SCSJ3323: Software Design and Architecture

System Design Document

AATRIM

Version 3.0

4th February 2023

School of Computing, Faculty of Engineering

Prepared by: Group 7 <AATRIM>

Revision Page

a. Overview

The current version describes the system design. It includes component model, and the models are divided into subsystems, each subsystem containing class diagrams and sequence diagrams. Overall, this document can be used for properly designing the system.

b. Target Audience

The target audience for the proposed system in this software design document are as follows:

- Applicant
- Student Recruiting Agents
- Faculty Staff
- SRAD Staff

c. Project Team Members

1. Adib Bin Morshed (A20EC4008)

2. Aaraf Islam (A20EC4001)

3. Ibrahim Elawady (A20EC4059)

4. Islam Mohammed Ruzhan (A20EC4028)

5. Musab Mudathir Altayeb (A20EC4077)

6. Mir Tamzid Hasan (A20EC4037)

Member	Task	Status
Adib Bin Morshed(A20EC4008)	Application Monitoring Subsystem (UC004, UC005)	Complete
Aaraf Islam (A20EC4001)	Application Registration Subsystem (UC002, UC003)	Complete
Ibrahim Elawady (A20EC4059)	Application Assessment Subsystem (UC007, UC008)	Complete
Islam Mohammed Ruzhan (A20EC4028)	Application Management Subsystem (UC011, UC012)	Complete
Musab Mudathir Altayeb (A20EC4077)	Application Assessment Subsystem (UC006, UC009)	Complete
Mir Tamzid Hasan (A20EC4037)	Finance Subsystem (UC001), Application Management Subsystem (UC10)	Complete

d. Version Control History

Version	Primary Author(s)	Description of Version	Date Completed
1.0	Aaraf Islam	1. Make SRS enhancement 2. Make updated sub- system module 3. Revise file	03/01/2023

Table of Contents

1.Introduction

1.1.Purpose

This SDD document describes the product perspective, product function, user characteristics, constraints, assumptions, and dependencies, apportioning of requirements, external interface requirements, system features, performance requirements, design constraints, software system attributes and other requirements of the AATRIM system. The intended audience of this document is the stakeholders, project manager and development team.

1.2.Scope

Software design documents (SDDs), also known as technical specification papers or software design documents (SDSs), explain the general architecture of a software product.

The goal of the AATRIM system's intended scope is to make links between the student admission process and other processes considerably easier and more sophisticated.

The scope of the system will include the following:

1. A module that streamlines the registration process for recruitment agents. In order for agents to assist potential students in applying to programs, they must first register with UTM. The agents can then move through with submitting student applications on behalf of the students after they are connected to the system, for the convenience of the students.
2. A module that acts as an interface for the registration of graduate and undergraduate students. Where each of the many categories of students may profit from their specific interests and needs.
3. A system monitoring module in which after applications are submitted to the faculty for approval, SRAD has to keep track of them using this module. By doing this, it is ensured that the applications can be handled in the time allotted.

4. Given that SRAD is expected to produce a statistical report for the university administration about student admittance. As a result, the UTM AATRIM system includes the statistics module which is a module for administration which permits SRAD to handle the application.
5. Module to ensure that candidates complete out the form accurately and without creating mistakes, and to provide them with their chosen user interface when they ask for it.
6. Provide applicants with a user-friendly interface so they may pay their application fees using the approved online banking services. Additionally, they are not required to accept the traditional method of scheduling appointments or traveling from bank to bank to make cash payments.

Some elements will profit from the system's engagement. These are the aspects:

1. Provide applicants and agents with secure login and menu interfaces.
2. A convenient and usable method of payment.
3. Easily view the candidates' statistics report.
4. It instantly recognizes any blank fields left behind from errors made when filling out the form, making error correction simple.

The objectives relating the system:

1. Merge the registration procedures for graduate and undergraduate students into a single system.
2. Create a mechanism for tracking performance and progress. The status of an application may be checked and tracked by agents and applicants.

1.3. Definitions, Acronyms and Abbreviation

SRAD: Student Recruitment and Admission Division

SDD: System Design Document

UTM: Universiti Teknologi Malaysia

1.4.References

- https://creately.com/blog/diagrams/class-diagram-tutorial/
- https://blog.bit.ai/software-design-document/
- https://www.figma.com/file/aNvkDZMOpMIcD5xDnxfoH5/SDA?node-id=0%3A1&t=kZIua5L7S1CbnmLK-0

1.5.System Overview

The system is about the student admission system. Through this system the students or agents can register and apply for admission. They can also make payments. The applicants can check the application status and the faculty can send the approval.

To design the system, it is divided into 5 subsystems, each with their own functionality. The basic architectural model depicts the overall design of the system. Each subsystem has been described in detail using package diagrams, class diagrams and sequence diagrams.

Lastly, there are data descriptions and data dictionaries to give a brief description of the entities and data involved in the system.

2. System Architectural Design

2.1.Architecture Style and Rationale

This system's architectural design pattern will be layered architecture. This strategy was chosen because, assuming the interface is kept up to par, it makes it possible to replace the whole layer. By combining layers of components with the same function, the structure is also easy to understand. Adding new functions and business rules is easier as long as the process and business logic are not scattered across the code. This type of architecture will also make debugging and tracing easier because the code and modules are organized and intuitively discovered.

In this case study and project, three layers—the view layer, the controller layer, and the data access layer—will be employed. The view layer will refer to the controller layer, and the controller layer will refer to the data access layer. To simplify the code and system, only closed layers will be utilized in this project.

Layered architecture is one of the most often used architectural designs. The horizontal stacking of modules or components with equivalent functionality is the idea behind layered architecture. So, each layer serves a certain purpose inside the software.

There is no restriction on the number of levels an application may have because the purpose of the layered architectural style is to have layers that support the notion of separation of concerns. The layered architecture method abstracts the system's overall viewpoint while providing enough details to understand the roles and connections among the numerous levels.

2.2.**Architecture Model**

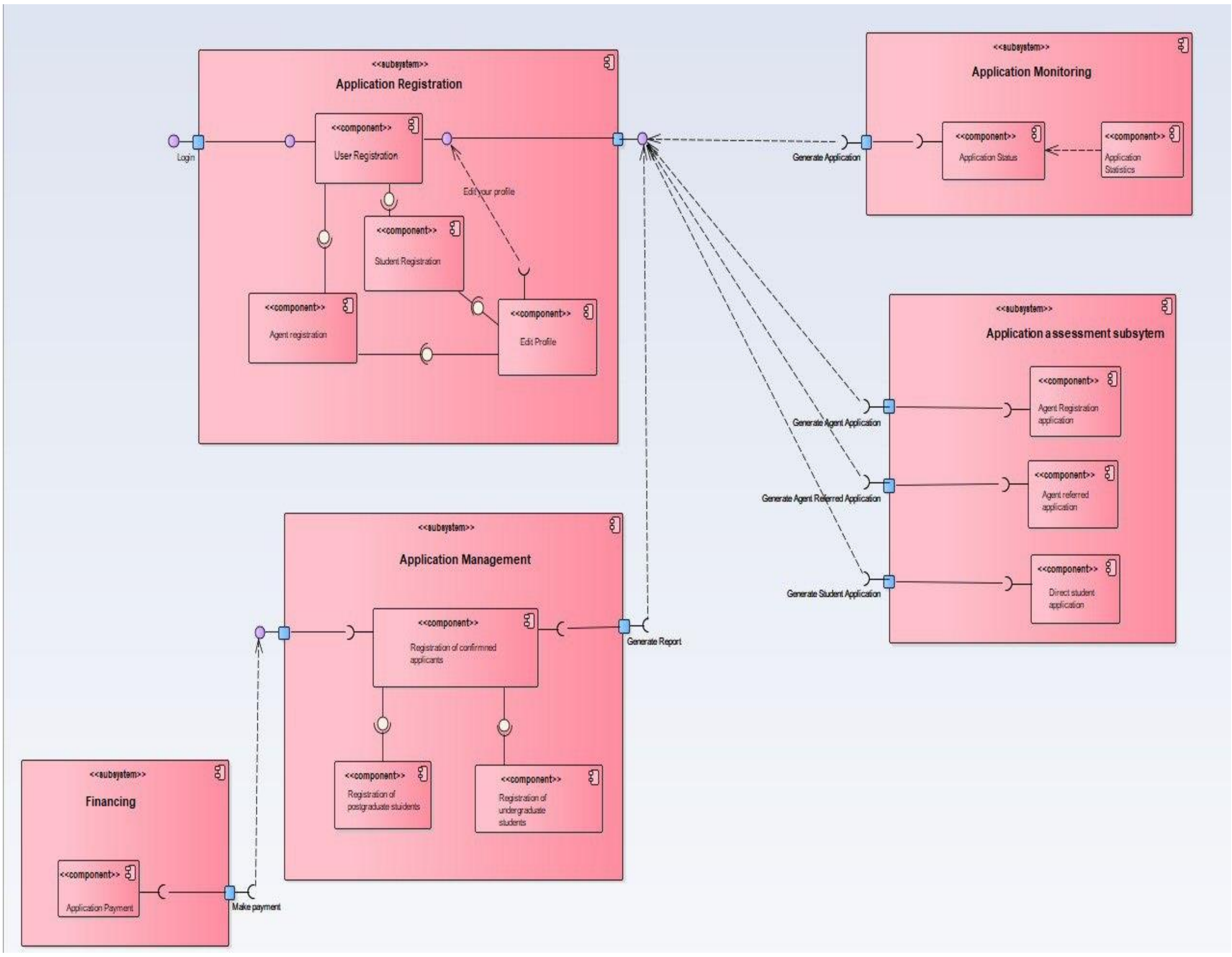


Figure: Component Model of <AATRIM System>

2.3.Use Case Diagram

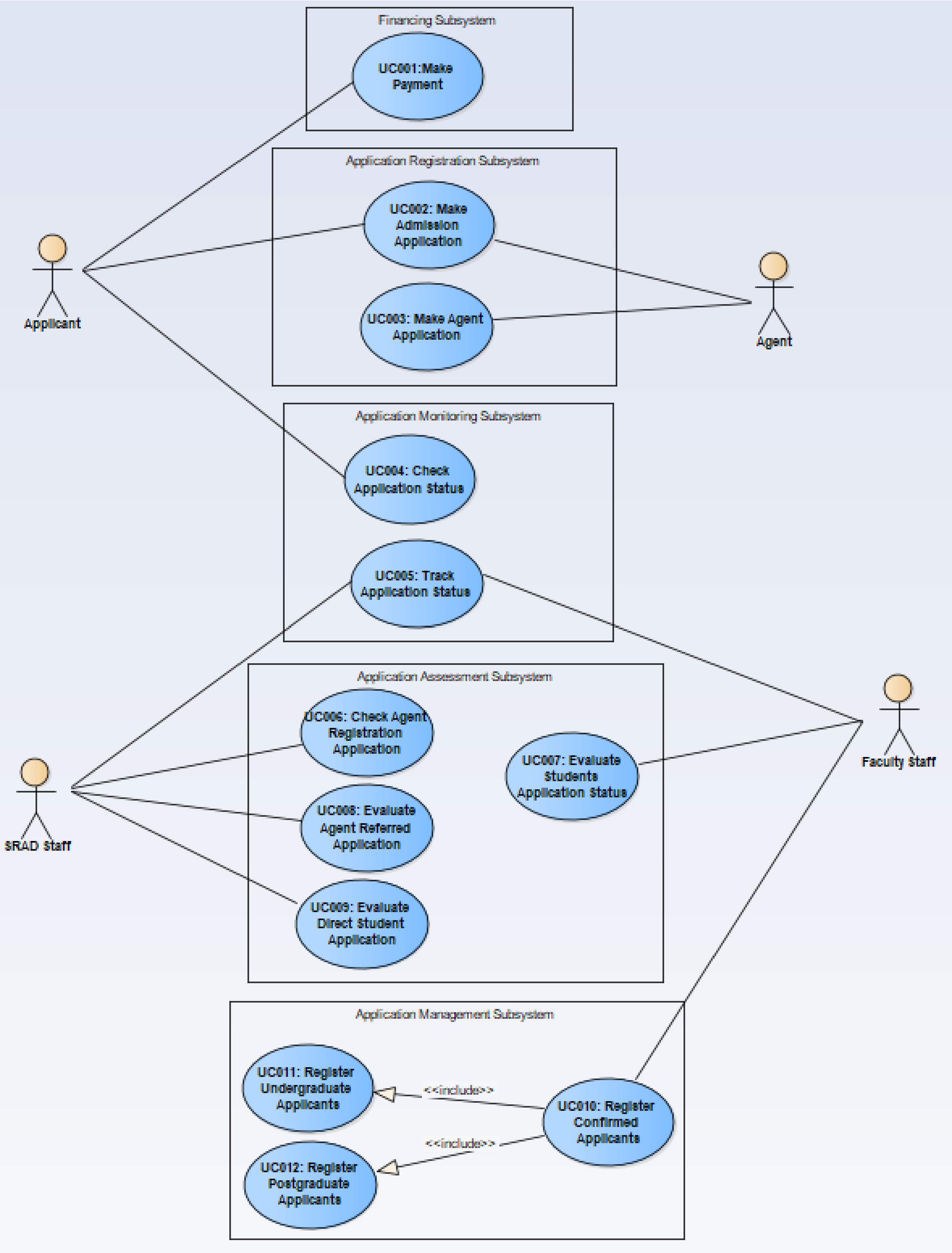

Financing Subsystem
UC001:Make Payment
Application Registration Subsystem
UC002: Make Admission Application
UC003: Make Agent Application
Applicant
Agent
Application Monitoring Subsystem
UC004: Check Application Status
UC005: Track Application Status
Application Assessment Subsystem
UC006: Check Agent Registration Application
UC007: Evaluate Students Application Status
UC008: Evaluate Agent Referred Application
UC009: Evaluate Direct Student Application
SRAD Staff
Faculty Staff
Application Management Subsystem
UC011: Register Undergraduate Applicants
UC012: Register Postgraduate Applicants
UC010: Register Confirmed Applicants
<<include>>
<<include>>

