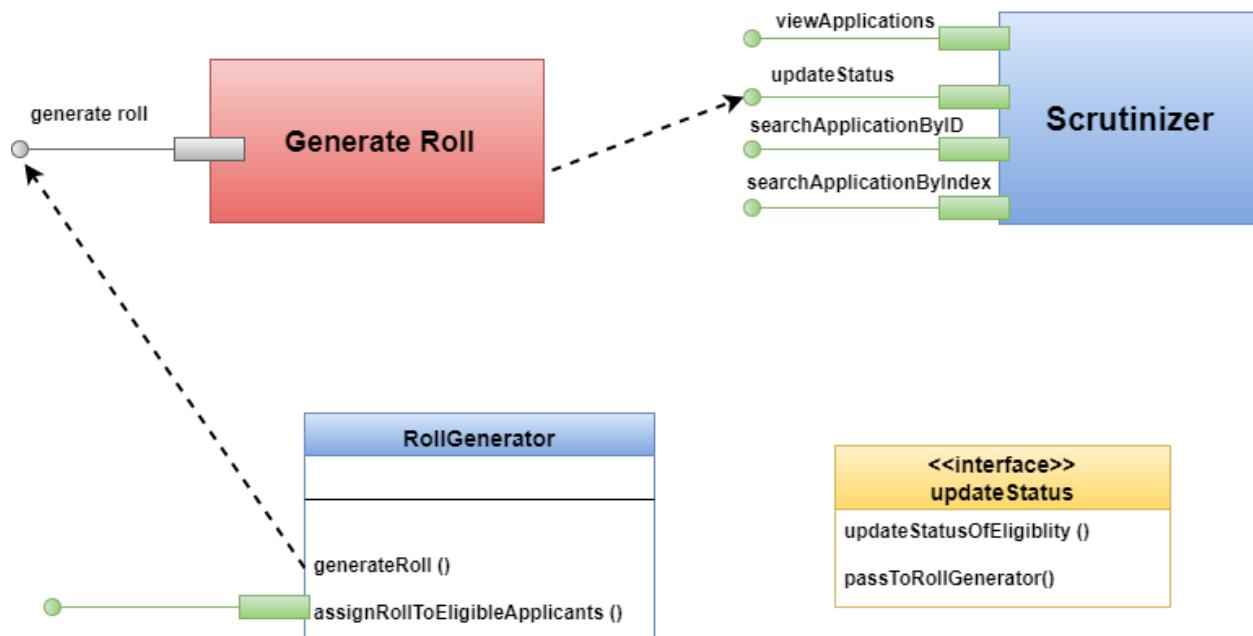


Figure 92 RollGenerator class

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 93 shows `ReportCard` class.

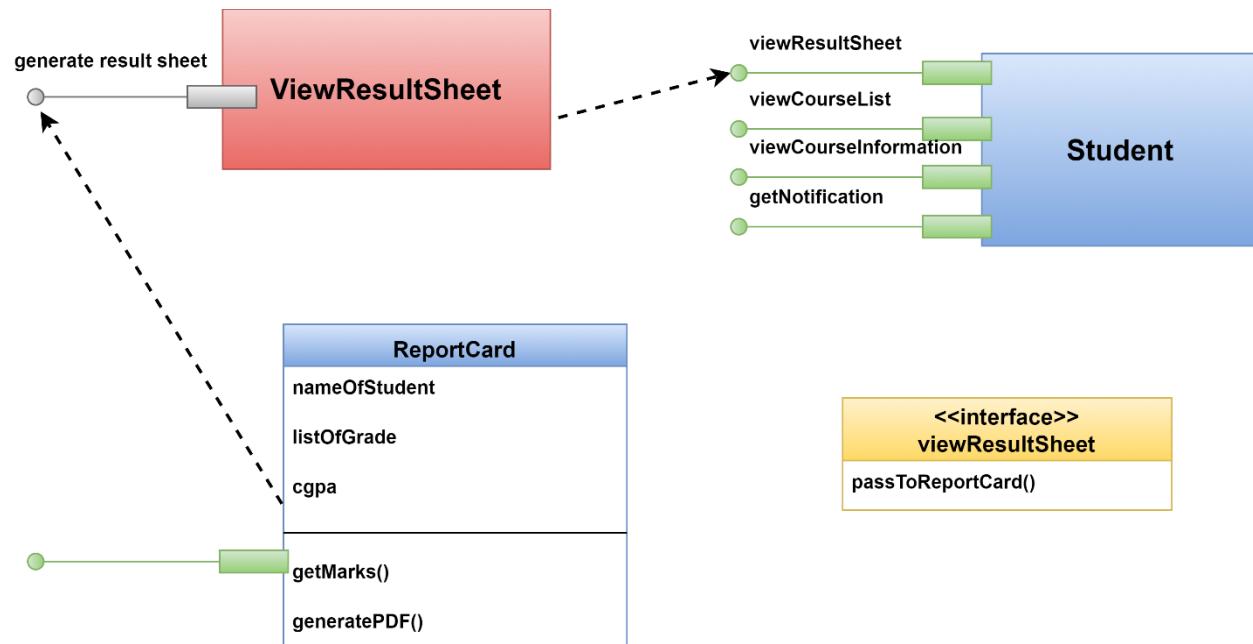


Figure 93 ReportCard class

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 94 shows RoomHandler class.

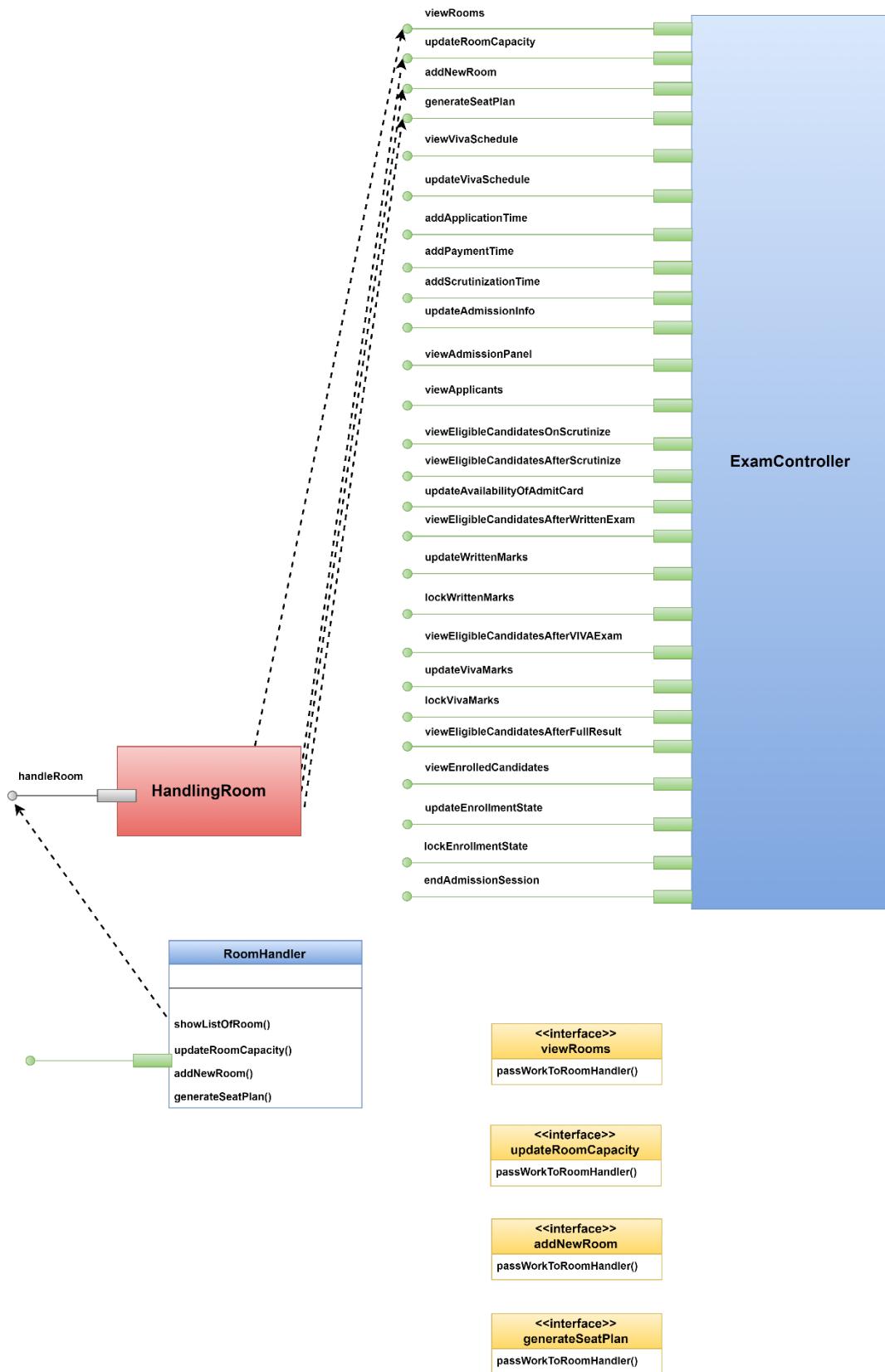


Figure 94 RoomHandler Class

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 95 shows DisplayHeader class.

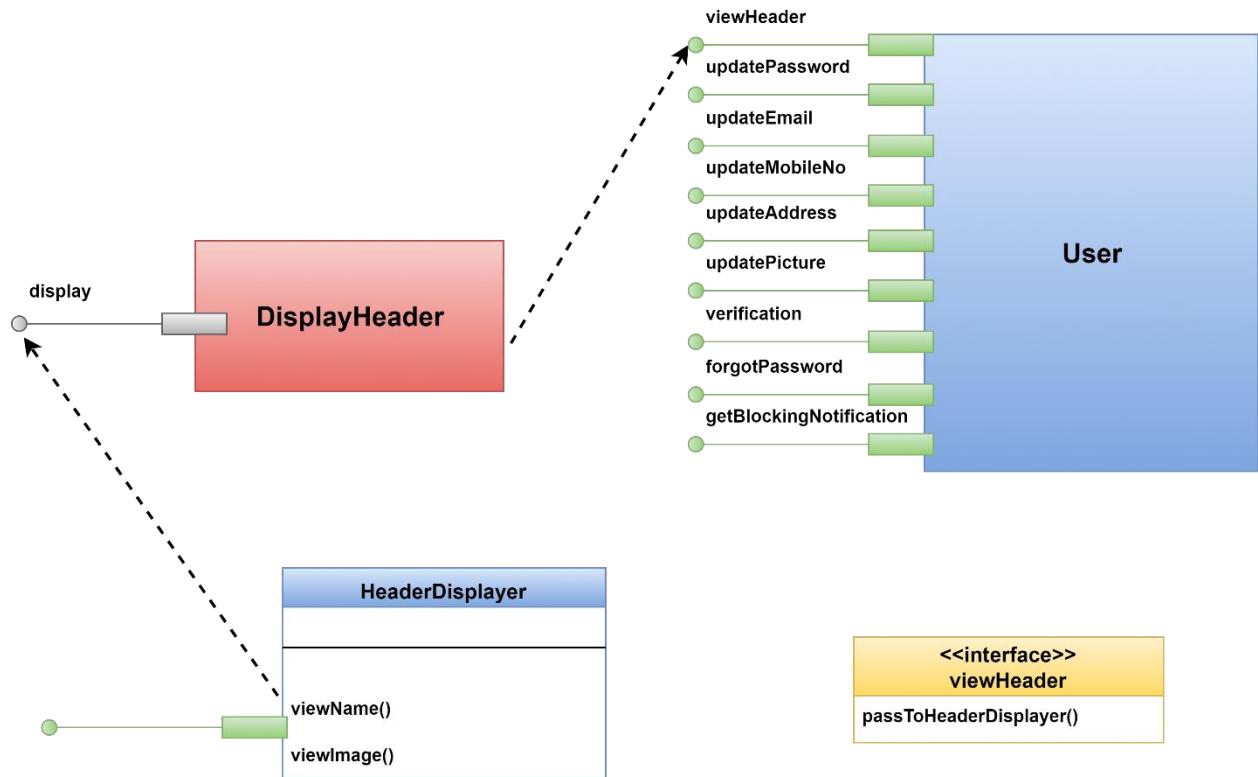


Figure 95 HeaderDisplayer class

It is not the task of Teacher class to generate PDF. So PDFHandler class has been introduced which is shown in figure 96.

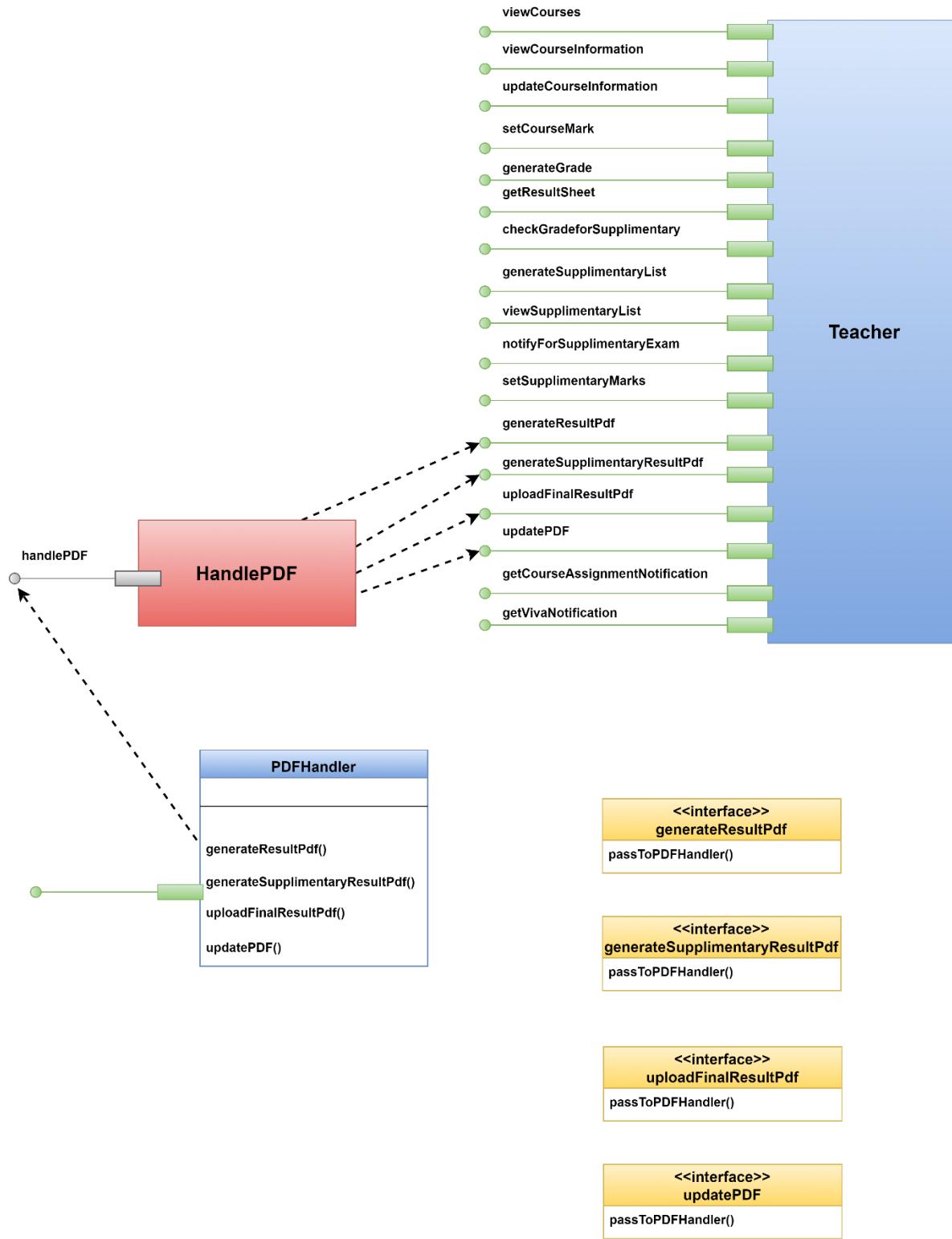


Figure 96 PDFHandler class

### 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	cpga	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 Admission	Applicant	boolean	false	Either true or false
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 97-100 show UML activity diagrams for Authentication class.

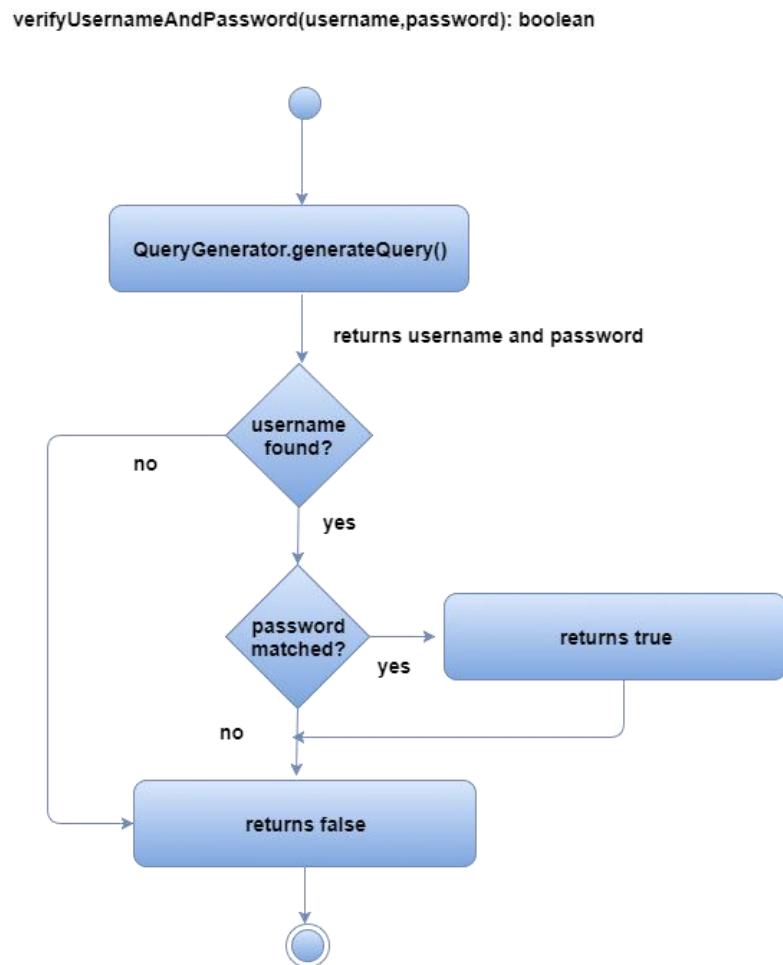


Figure 97 verifyUsernameAndPassword() of Authentication

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

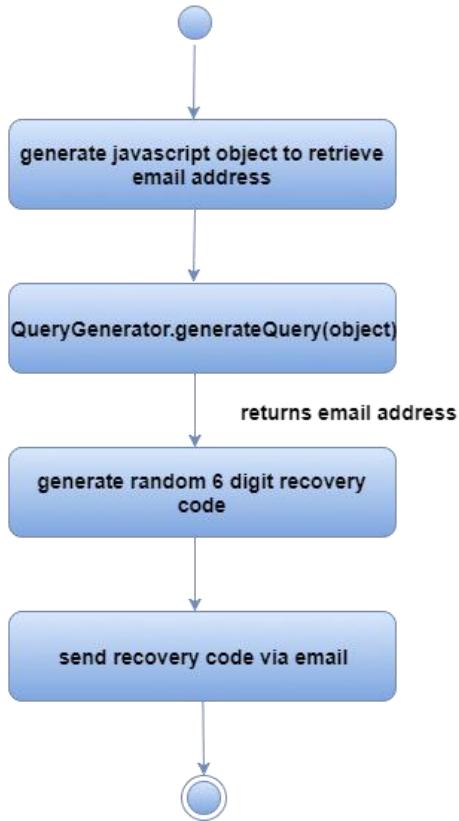


Figure 98 `forgotPassword()` of Authentication

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

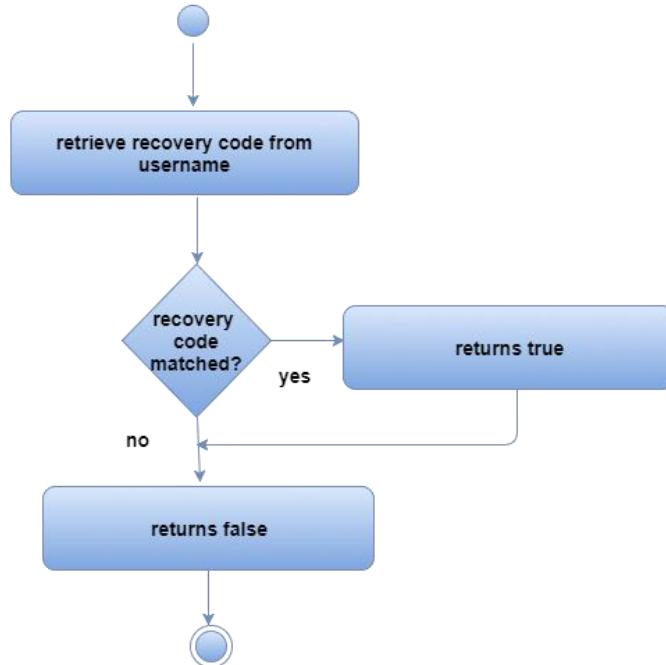


Figure 99 `verifyUsernameAndRecoveryCode()` of Authentication

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

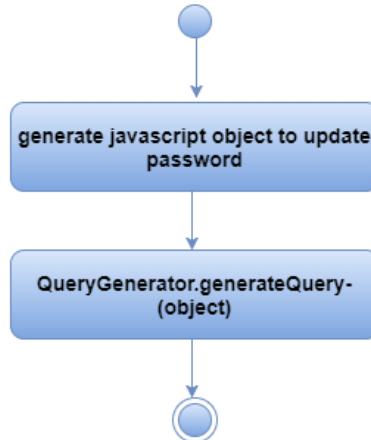


Figure 100 `setNewPassword()` of Authentication

Figure 101-103 show UML activity diagrams for Applicant class.