Figure: Use Case Diagram of <AATRIM System>

3. Detailed Description of Components

3.1. Complete Package Diagram

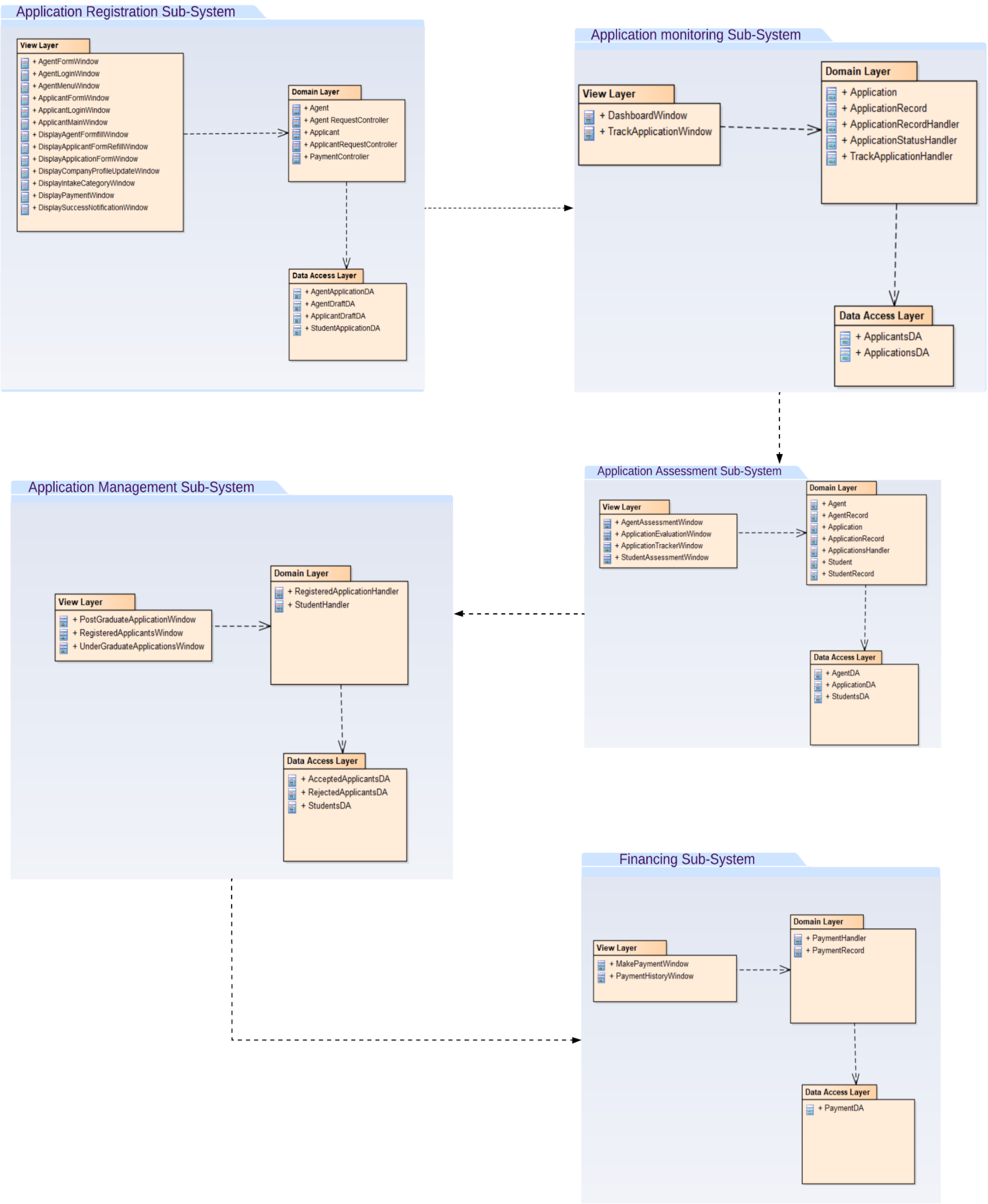


Figure: Complete package diagram of all sub-systems

3.2.Detailed Description

3.2.1.Module <Financing>

3.2.1.1.P001: Package <Make Payment>

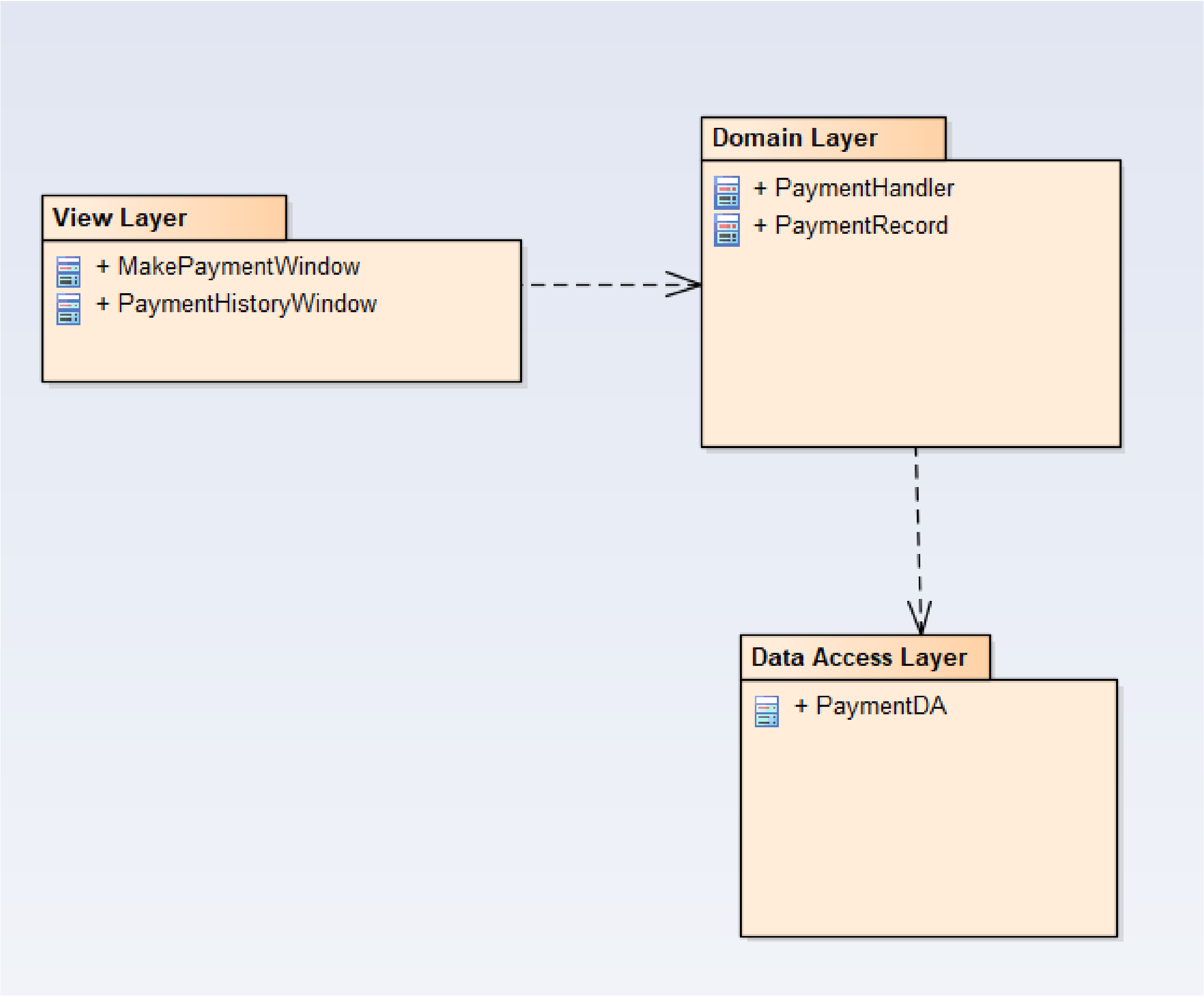


Figure: Package diagram for make payment

3.2.1.2.Class Diagram

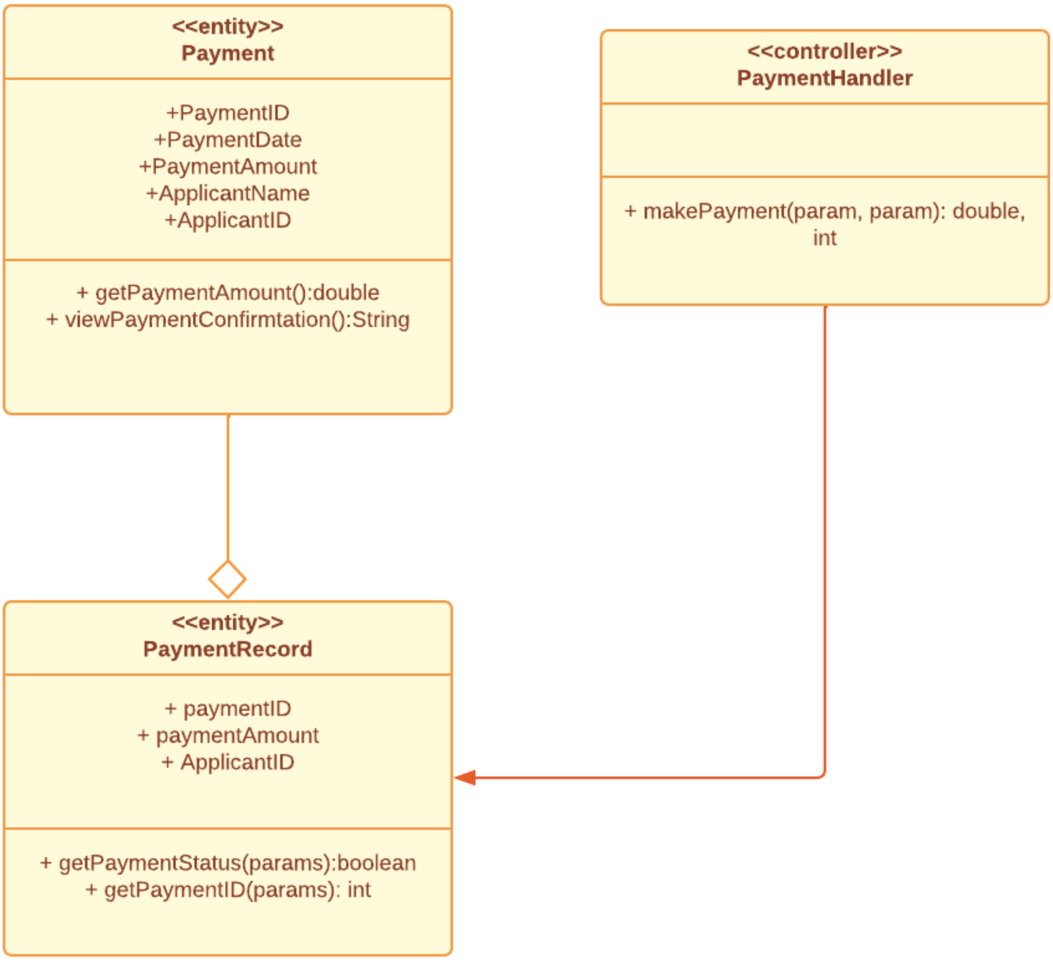


Figure: Class diagram for Financing

Entity Name	Payment
Method Name	getPaymentAmount
Input	-
Output	double

Algorithm	1. Start 2. Return payment amount 3. End

Entity Name	Payment
Method Name	viewPaymentConfirmation()
Input	-
Output	String
Algorithm	Start Check if the payment is done. Return the payment confirmation. End

Entity Name	PaymentRecord
Method Name	getPaymentStatus
Input	-
Output	boolean
Algorithm	1. Start 2. Check if payment confirmed.

	3. If payment confirmed return true. 4. Else return false. 5. End

Entity Name	Payment
Method Name	getPaymentID()
Input	**-**
Output	int
Algorithm	1. Start 2. Get the payment ID 3. return the payment ID 4. End

3.2.1.3.Sequence Diagrams

a) **SD001:** Sequence diagram for Making Payment

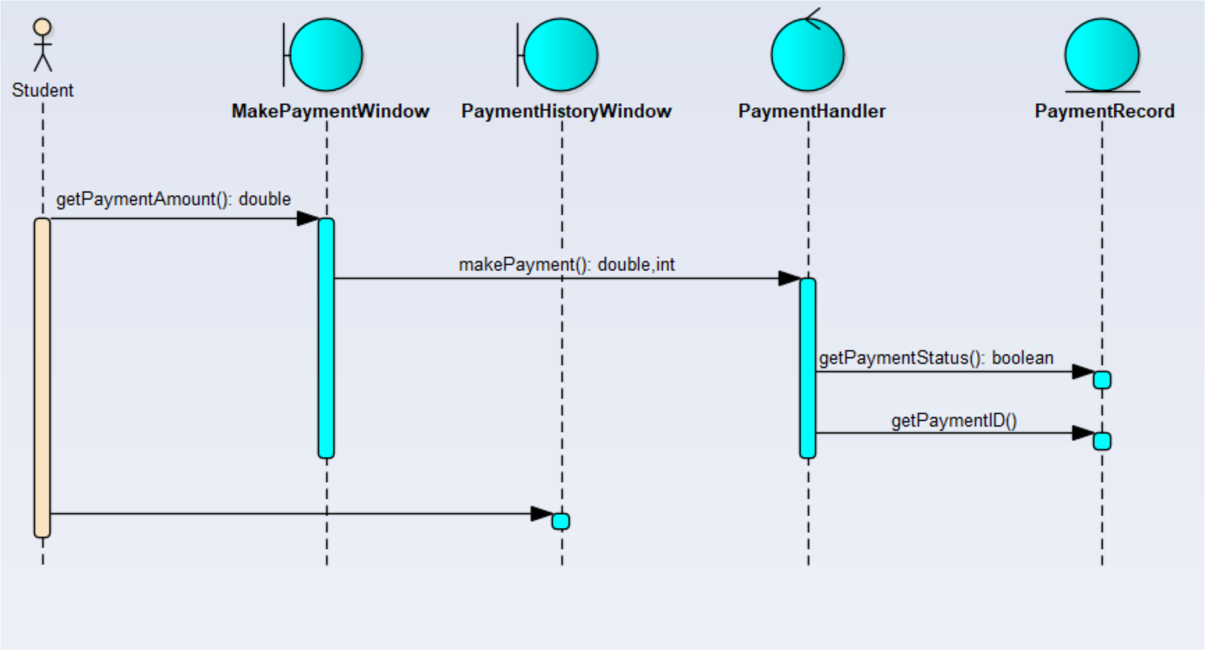


Figure: Sequence Diagram of < Make Payment scenario>

3.2.2.Module <Application Registration>

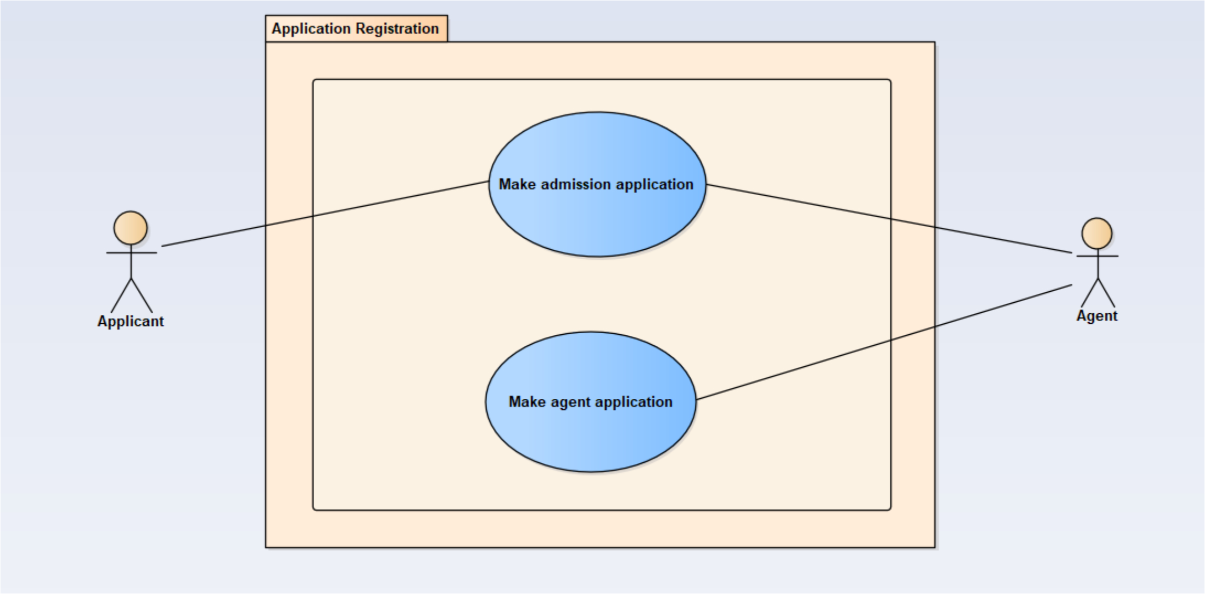


Figure: Application registration module

3.2.2.1.P002: Package <Application Registration>

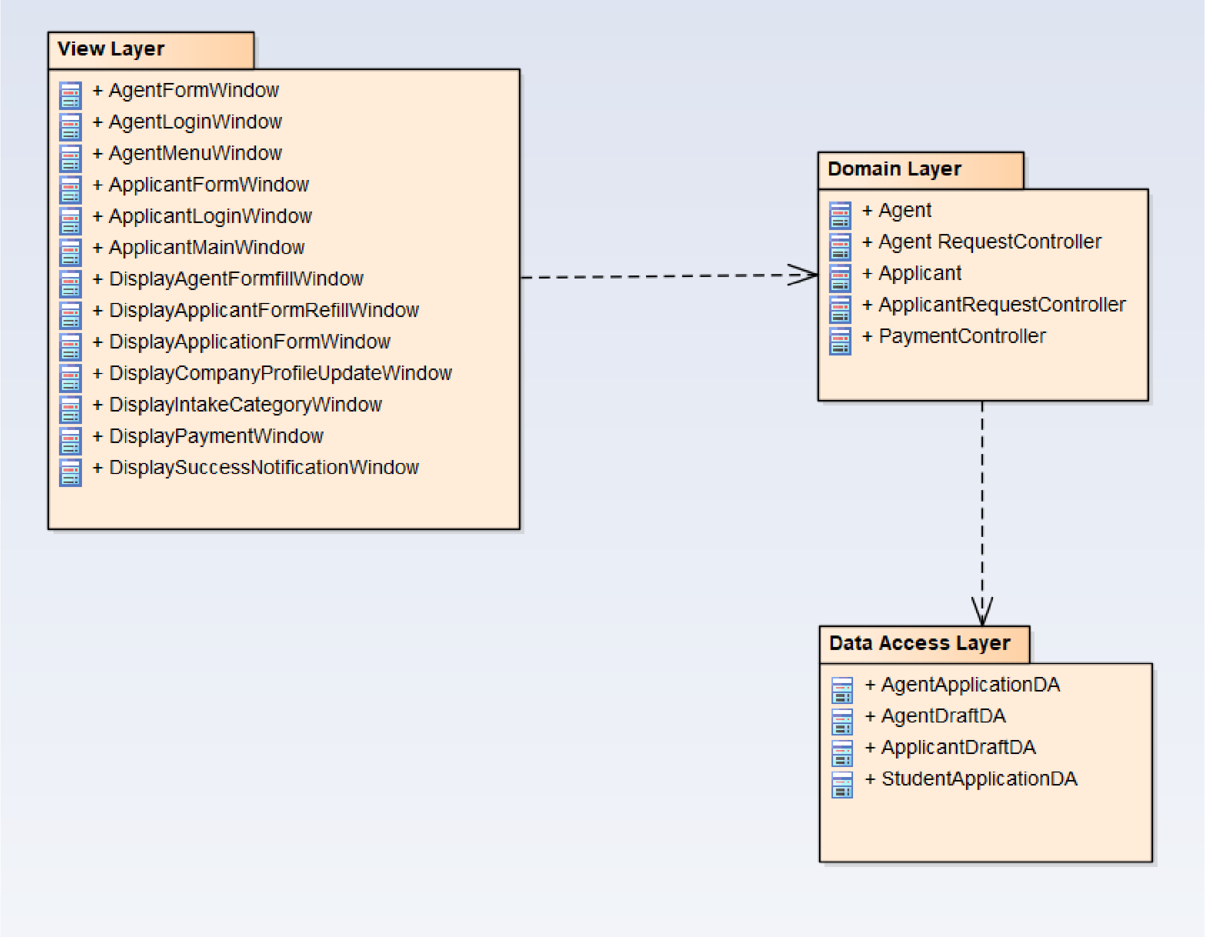


Figure: Package diagram for <Application registration>

3.2.2.2.Class Diagram

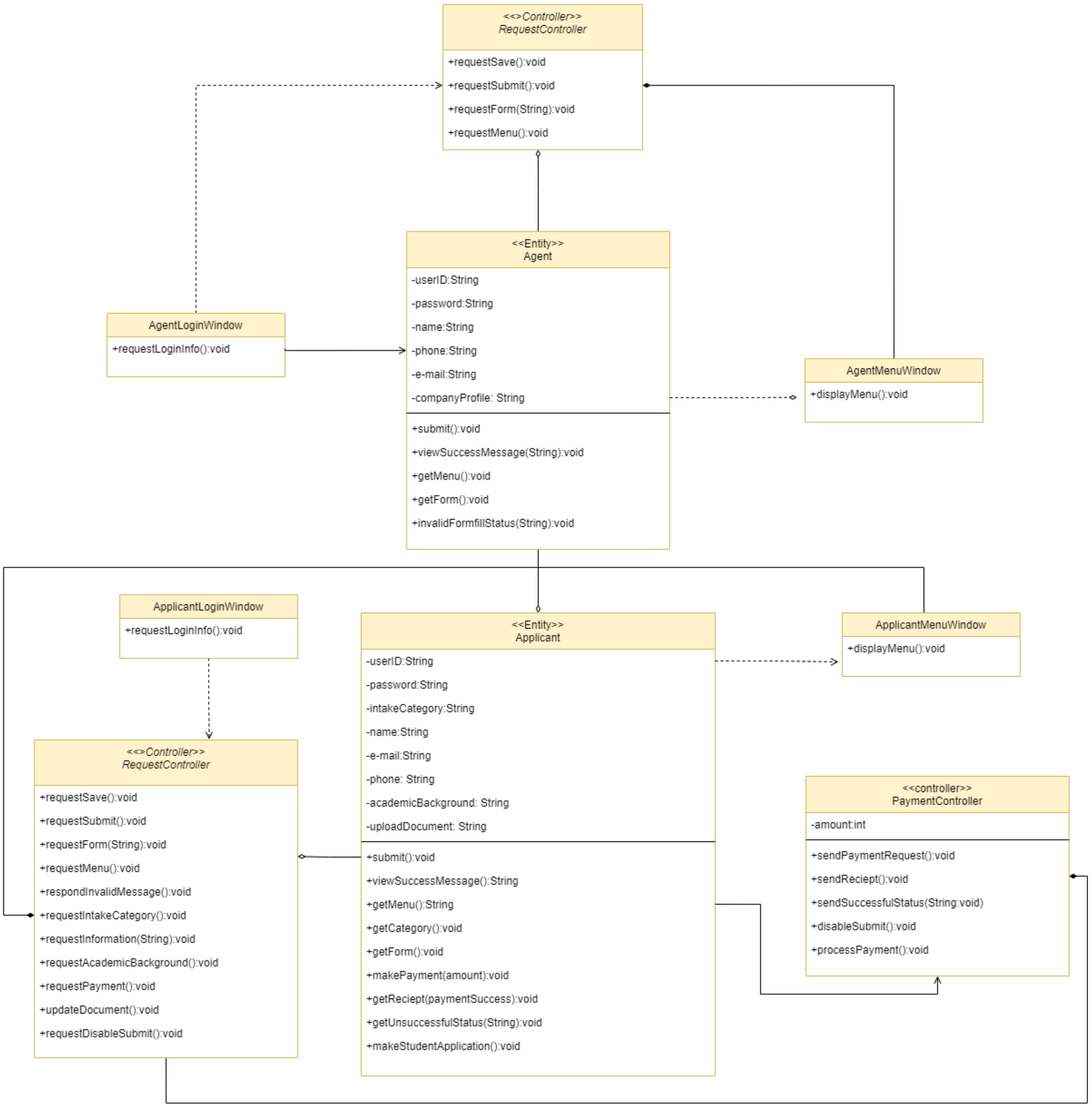


Figure: Class diagram for <Application registration>

Entity name	Agent
Method name	Submit()
Input	String

Output	viewSuccessMessage(), invalidFormfillStatus()
Algorithm	Start 1. Login to system 2. Request menu 3. Select agent application 4. View form 5. Input personal information 6. Submit 7. Continue normal flow if no refill notification shows If Invalid form fill status shows 1. View form 2. Input empty or solve incorrect information 3. Submit and follow normal flow Continuing to normal flow 1. Input company profile 2. Submit company profile 3. View success status message End

Entity name	Agent, Applicant

Method name	Submit()
Input	String
Output	viewSuccessMessage(), getUnsuccessfulStatus(), getForm()
Algorithm	Start 1. Login to system i. Agent login ii. Applicant login 2. Request menu 3. Select make student application 4. View intake category 5. Input intake category 6. Get form 7. Fill up form with information 8. Save 9. Select course 10. Input academic background 11. Submit 12. Proceed to payment If Invalid form fill status shows 1. View form 2. Input empty or solve incorrect information 3. Save and follow from normal flow 9

	End
Method name	makePayment()
Input	Amount
Output	getReciept(), paymentSuccess()