`applyForAdmission(application data): String`

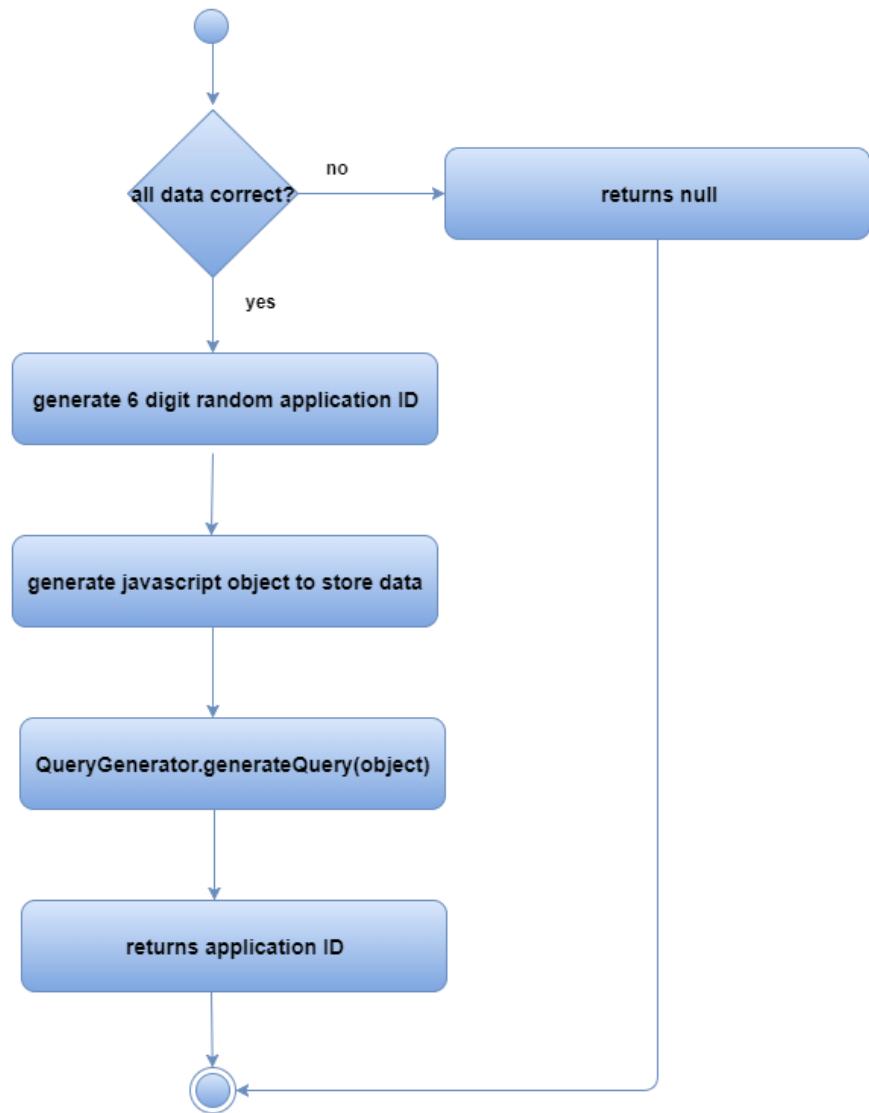


Figure 101 `applyForAdmission()` of Applicant

```
checkAvailabilityOfAdmitCard(ApplicationID): boolean
```

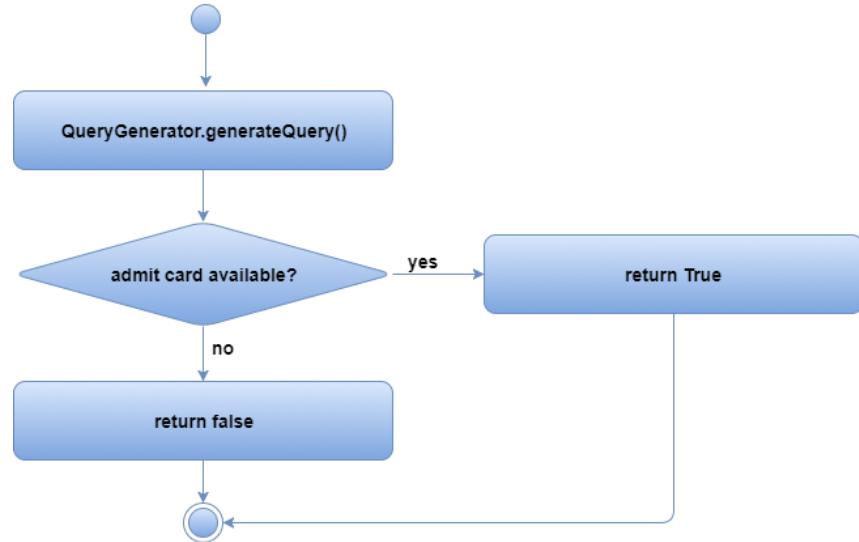


Figure 102 `checkAvailabilityOfAdmitCard()` of Applicant

```
generateAdmitCard(applicationID): void
```

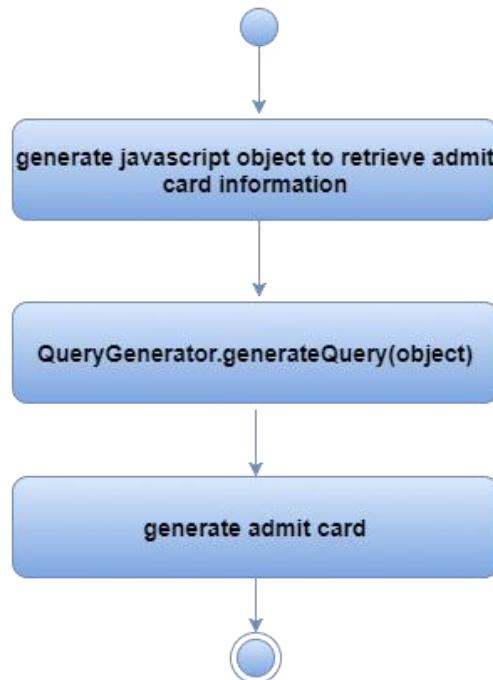


Figure 103 `generateAdmitCard()` of Applicant

Figure 104 shows UML activity diagram for `QueryGenerator` class.

```
generateQuery(javascript object): resultSet
```

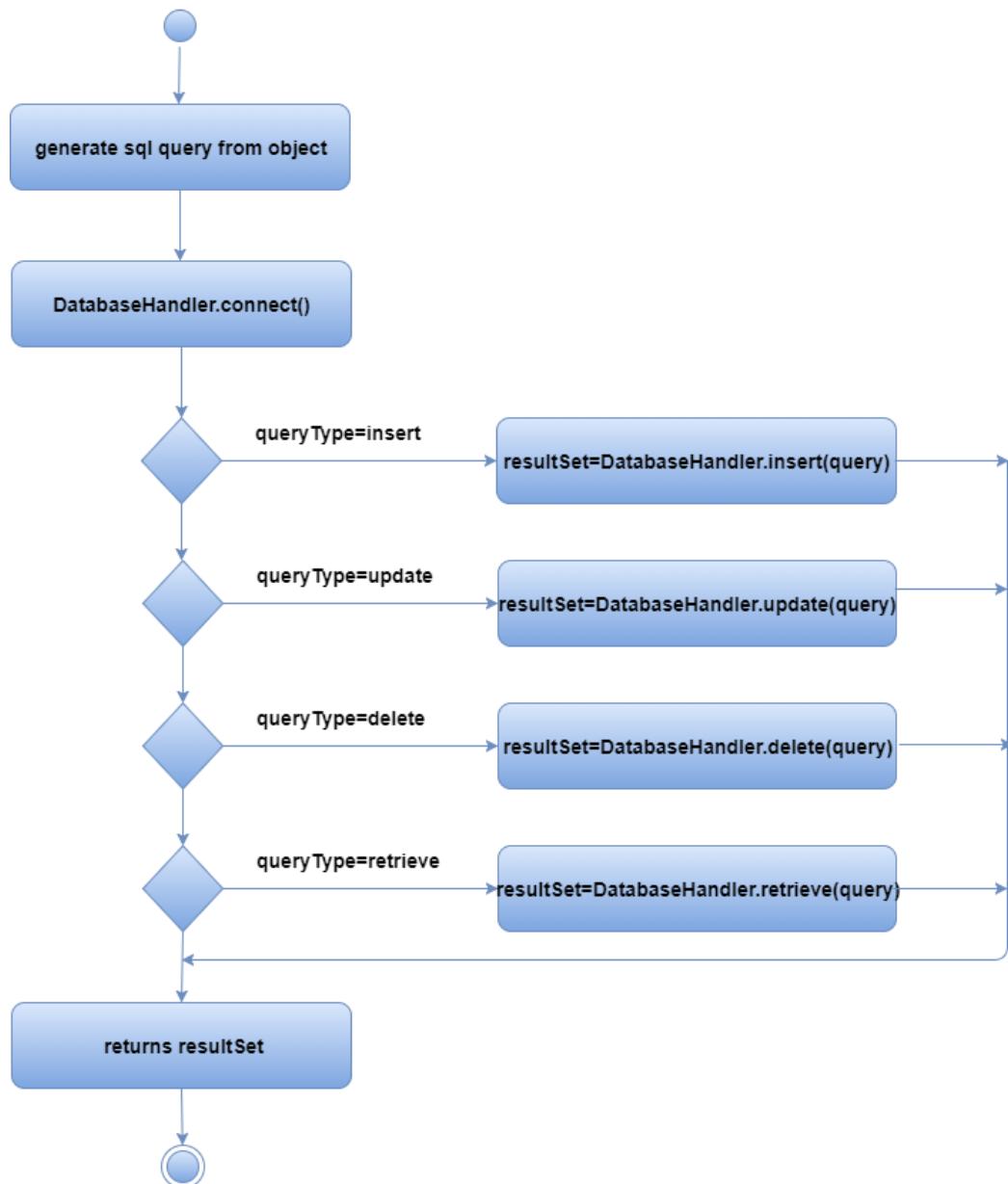


Figure 104 `generateQuery()` of `QueryGenerator`

Figure 105-108 show UML activity diagrams for User class.

`updateAddress(username): void`

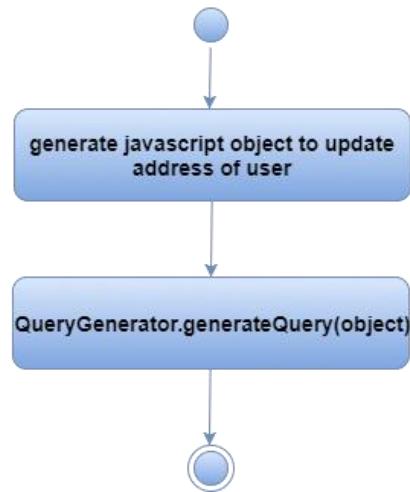


Figure 105 `updateAddress()` of User

`updateMobileNo(username): void`

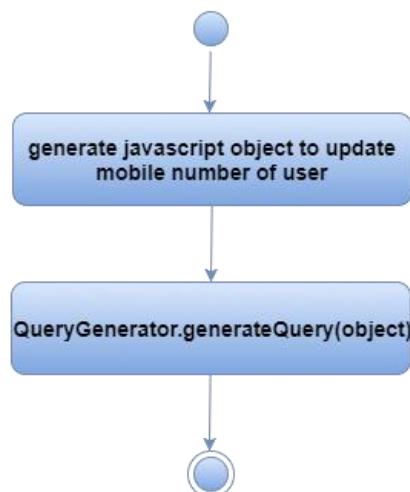


Figure 106 `updateMobileNo()` of User

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

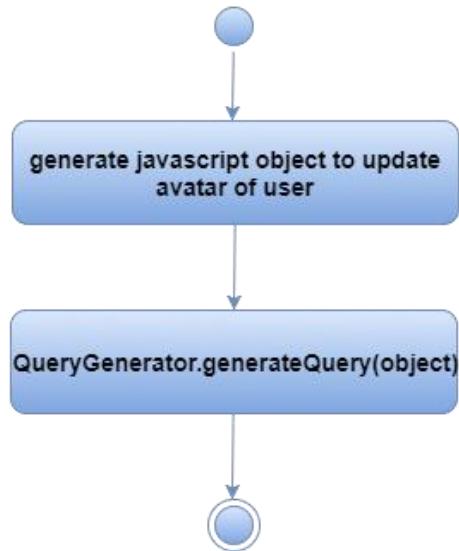


Figure 107 updatePicture() of User

```
getBlockingNotification(): notificationMessage
```

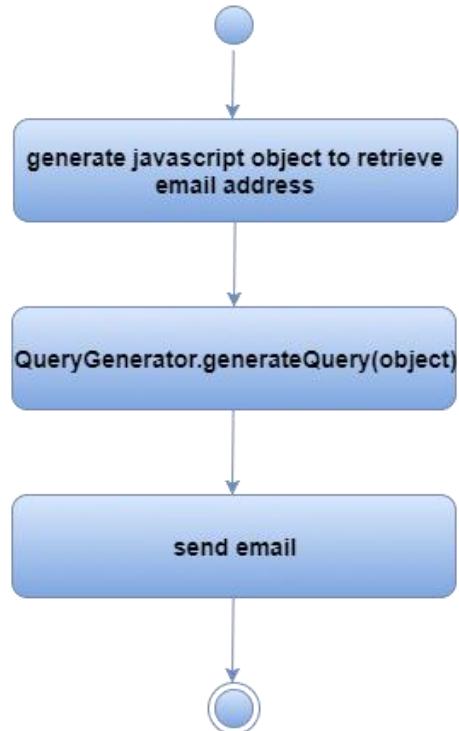


Figure 108 getBlockingNotification() of User

Figure 109-110 show UML activity diagrams for HeaderDisplayer class.

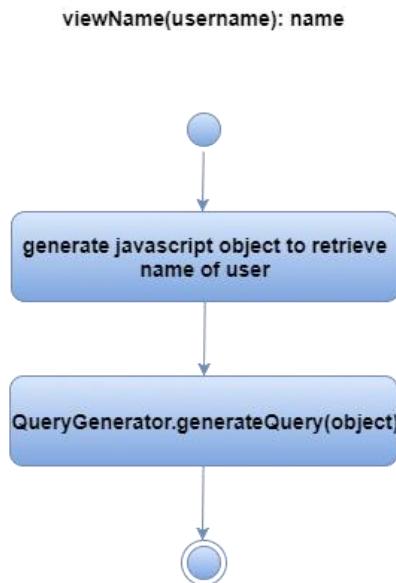


Figure 109 viewName() of HeaderDisplayer

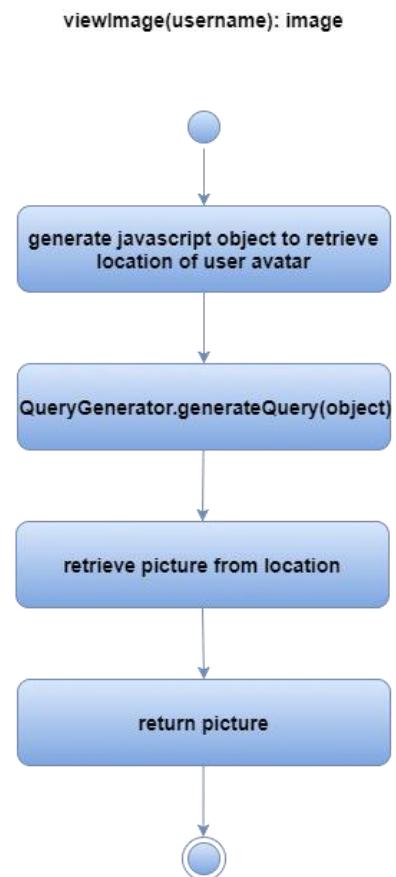


Figure 110 viewImage() of HeaderDisplayer

Figure 111-121 show UML activity diagrams for ProgramChairperson class.

`viewListOfUsers(): void`

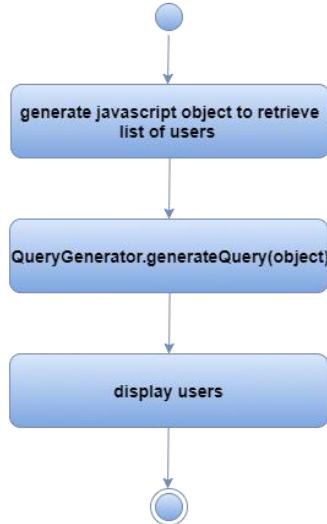


Figure 111 `viewListOfUsers()` of ProgramChairperson

`updatefullName(username): void`

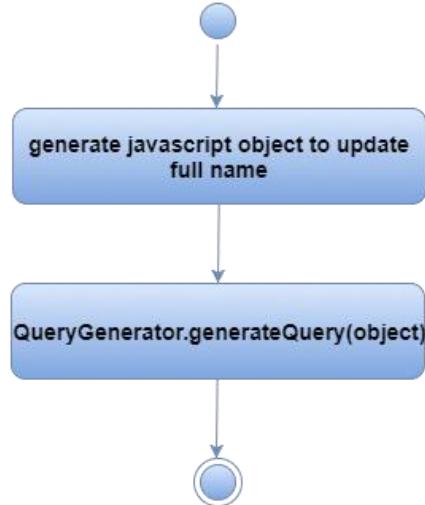


Figure 112 `updateFullName()` of ProgramChairperson

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

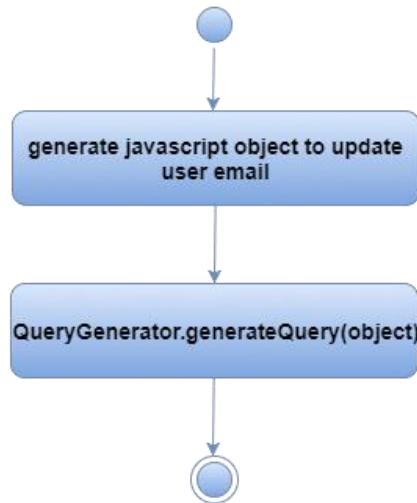


Figure 113 updateUserEmail () of ProgramChairperson

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

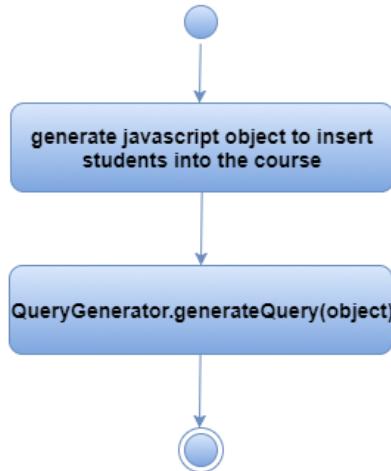


Figure 114 addNewStudentsIntoCourse() of ProgramChairperson

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

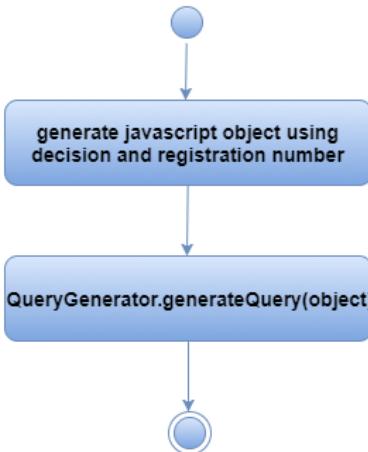


Figure 115 approveAccountRequest() of ProgramChairperson

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

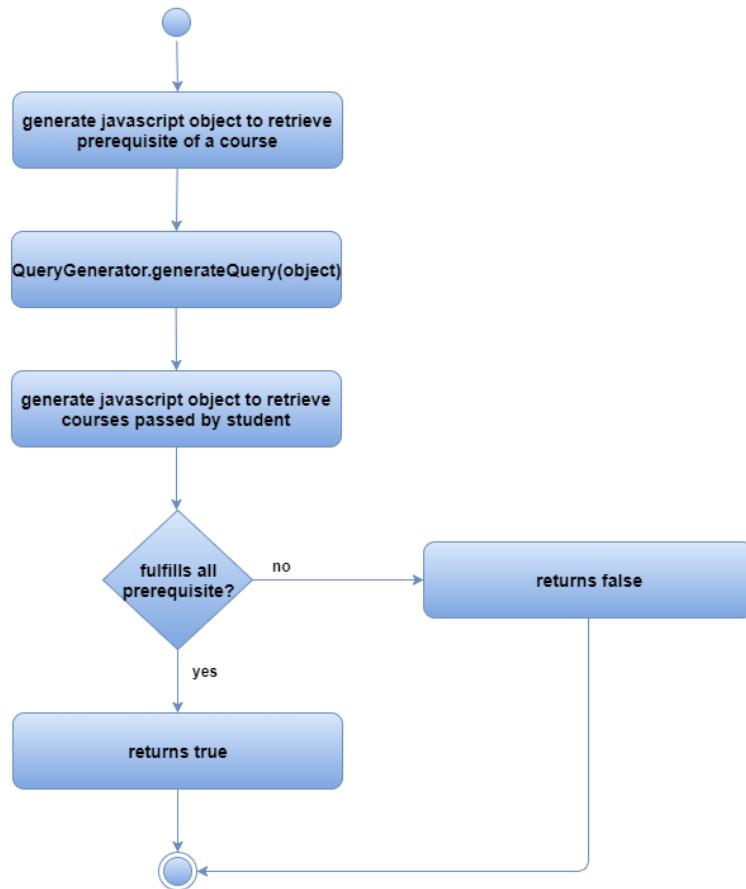


Figure 116 checkFulfilmentOfPayment() of ProgramChairperson

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

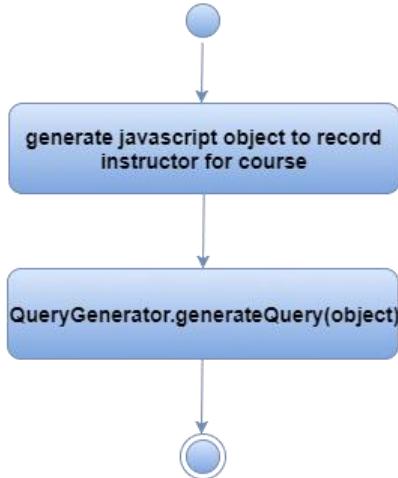


Figure 117 `selectInstructor()` of ProgramChairperson

```
createStudentAccount(): void
```