3.2.2.3.Sequence Diagrams

a) SD001: Sequence diagram for making agent application

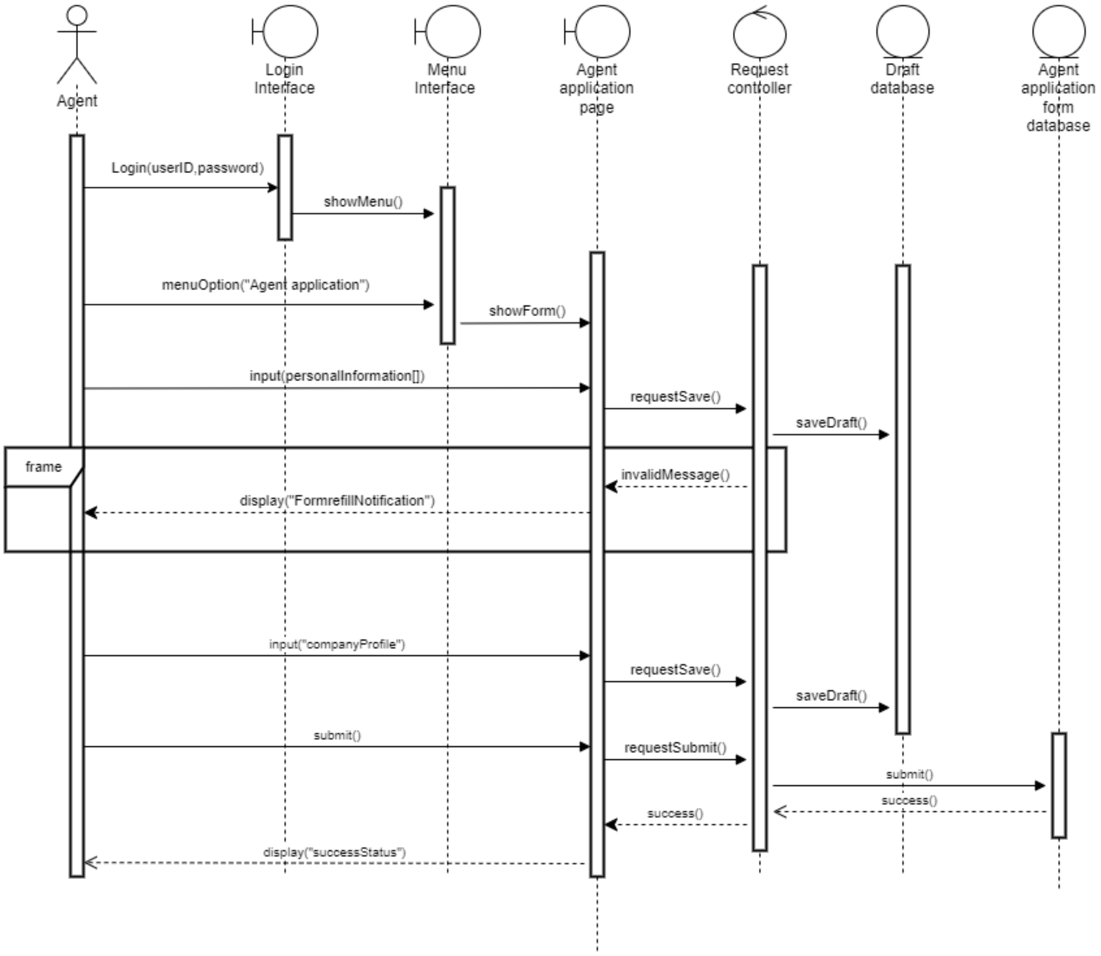


Figure: Sequence diagram of <make agent application>

b) SD002: Sequence diagram for making student application

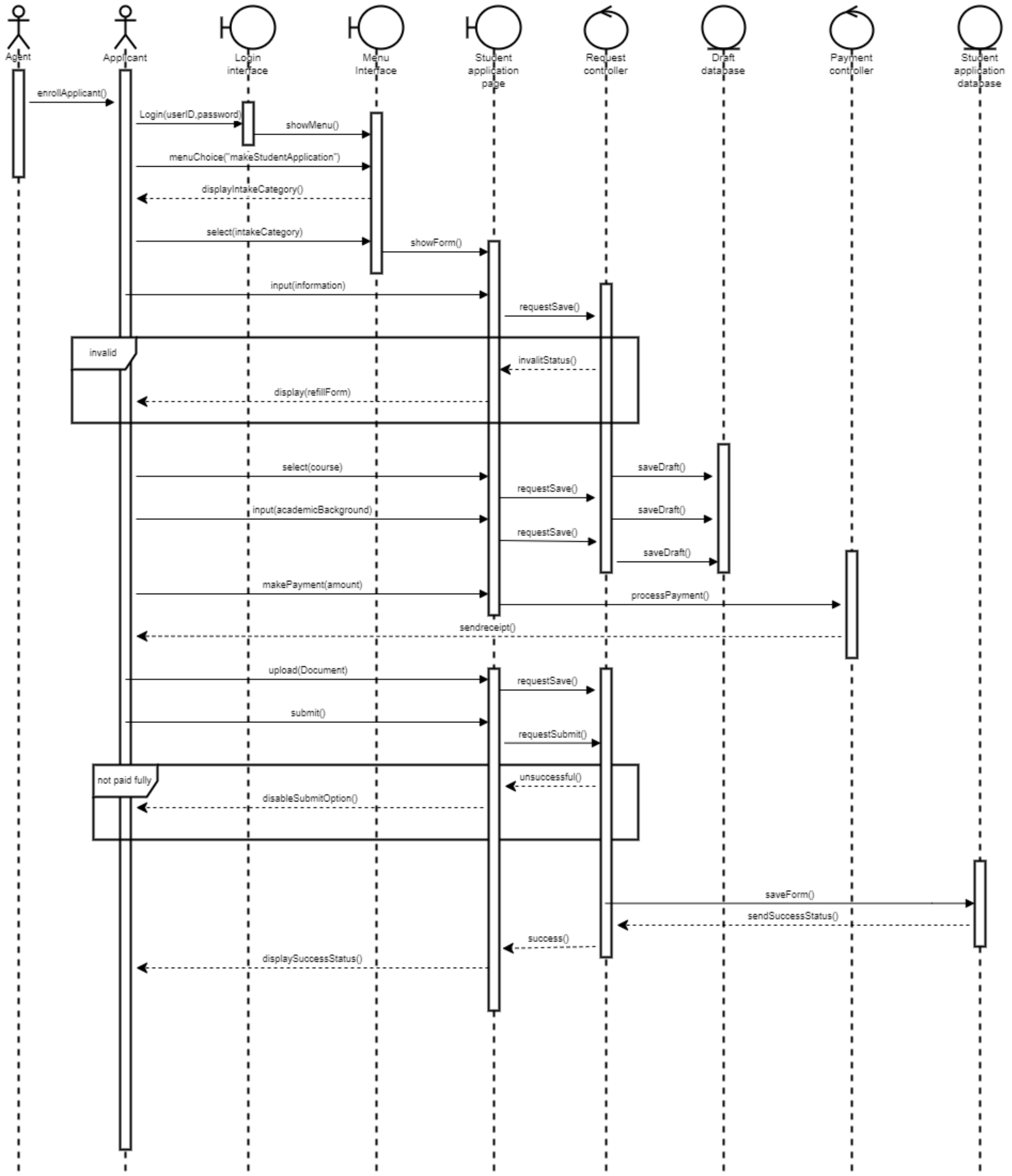


Figure: Sequence diagram of <make student application>

3.2.3.Module <Application Monitoring>

3.2.3.1.P003: Package<File Monitoring>

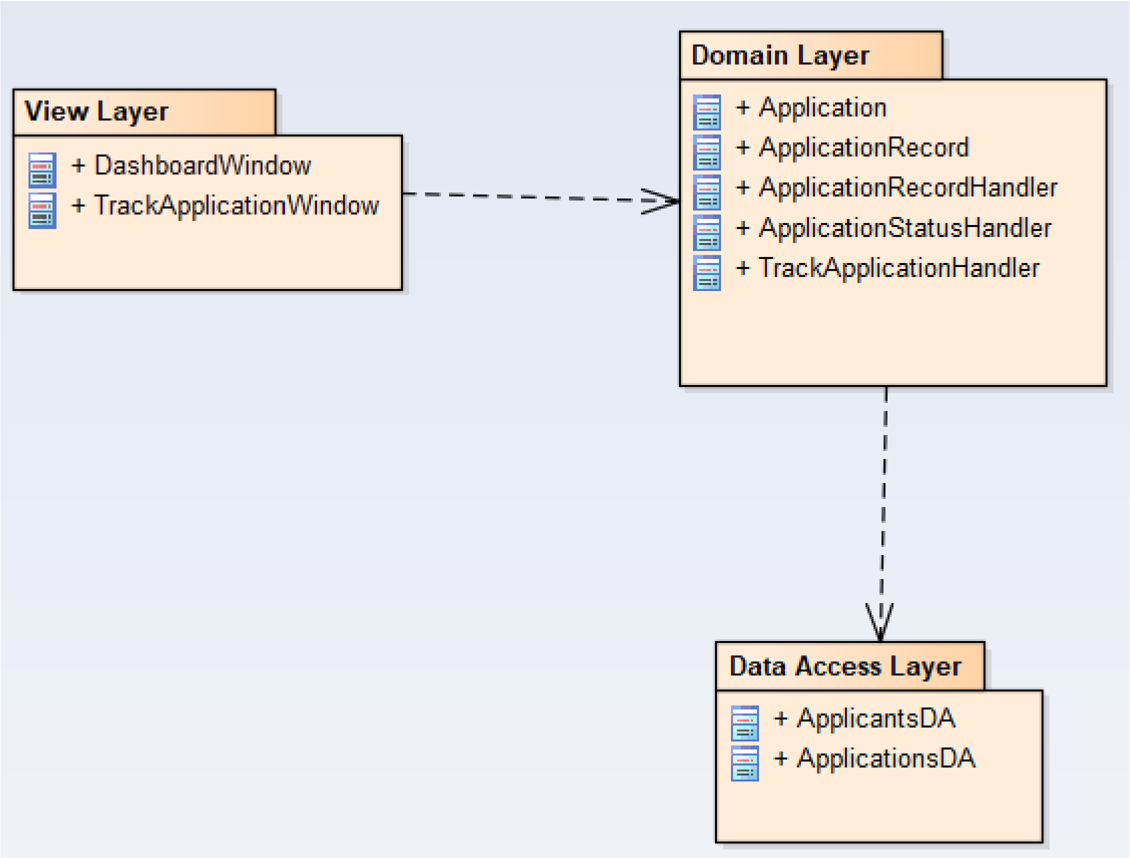


Figure: Package diagram for <File Monitoring>

3.2.3.2.Class Diagram

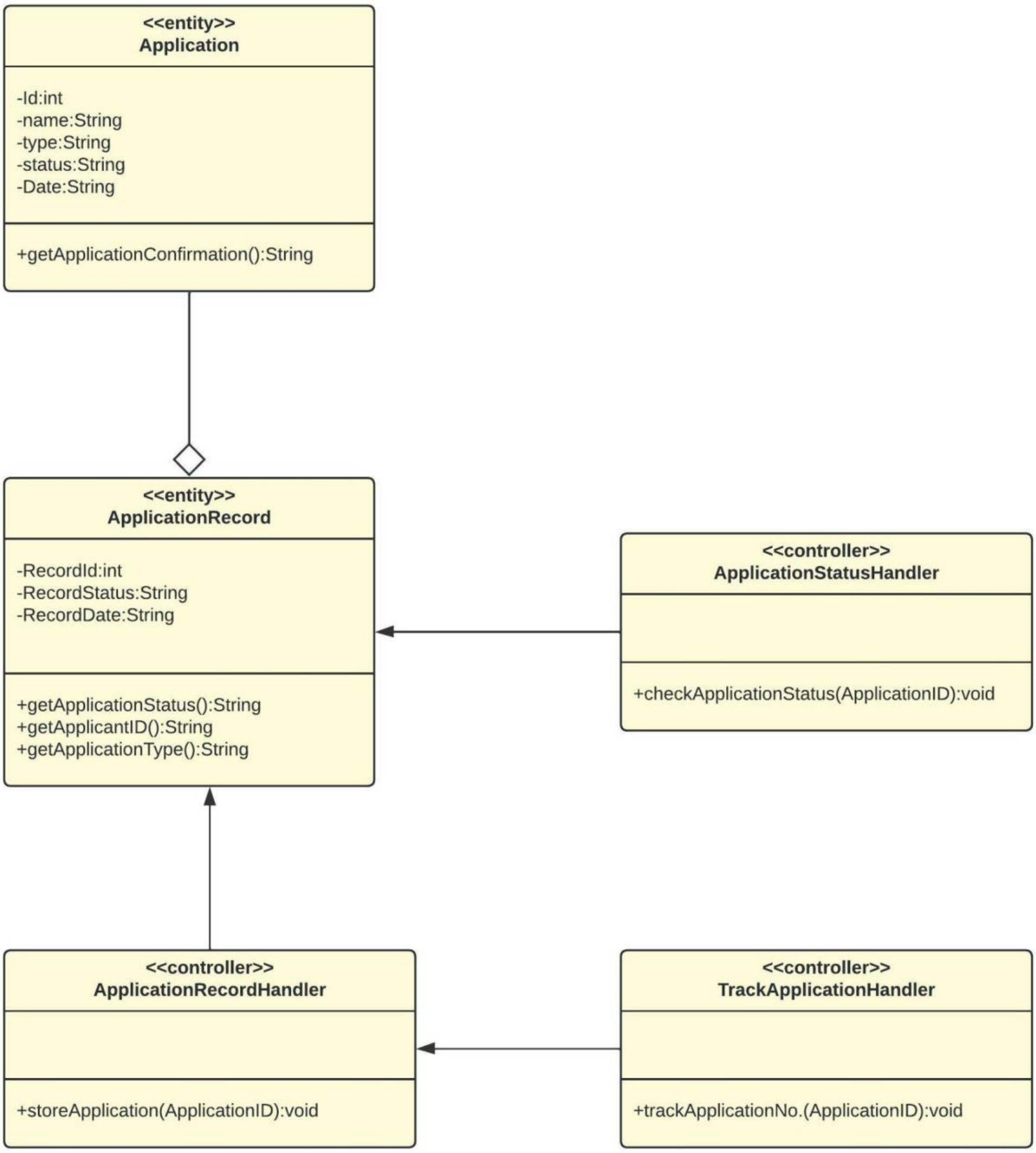


Figure: Class diagram for <File Monitoring>

Entity Name	Application
Method Name	getApplicationConfirmation
Input	-
Output	String

Algorithm	1. Start 2. Return confirmation 3. End

Entity Name	ApplicationRecord
Method Name	getApplicationStatus
Input	-
Output	String
Algorithm	1. Start 2. Return application status 3. End

Entity Name	ApplicationRecord
Method Name	getApplicationID
Input	-
Output	String
Algorithm	1. Start 2. Return ID 3. End

Entity Name	ApplicationRecord
Method Name	getApplicationType
Input	-
Output	String
Algorithm	1. Start 2. Return Application type 3. End

3.2.3.3.Sequence Diagrams

a) SD004: Sequence diagram for Check Application Status

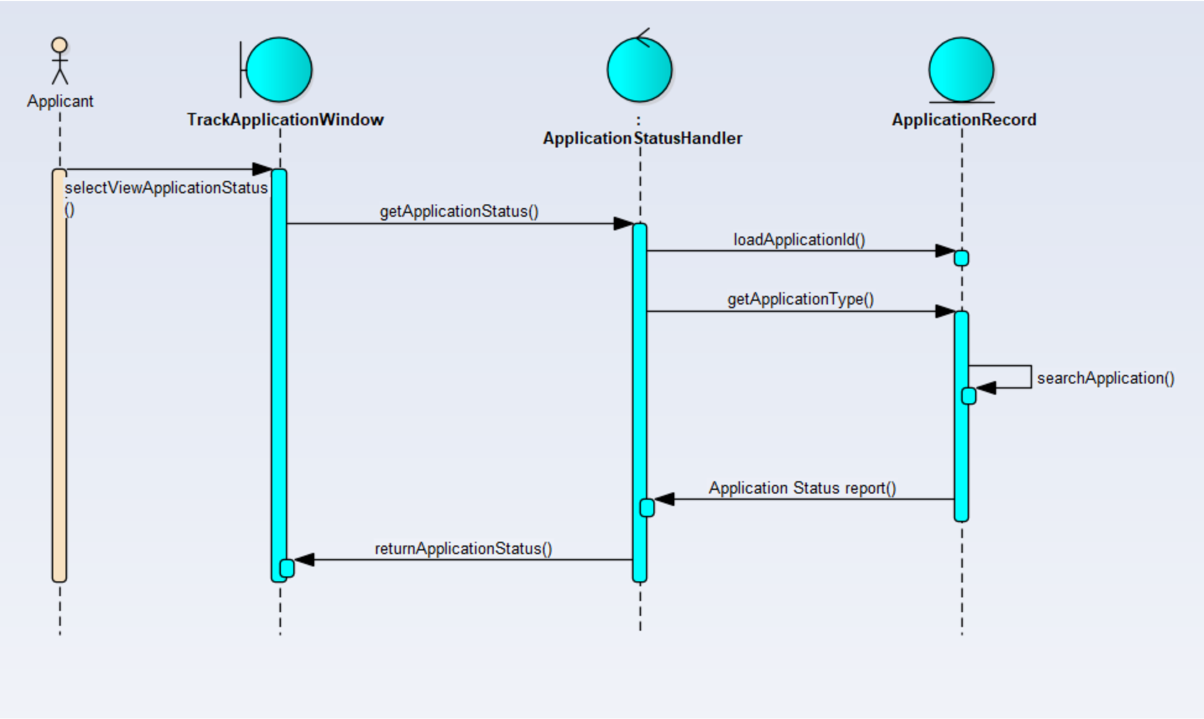


Figure: Sequence diagram for <Check application status>

b) SD005: Sequence Diagram for Track Application Statistics

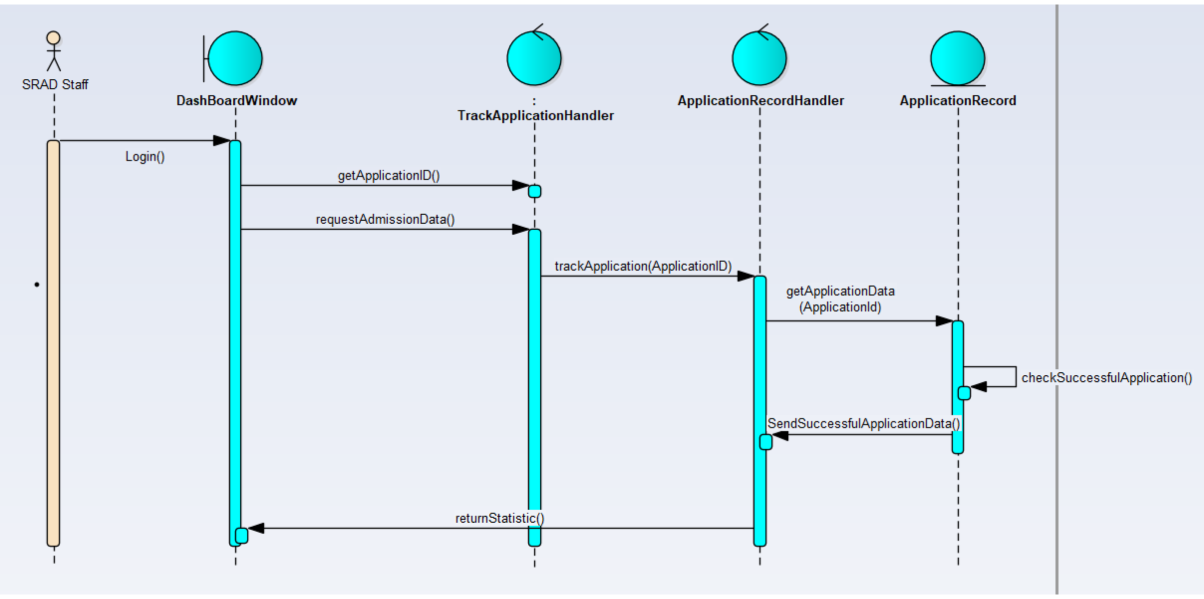

SRAD Staff
DashBoardWindow
: TrackApplicationHandler
ApplicationRecordHandler
ApplicationRecord
Login()
getApplicationID()
requestAdmissionData()
trackApplication(ApplicationID)
getApplicationData (ApplicationId)
checkSuccessfulApplication()
SendSuccessfulApplicationData()
returnStatistic()