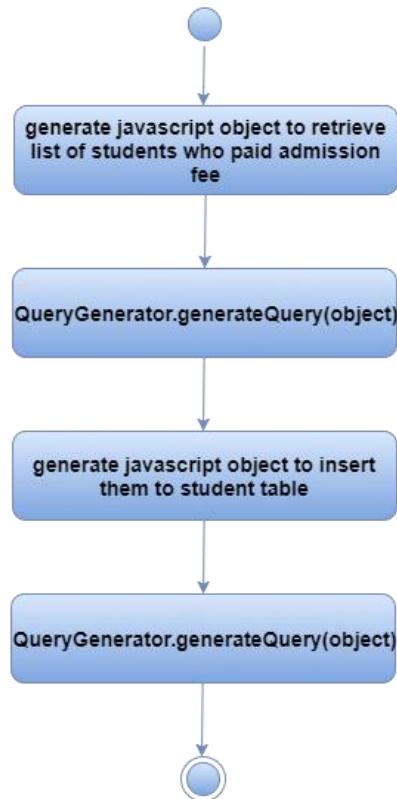


Figure 118 `createStudentAccount()` of ProgramChairperson

`viewResult(choice,value): void`

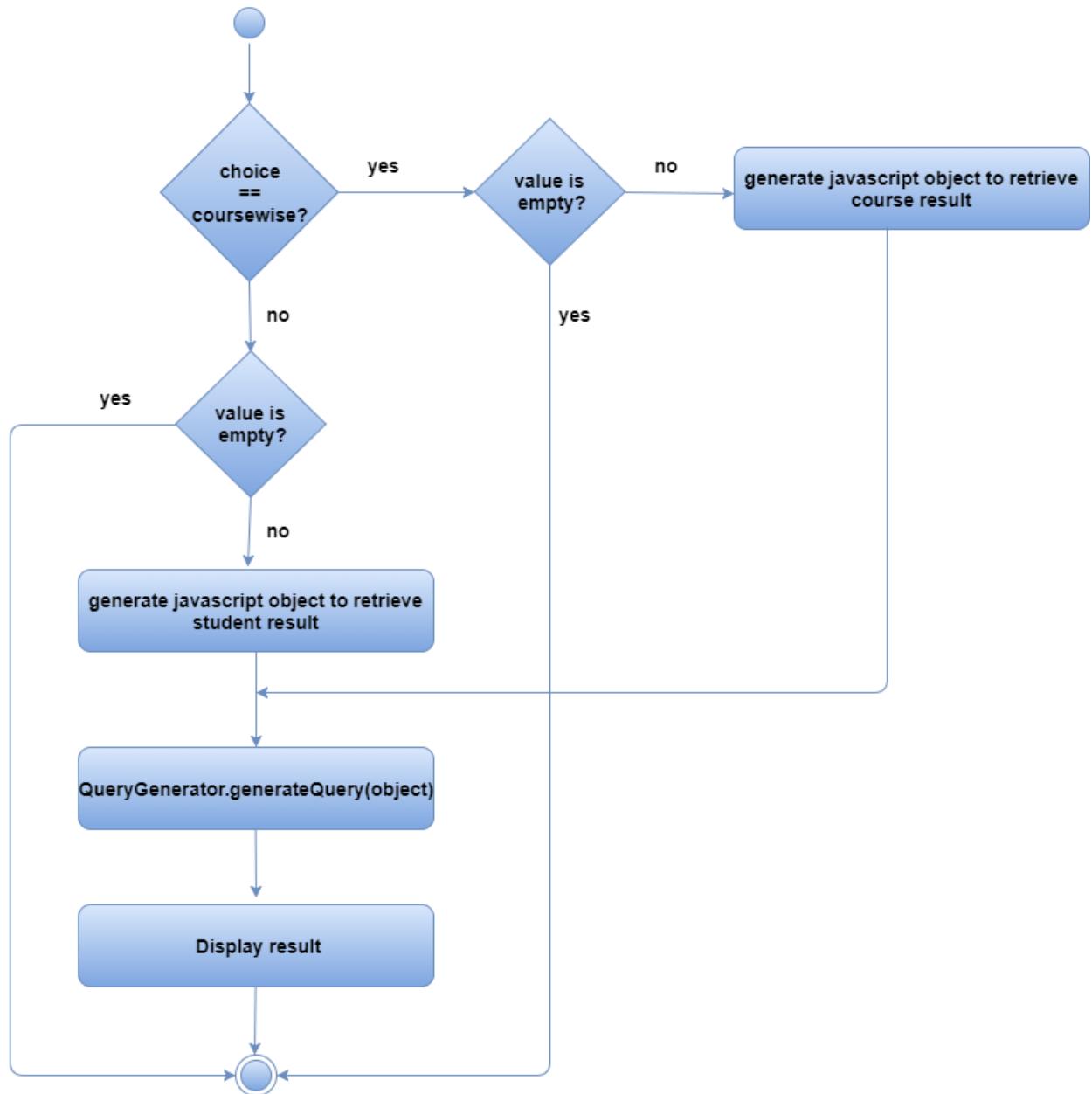


Figure 119 `viewResult()` of `ProgramChairperson`

`viewInformationOfOneUser(username): void`

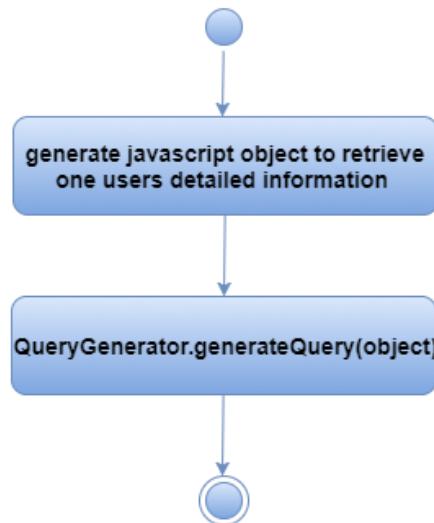


Figure 120 `viewInformationOfOneUser()` of ProgramChairperson

`viewListOfCourses(): void`

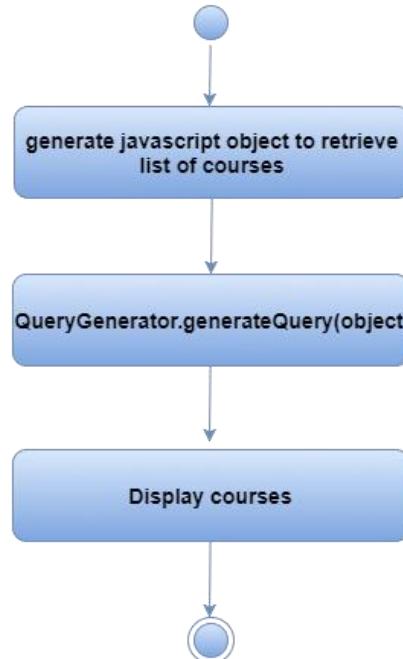


Figure 121 `viewListOfCourses ()` of ProgramChairperson

Figure 122-134 show UML activity diagrams for Teacher class.

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

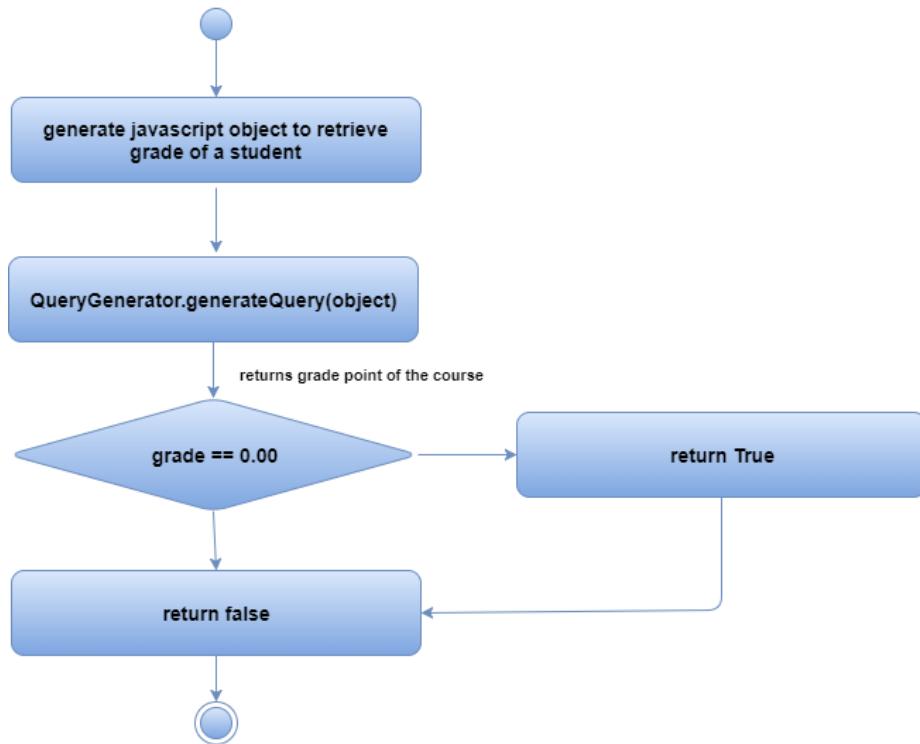


Figure 122 checkGradeForSupplementary() of Teacher

```
generateResultPdf(): void
```

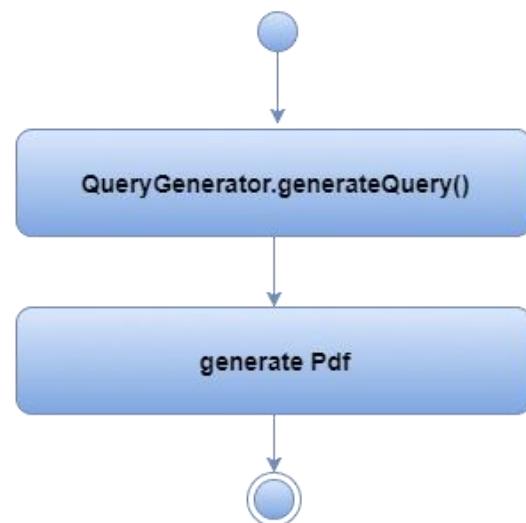


Figure 123 generateResultPdf() of Teacher

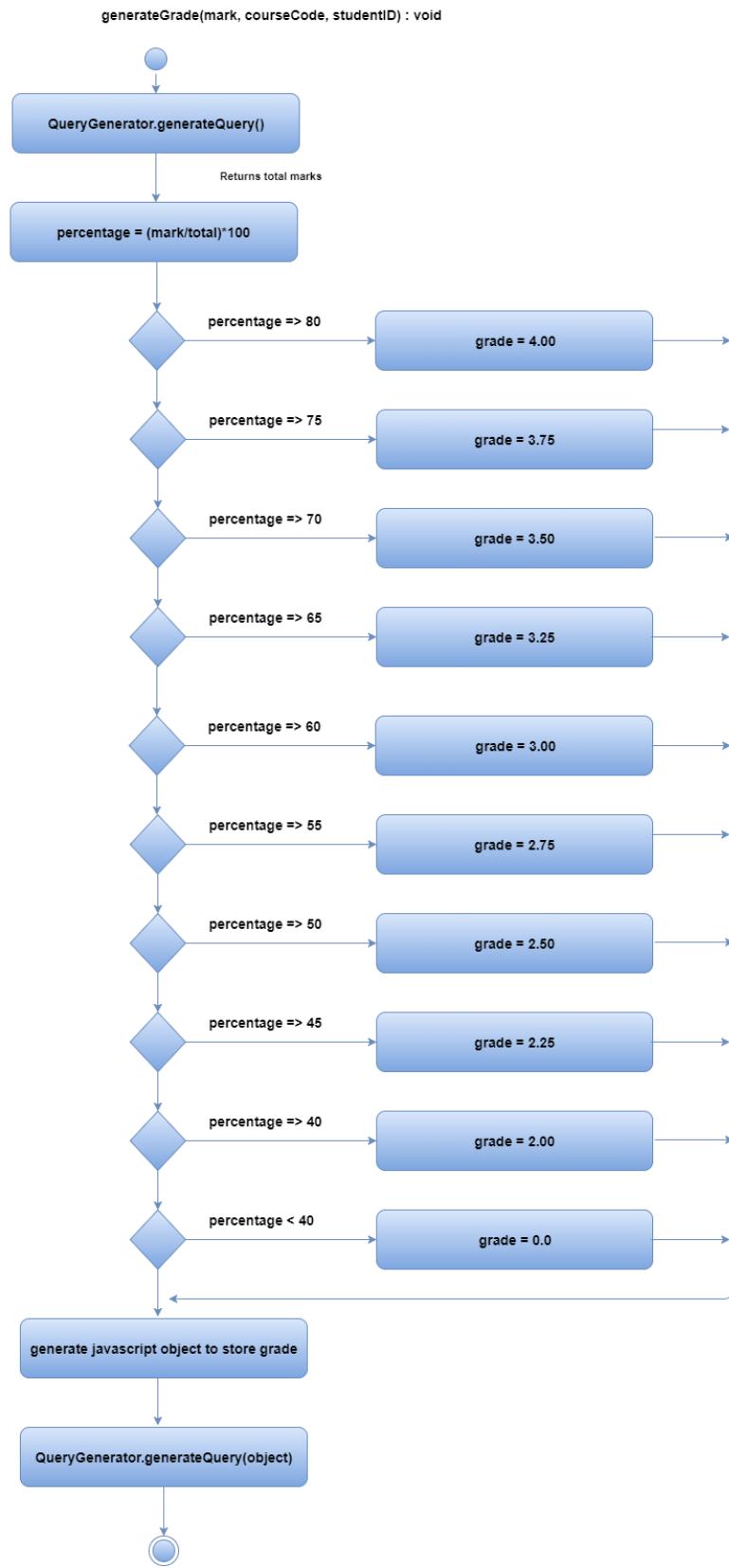


Figure 124 generateGrade() of Teacher

`generateSupplementaryList(courseID): supplementaryList`

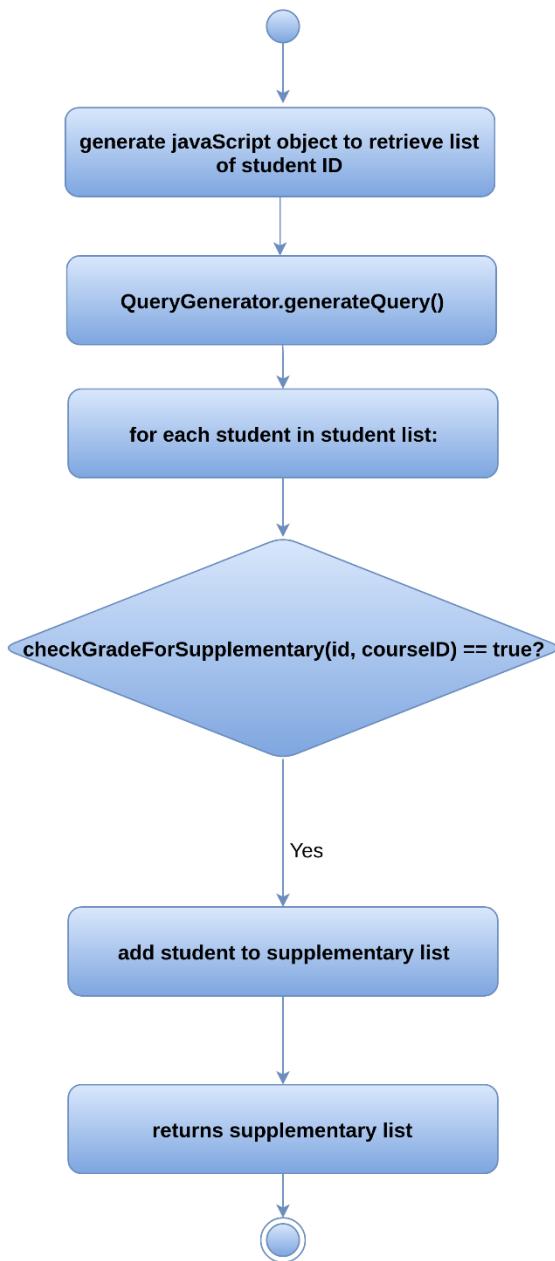


Figure 125 `generateSupplementaryList()` of Teacher

`generateSupplementaryResultPdf(): void`

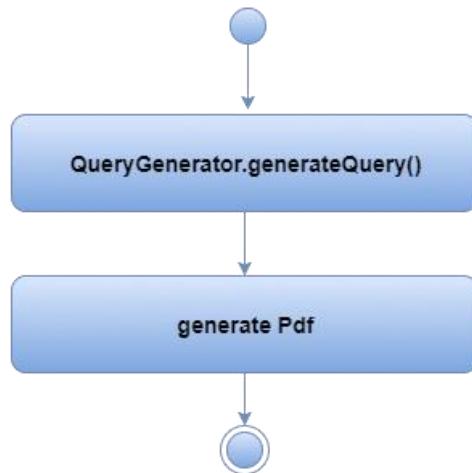


Figure 126 `generateSupplementaryResultPdf()` of Teacher

`getResultSet(): result sheet`

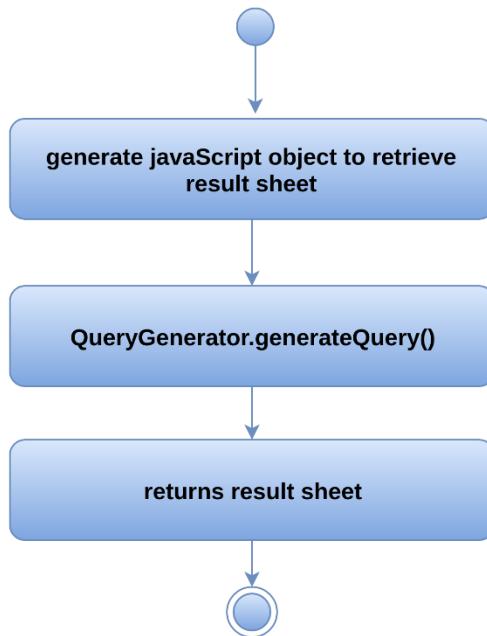


Figure 127 `getResultSet()` of Teacher

`notifyForSupplementaryExam(): void`

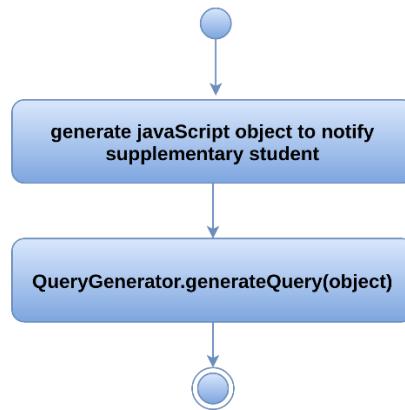


Figure 128 `notifyForSupplementaryExam()` of Teacher

`setCourseMark(mark) : void`

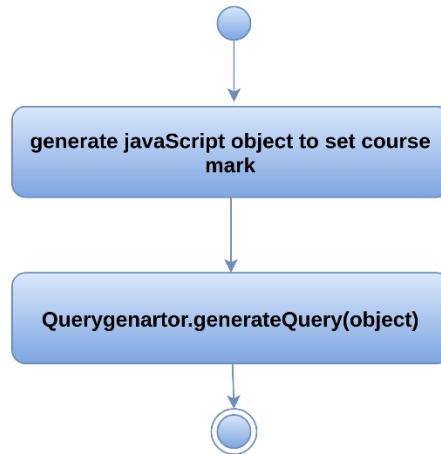


Figure 129 `setCourseMark()` of Teacher

`setSupplementaryMark(mark): void`

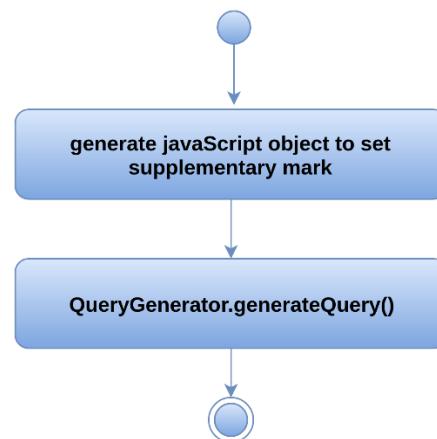


Figure 130 `setSupplementaryMark()` of Teacher

`updateCourseInformation(course ID, information):void`

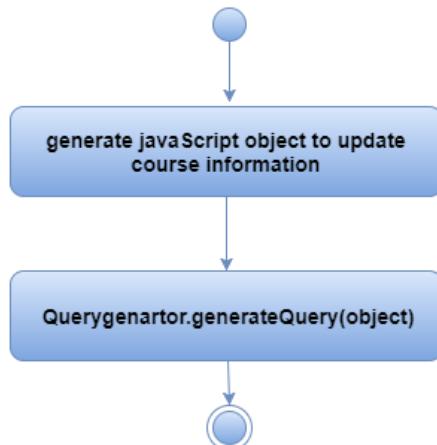


Figure 131 `updateCourseInformation()` of Teacher

`viewCourseInformation(courseCode): courseInformation`

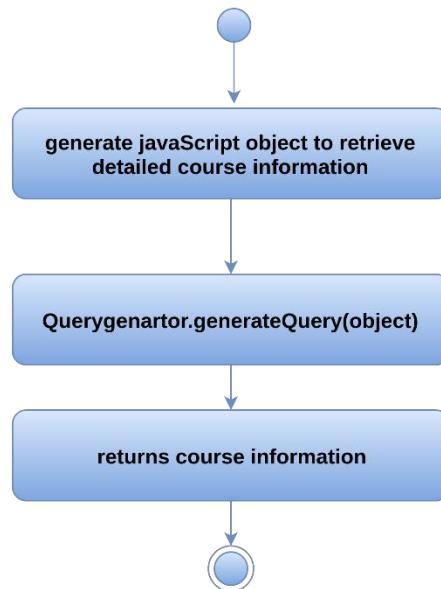


Figure 132 `viewCourseInformation()` of Teacher

`viewListOfCourse(): courseList`

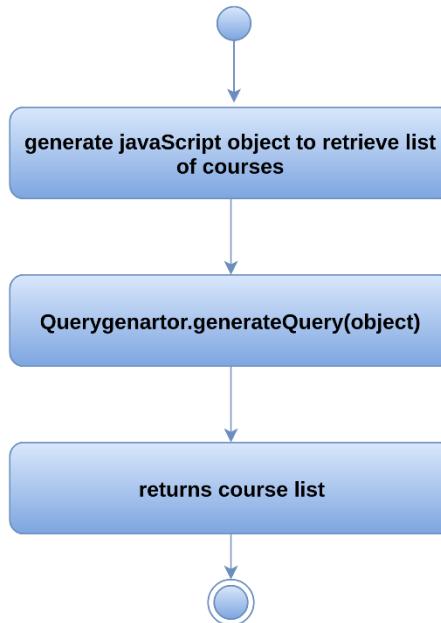


Figure 133 `viewListOfCourse()` of Teacher

`viewSupplementaryList():void`

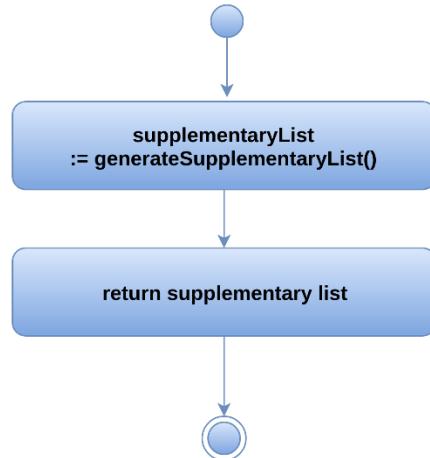


Figure 134 `viewSupplementaryList()` of Teacher

Figure 135-138 show UML activity diagrams for `PDFHandler` class.

`generateResultPdf(result): resultPDF`

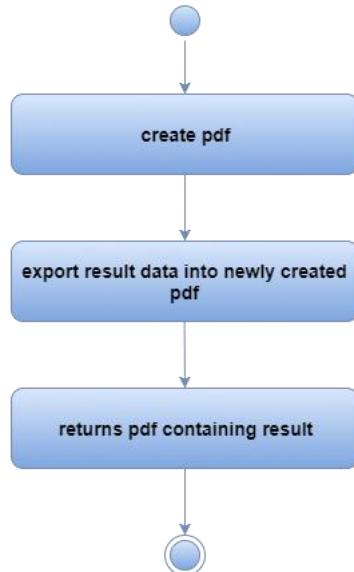


Figure 135 `generateResultPdf()` of `PDFHandler`

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

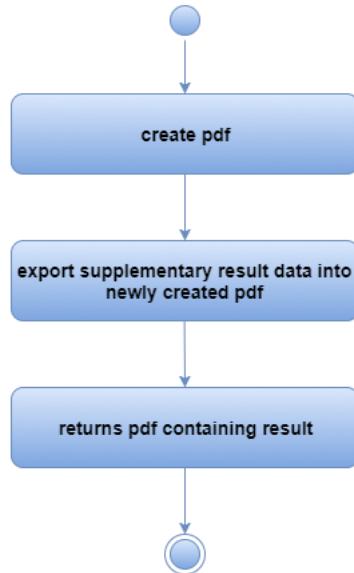


Figure 136 generateSupplementaryResultPdf() of PDFHandler

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

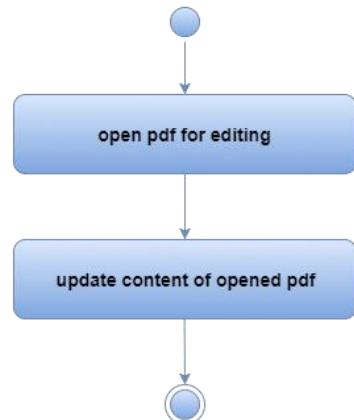


Figure 137 updatePdf() of PDFHandler

```
uploadFinalResultPdf(finalResult): resultPDF
```

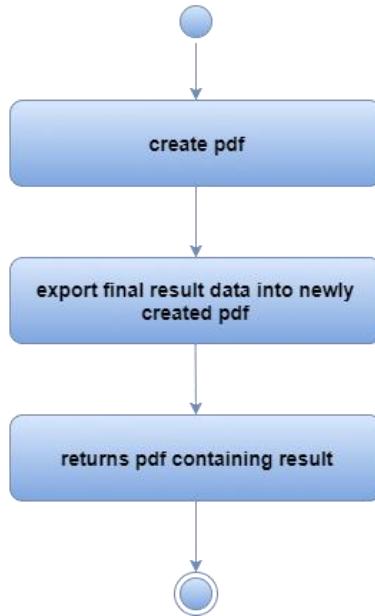


Figure 138 uploadFinalResultPdf() of PDFHandler

Figure 139-141 show UML activity diagrams for Student class.

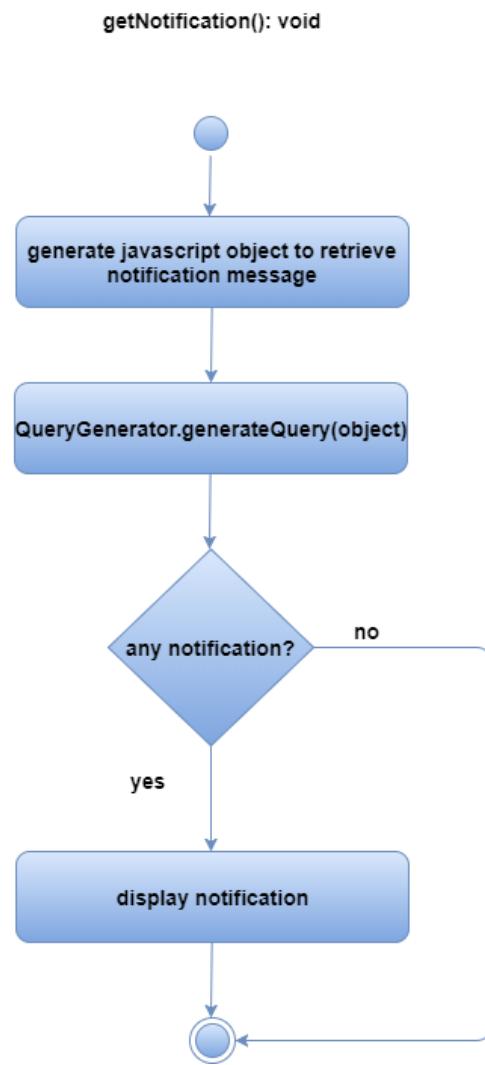


Figure 139 `getNotification()` of Student

`viewCourseDetails(courseID): courseInformation`

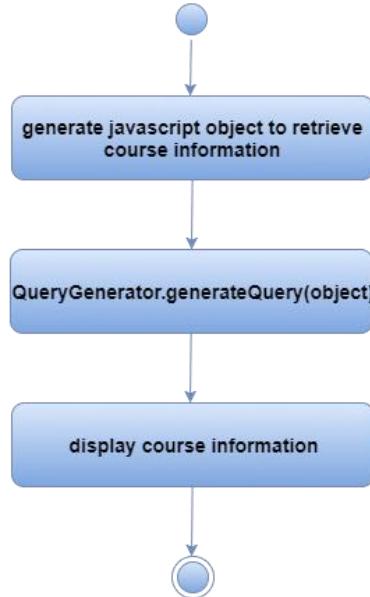


Figure 140 `viewCourseDetails()` of Student

`viewCourseList(): courseList`

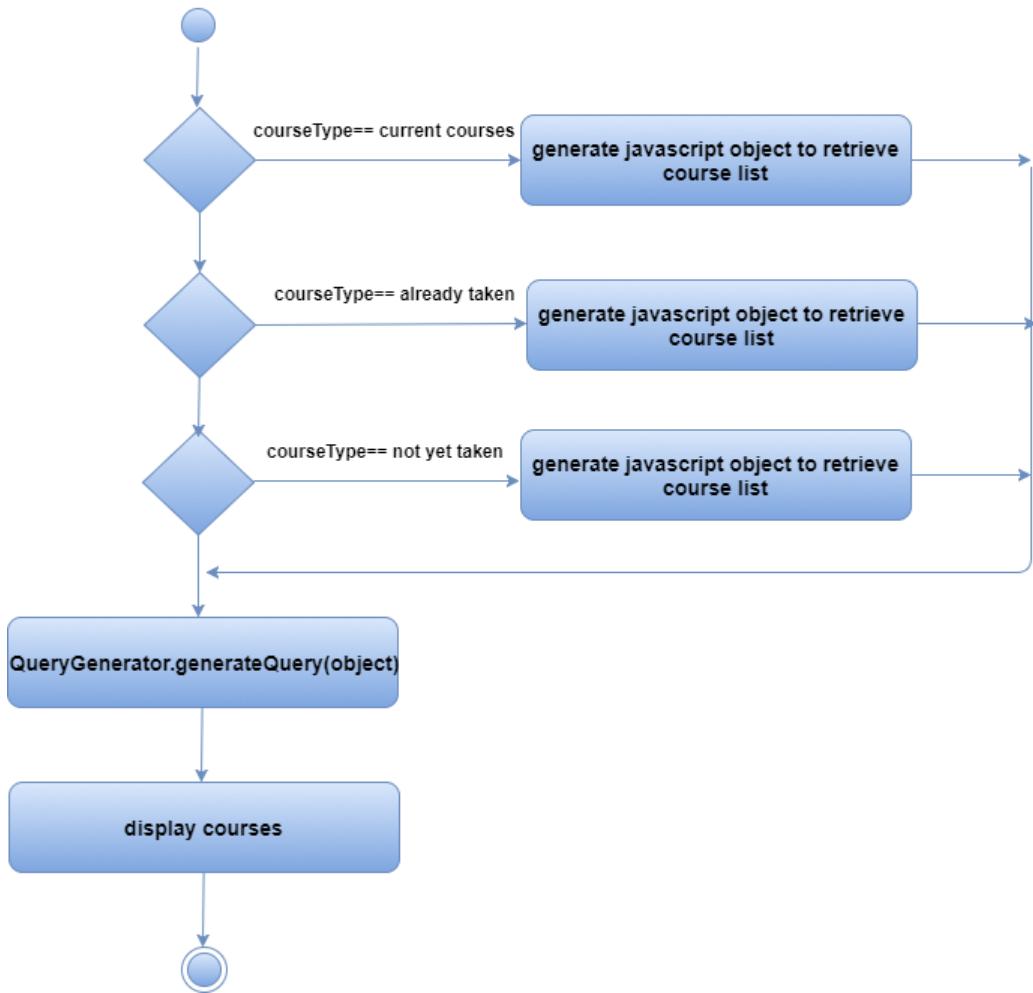


Figure 141 `viewCourseList()` of Student

Figure 142-143 show UML activity diagrams for ReportCard class.

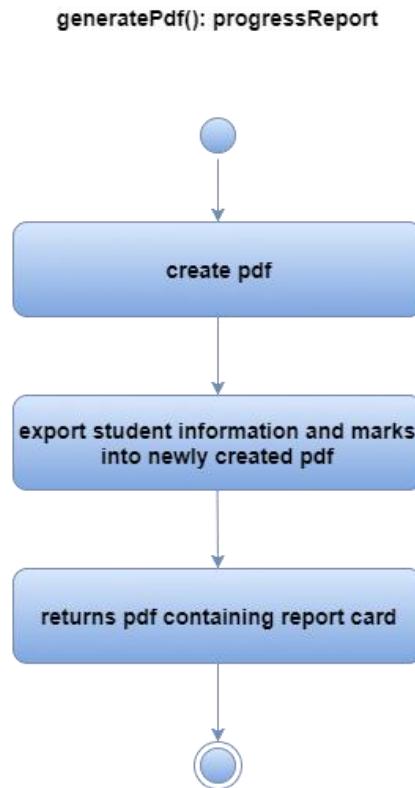


Figure 142 generatePdf() of ReportCard

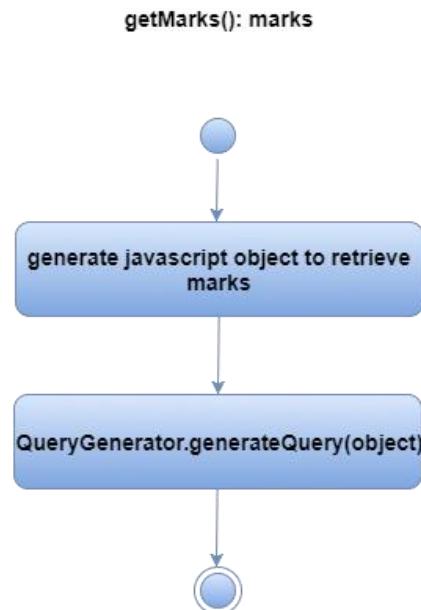


Figure 143 getMarks() of ReportCard

Figure 144-156 show UML activity diagrams for ExamController class.

`addApplicationTimeRange(timeRange): void`

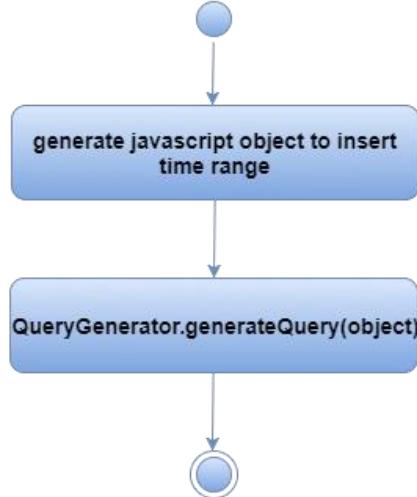


Figure 144 addApplicationTimeRange() of Exam Controller

`addPaymentTimeRange(timeRange): void`

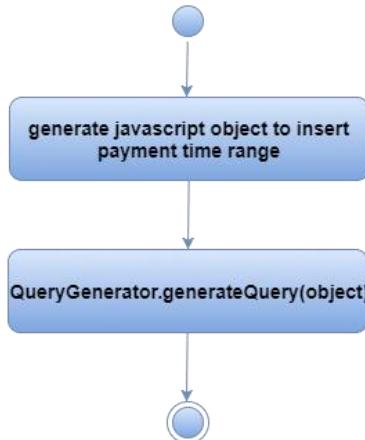


Figure 145 addPaymentTimeRange() of Exam Controller

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

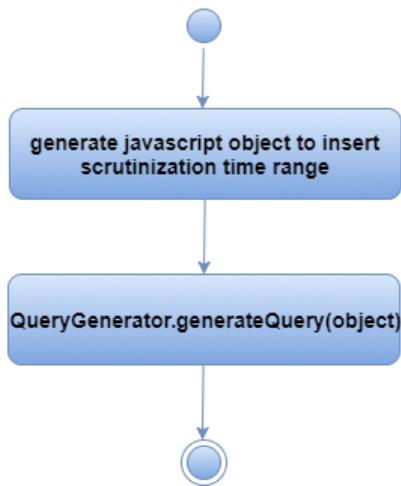


Figure 146 addScrutinizationTimeRange() of Exam Controller

```
endAdmissionSession(): void
```

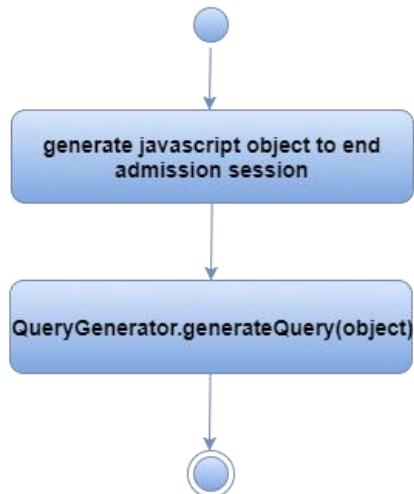


Figure 147 endAdmissionSession() of ExamController

`lockVivaMarks(): void`

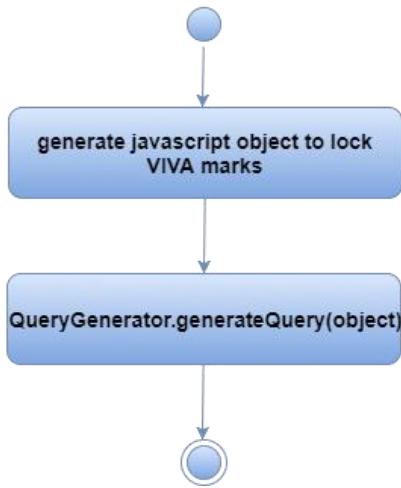


Figure 148 lockVivaMarks() of ExamController

`lockWrittenMarks(): void`

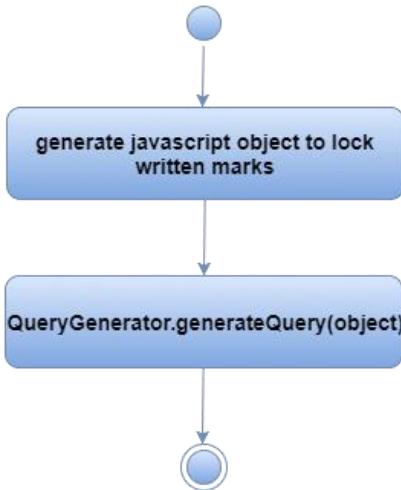


Figure 149 lockWrittenMarks() of ExamController

**updateAvailabilityOfAdmitCard(): void**

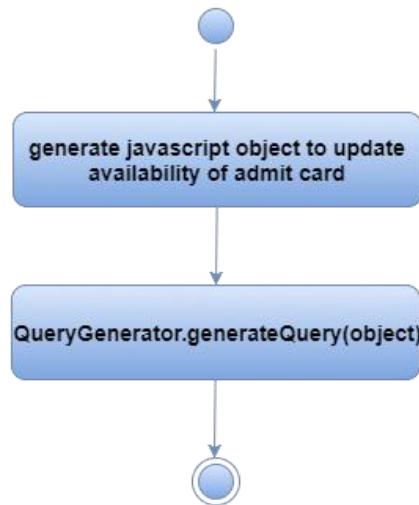


Figure 150 `updateAvailabilityOfAdmitCard()` of ExamController

**updateEnrollmentState(): void**

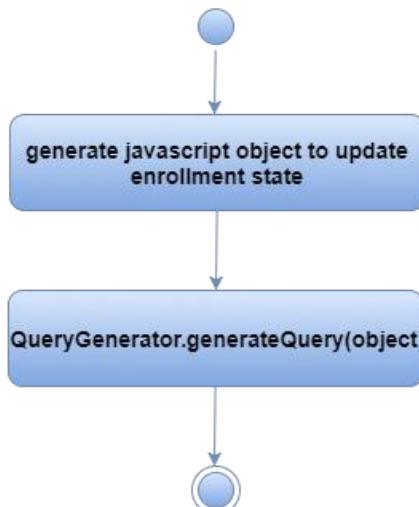


Figure 151 `updateEnrollmentState()` of ExamController

`updateVivaSchedule(): void`

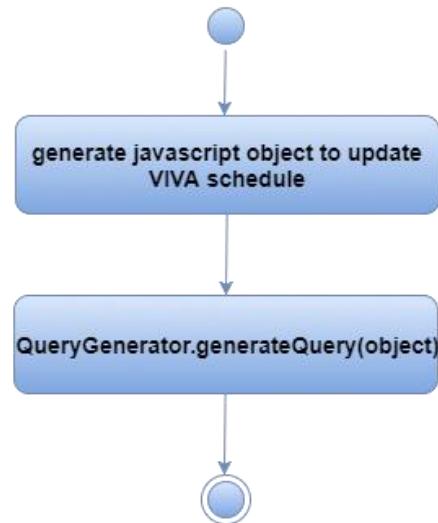


Figure 152 `updateVivaSchedule()` of ExamController

`updateWrittenMarks(): void`

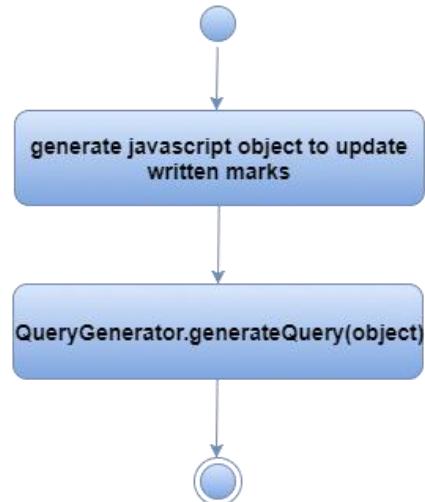


Figure 153 `updateWrittenMarks()` of ExamController

```
viewListOfApplicants(): void
```

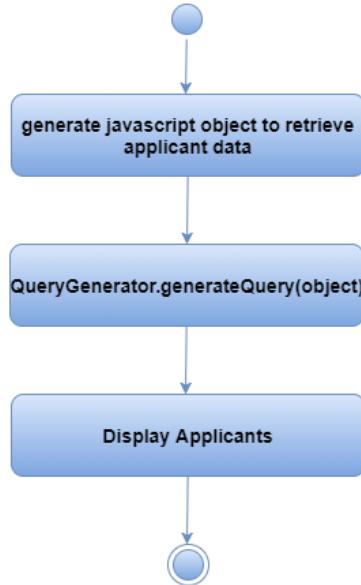


Figure 154 `viewListOfApplicants()` of ExamController

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

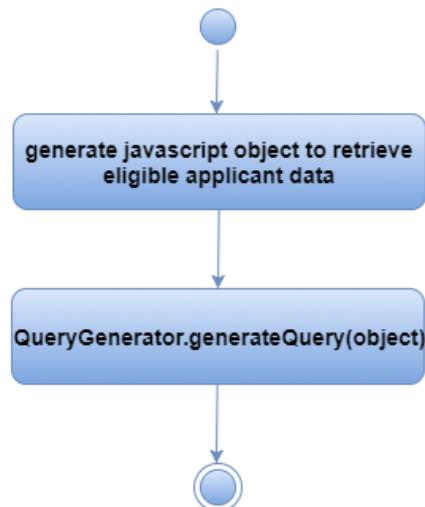


Figure 155 `viewListOfEligibleCandidateAfterFullResult()` of ExamController

`viewListOfEnrolledCandidates(): List<Applicants>`

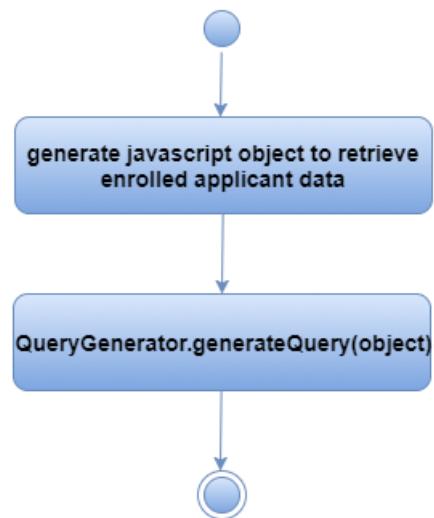


Figure 156 `viewListOfEnrolledCandidates()` of ExamController

Figure 157-160 show UML activity diagrams for RoomHandler class.

`addNewRoom(roomNumber, capacity): void`

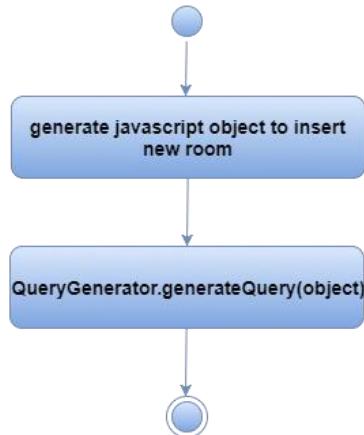


Figure 157 `addNewRoom()` of RoomHandler

**generateSeatPlan(): seatPlan**

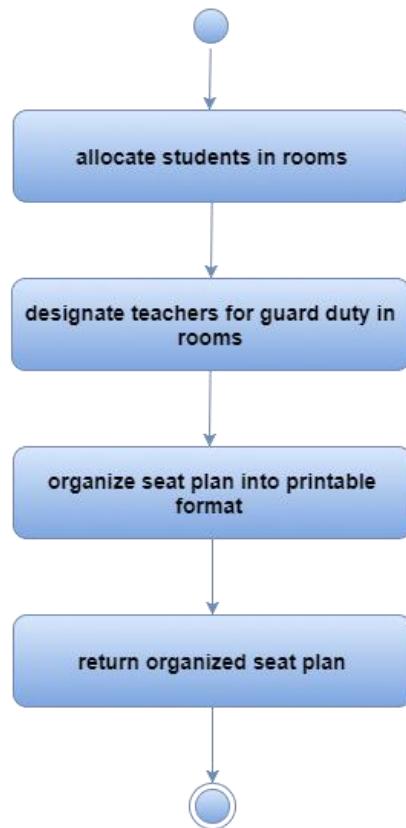


Figure 158 generateSeatPlan() of RoomHandler

**showListOfRoom(): roomList**

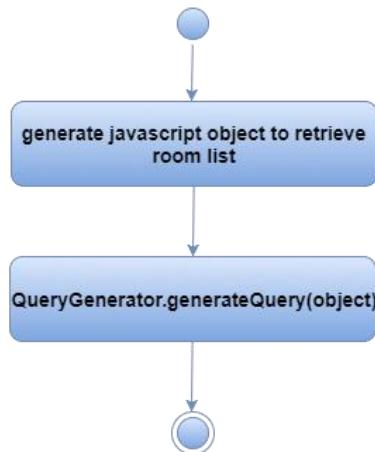


Figure 159 showListOfRoom() of RoomHandler

`updateRoomCapacity(roomNumber, capacity): void`

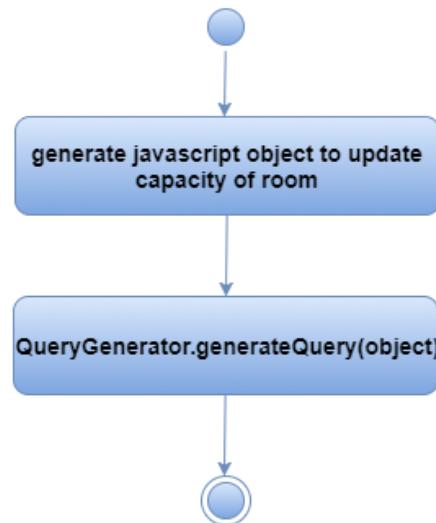


Figure 160 `updateRoomCapacity()` of RoomHandler

Figure 161-163 show UML activity diagrams for Scrutinizer class.

`viewFilteredApplications(listOfStudents):void`

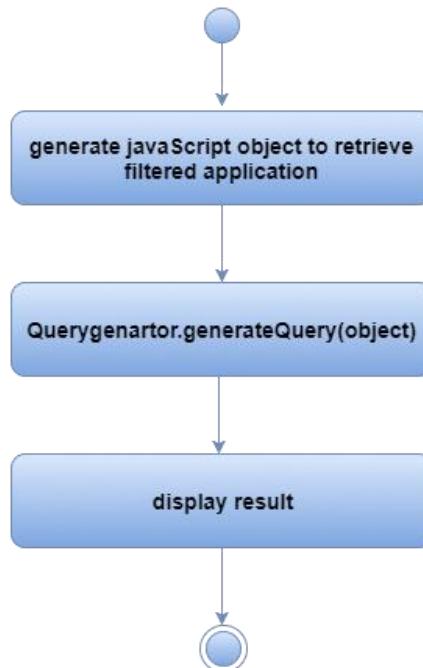


Figure 161 `viewFilteredApplications()` of Scrutinizer

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

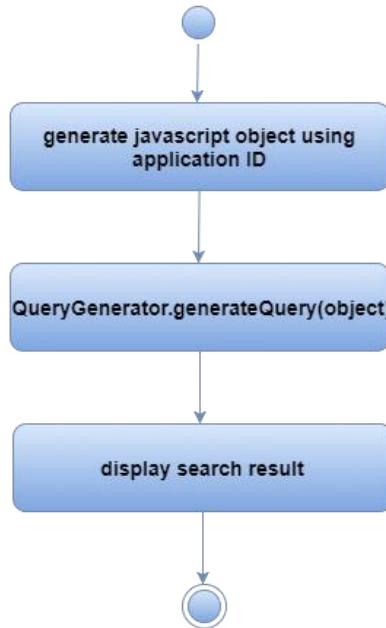


Figure 162 `searchByApplicationID ()` of Scrutinizer

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

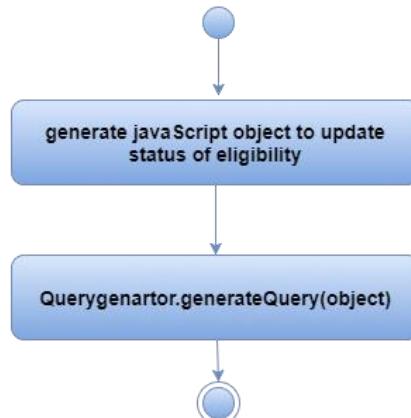


Figure 163 `updateStatusOfEligibility()` of Scrutinizer

Figure 164-165 show UML activity diagrams for RollGenerator class.