Figure: Sequence diagram for <Track application status>

3.2.4.Module <Application Assessment>

3.2.4.1.P004: Package<Application assessment>

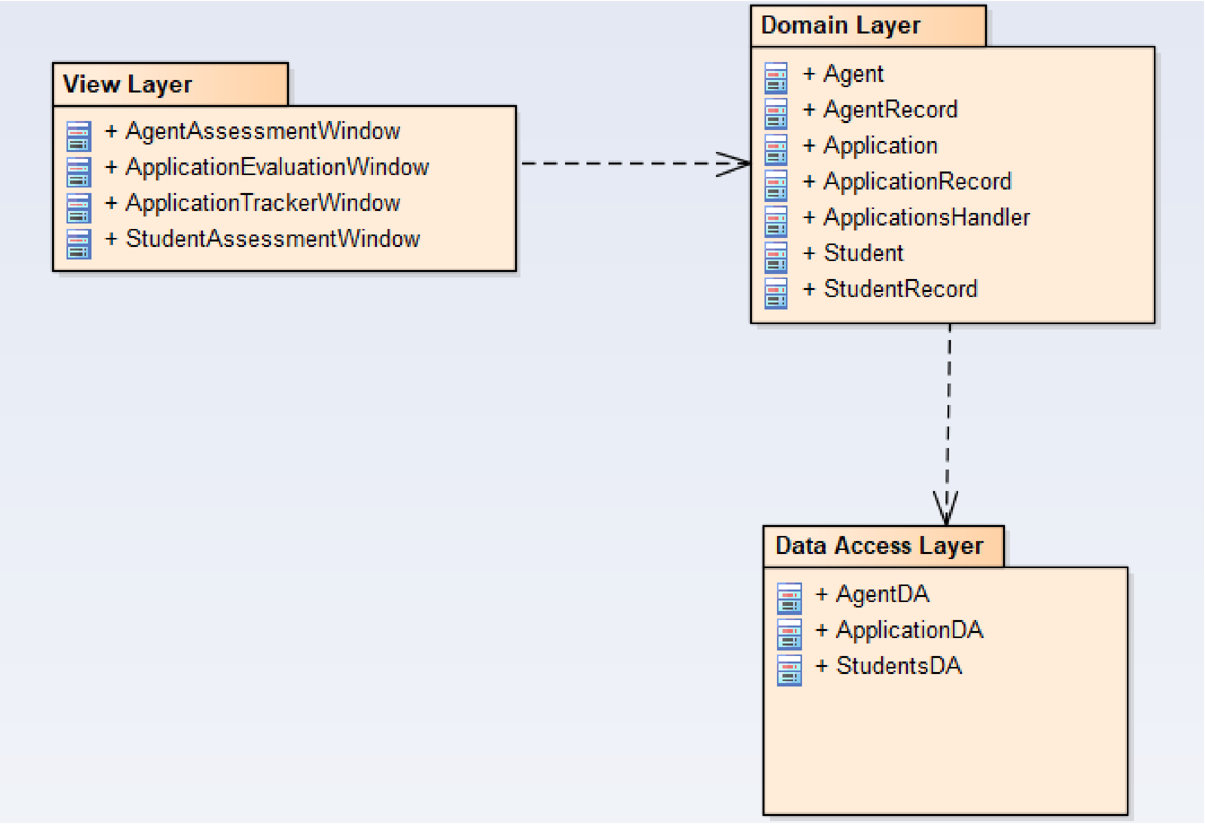
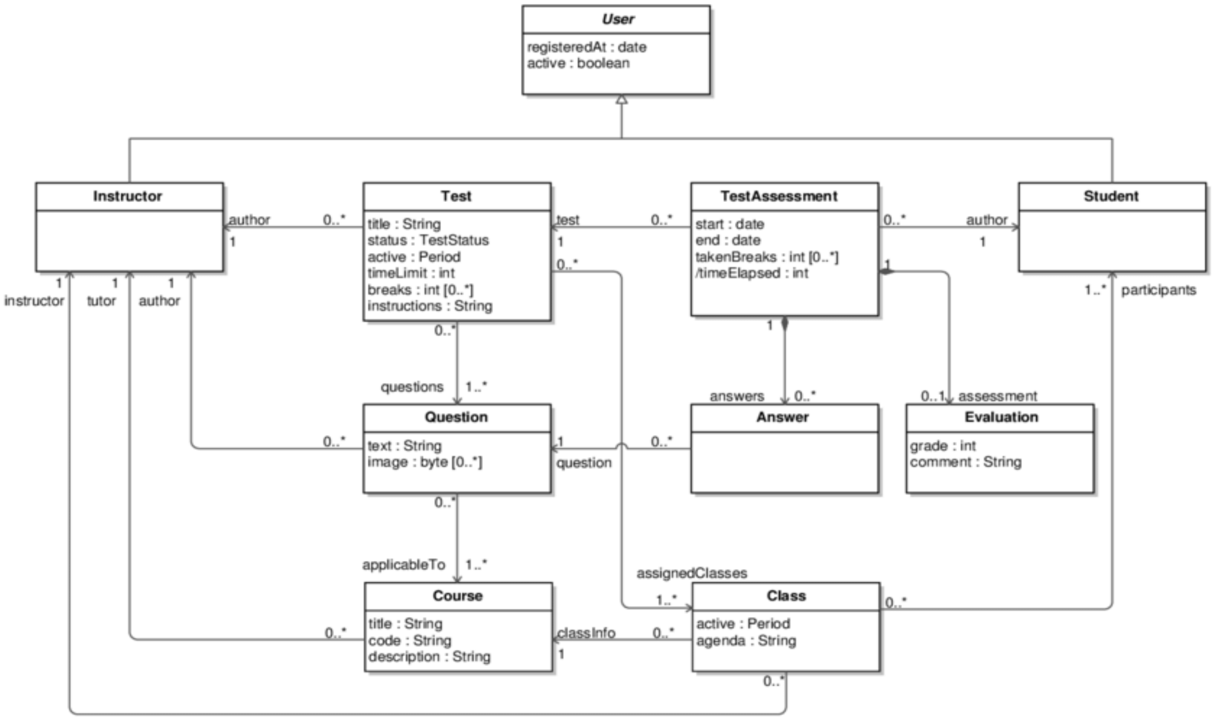