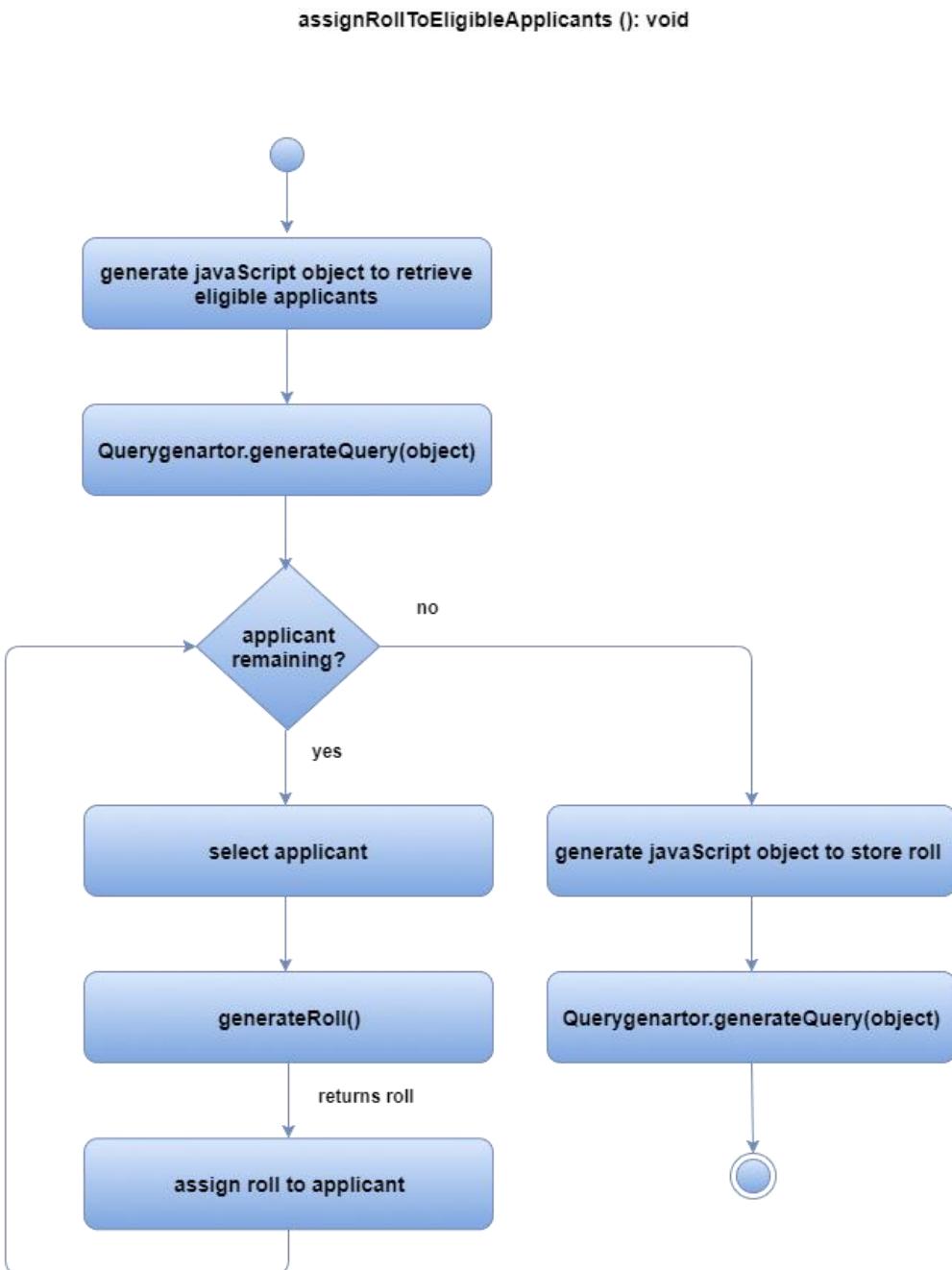


Figure 164 assignRollToEligibleApplicants() of RollGenerator

`generateRoll():string`

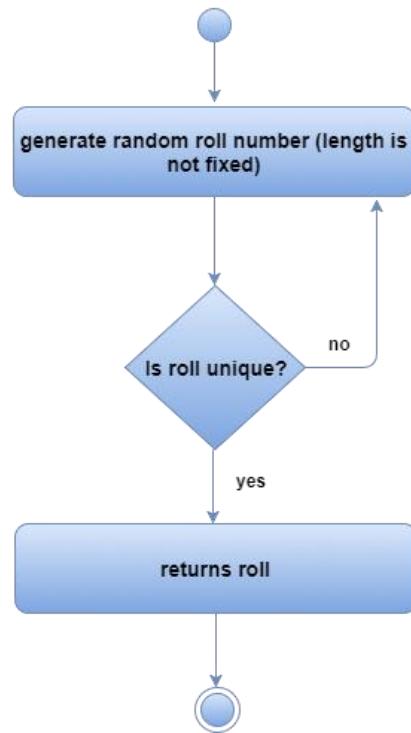


Figure 165 `generateRoll()` of `RollGenerator`

Figure 166-167 show UML activity diagrams for Receptionist class.

`updateStatusOfPayment():void`

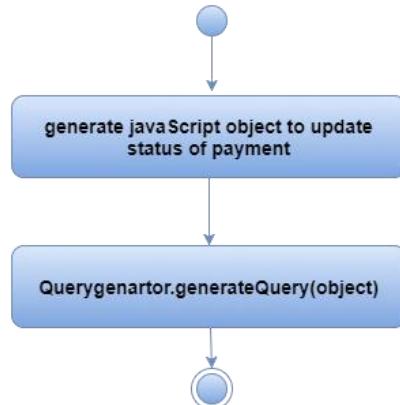


Figure 166 `updateStatusOfPayment()` of `Receptionist`

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

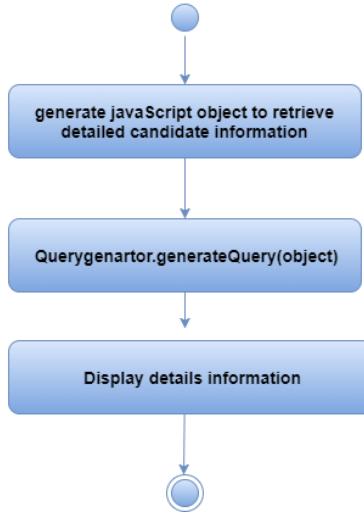


Figure 167 viewApplicantInformation() of Receptionist

### 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 168).



Figure 168 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 169 shows state diagram of the Applicant class.

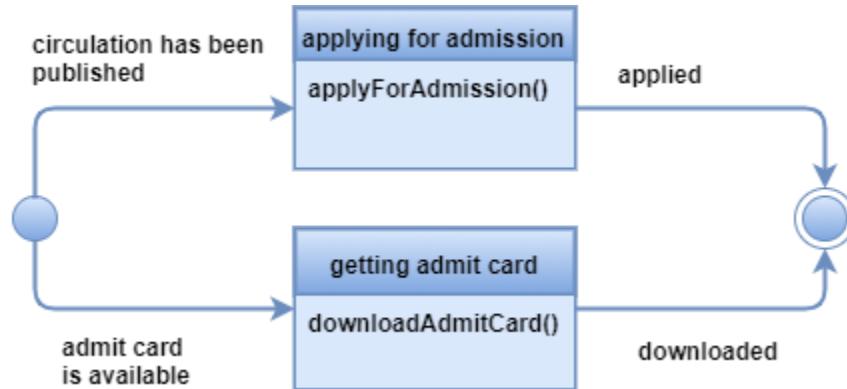


Figure 169 State Diagram of the Applicant class

Figure 170 shows state diagram of the Authentication class.

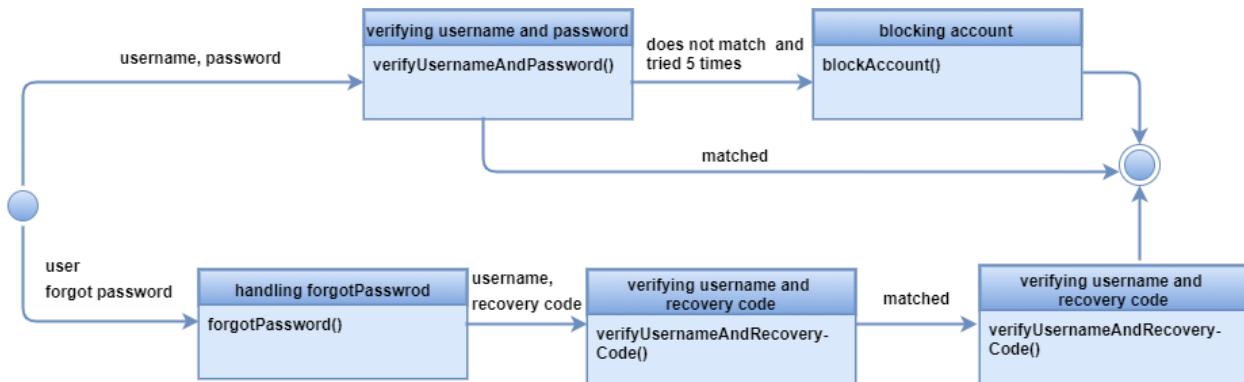


Figure 170 State Diagram of the Authentication class

Figure 171 shows state diagram of the User class.

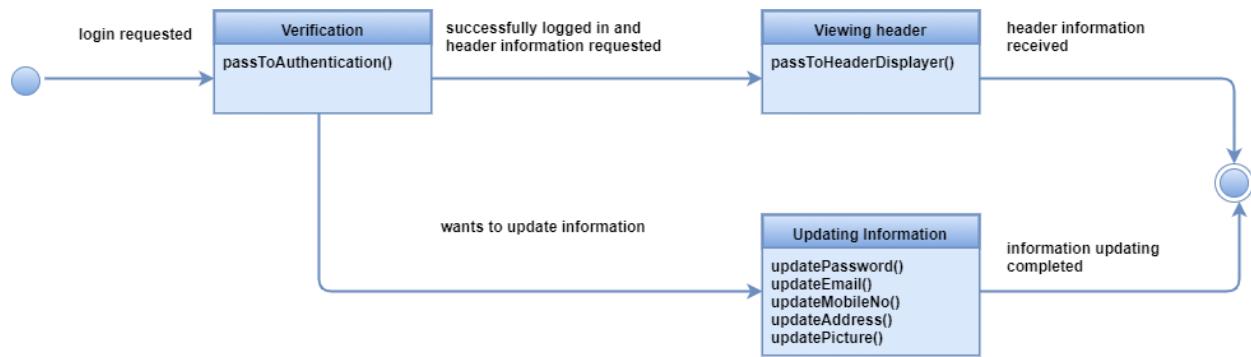


Figure 171 state diagram of the Authentication class

Figure 172 shows state diagram of the HeaderDisplayer class.



Figure 172 State Diagram of the HeaderDisplayer class

Figure 173 shows state diagram of the ProgramChairperson class.

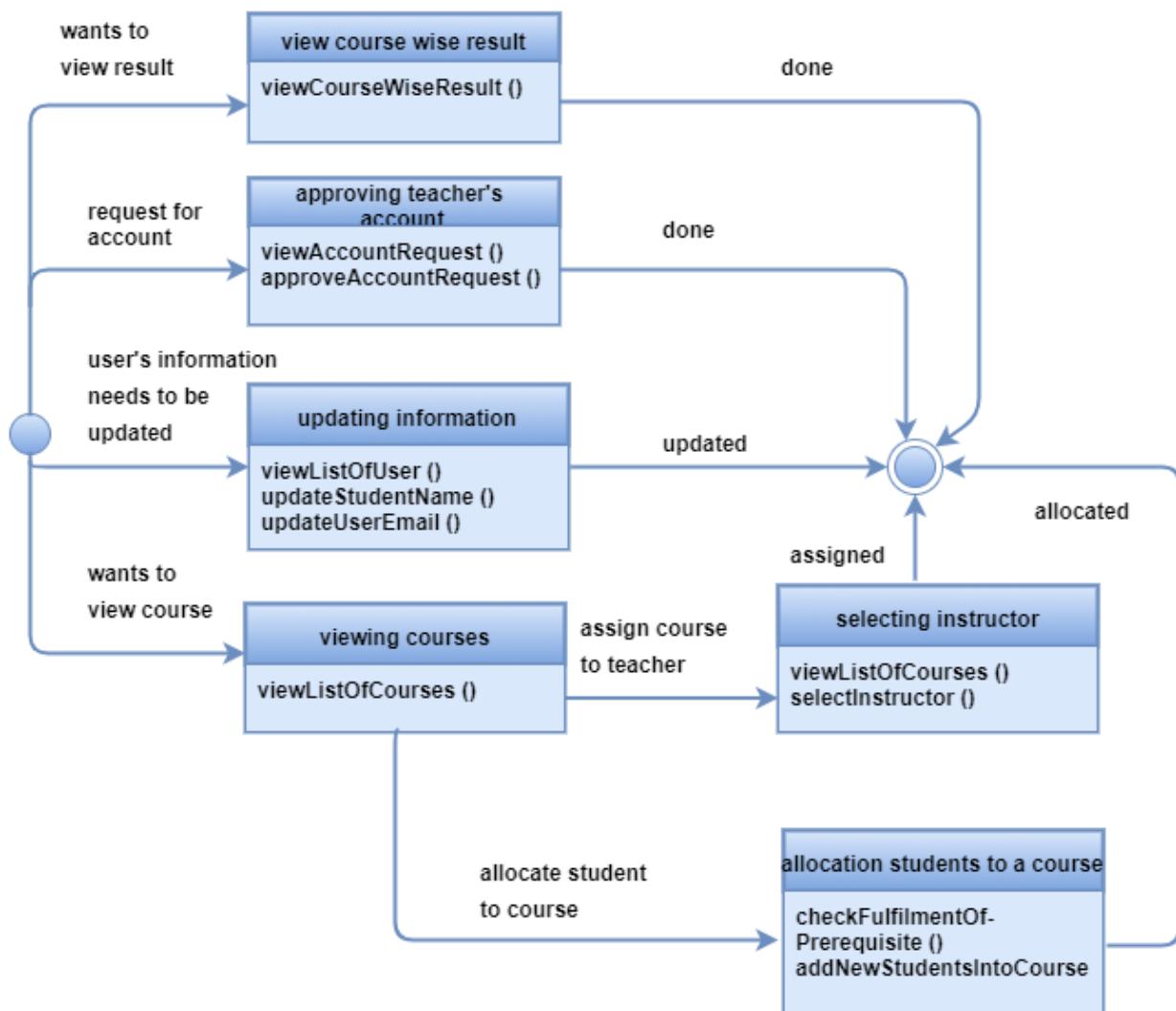


Figure 173 State Diagram of the ProgramChairperson class

Figure 174 shows state diagram of the ExamController class.

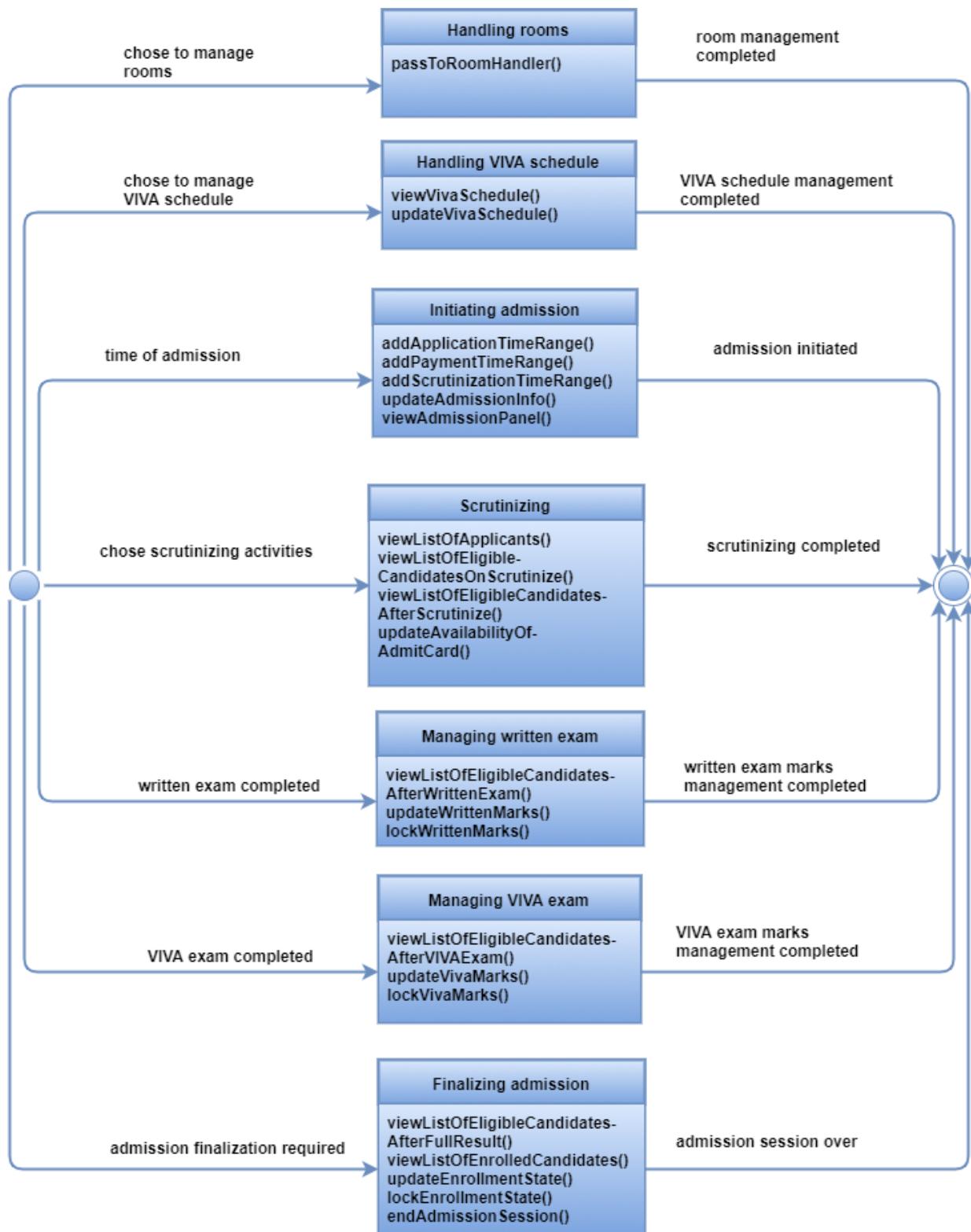


Figure 174 State Diagram of the ExamController class

Figure 175 shows state diagram of the RoomHandler class.

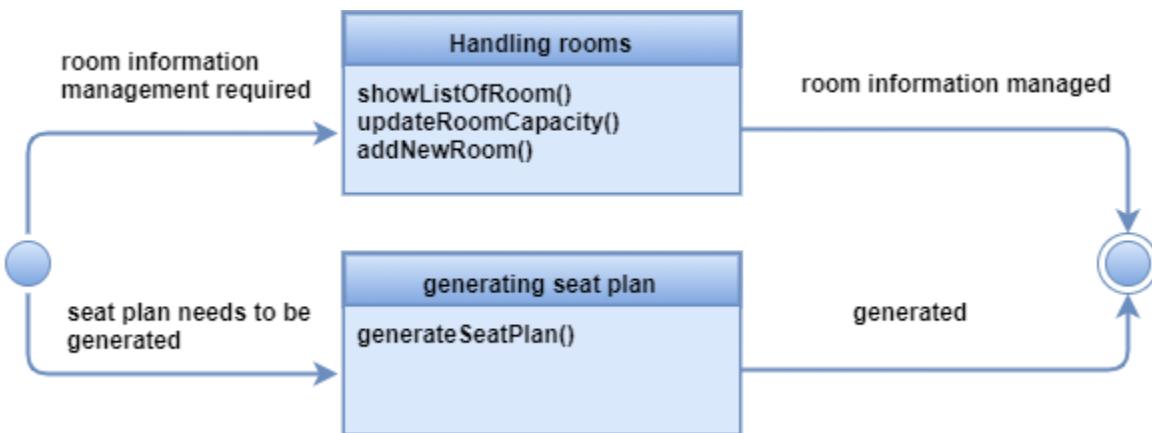


Figure 175 State Diagram of the RoomHandler class

Figure 176 shows state diagram of the Receptionist class.

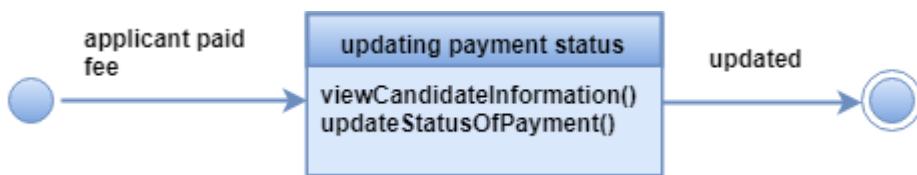


Figure 176 State Diagram of the Receptionist class

Figure 177 shows state diagram of the Scrutinizer class.

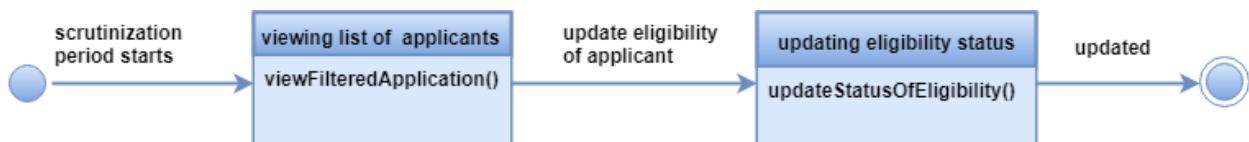


Figure 177 State Diagram of the Scrutinizer class

Figure 178 shows state diagram of the RollGenerator class.

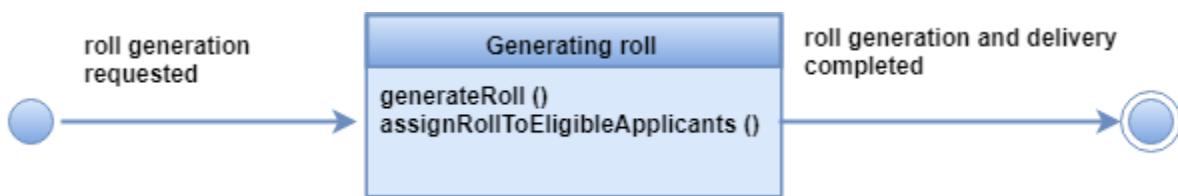


Figure 178 State Diagram of the RollGenerator class

Figure 179 shows state diagram of the Teacher class.

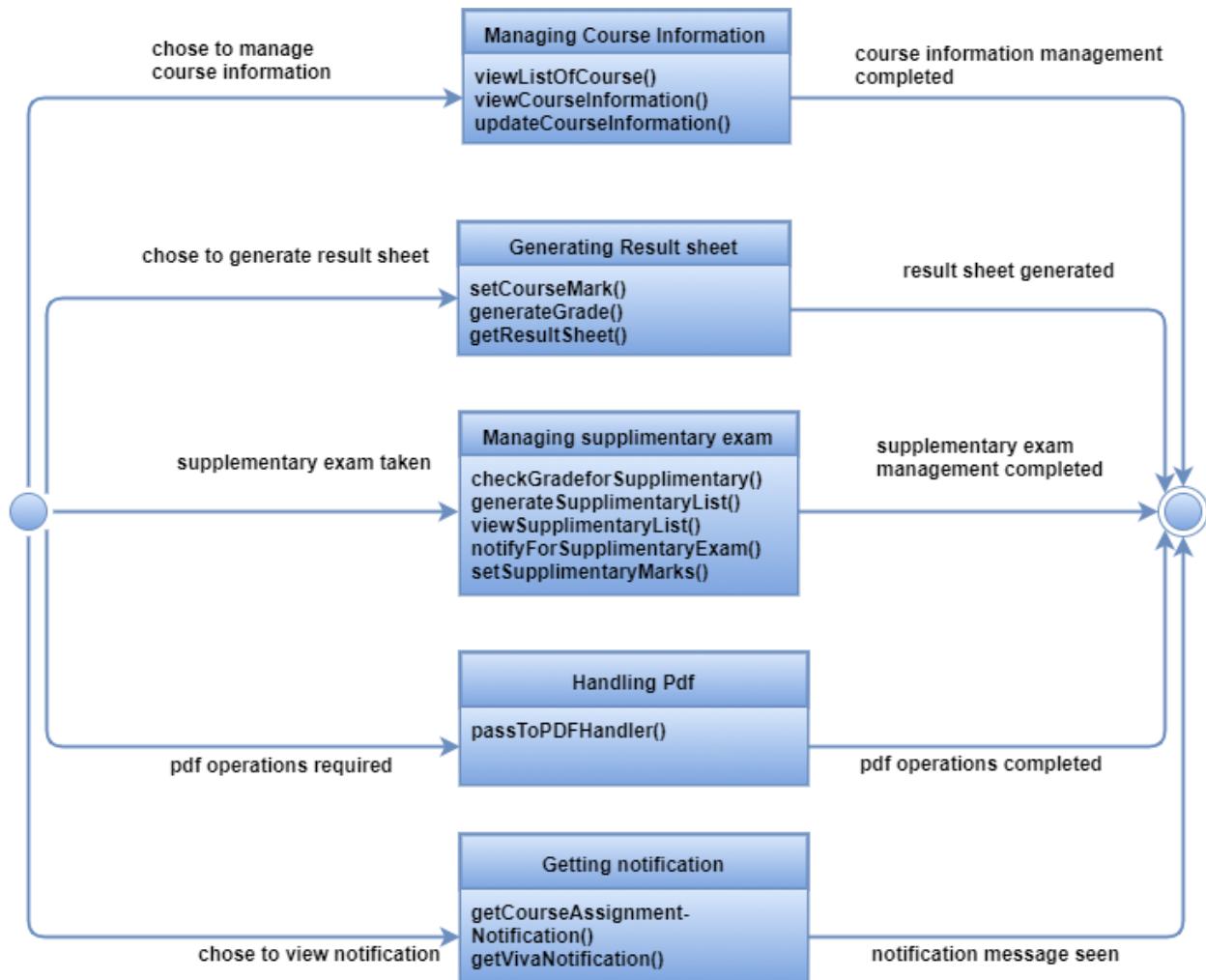


Figure 179 State Diagram of the Teacher class

Figure 180 shows state diagram of the PDFHandler class.

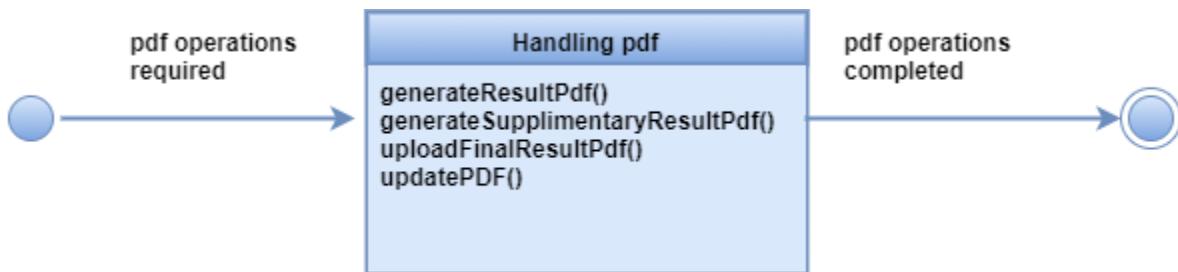


Figure 180 State Diagram of the PDFHandler class

Figure 181 shows state diagram of the Student class.

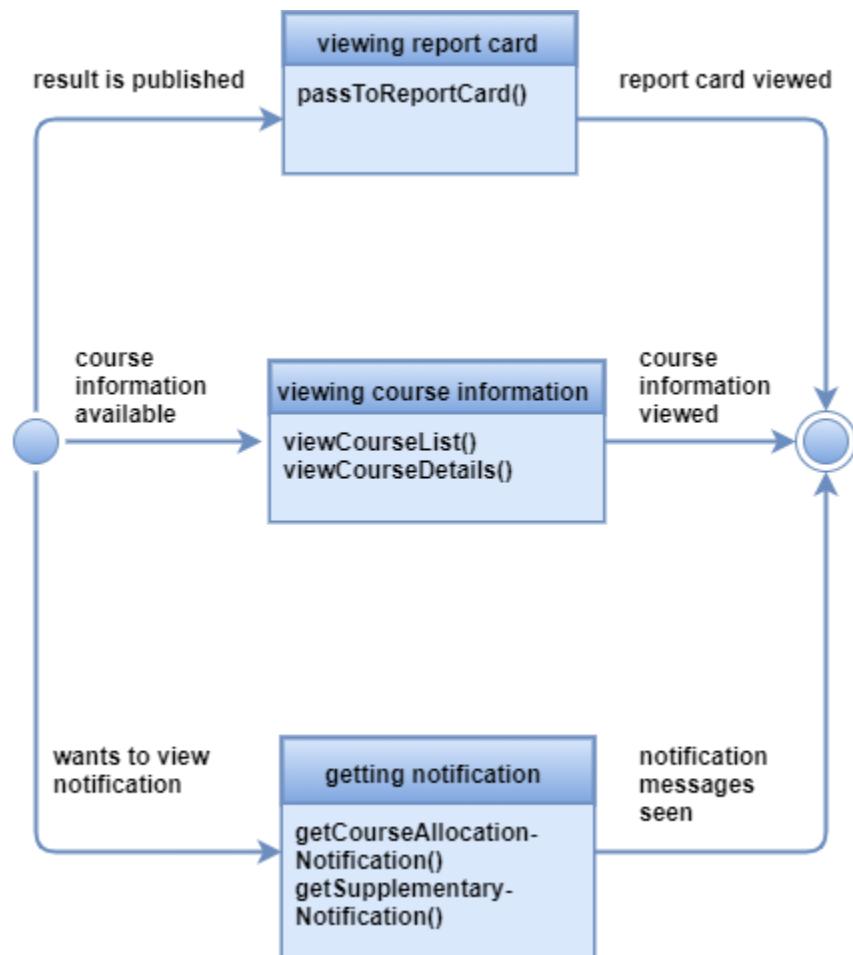


Figure 181 State Diagram of the Student class

Figure 182 shows state diagram of the ReportCard class.

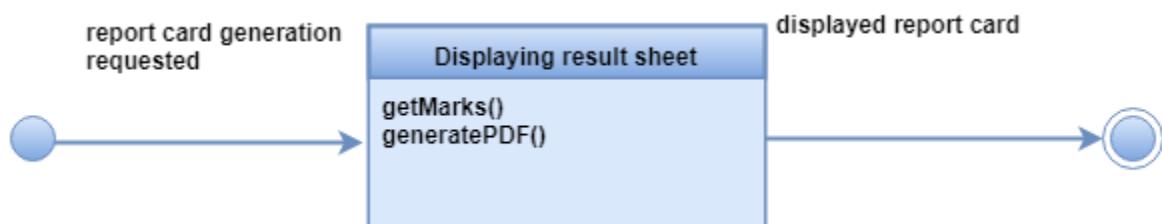


Figure 182 State Diagram of the ReportCard class

### 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 183 shows deployment diagram of PGDIT Automation System.

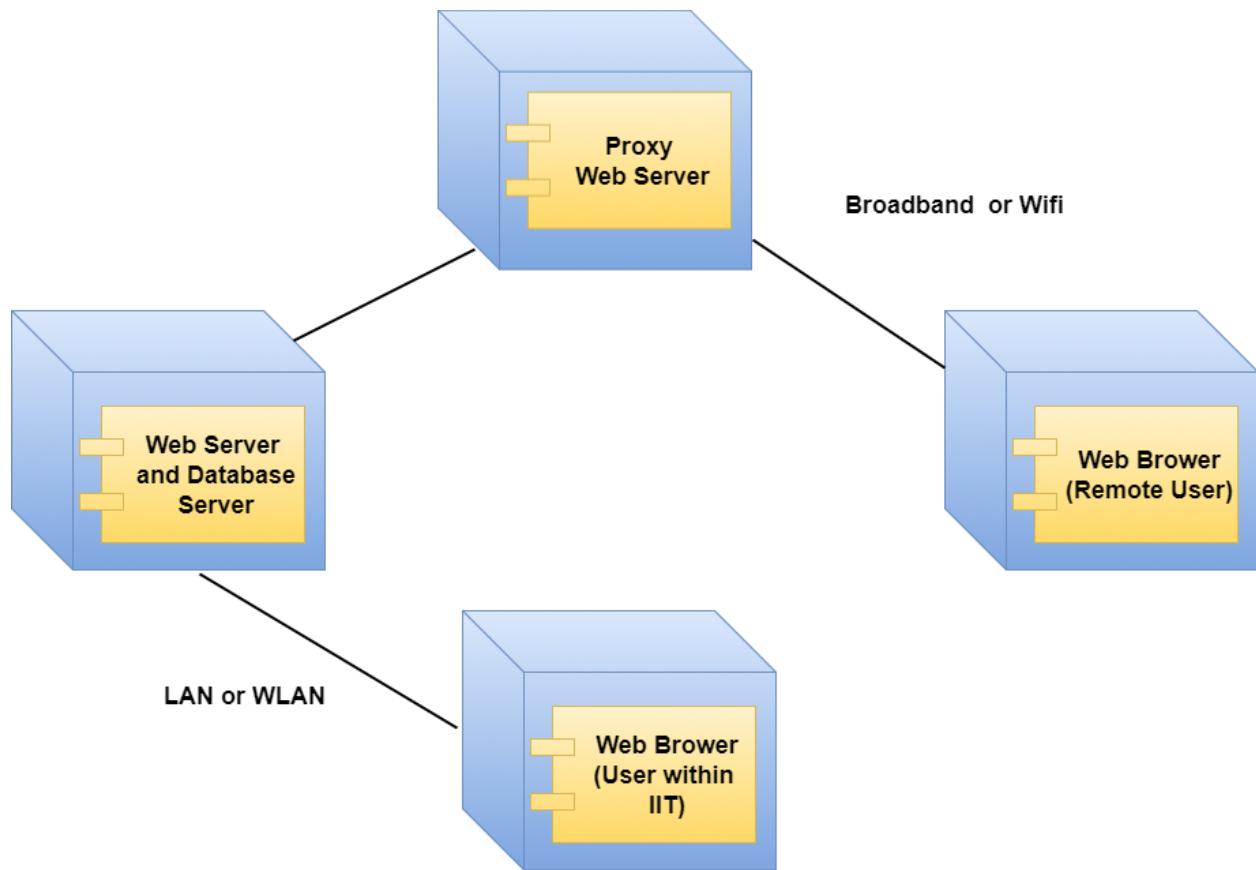


Figure 183 Deployment Diagram of PGDIT Automation System

# 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 184.

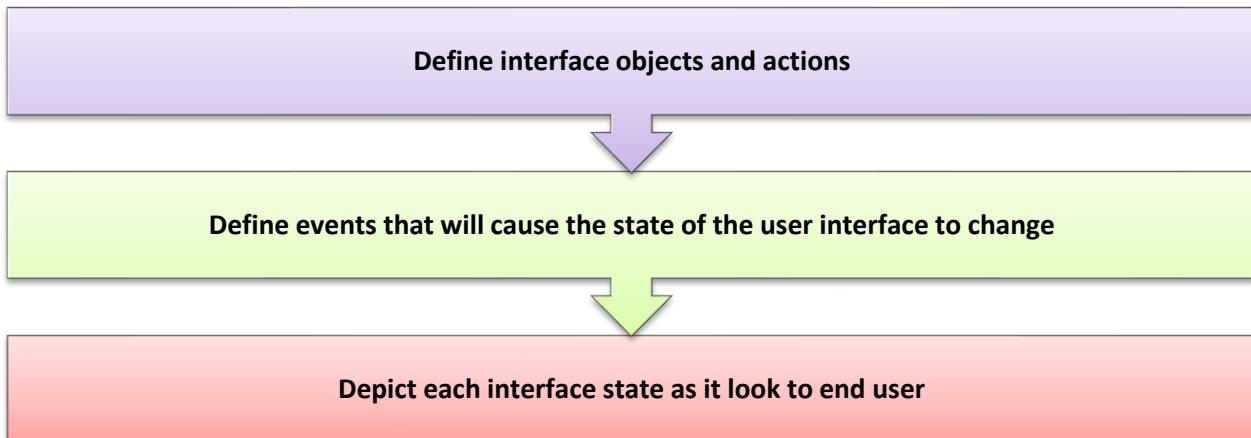


Figure 184 Interface Design Steps

#### 4.2.1 Define Interface Objects and Actions

We identified following objects and actions for the user interface.

##### External

- Home

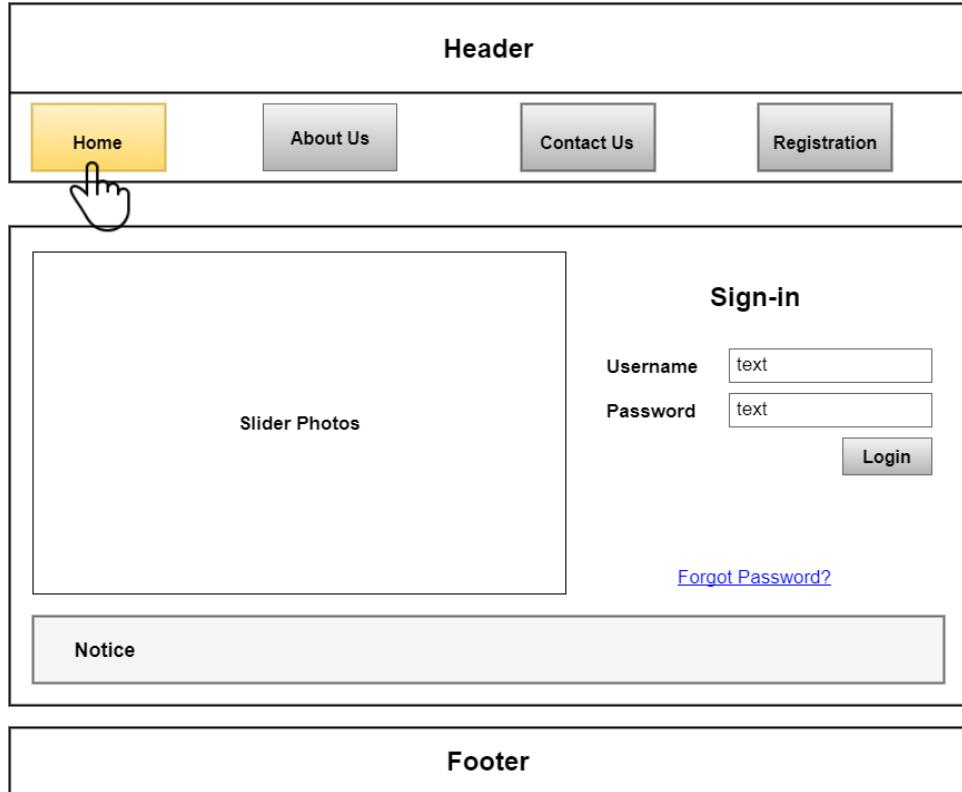


Figure 185 Home

- About us

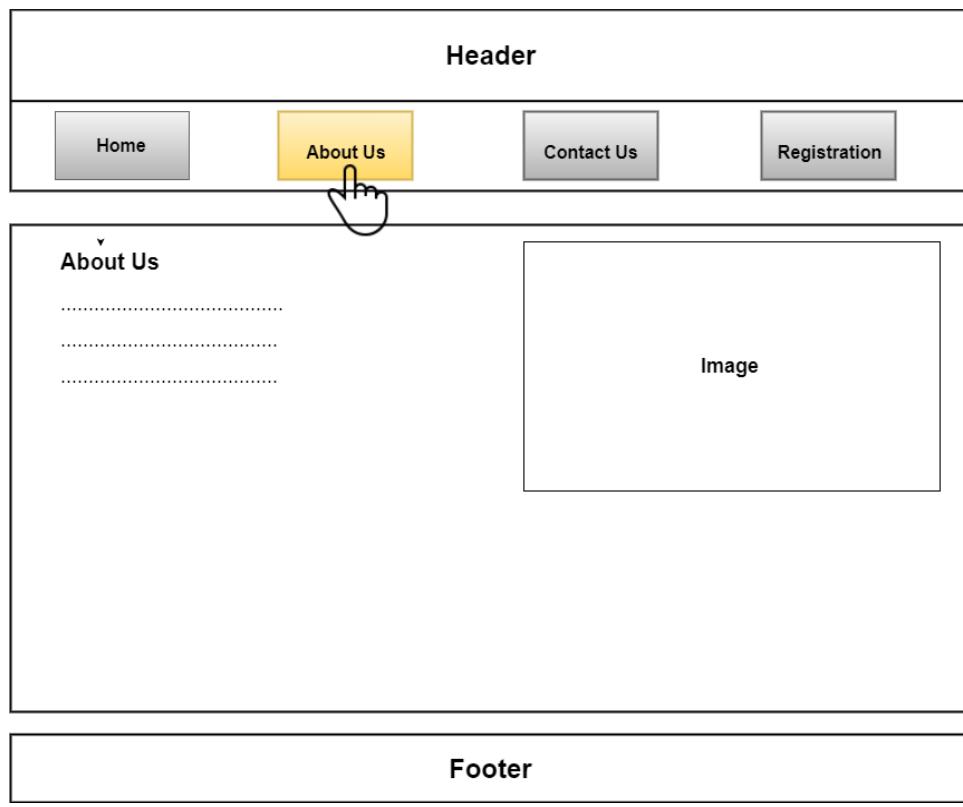


Figure 186 About us

- Contact us

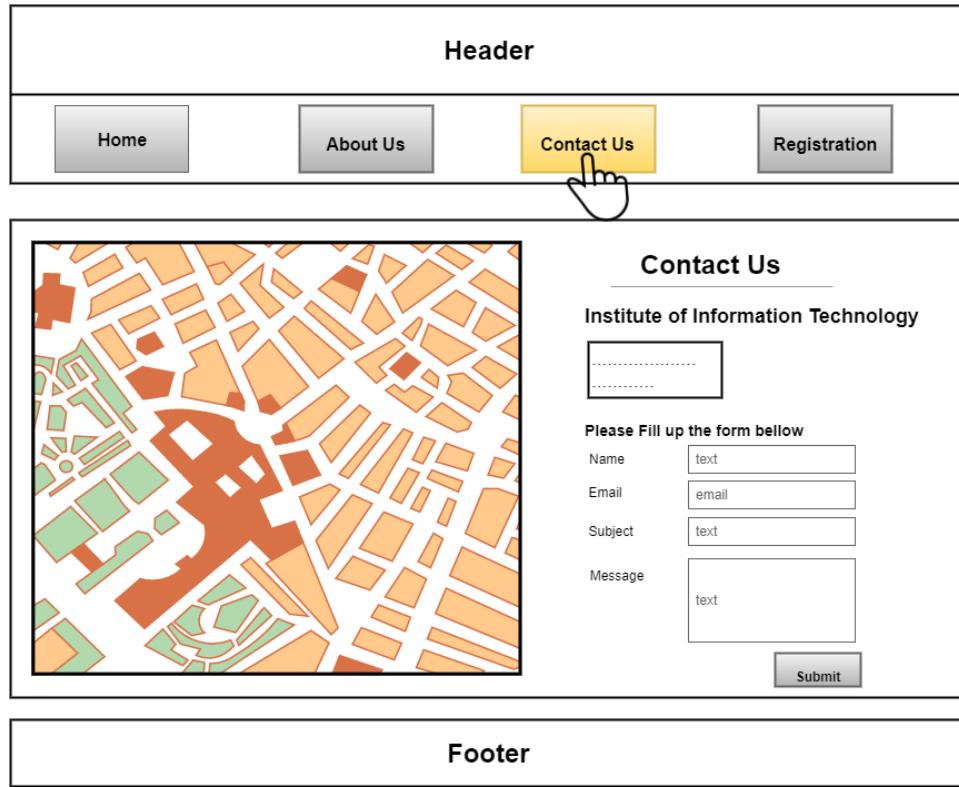


Figure 187 Contact us

- Registration

Header			
Home	About Us	Contact Us	Registration
<b>Sign Up</b>			
Full Name	<input type="text"/>		
Username	<input type="text"/>		
Email Address	<input type="email"/>		
Teacher's Registration No.	<input type="number"/>		
Father's Name	<input type="text"/>		
Mother's Name	<input type="text"/>		
Address	<input type="text"/>		
Password	<input type="text"/>		
Confirm Password	<input type="text"/>		
<input type="button" value="Send"/>			
Footer			

Figure 188 Registration

- Forgot Password

<b>Header</b>			
<a href="#">Home</a>	<a href="#">About Us</a>	<a href="#">Contact Us</a>	<a href="#">Registration</a>
<b>Forgot Password?</b>			
<b>Enter Your Username</b>			
<input type="text" value="Username"/>			
<input type="button" value="Send"/>			
<b>Footer</b>			

Figure 189 Forgot Password

<b>Header</b>			
<a href="#">Home</a>	<a href="#">About Us</a>	<a href="#">Contact Us</a>	<a href="#">Registration</a>
<b>Reset Password</b>			
Username	<input type="text"/>		
Recovery Code	<input type="text"/>		
<input type="button" value="Submit"/>			
<b>Footer</b>			

Figure 190 Reset Password

<b>Header</b>			
<a href="#">Home</a>	<a href="#">About Us</a>	<a href="#">Contact Us</a>	<a href="#">Registration</a>
<b>New Password</b>			
New Password	<input type="text"/>		
Confirm Password	<input type="text"/>		
<input type="button" value="Submit"/>			
<b>Footer</b>			

Figure 191 New Password

- 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>																

Figure 192 Admission form

- Download Admit card

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

Figure 193 Download Admit Card

■ Internal

- Program Chairperson
  - Profile

**Header**

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

**User name**

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

**Change Photo**

**Footer**

Figure 194 Profile of Program Chairperson

**X**

**User name** text

**Full Name** text

**Email address** text

**Father's name** text

**Mother's name** text

**Mobile no** number

**Address** text

**Password** \*\*\*\*

**Save**

Figure 195 Update Profile

- Course Management

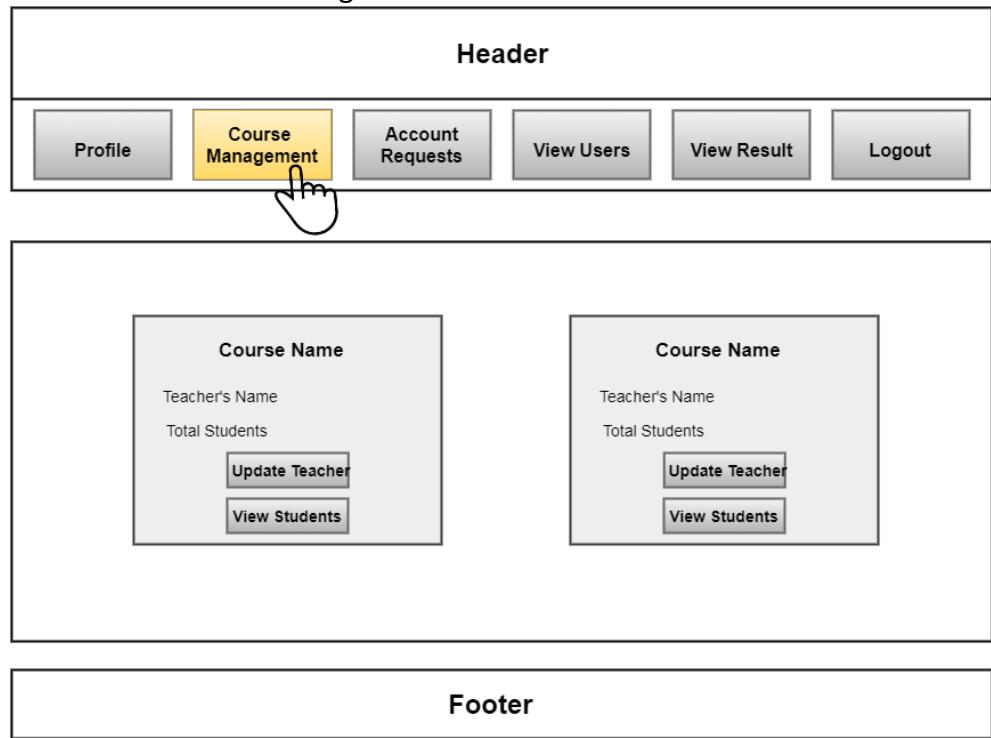


Figure 196 Course Management

A modal dialog box titled "Course name (Course Code)" is shown. It contains a "Select Teacher" label and a dropdown menu labeled "teacher's name▼". A "Save" button is at the bottom right. A close button (an "X" in a circle) is located in the top right corner of the dialog box.