Figure: Package diagram for <Application Assessment>

3.2.4.2. Class Diagram



3.2.4.3. Sequence Diagram

a) SD006: Sequence Diagram for Check agent registration application

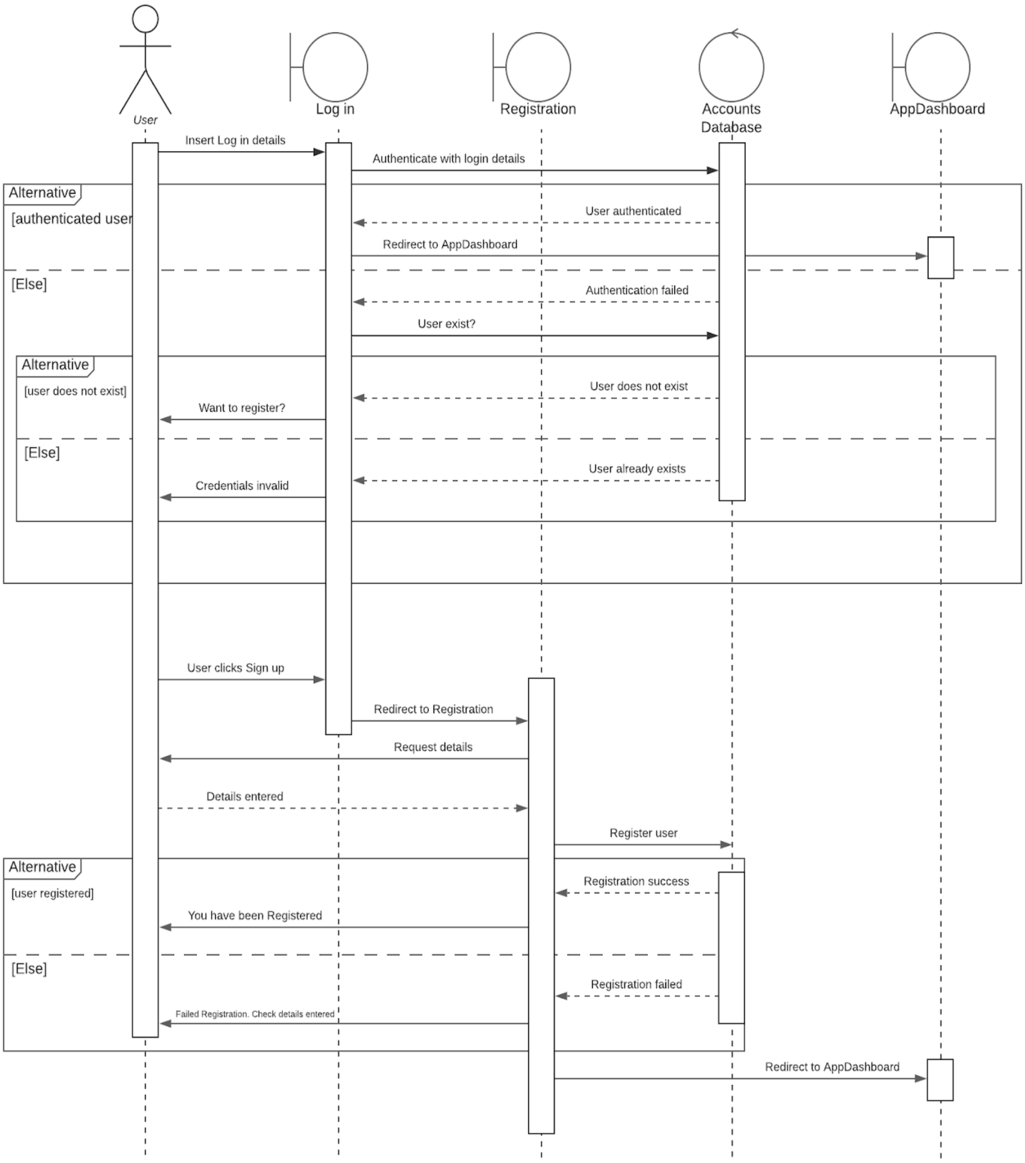


Figure: Sequence diagram for <Check agent registration application>

b) SD007: Sequence diagram for Evaluate student application

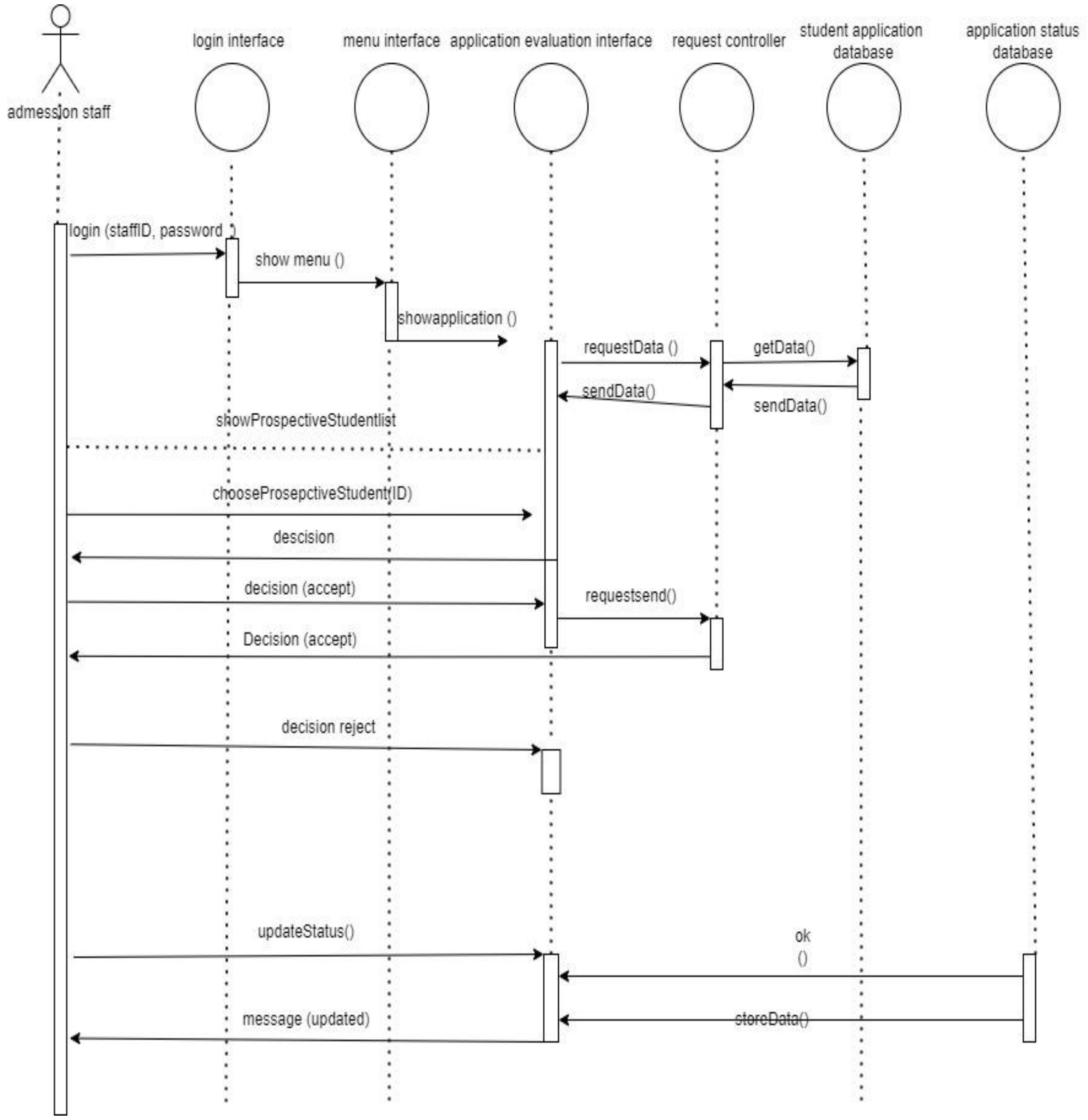


Figure: Sequence Diagram of <Evaluate student application>

***c)*SD008:** Sequence diagram for Evaluate Agent registration application

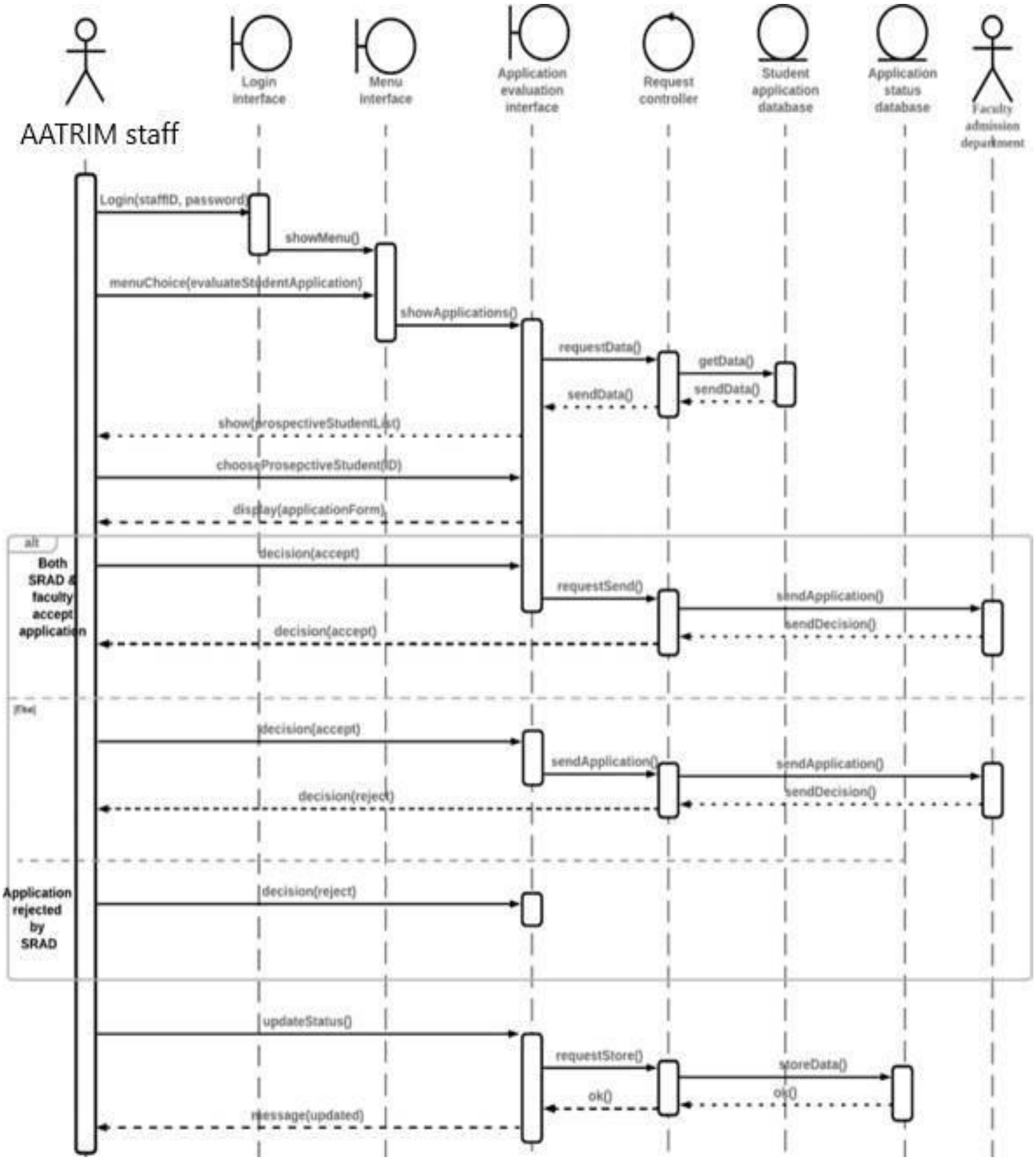


Figure: Sequence Diagram of < Evaluate Agent application>

d)SD009: Sequence diagram for Evaluate Direct student Application

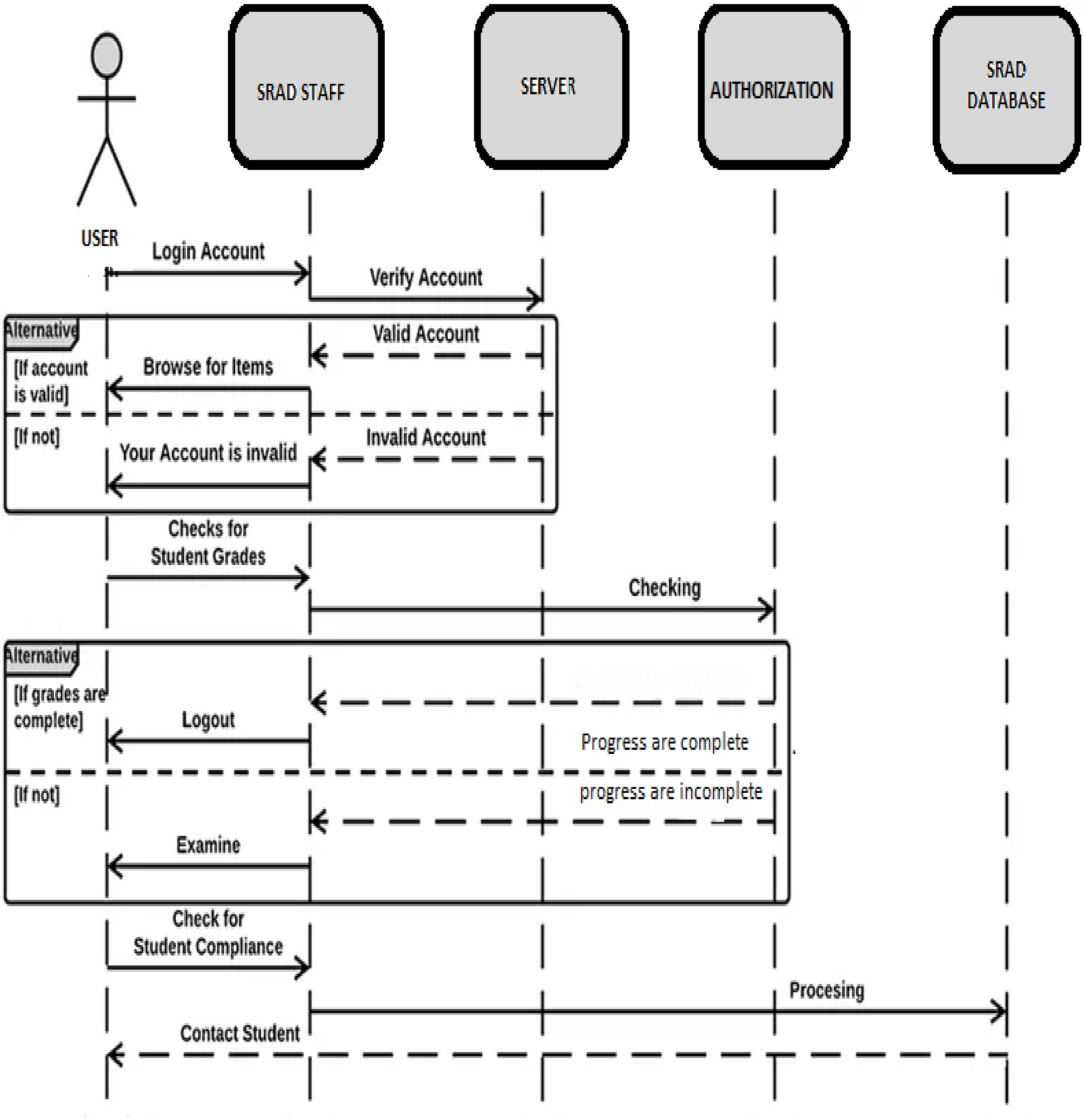


Figure: Sequence Diagram of <Evaluate Direct student Application >

3.2.5. Module <Application Management>

3.2.5.1. P005: Package <Application Management>

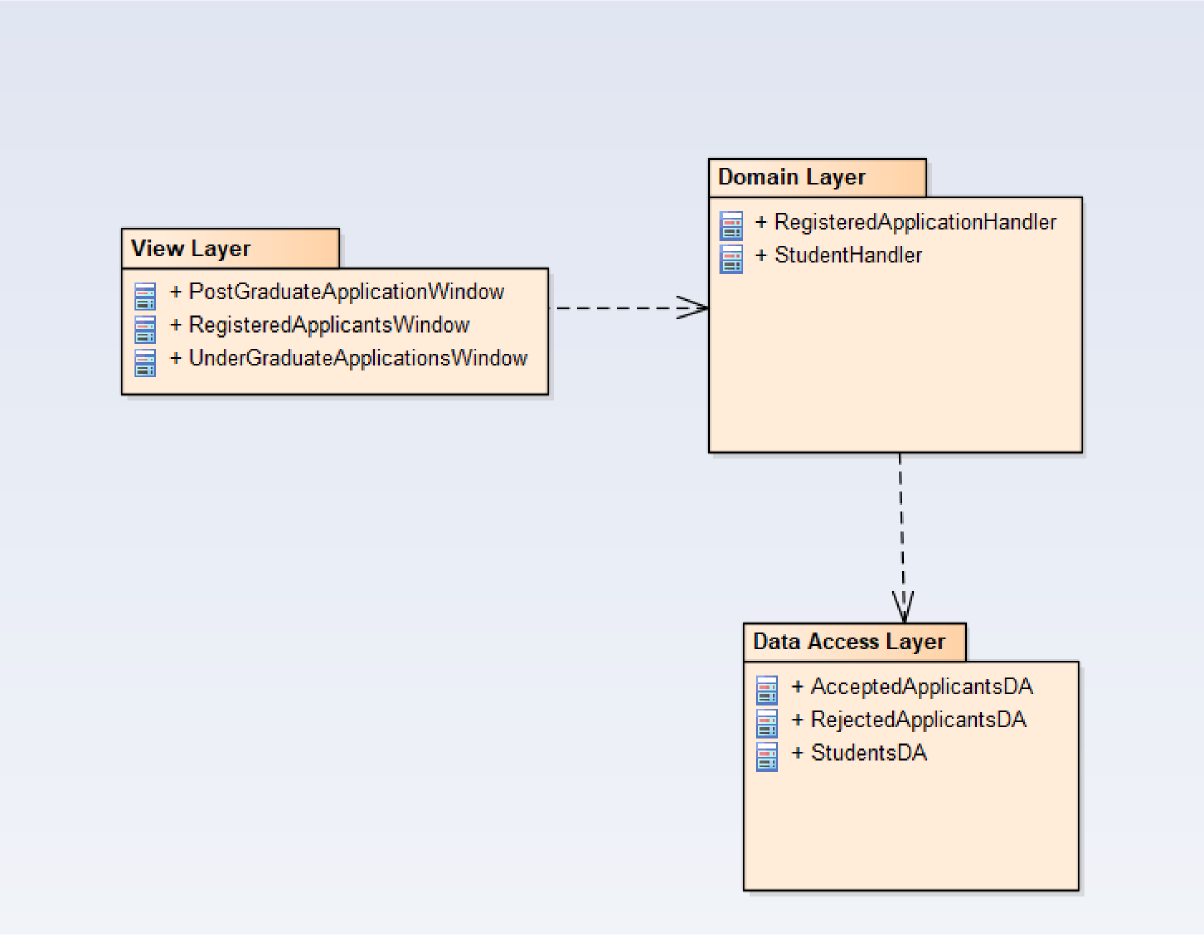
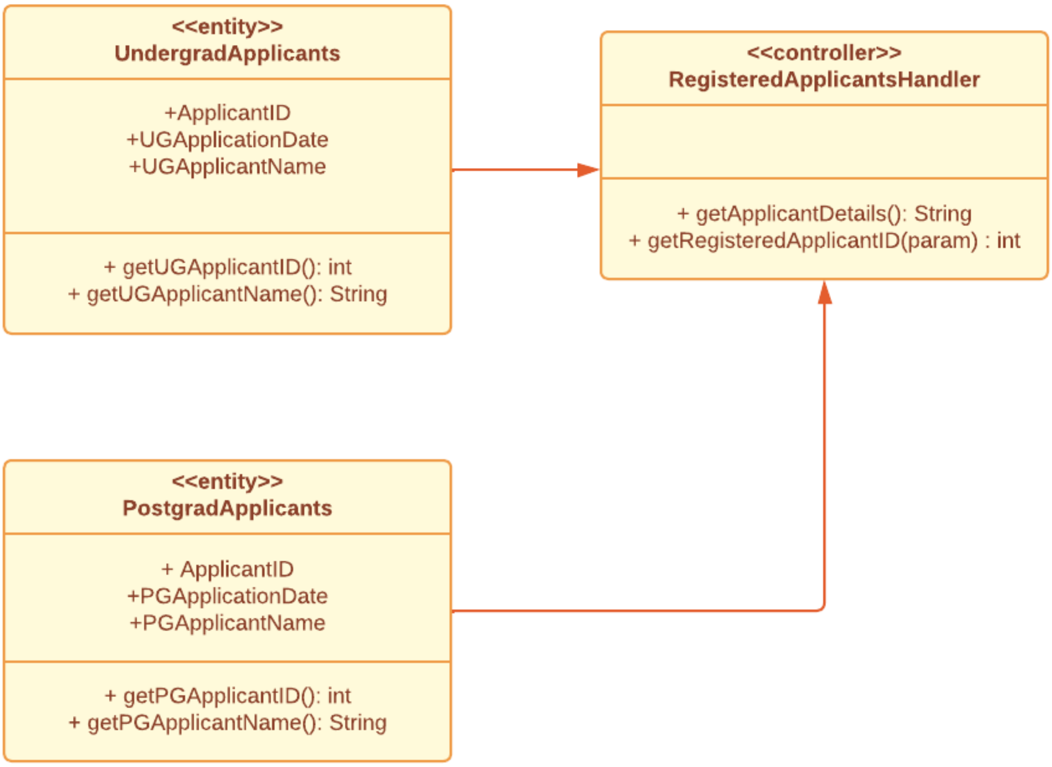

View Layer
+ PostGraduateApplicationWindow
+ RegisteredApplicantsWindow
+ UnderGraduateApplicationsWindow
Domain Layer
+ RegisteredApplicationHandler
+ StudentHandler
Data Access Layer
+ AcceptedApplicantsDA
+ RejectedApplicantsDA
+ StudentsDA

Figure: Package diagram for <Application Management>

3.2.5.2 Class diagram



<<entity>>
UndergradApplicants
+ApplicantID
+UGApplicationDate
+UGApplicantName
+ getUGApplicantID(): int
+ getUGApplicantName(): String
<<controller>>
RegisteredApplicantsHandler
+ getApplicantDetails(): String
+ getRegisteredApplicantID(param) : int
<<entity>>
PostgradApplicants
+ ApplicantID
+PGApplicationDate
+PGApplicantName
+ getPGApplicantID(): int
+ getPGApplicantName(): String

Figure: Class diagram for <Application Management>

Entity Name	UndergradApplicants
Method Name	getUGApplicantID
Input	-
Output	int

Algorithm	1. Start 2. Return Applicant ID 3. End

Entity Name	UndergradApplicants
Method Name	getUGApplicantName
Input	-
Output	String
Algorithm	1. Start 2. Return Applicant Name 3. End

Entity Name	PostgradApplicants
Method Name	getPGApplicantName
Input	-
Output	String
Algorithm	1. Start 2. Return Applicant Name 3. End

Entity Name	PostgradApplicants
Method Name	getPGApplicantID
Input	-
Output	int
Algorithm	1. Start 2. Return Applicant ID 3. End

3.2.5.3 Sequence diagrams

a) UC010: Sequence diagram Register Confirmed Applicants

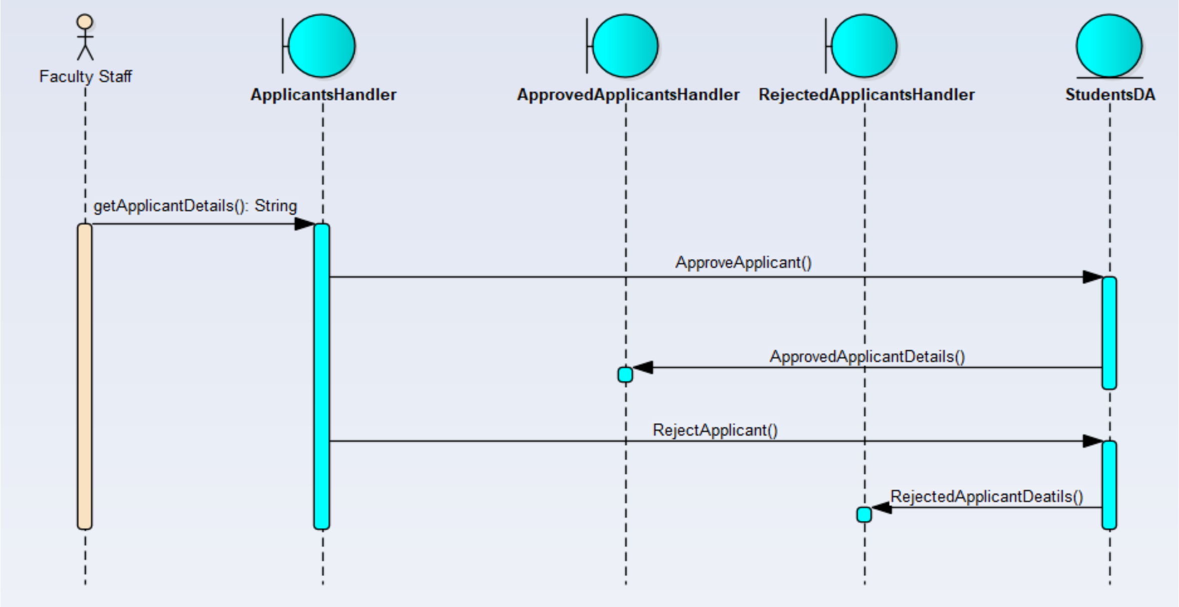

Faculty Staff
ApplicantsHandler
ApprovedApplicantsHandler
RejectedApplicantsHandler
StudentsDA
getApplicantDetails(): String
ApproveApplicant()
ApprovedApplicantDetails()
RejectApplicant()
RejectedApplicantDeatils()

Figure: Sequence diagram for < Register Confirmed Applicants>

b) UC011: Sequence diagram for Register Undergraduate Students

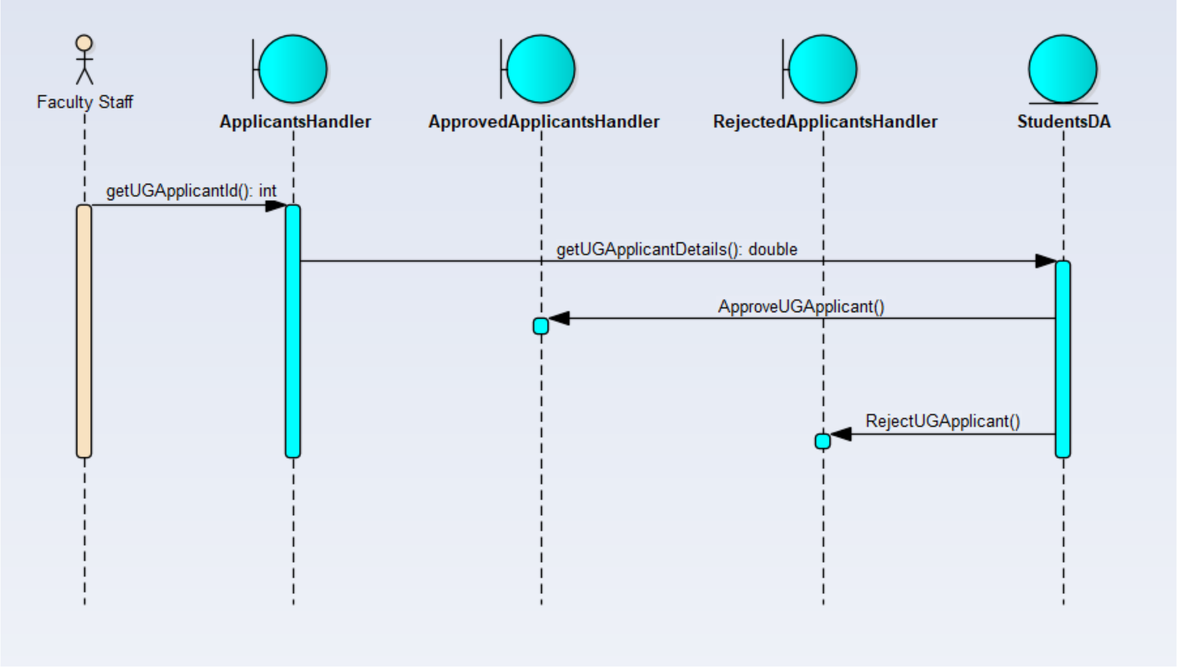


Figure: Sequence diagram for < Register Undergraduate Students >

c) UC012: Sequence diagram for Register Postgraduate Students

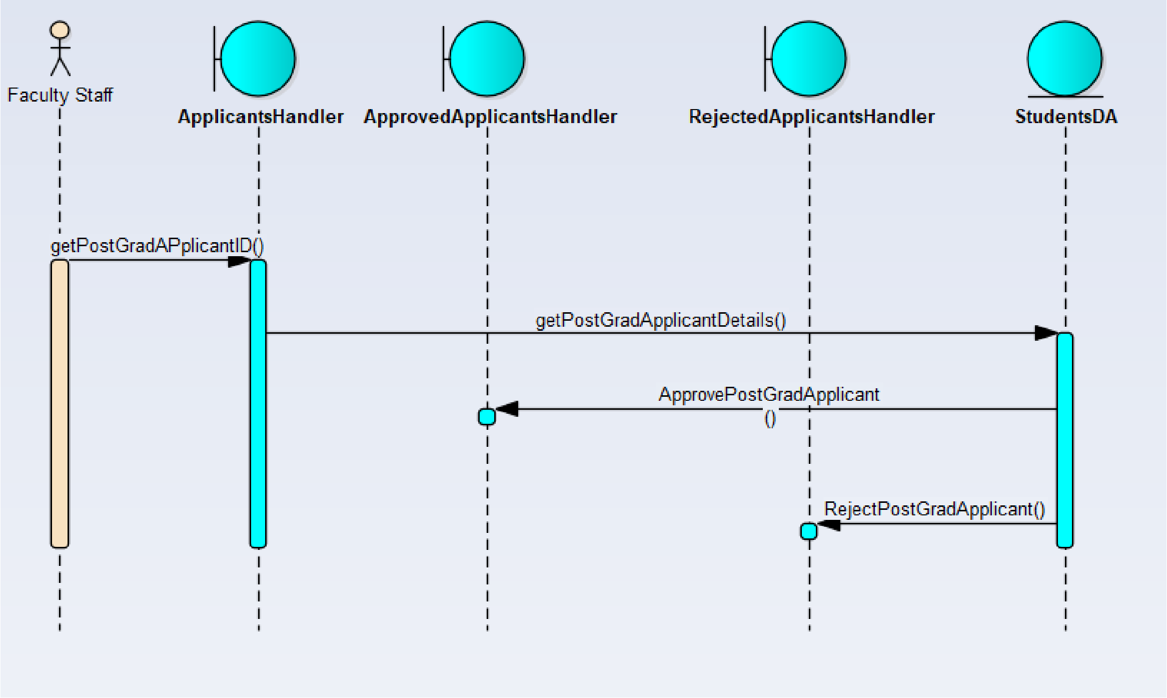

Faculty Staff
ApplicantsHandler
ApprovedApplicantsHandler
RejectedApplicantsHandler
StudentsDA
getPostGradAPplicantID()
getPostGradApplicantDetails()
ApprovePostGradApplicant
()
RejectPostGradApplicant()

Figure: Sequence diagram for < Register Postgraduate Students >

4. Data Design

4.1. Data Description

No.	Entity Name	Description
1	Agent	This entity represents the agent who needs to register to the system and who can assist the student in making an application.
2	Faculty Staff	This entity evaluates the documents provided by the student and approves the admission of students.
3	Undergrad Applicants	An entity that represents the students who apply for the undergraduate program.
4	Postgrad Applicants	An entity that represents the students who apply for the postgraduate program.
5	Payment	An entity that represents the amount to be paid by the applicant through the system for processing the application.
6	Payment Record	This entity stores the data of the details of the payments that have been made.
7	Application	This entity is used to make applications and represents the application of the students and agents done by the system.
8	Application Record	An entity that stores the details of the students and

		agents that have made applications through the system.

4.2. Data Dictionary

4.2.1 Entity <Agent>

Attribute Name	Type	Description
agentID	INTEGER	Primary key created for the agent after application
Name	VARCHAR(50)	Name of the agent
telephone	INTEGER	Telephone no. of the agent
email	VARCHAR(50)	E-mail address of the agent
yearlyKPI	DOUBLE	Yearly KPI data of the agent

4.2.2 Entity <Student>

Attribute Name	Type	Description
Email	VARCHAR(50)	E-mail of the student
Name	VARCHAR(50)	Name of the applicant
Phonenum	INTEGER	Phone no. of the student
studentID	INTEGER	Primary key which is assigned to a student.

4.2.3 Entity <Application >

Attribute Name	Type	Description
ApplicantAddress	VARCHAR(50)	Address of the applicant
ApplicantEmail	VARCHAR(50)	E-mail of the applicant

ApplicantName	VARCHAR(50)	Name of the applicant
isDraft	BOOL	Drafted information
submitDate	Date	Date of submission
type	VARCHAR(50)	Type of the application
applicationID	INTEGER	Primary key for the application
agentID	INTEGER	Foreign key for the agent
studentID	INTEGER	Foreign key for the student
staffID	INTEGER	Foreign key for the staff
paymentID	INTEGER	Payment ID required for transaction

4.2.4 Entity <SRAD Staff >

Attribute Name	Type	Description
email	VARCHAR(50)	E-mail of the staff
ID	INTEGER	Primary key for the staff
Name	VARCHAR(50)	Name of the staff
tellNo	VARCHAR(50)	Telephone no. of the staff

4.2.5 Entity <Faculty Admission department staff>

Attribute Name	Type	Description
e-mail	VARCHAR(50)	Email of the staff
id	INTEGER	Foreign key for the id
Name	VARCHAR(50)	Name of the staff
tellNo	VARCHAR(50)	Telephone no. of the staff
staffID	INTEGER	Primary key for the staff

5. User Interface Design

5.1. Overview of User Interface

The user will need to register as an undergraduate or postgraduate student first. The user will have the option of registering as a student or as an agent. The registration page will consist of the option to choose to register as a student or as an agent. There will be an admin login page as well where the admin will be redirected to the admin dashboard. Meanwhile, the users will be redirected to the user dashboard. From the user's point of view as a student, the student himself will be able to make the application or the agent can make the application on behalf of the student. The application form will be displayed on the screen for the student to fill up. The agent can fill up the form on behalf of the student. After submitting the form, it will be evaluated by the admin. From the user’s point of view as an admin, he’ll be able to view the applications made, can evaluate the applications, and reject the application if there are any problems. The SRAD staff will also be able to track the number of applications made by the students/agents. There will be an admission tracking page where the students or the agents will be able to check the admission status. They can also make changes needed to the application form.

5.2. Screen Images

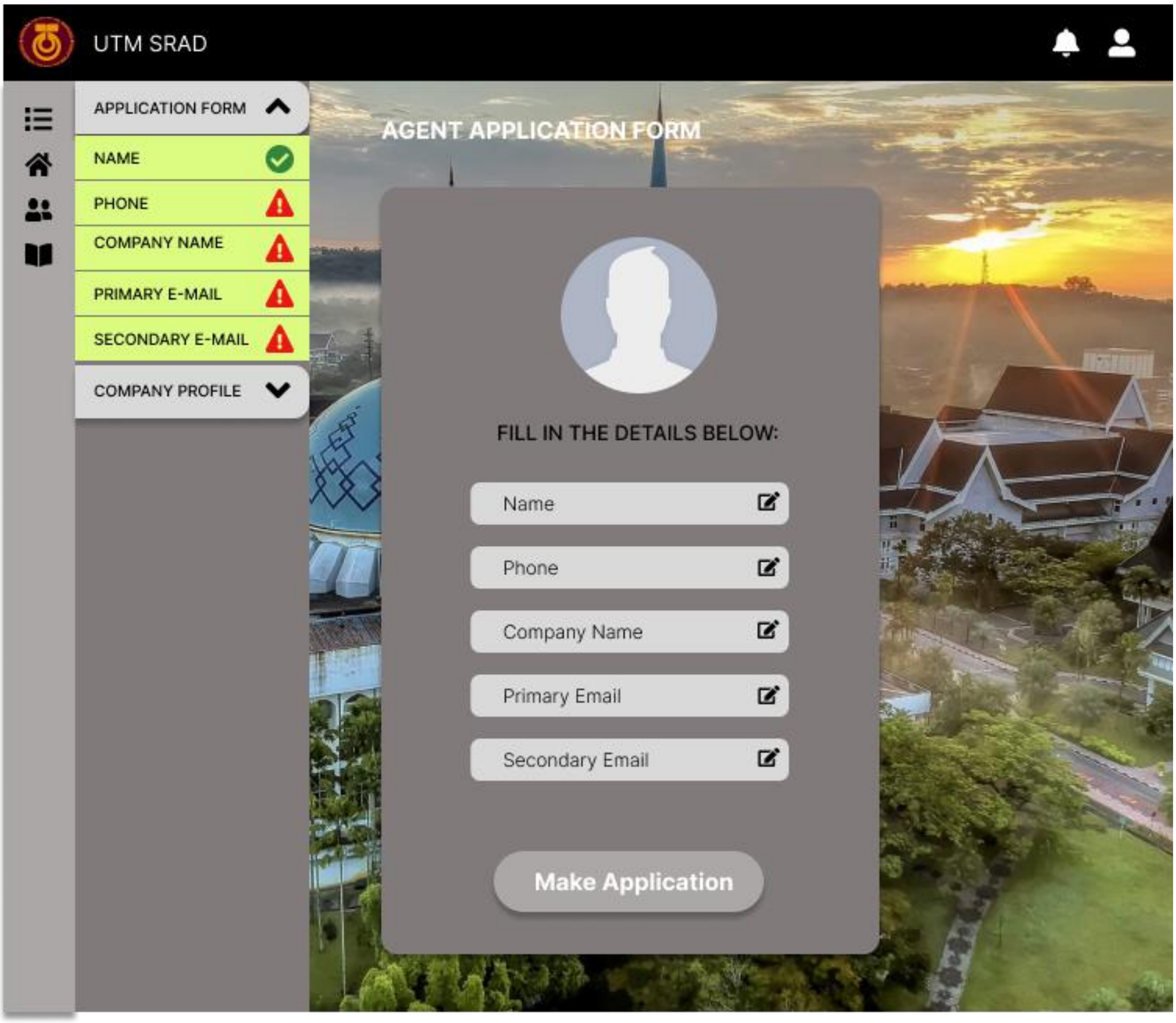


Image: Interface for view agent application form

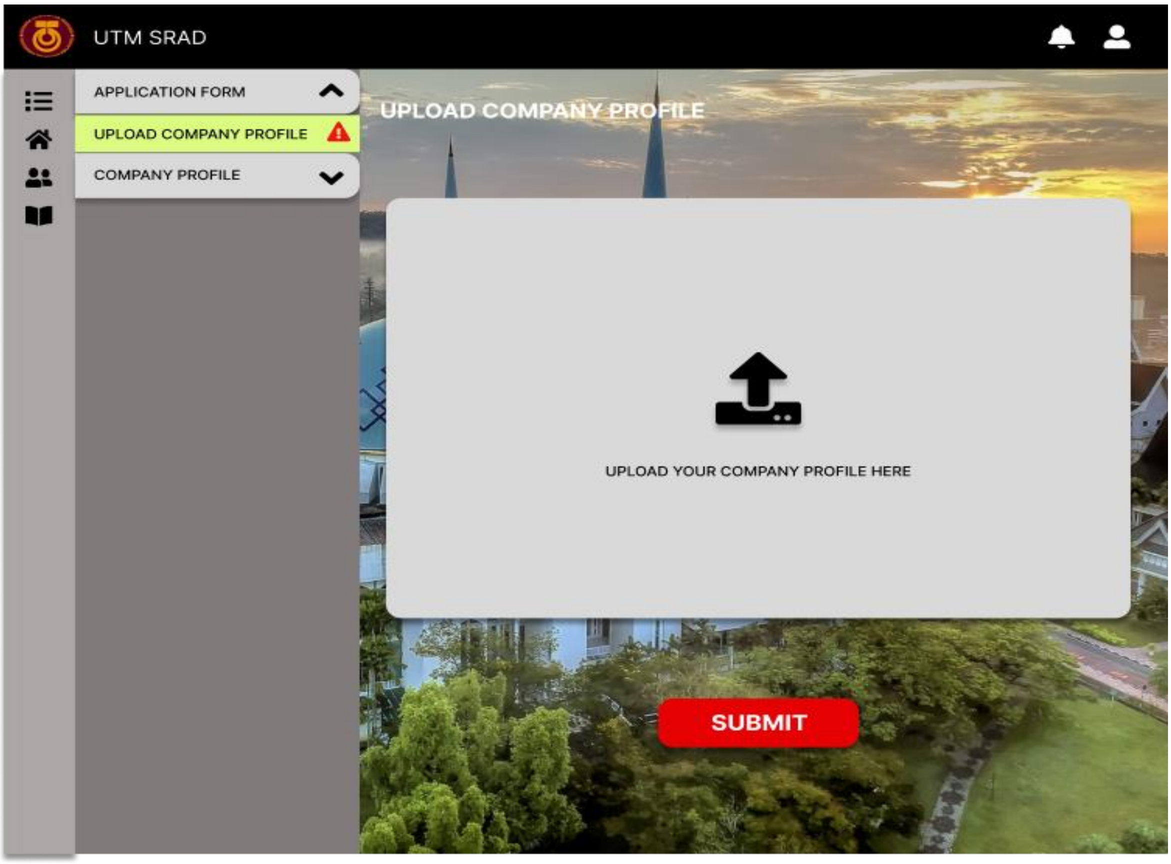
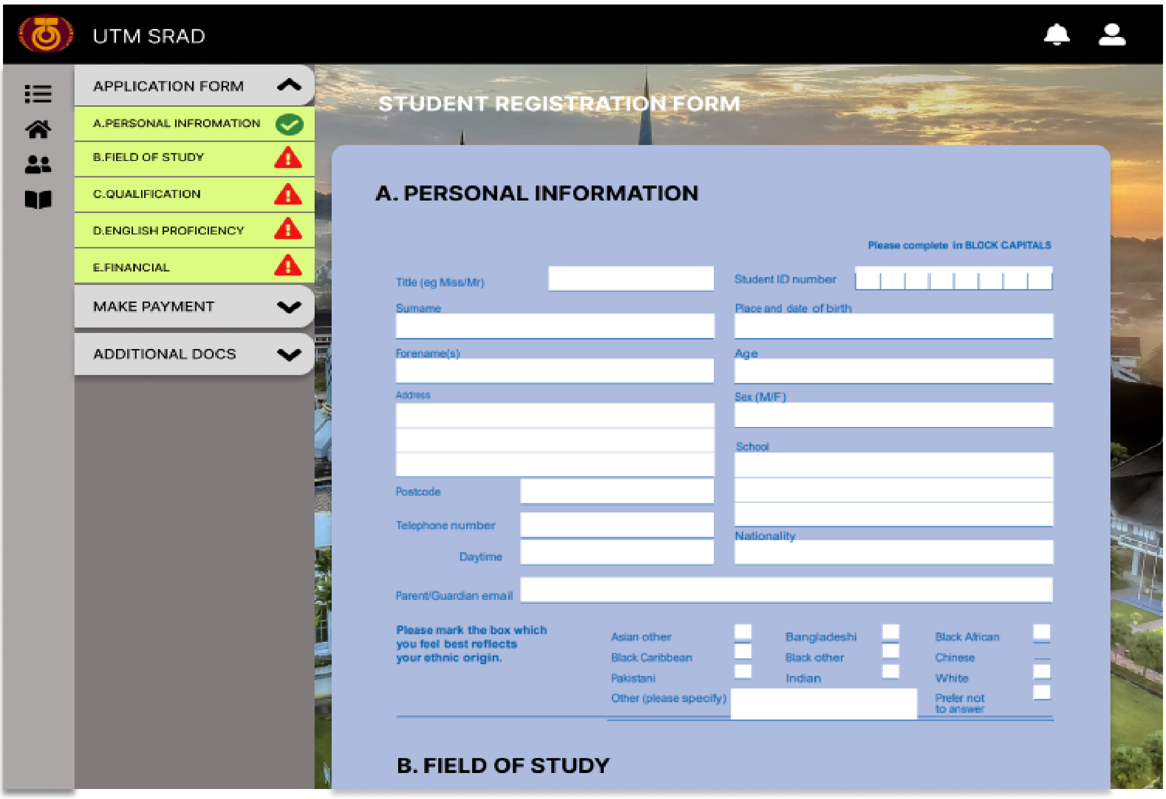


Image: Interface for Update agent's company profile



A. PERSONAL INFORMATION

Please complete in BLOCK CAPITALS

Title (eg Miss/Mr)

Student ID number

Surname

Place and date of birth

Forename(s)

Age

Address

Sex (M/F)

School

Postcode

Telephone number

Daytime

Nationality

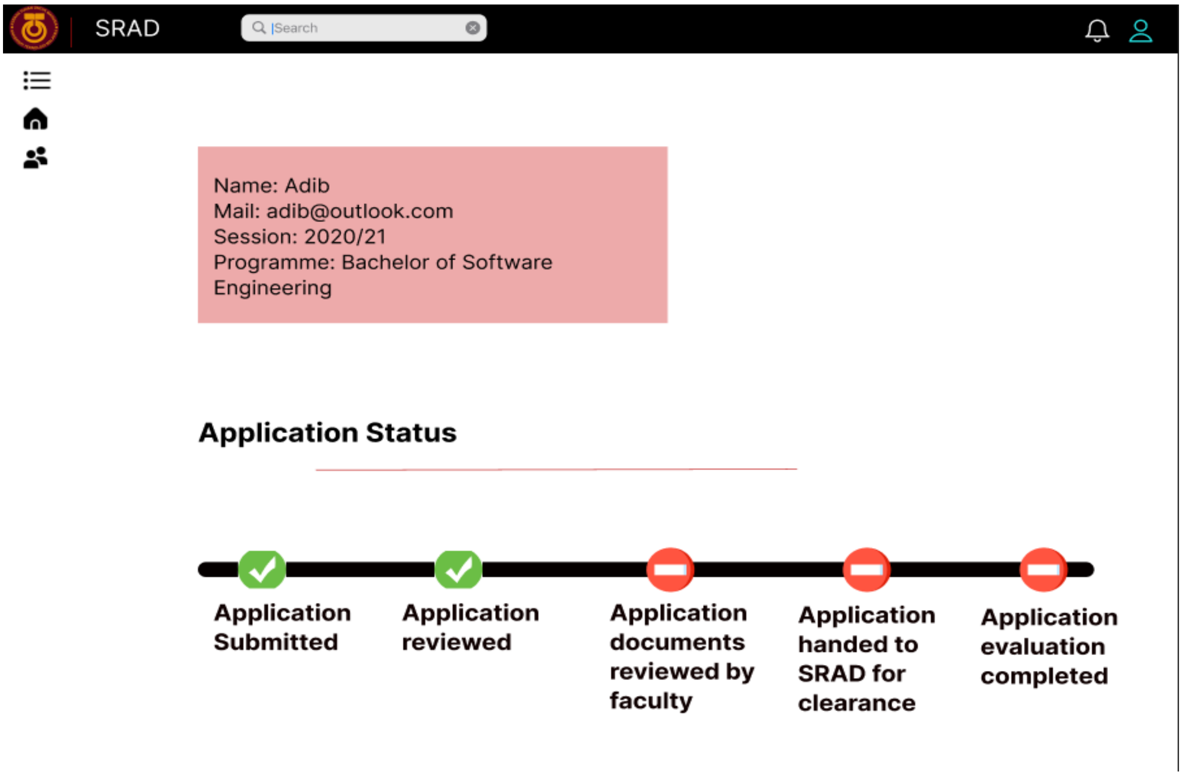
Parent/Guardian email

Please mark the box which you feel best reflects your ethnic origin.

Asian other	□	Bangladeshi	□	Black African	□
Black Caribbean	□	Black other	□	Chinese	
Pakistani	□	Indian	□	White	□
Other (please specify)				Prefer not to answer	□

B. FIELD OF STUDY

Image: Interface for student registration form

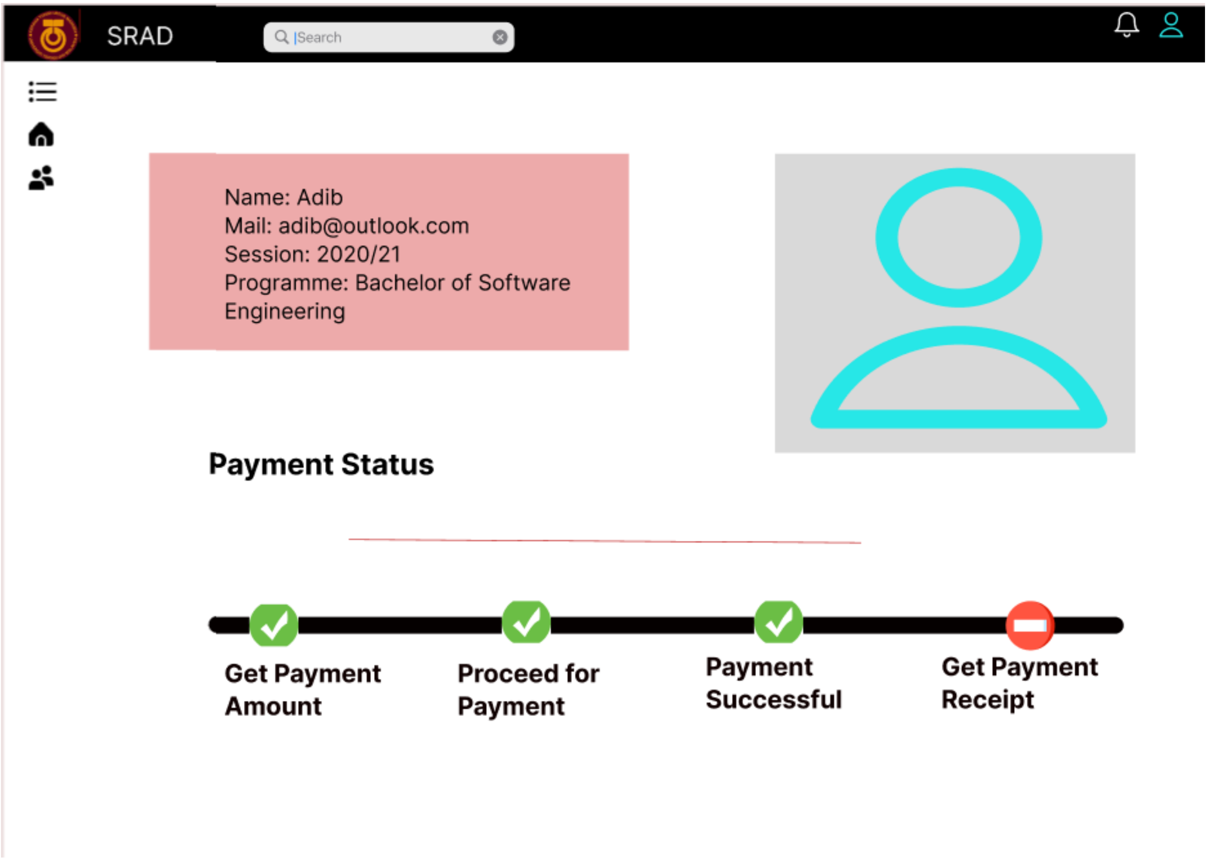


Name: Adib
Mail: adib@outlook.com
Session: 2020/21
Programme: Bachelor of Software Engineering

Application Status

✓ Application Submitted
✓ Application reviewed
⛔ Application documents reviewed by faculty
⛔ Application handed to SRAD for clearance
⛔ Application evaluation completed