Figure 197 Select Teacher

The screenshot shows a user interface for managing students of a course. At the top, the title "Course name (Course Code)" is displayed. Below it is a search bar with the placeholder "Search" and a magnifying glass icon. To the right of the search bar is a circular close button with a white "X".

Roll no	Student name
text	text
text	text

At the bottom of the interface is a grey rectangular button labeled "Allocate Student".

Figure 198 View Students of a Course

X

## Students who fulfilled Prerequisite

Search:

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

**Allocate to Course**

Figure 199 Allocate Students

- Account Requests

Header					
Profile	Course Management	Account Requests	View Users	View Result	Logout
					
Full name	Email Address	Registration No			
text	text	text	approve	decline	
text	text	text	approve	decline	

Footer

Figure 200 Account Requests

- View Users

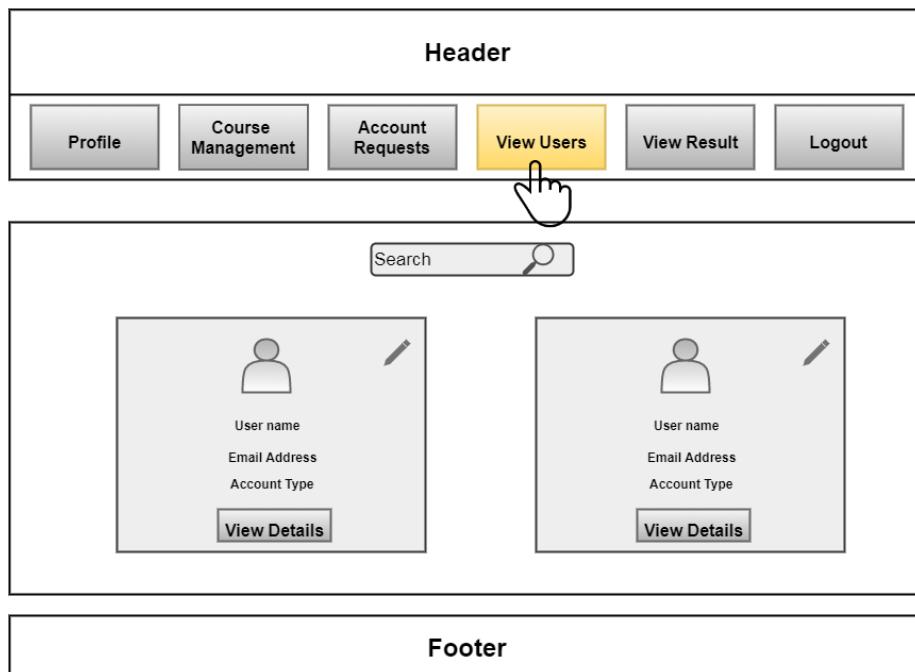


Figure 201 View Users

The diagram shows a detailed view of a user profile form. At the top is a large, empty placeholder for a user icon or profile picture. Below this placeholder is the label "User name". To the right of the placeholder is a circular icon containing a white "X". The form then lists seven data fields with their types:

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

Figure 202 View User Details



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>	

Figure 203 Edit User Information

- View Result
  - Course wise Result

<b>Header</b>																							
<a href="#">Profile</a>	<a href="#">Course Management</a>	<a href="#">Account Requests</a>	<a href="#">View Users</a>	<a href="#" style="background-color: yellow;">View Result</a>	<a href="#">Logout</a>																		
<a href="#" style="background-color: yellow; color: black; padding: 2px 10px;">Course wise Result</a> <a href="#" style="color: black; padding: 2px 10px;">Student wise Result</a> 																							
<table border="1" style="width: 100%; border-collapse: collapse;"> <tr> <td style="padding: 5px;">Select Course</td> <td style="padding: 5px;"><input style="border: 1px solid #ccc; width: 150px; height: 25px;" type="button" value="Course Name"/></td> <td style="padding: 5px;"><input style="border: 1px solid #ccc; width: 100px; height: 25px;" type="button" value="Submit"/></td> </tr> <tr> <td colspan="6" style="text-align: center; padding-top: 20px;"> <input style="width: 150px; height: 25px; margin-right: 10px;" type="text"/>  </td> </tr> <tr> <td style="width: 25%;">Roll number</td> <td style="width: 25%;">Student Name</td> <td style="width: 50%;">Obtained Marks</td> </tr> <tr> <td><input type="text"/></td> <td><input type="text"/></td> <td><input type="text"/></td> </tr> <tr> <td><input type="text"/></td> <td><input type="text"/></td> <td><input type="text"/></td> </tr> </table>						Select Course	<input style="border: 1px solid #ccc; width: 150px; height: 25px;" type="button" value="Course Name"/>	<input style="border: 1px solid #ccc; width: 100px; height: 25px;" type="button" value="Submit"/>	<input style="width: 150px; height: 25px; margin-right: 10px;" type="text"/> 						Roll number	Student Name	Obtained Marks	<input type="text"/>					
Select Course	<input style="border: 1px solid #ccc; width: 150px; height: 25px;" type="button" value="Course Name"/>	<input style="border: 1px solid #ccc; width: 100px; height: 25px;" type="button" value="Submit"/>																					
<input style="width: 150px; height: 25px; margin-right: 10px;" type="text"/> 																							
Roll number	Student Name	Obtained Marks																					
<input type="text"/>	<input type="text"/>	<input type="text"/>																					
<input type="text"/>	<input type="text"/>	<input type="text"/>																					
<b>Footer</b>																							

Figure 204 Course wise Result

- Student wise Result

Header																																												
Profile	Course Management	Notifications	View Users	View Result	Logout																																							
<div style="border: 1px solid #ccc; padding: 5px; display: inline-block;">Course wise Result</div> <div style="border: 1px solid #ccc; padding: 5px; display: inline-block; background-color: yellow;">Student wise Result</div>																																												
																																												
<table><tr><td>Select Student</td><td>Student username <input type="text"/></td><td>Submit <input type="button" value="Submit"/></td></tr><tr><td colspan="6"> </td></tr><tr><td>Search</td><td colspan="5"><input type="text"/> </td></tr><tr><td colspan="6"> </td></tr><tr><th>Course Name</th><th>Obtained Marks</th><th>Obtained Grade</th><th></th><th></th><th></th></tr><tr><td>text</td><td>marks</td><td>grade</td><td></td><td></td><td></td></tr><tr><td>text</td><td>marks</td><td>grade</td><td></td><td></td><td></td></tr></table>						Select Student	Student username <input type="text"/>	Submit <input type="button" value="Submit"/>	 						Search	<input type="text"/> 					 						Course Name	Obtained Marks	Obtained Grade				text	marks	grade				text	marks	grade			
Select Student	Student username <input type="text"/>	Submit <input type="button" value="Submit"/>																																										
Search	<input type="text"/> 																																											
Course Name	Obtained Marks	Obtained Grade																																										
text	marks	grade																																										
text	marks	grade																																										
Footer																																												

Figure 205 Student wise Result

- Teacher
  - Profile

**Header**

Profile      View Course      Manage Results      Notifications      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**

Figure 206 Teacher's Profile

- View Course

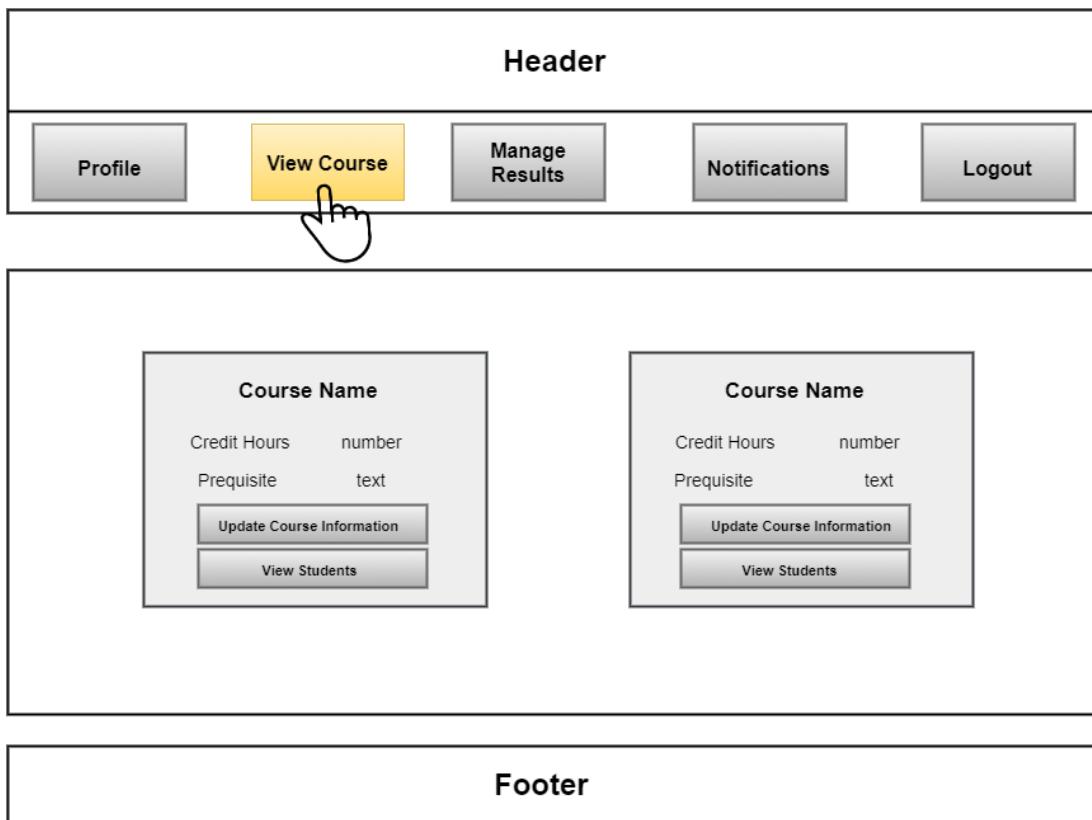


Figure 207 View Courses

**Course name (Course Code)**

Search

X

<b>Roll no</b>	<b>Student name</b>
text	text
text	text

Figure 208 View Students

- Manage Results

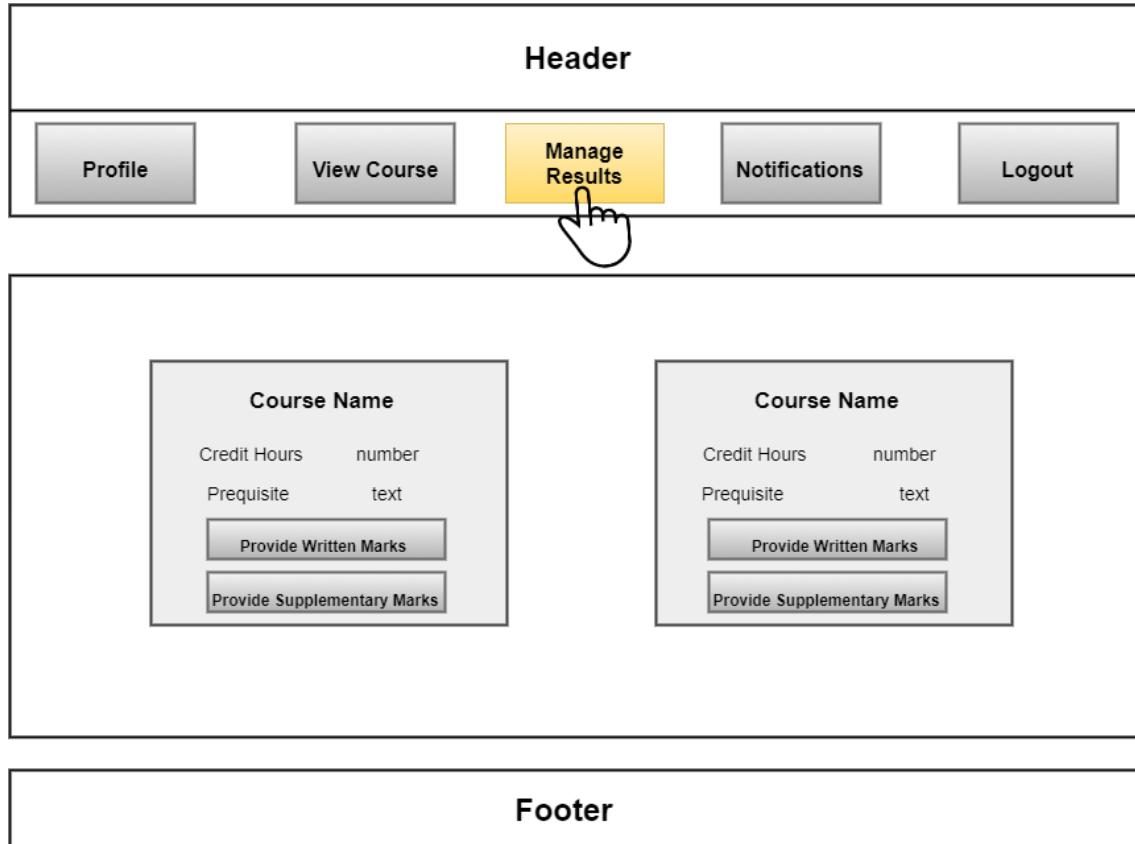


Figure 209 Manage Result



**Course (Course Code)**

Search:  

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

**Save**

Figure 210 Provide Marks

- Notifications

**Header**

[Profile](#) [View Course](#) [Manage Results](#) [Notification](#) [Logout](#)



**Viva Schedule**

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

**Footer**

Figure 211 Notifications

- Student
  - Profile

The diagram illustrates a student profile interface. At the top is a header bar containing four buttons: 'Profile' (highlighted in yellow), 'View Courses', 'Notification', and 'Logout'. A hand cursor is shown pointing at the 'Profile' button. Below the header is a main content area. On the left is a placeholder for a user photo with a 'Change Photo' button below it. To the right of the photo is the text 'User name' followed by a pencil icon. Underneath are five data fields with their respective types:

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

A footer section is located at the bottom of the main content area.

Figure 212 Student Profile

- Download Report Card

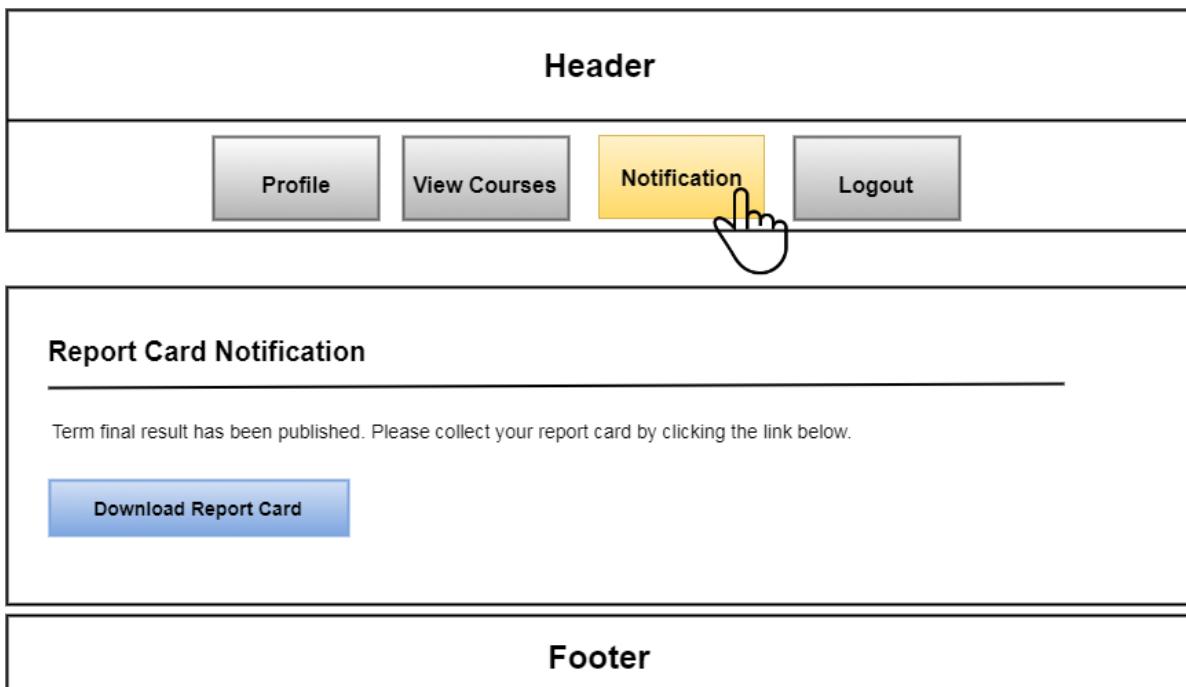


Figure 213 Download Report Card

▪ View Courses

The screenshot shows a user interface with a header containing 'Header' and four buttons: 'Profile', 'View Courses' (which is highlighted with a yellow background), 'Notification', and 'Logout'. A hand cursor is pointing at the 'View Courses' button. Below the header are two course cards. The left card is for 'Operating System Concepts & UNIX OS' with details: Course Code: PGD106, Instructor: Nawshin Nawar, Credit Hour: 03, Prerequisite: None. The right card is for 'Internet Programming' with details: Course Code: PGD102, Instructor: Sumon Ahmed, Credit Hour: 03, Prerequisite: None. At the bottom is a footer section with 'Footer'.

Figure 214 View Courses

- Exam Controller

- Profile

The screenshot shows a user interface for managing a profile. At the top, there is a horizontal navigation bar with several buttons: 'Profile' (highlighted with a yellow background), 'Manage Admission Initiation', 'Manage Admission Exam', 'Room Management', 'Term Result', and 'More'. Below this is a large central area containing a placeholder user icon and a 'Change Photo' button. To the right of the icon is a small edit icon. Below the photo area is a table with the following columns:

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

At the bottom of the page is a footer section.