Image: Interface for Check Payment Status

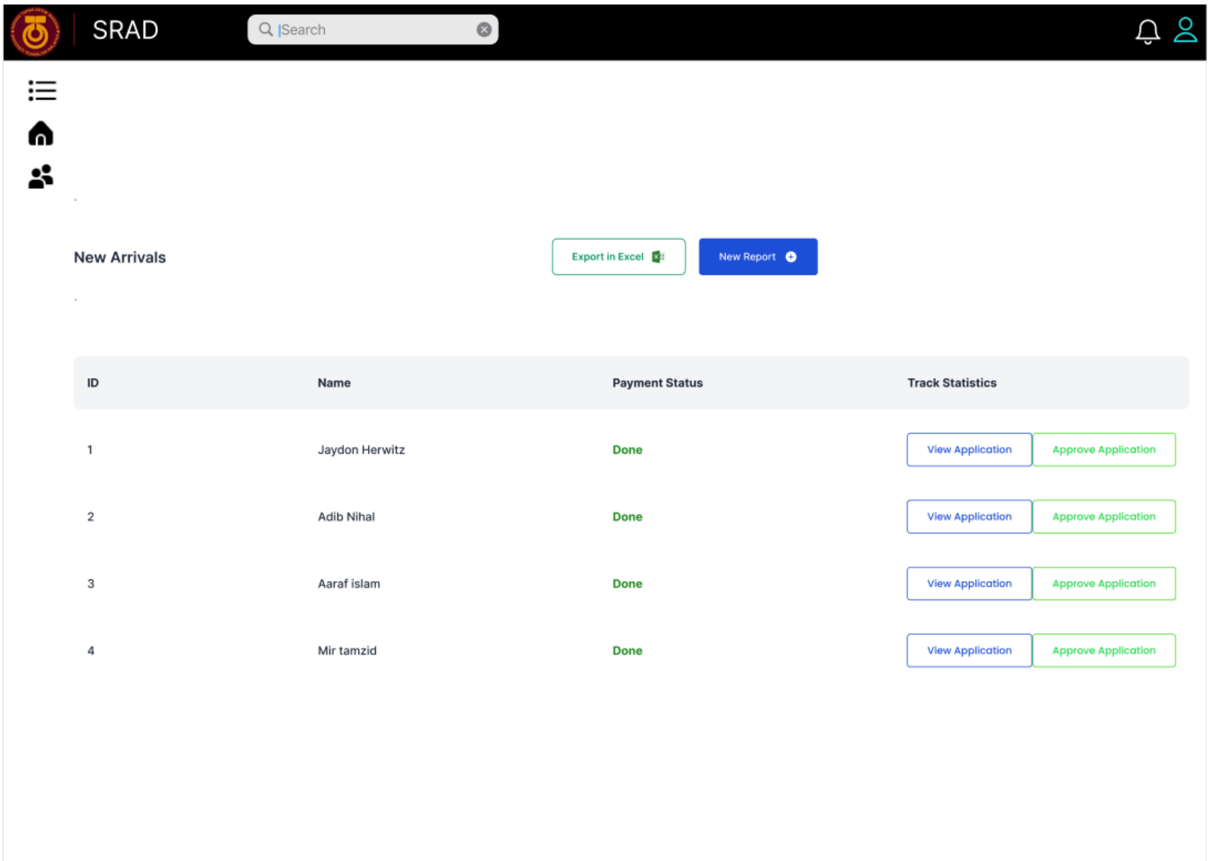


Name: Adib
Mail: adib@outlook.com
Session: 2020/21
Programme: Bachelor of Software Engineering

Payment Status

✓ Get Payment Amount
✓ Proceed for Payment
✓ Payment Successful
⛔ Get Payment Receipt

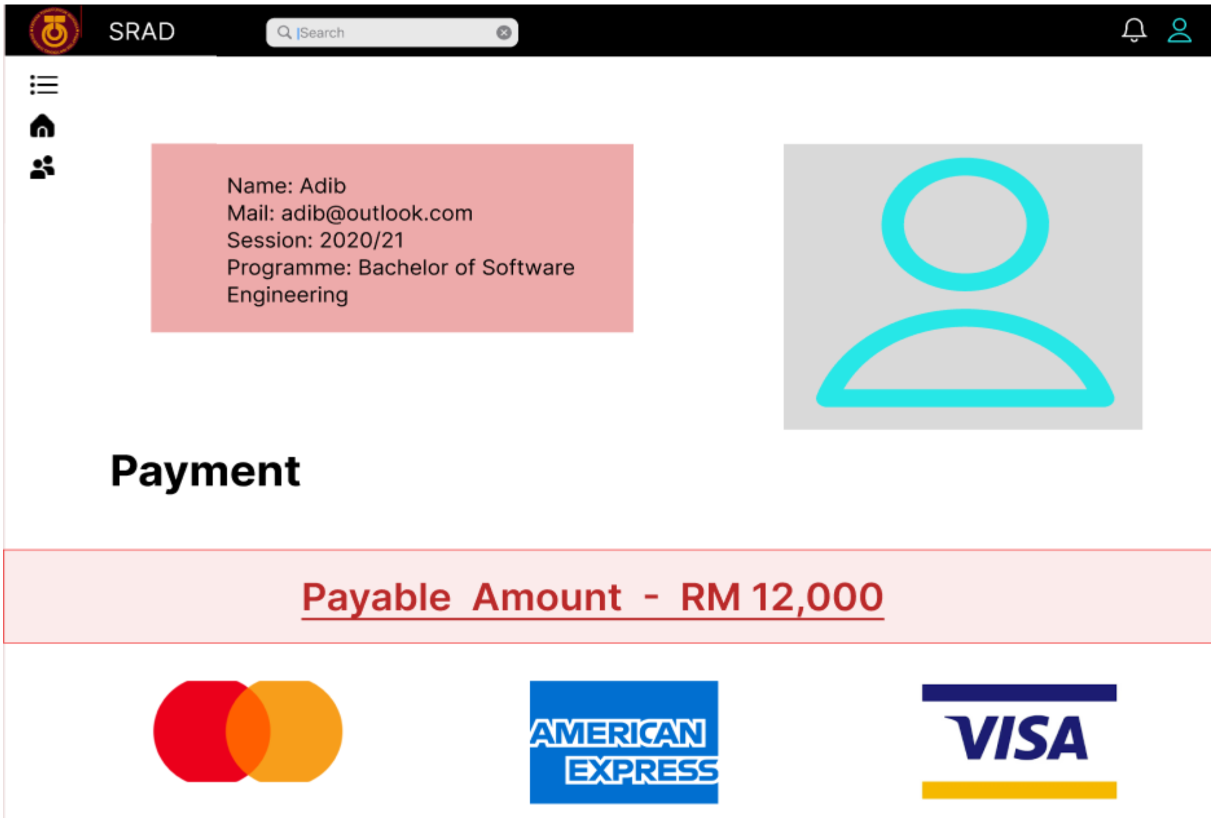
Image: Interface for Check Payment Status



New Arrivals

ID	Name	Payment Status	Track Statistics
1	Jaydon Herwitz	Done	View Application Approve Application
2	Adib Nihal	Done	View Application Approve Application
3	Aaraf islam	Done	View Application Approve Application
4	Mir tamzid	Done	View Application Approve Application

Image: Interface for Track Admission Statistics



Name: Adib
Mail: adib@outlook.com
Session: 2020/21
Programme: Bachelor of Software Engineering

Payment

Payable Amount - RM 12,000

Image: Interface for Make Payment

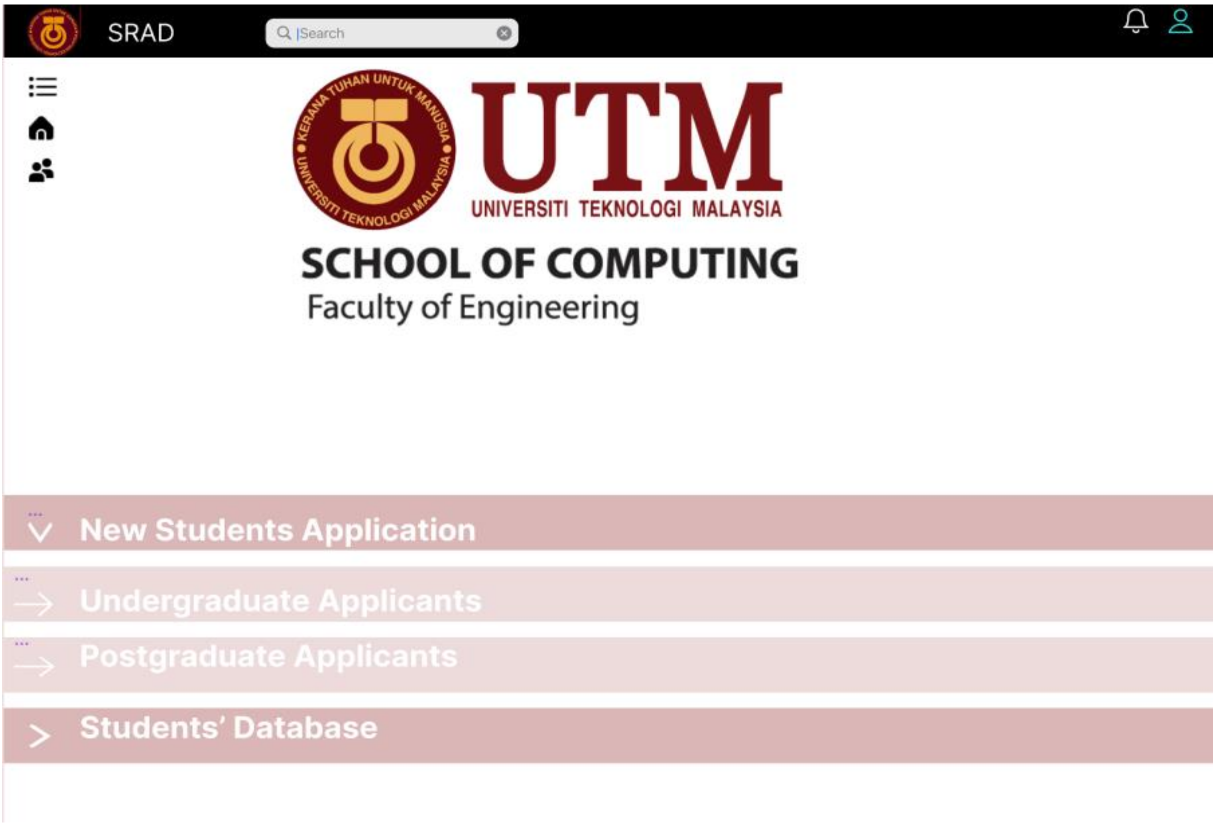
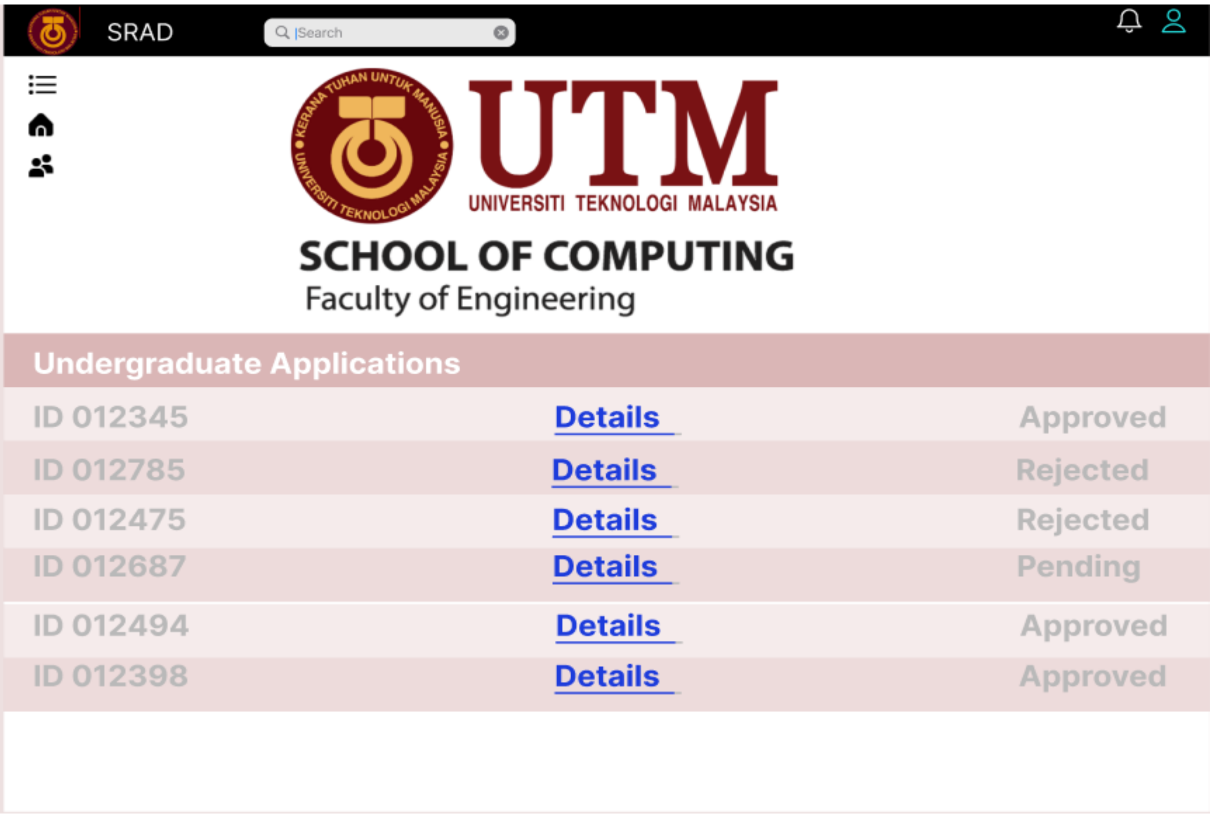


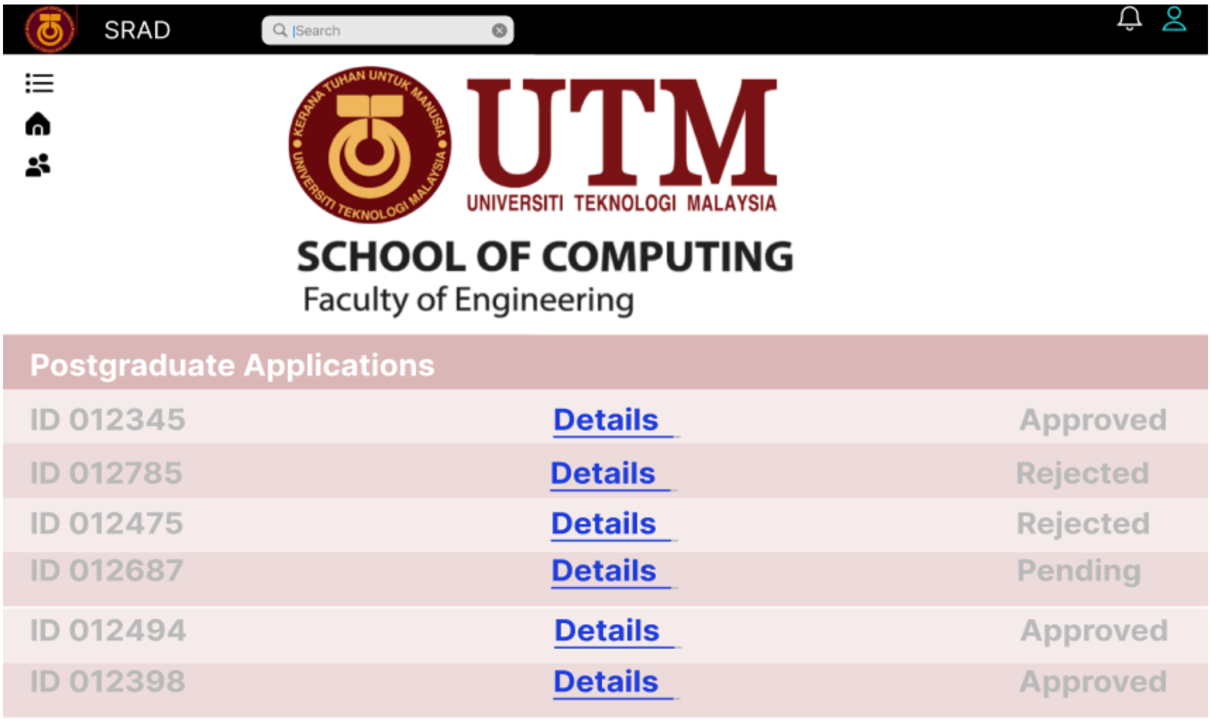
Image: Interface for Approve/Reject New Applicants



Undergraduate Applications

ID 012345	Details	Approved
ID 012785	Details	Rejected
ID 012475	Details	Rejected
ID 012687	Details	Pending
ID 012494	Details	Approved
ID 012398	Details	Approved

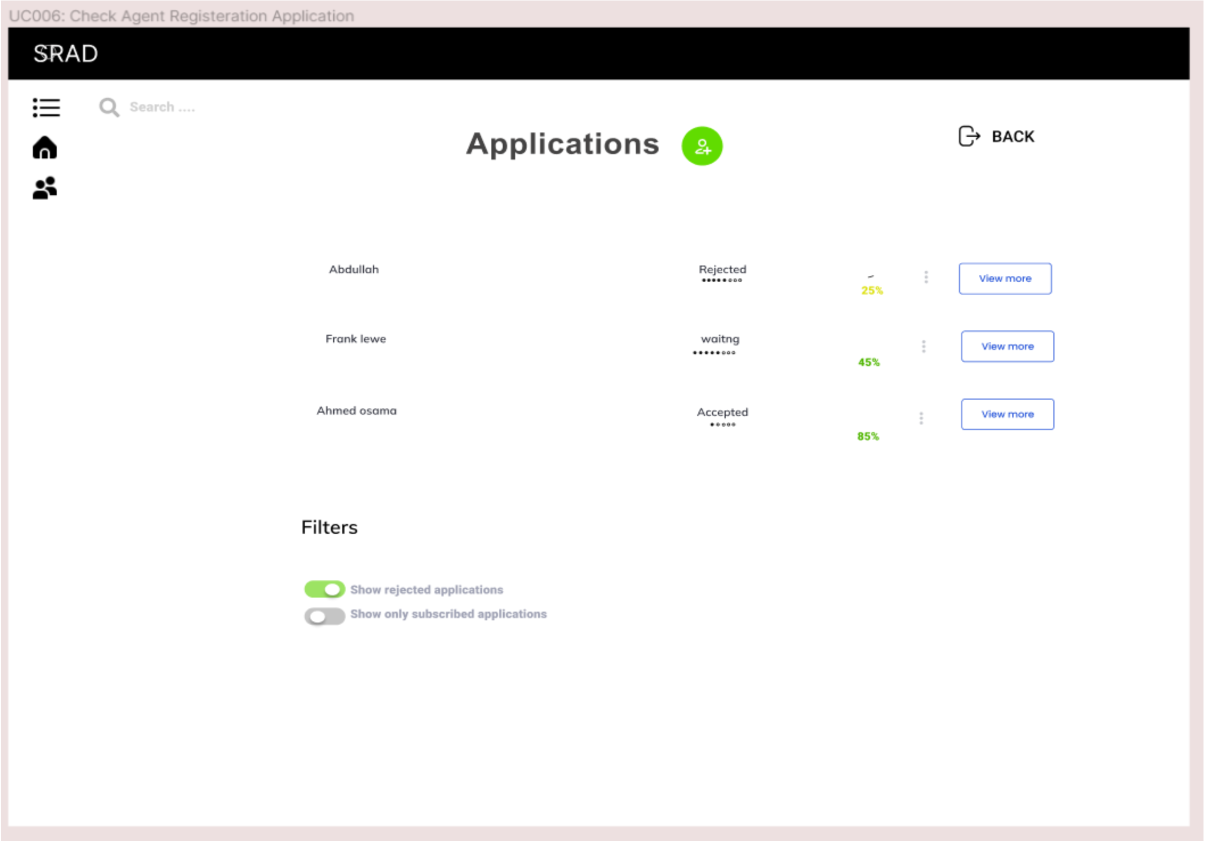
Image: Interface for Approve/Reject Undergraduate Applicants



Postgraduate Applications

ID 012345	Details	Approved
ID 012785	Details	Rejected
ID 012475	Details	Rejected
ID 012687	Details	Pending
ID 012494	Details	Approved
ID 012398	Details	Approved