Figure 215 Exam Controller's Profile

- 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

**Footer**

Figure 216 Manage Admission Initiation

X

**Introduce New Admission Session**

Application Submission Start:	<input type="button" value="▼"/>	Application Submission End:	<input type="button" value="▼"/>
Application Payment Start:	<input type="button" value="▼"/>	Application Payment End:	<input type="button" value="▼"/>
Scrutinizing Start:	<input type="button" value="▼"/>	Scrutinizing End:	<input type="button" value="▼"/>
Admission Payment Start:	<input type="button" value="▼"/>	Admission Payment End:	<input type="button" value="▼"/>

Figure 217 Setting New Admission

- Manage Admission Exam

**Header**

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



**Written Examination Completed**

**Upload Marks**

**Footer**

Figure 218 Manage Admission Exam

X

**Written Exam Marks**

Search:

Application ID	Marks
text	
text	

**Save**

Figure 219 Provide Written Marks

- Room Management

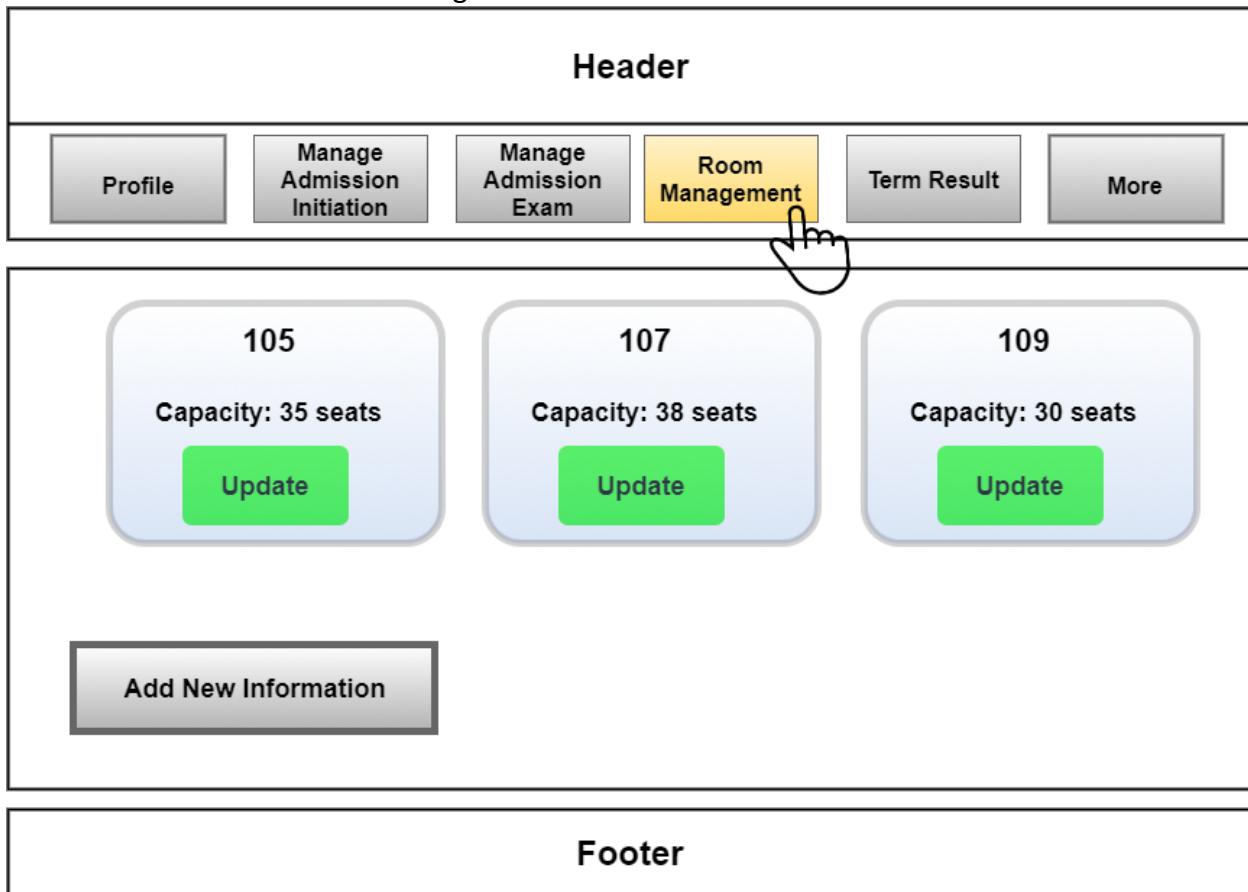


Figure 220 Room Management

- Term Result

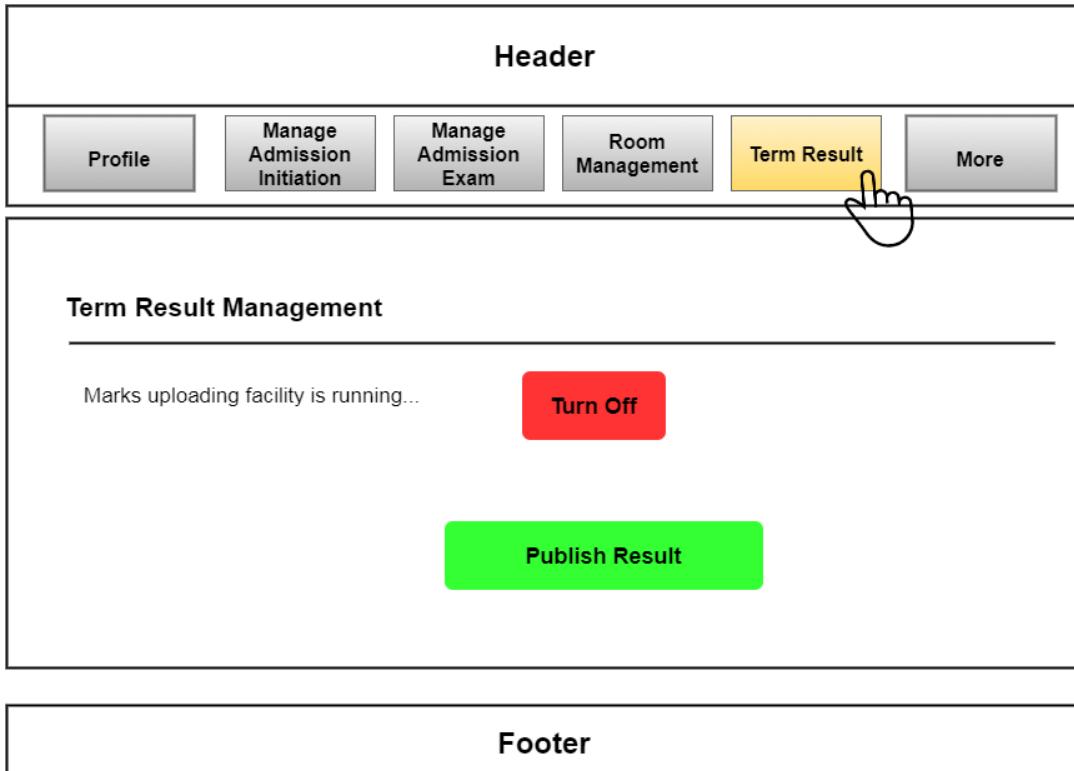


Figure 221 Term Result

▪ Update Signature

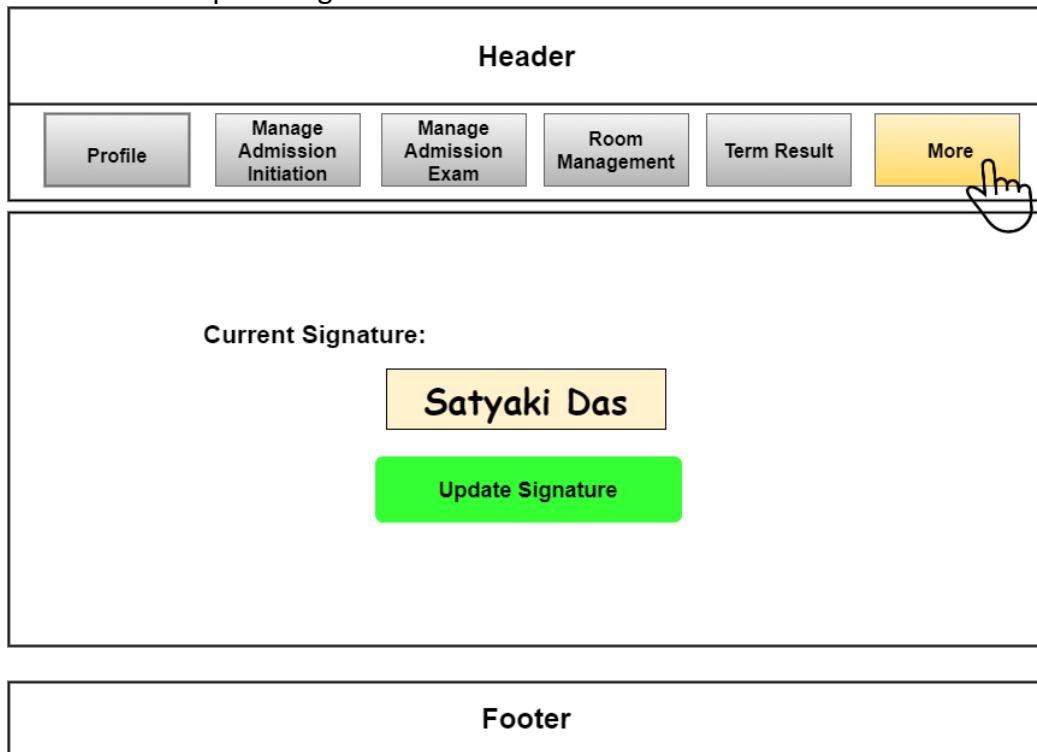


Figure 222 Update Signature

<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

**User name**

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

Figure 223 Receptionist Profile

■ View Applicants

Application ID number	Full Name text	Payment Status text	Options (Edit/Details)  Edit Details
number	text	text	 Edit Details

Figure 224 View Applicants

**Application ID (number)**

**Full Name**      **Applicant Name**

**Payment Status**

**Save**

Figure 225 Update Payment Status

- Scrutinizer
  - View Applicants

**Header**

<a href="#" style="background-color: #FFD700; color: black; text-decoration: none; font-weight: bold;">View Applicants</a>	<a href="#" style="background-color: #C0C0C0; color: black; text-decoration: none; font-weight: bold;">Logout</a>
--	---



**Search**  

Application ID	Full Name	Eligibility Status	Options (Edit/Details)
number	text	text	<a href="#">Edit</a> <a href="#">Details</a>
number	text	text	<a href="#">Edit</a> <a href="#">Details</a>

**Footer**

Figure 226 View Applicants (Scrutinizer)

X

**Application ID (number)**

**Full Name**

**Eligibility Status**

**Applicant Name**

Save

Figure 227 Update Eligibility Status

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

### External Object

#### Sign-in



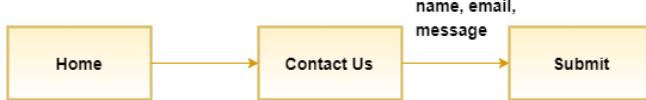
#### Registration



#### Recover Password



#### Contact Us



#### About us



#### Scrutinizer

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



##### **Edit Applicant**



## Program Chairperson

### Update Profile



### Set Course Teacher



### Allocate Students to a Course



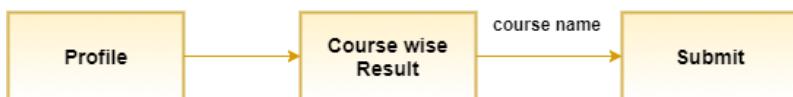
### Approve New Account



### 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

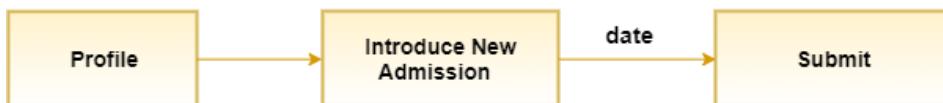


## Exam Controller

### Update Profile



### Introduce New Admission



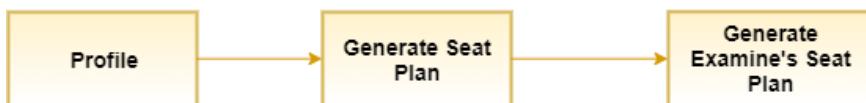
### Update Current Admission Session



### 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 shows the homepage of the PGDIT website. At the top left is the logo 'IIT University of Dhaka' and at the top right is the university crest. A navigation bar with tabs 'Home', 'About us', 'Contact us', and 'Registration' is visible. Below the navigation bar is a photograph of the PGDIT building entrance, featuring palm trees and a 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 a 'Notices' section with a message about the SEAT PLAN for the 2017-2018 admission test.

**IIT**  
University of Dhaka

**PGDIT**

**Sign in**

Username \*

Password \*

Send

Forgot Password?

**Notices**

The SEAT PLAN of PGDIT Admission Test 2017-2018 has been Published  
To Download PGDIT Admission Test SEAT PLAN [Click Here](#)

Figure 228 Home Page

## Registration Page



The image shows a registration page for IIT PGDIT. At the top left is the logo 'IIT' in blue with 'University of Dhaka' underneath. To the right is the text 'PGDIT'. Further right is the university's crest. A navigation bar at the top has links for 'Home', 'About us', 'Contact us', and 'Registration'. An orange arrow points to the 'Registration' link. The main form is titled 'Sign-up'. It includes a placeholder for a profile picture with an 'Upload Photo' button. Below are fields for 'Full Name', 'Username \*', 'Email Address \*', 'Teacher's Registration No \*', 'Father's Name', 'Mother's Name', 'Address', 'Password \*', and 'Confirm Password \*'. A 'Send' button is at the bottom right.

**IIT**  
University of Dhaka

**PGDIT**

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 229 Registration Page

## About us Page

**IIT**  
University of Dhaka

**PGDIT**



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

---

## 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 230 About us Page

## Contact us Page

The screenshot shows the 'Contact us' page of the IIT PGDIT website. At the top, there is a header with the IIT logo and the text 'University of Dhaka'. Below the header is a navigation bar with links for 'Home', 'About us', 'Contact us' (which is highlighted in orange), and 'Registration'. To the right of the navigation bar is the university's crest. The main content area contains a map of Dhaka, specifically the northern part around the University of Dhaka. A red marker indicates the university's location. To the right of the map, there is a section titled 'Contact Us' with the heading 'Institute of Information Technology'. Below this, there is contact information: 'Suhrawardi Udyan Rd, Dhaka 1200, Bangladesh' and 'Mobile: 8801779482994'. There is also a note: 'Please fill out the form below and we will contact you shortly:'. Below this note are four input fields labeled 'Name \*', 'Email \*', 'Subject', and 'Message'. At the bottom right of the form is a 'Send' button.

Figure 231 Contact us Page

## Program Chairperson: Profile

**Moumita Asad**

---

User type: Program Chairperson

Email address: moumita.asad@yahoo.com

Father's name: Asadul Ahad

Mother's name: Mahfuza Khanam

Mobile no: 01815442207

Address: 5/A, Fuller Road, University of Dhaka

Change Photo

Figure 232 Program Chairperson:Profile

Program Chairperson: Course Management

The screenshot shows a web-based course management system. At the top, there is a header with the text "PGDIT" in large blue letters, "University of Dhaka" below it, and a small logo on the right. Below the header is a navigation bar with links: "Profile", "Course Management" (which is highlighted in orange), "Account Requests", "View Users", "View Result", and "Log out". To the left of the main content area is a search bar with the placeholder "Search User". The main content area displays three course cards, each with a title, teacher information, and two buttons: "Update Teacher" and "View Students".

Course Title	Code	Teacher	Current Students
Computer Fundamentals and Office Automation	(PGD101)	Nadia Nahar	5
Introduction to Software Engineering	(PGD105)	Nawshin Nahar	21
Structured Programming	(PGD104)	Zerina Begum	17

Figure 233 Program Chairperson: Course Management

Program Chairperson: Account Requests

The screenshot shows a web-based application for managing account requests. At the top, there is a header with the logo of IIT University of Dhaka on the left and the text "PGDIT" in large blue letters. To the right of "PGDIT" is the university's crest. Below the header is a navigation bar with several links: "Profile", "Course Management", "Account Requests" (which is highlighted in orange), "View Users", "View Result", and "Log out". A yellow downward-pointing arrow is positioned below the "Account Requests" link. The main content area features a user profile box for "Alim Ul Gias". It includes a thumbnail photo of a man with dark hair and a beard, his name "Alim Ul Gias" in bold blue text, his email "Email: alim@du.ac.bd", and his registration number "Registration No: 140106". To the right of this information are two buttons: a green "Accept" button and a red "Decline" button.

Figure 234 Program Chairperson: Account Requests

## Program Chairperson: View Users

The screenshot shows a web application interface for managing users. At the top, there is a logo for "IIT University of Dhaka" on the left and "PGDIT" in large blue letters on the right. To the right of "PGDIT" is a red shield-shaped seal with a torch and other symbols. Below the header is a dark blue navigation bar with white text containing links: "Profile", "Course Management", "Account Requests", "View Users" (which is highlighted in orange), "View Result", and "Log out". A small orange downward-pointing arrow is positioned under the "View Users" link. Below the navigation bar is a search bar with the placeholder text "Search User". The main content area displays two user profiles. On the left, there is a portrait of a woman named Afrina Sharmin, described as a Receptionist with the email address afrina@yahoo.com. On the right, there is a portrait of a man named Reshad Mollick, described as a Teacher with the email address reshad@yahoo.com. Each profile has a teal "View Details" button at the bottom. There are also small edit icons (pencils) next to each portrait.

Afrina Sharmin  
Receptionist  
afrina@yahoo.com

View Details

Reshad Mollick  
Teacher  
reshad@yahoo.com

View Details

Figure 235 Program Chairperson: View Users

Teacher: Profile

**IIT**  
University of Dhaka

**PGDIT**

Profile View Course Manage results Notifications Log out

 Change Photo

## 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 236 Teacher: Profile

Teacher: View Course

**IIT**  
University of Dhaka

**PGDIT**



Profile    **View Course**    Manage results    Notifications    Log out

**Search**   

**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 237 Teacher: View Course

## Teacher: Manage Results

The screenshot shows the PGDIT Teacher: Manage Results interface. At the top, there is a header with the IIT University of Dhaka logo, the PGDIT logo, and a red shield emblem. Below the header is a navigation bar with links: Profile, View Course, Manage results (which is highlighted with an orange background), Notifications, and Log out. The main content area is titled "Search" and features a search bar with the placeholder "Search Course". Below the search bar are three course cards:

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

Figure 238 Teacher: Manage Results

## Teacher: Notifications

The screenshot shows the PGDIT Teacher: Notifications interface. At the top, there is a header with the IIT University of Dhaka logo, the PGDIT logo, and a red shield emblem. Below the header is a navigation bar with links: Profile, View Course, Manage results, Notifications (which is highlighted with an orange background), and Log out. The main content area is titled "Viva Schedule" and displays a single notification card:

- Nov 20, 2017 10.30 am  
**Viva Duty**  
Room no: 107

Figure 239 Teacher: Notifications



## Exam Controller: Profile

**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 240 Exam Controller: Profile

### Exam Controller: Introduce New Admission

**IIT**  
University of Dhaka

**PGDIT**



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

#### Introduce New Admission Session

Application Submission Start:	▼	Application Submission End:	▼
Application Payment Start:	▼	Application Payment End:	▼
Scrutinization Start:	▼	Scrutinization End:	▼
Admission Payment Start:	▼	Admission Payment End:	▼

**Submit**

Figure 241 Exam Controller: Introduce New Admission

### Exam Controller: Update Admission Information

**IIT**  
University of Dhaka

**PGDIT**



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

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

**Update**

Figure 242 Exam Controller: Update Admission Information

Exam Controller: Additional Admission Initiation Activities

The screenshot shows a web-based application interface for managing admissions. At the top, there is a logo for "IIT University of Dhaka" and a shield emblem with text. Below the header is a navigation bar with links: Profile, Manage Admission Initiation, Manage Admission Exam, Room Management, Term Result, and More. The main content area has two main sections:

- Scrutinization Session:** A message says "Scrutinization Session is ongoing...". There are two buttons: a red "Lock" button and a green "Unlock" button.
- Admit Card:** A message says "Admission payment completed for all applications.". There is a green button labeled "Make Admit Card Available".

Figure 243 Exam Controller: Additional Admission Initiation Activities

Exam Controller: Upload Marks written/VIVA

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

## Written Examination Completed

[Upload Marks](#)

Figure 244 Exam Controller: Upload Marks written/VIVA

## Exam Controller: Generate Seat Plan

The screenshot shows a web-based application for generating seat plans. At the top, there is a header with the IIT PGDIT logo and the University of Dhaka name. Below the header is a navigation bar with links: Profile, Manage Admission Initiation, Manage Admission Exam, Room Management, Term Result, and More. The main content area features two large blue rectangular buttons: "Generate Examinees' Seat Plan" and "Generate VIVA Schedule and Notify Teachers".

Figure 245 Exam Controller: Generate Seat Plan

## Exam Controller: Update Enrollment State

The screenshot shows a web-based application for updating enrollment states. At the top, there is a header with the IIT PGDIT logo and the University of Dhaka name. Below the header is a navigation bar with links: Profile, Manage Admission Initiation, Manage Admission Exam, Room Management, Term Result, and More. The main content area displays a table with four rows. The columns are labeled "Roll", "Total Marks", and "Enrollment". The first three rows show enrollment status as "Enrolled", while the fourth row shows an "Enroll" button instead of the status. A search bar is located at the top right of the table.

Roll	Total Marks	Enrollment
123456	92	Enrolled
123480	88	Enroll
123471	86	Enrolled
123450	81	Enroll

Figure 246 Exam Controller: Update Enrollment State

Exam Controller: Manage Rooms

The screenshot shows a web-based application for managing rooms. At the top, there is a header with the IIT PGDIT logo and the text "University of Dhaka". Below the header is a navigation bar with links: Profile, Manage Admission Initiation, Manage Admission Exam, Room Management (which is highlighted in orange), Term Result, and More. The main content area displays three room details in rounded blue boxes:

- Room 105:** Capacity: 35 seats. Includes an "Update" button.
- Room 107:** Capacity: 40 seats. Includes an "Update" button.
- Room 402:** Capacity: 36 seats. Includes an "Update" button.

At the bottom left of the content area, there is a button labeled "Add New Room Information".

Figure 247 Exam Controller: Manage Rooms

## Exam Controller: Term Result Management

The screenshot shows the 'Term Result Management' section of the Exam Controller interface. At the top, there is a header with the IIT University of Dhaka logo, the PGDIT logo, and the university's crest. Below the header is a navigation bar with links: Profile, Manage Admission Initiation, Manage Admission Exam, Room Management, Term Result (which is highlighted in orange), and More. The main content area has a light blue background. It displays a message 'Marks uploading facility is running...' and a red 'Turn Off' button. In the center, there is a green 'Publish Result' button. The overall layout is clean and professional.

Figure 248 Exam Controller: Term Result Management

## Exam Controller: Update Director's Signature

The screenshot shows the 'Update Director's Signature' section of the Exam Controller interface. At the top, there is a header with the IIT University of Dhaka logo, the PGDIT logo, and the university's crest. Below the header is a navigation bar with 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 a section titled 'Current Signature:' with a placeholder text 'Satyaki Das'. Below this, there is a green 'Upload New Signature' button. The overall layout is clean and professional.

Figure 249 Exam Controller: Update Director's Signature

**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 250 Exam Controller: Notifications

Student: Profile

The screenshot shows the PGDIT Student Profile page for a user named Nafis Faysal. At the top left is the IIT University of Dhaka logo. To the right is the PGDIT logo and a purple university crest. A navigation bar at the top includes links for Profile (which is highlighted in orange), View Courses, Notification, and Logout. Below the navigation bar is a profile section featuring a photo of the student, Nafis Faysal, wearing glasses and a blue shirt. A "Change Photo" button is located below the photo. To the right of the photo is the student's name, "Nafis Faysal", followed by a pencil icon for editing. A horizontal line separates this from the contact information table. The table contains the following data:

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

Figure 251 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 252 Student: View Running Courses

Student: View Courses Not Yet Taken

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

[Current Courses](#)

[Courses Not Yet Taken](#)

[Courses Already Taken](#)

**Object Oriented Programming**

**Data Structure & Algorithm**

**Course Code:** PGD201

**Instructor:** Shah Mostafa Khaled

**Credit Hour:** 03

**Prerequisite:** None

**Course Code:** PGD202

**Instructor:** Md. Mainul Islam

**Credit Hour:** 03

**Prerequisite:** PGD 101, PGD 104

Figure 253 Student: View Courses Not Yet Taken

Student: View Courses Already Taken

**IIT**  
University of Dhaka

**PGDIT**



Profile    View Courses    Notification    Logout

**Structured Programming**

<b>Course Code:</b>	PGD104
<b>Instructor:</b>	Nadia Nahar
<b>Credit Hour:</b>	03
<b>Prerequisite:</b>	None

**Computer Fundamentals  
and Office Automation**

<b>Course Code:</b>	PGD101
<b>Instructor:</b>	Rezvi Shahriar
<b>Credit Hour:</b>	03
<b>Prerequisite:</b>	None

Figure 254 Student: View Courses Already Taken

## Student: Notifications

The screenshot shows a web application for students. At the top, there is a logo for "IIT University of Dhaka" on the left and "PGDIT" in large blue letters on the right. To the right of "PGDIT" is the university's crest. Below the header is a dark navigation bar with four items: "Profile", "View Courses", "Notification" (which is highlighted in orange), and "Logout". The main content area has a light blue background. It displays a message: "Term final result has been published. Please collect your report card by clicking the link below." Below this message is a blue button labeled "Download Report Card".

Figure 255 Student: Notifications

## Scrutinizer: View Applicants

The screenshot shows a web application for scrutineers. At the top, there is a logo for "IIT University of Dhaka" on the left and "PGDIT" in large blue letters on the right. To the right of "PGDIT" is the university's crest. Below the header is a dark navigation bar with three items: "View Applicants" (which is highlighted in orange), "Logout", and a search bar labeled "Search" with a placeholder "Search Site". The main content area has a light blue background. It features a search bar at the top. Below the search bar is a table titled "View Applicants" with the following data:

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 256 Scrutinizer: View Applicants

Receptionist: Profile

**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 257 Receptionist: Profile

Receptionist: View Applicants

The screenshot shows a web-based application interface for a receptionist. At the top left is the logo of IIT University of Dhaka, followed by the text "PGDIT". To the right is the university's crest. A navigation bar at the top has two items: "Profile" and "View Applicants", with "View Applicants" being highlighted by a yellow arrow pointing to it. Below the navigation bar is a search bar labeled "Search" with a placeholder "Search". The main content area displays a table of applicant data:

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 258 Receptionist: View Applicants

## 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