Image: Interface for Approve/Reject Postgraduate Applicants



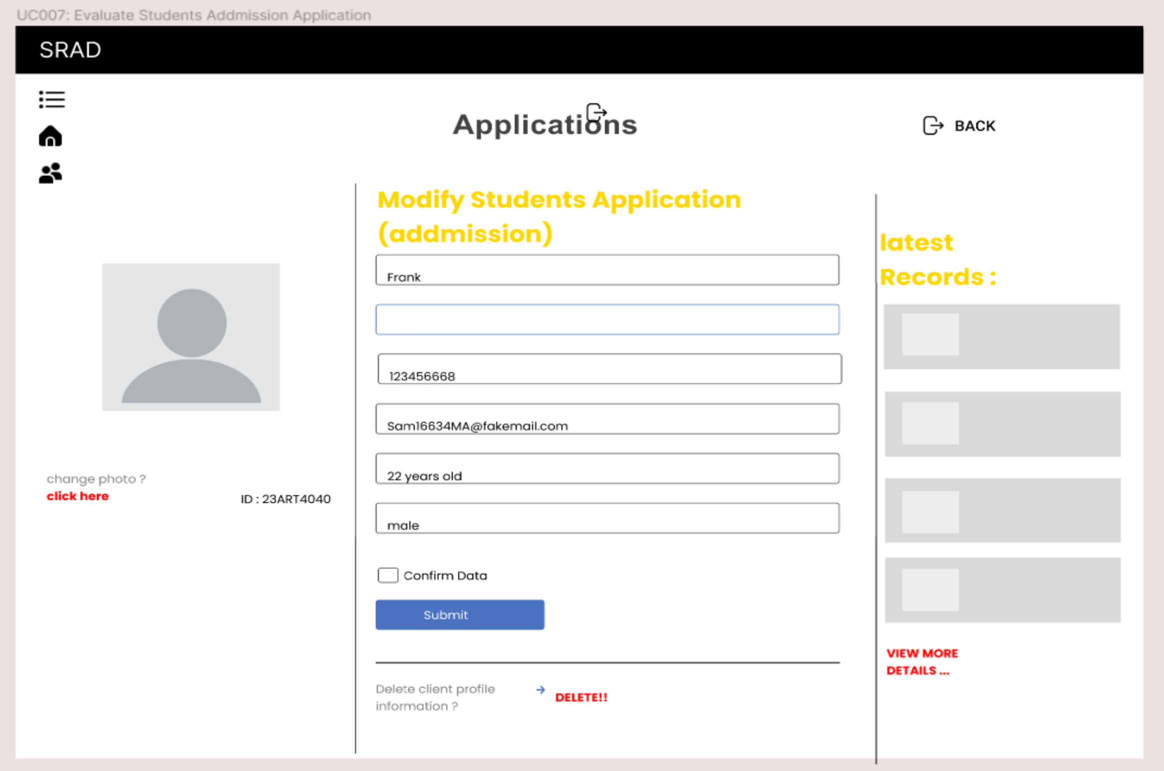
Applications

Abdullah	Rejected	25%	View more
Frank lewe	waitng	45%	View more
Ahmed osama	Accepted	85%	View more

Filters

- Show rejected applications
- Show only subscribed applications

Image: Interface for Check Agent Registration application



UC007: Evaluate Students Addmission Application

SRAD

Applications

Modify Students Application (addmission)

Frank

123456668

Sam16634MA@fakemail.com

22 years old

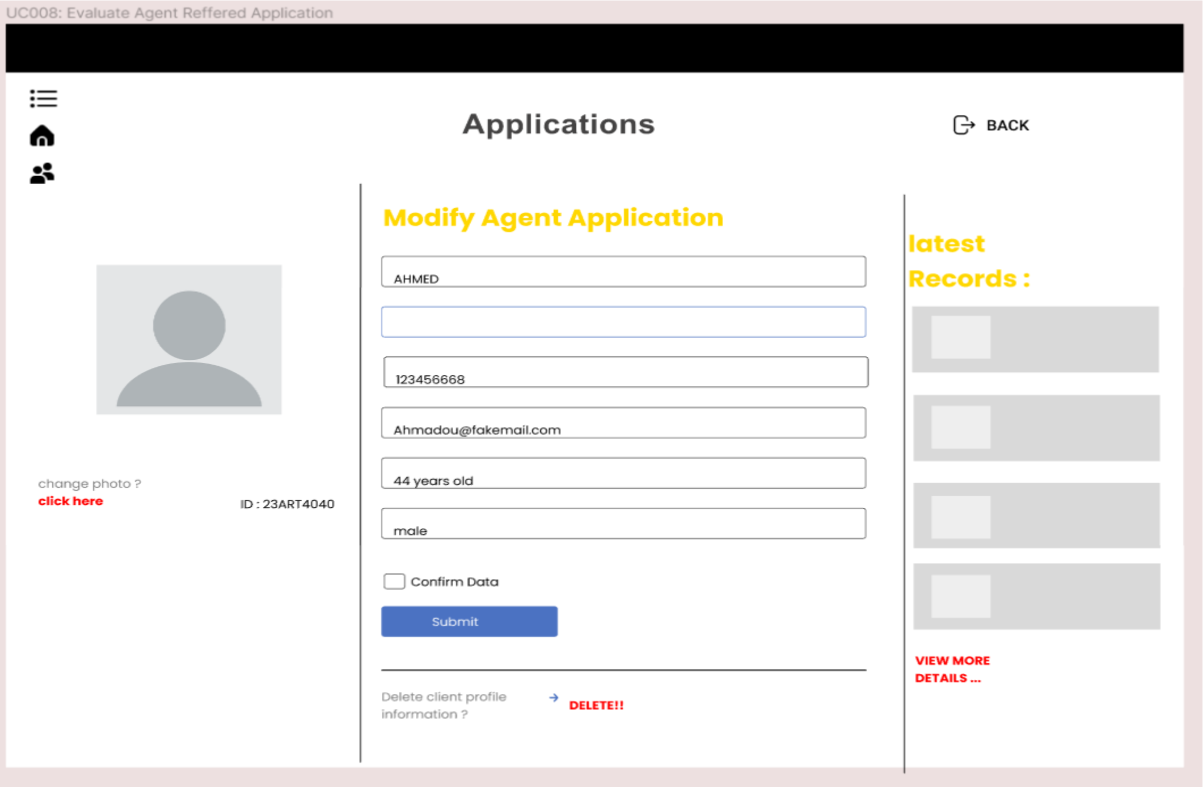
male

□ Confirm Data

ID : 23ART4040

latest Records :

Image: Interface for Evaluation students Admission Application



UC008: Evaluate Agent Reffered Application

Applications

Modify Agent Application

AHMED

123456668

Ahmadou@fakemail.com

44 years old

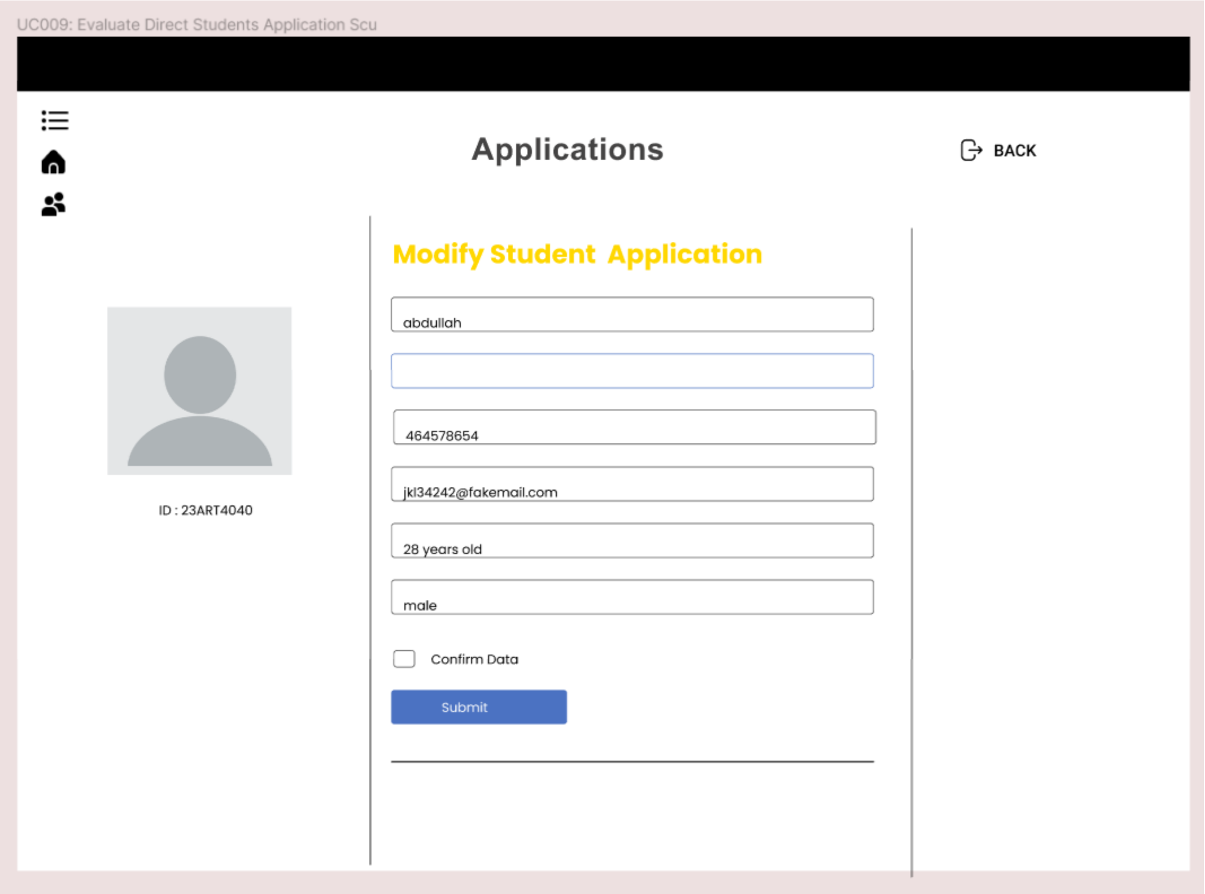
male

□ Confirm Data

ID : 23ART4040

latest Records :

Image: Interface for Evaluate agents Referred Application



UC009: Evaluate Direct Students Application Scu

Applications

Modify Student Application

abdullah

464578654

jkl34242@fakemail.com

28 years old

male

□ Confirm Data

ID : 23ART4040

Image: Interface for Evaluate direct students' application

6. Requirements Matrix

	P001	P002	P003	P004	P005
Module 1, UC001	X				
Module 2, UC002		X			
Module 2, UC003		X			
Module 3, UC004			X		
Module 3, UC005			X		
Module 4, UC006				X	
Module 4, UC007				X	
Module 4, UC008				X	
Module 4, UC009				X	
Module 5, UC010					X
Module 5, UC011					X
Module5, UC012					X

7. Design Pattern

7.1. Factory Design Pattern

For our approach we will be implementing the Factory Design Pattern. The Factory Design Pattern is used to create objects in a superclass but allow subclasses to alter the type of objects that will be created. This allows for greater flexibility in the types of objects that can be created and eliminates the need for the user to know the specific types that can be created. It abstracts the object creation process, making the code easier to maintain and change in the future, as new types of objects can be added or removed without affecting the rest of the code.

We implemented the Factory Design Pattern in PHP where the design pattern is used to create objects in a superclass but allow subclasses to alter the type of objects that will be created. The Factory Design Pattern abstracts the object creation process, making the code easier to maintain and change in the future, as new types of objects can be added or removed without affecting the rest of the code.

In PHP, the Factory Design Pattern is implemented using a factory class that has a static method that returns an instance of the desired object based on the input parameters. The client code calls the factory method, passing in the necessary information, and stores the returned object in a variable. The client code does not need to know the specific implementation of the object that it is working with.

The Factory Design Pattern is useful for reducing the amount of code that is required to create objects, and for making the code more flexible and maintainable by allowing for changes to be made to the specific types of objects that can be created without affecting the rest of the code.

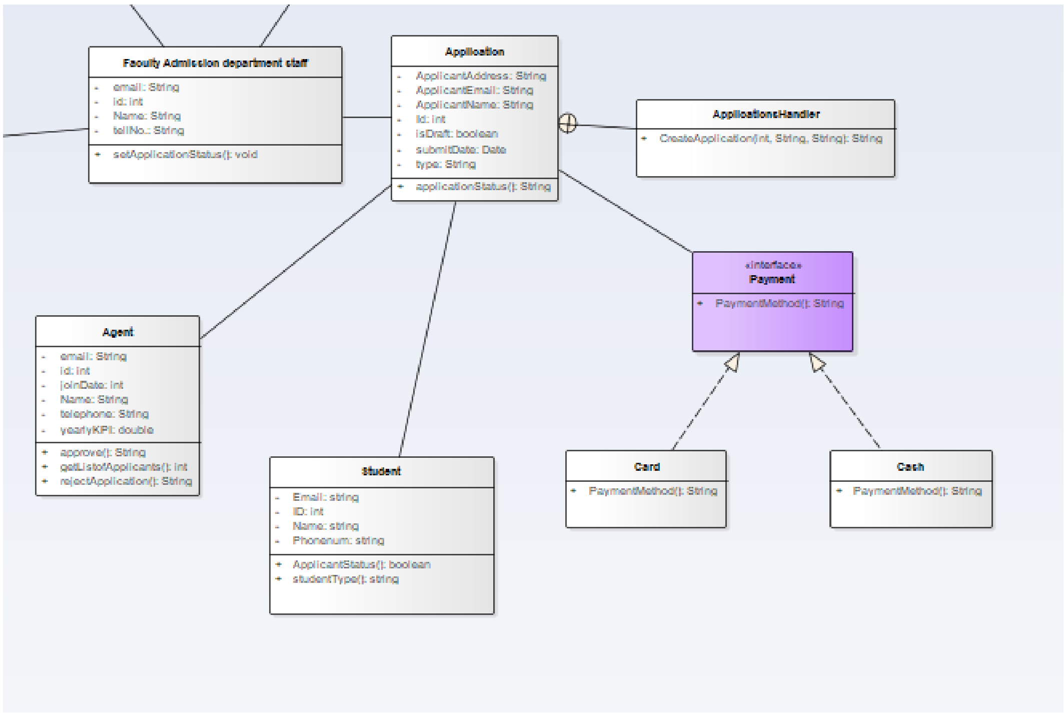


Figure: Factory design pattern of implementation

7.2. Iterator Design Pattern

We implemented the iterator design pattern because in this design pattern that allows traversal of elements in an aggregate object (such as an array) without exposing the underlying representation.

In PHP, the Iterator Design Pattern can be implemented by creating a custom iterator class that implements the Iterator interface, which defines methods such as rewind(), current(), key(), next(), and valid(). The aggregate object, for example an array, can be encapsulated within the iterator class and its elements can be retrieved through these methods.

The Iterator pattern defines a next() method that returns the next element in the collection and a hasNext() method that returns a Boolean indicating whether there are more elements in the collection. By using the Iterator pattern, we can access the elements of a collection one at a time, without having to worry about the underlying implementation of the collection. This makes the collection more flexible and easier to change in the future.

The Iterator Design Pattern is often used in object-oriented programming to provide a standard way of accessing and manipulating elements in a collection, and it helps to ensure that the collection remains unchanged as the elements are accessed and manipulated. It is particularly useful when working with large collections or when you need to traverse elements in a specific order.

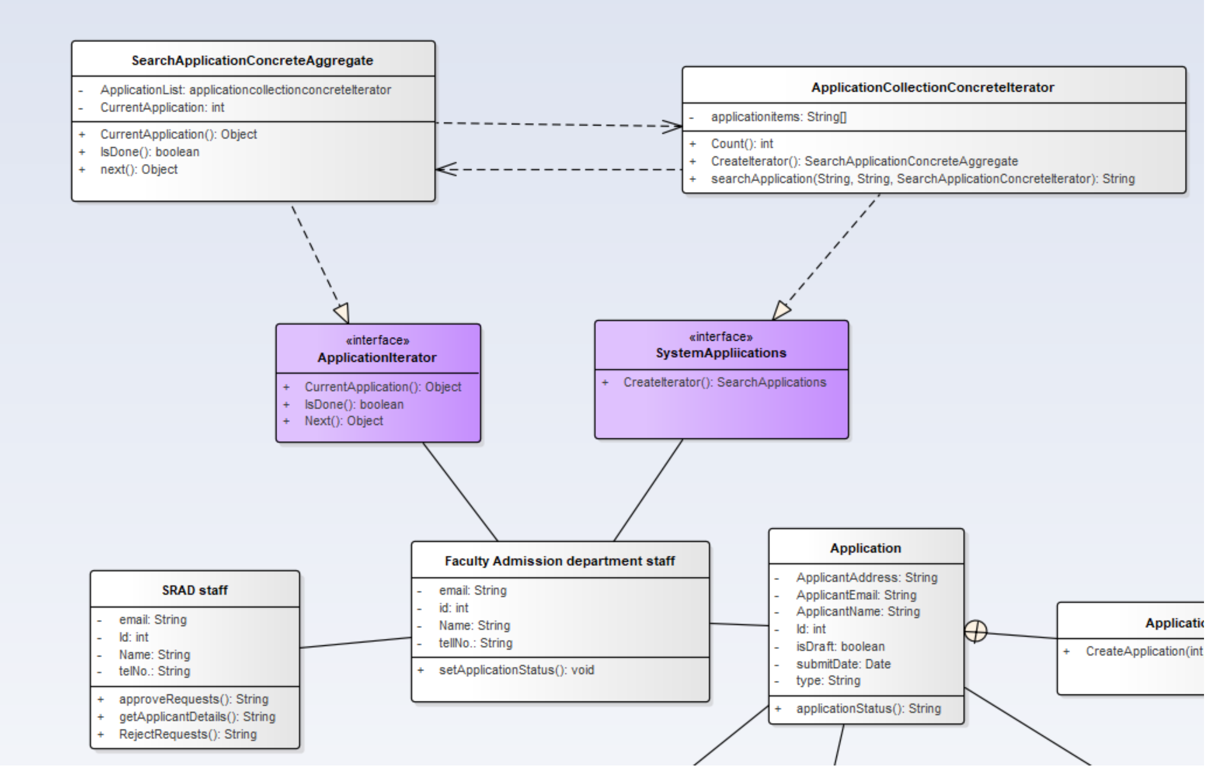

SearchApplicationConcreteAggregate
- ApplicationList: applicationcollectionconcreteIterator
- CurrentApplication: int
+ CurrentApplication(): Object
+ IsDone(): boolean
+ next(): Object
ApplicationCollectionConcreteIterator
- applicationitems: String[]
+ Count(): int
+ CreateIterator(): SearchApplicationConcreteAggregate
+ searchApplication(String, String, SearchApplicationConcreteIterator): String
«interface»
ApplicationIterator
+ CurrentApplication(): Object
+ IsDone(): boolean
+ Next(): Object
«interface»
SystemAppliications
+ CreateIterator(): SearchApplications
SRAD staff
- email: String
- Id: int
- Name: String
- telNo.: String
+ approveRequests(): String
+ getApplicantDetails(): String
+ RejectRequests(): String
Faculty Admission department staff
- email: String
- id: int
- Name: String
- tellNo.: String
+ setApplicationStatus(): void
Application
- ApplicantAddress: String
- ApplicantEmail: String
- ApplicantName: String
- Id: int
- isDraft: boolean
- submitDate: Date
- type: String
+ applicationStatus(): String
Applicati
+ CreateApplication(int

Figure: Iterator design pattern implementation

8. Construction Design

8.1. Flow Based Diagram

We utilized the same flow-based design that is used in entity applications for the construction design. By selecting a programme from the system and then picking an application function within the system, the actor applicant begins their participation with the initial action. Following completion of the application process and submission of the request, the SRAD staff will review the request and, if all is in order, will forward it to the faculty staff. The faculty staff member will then proceed with approving or rejecting the request, and this will mark the activity as complete once the status is displayed.

In this approach, the design process is focused on creating a smooth, seamless flow of materials, people, and information from the beginning of the project through to its completion. The goal of flow-based design is to improve the overall efficiency of the construction process, while also enhancing the quality of the finished product.

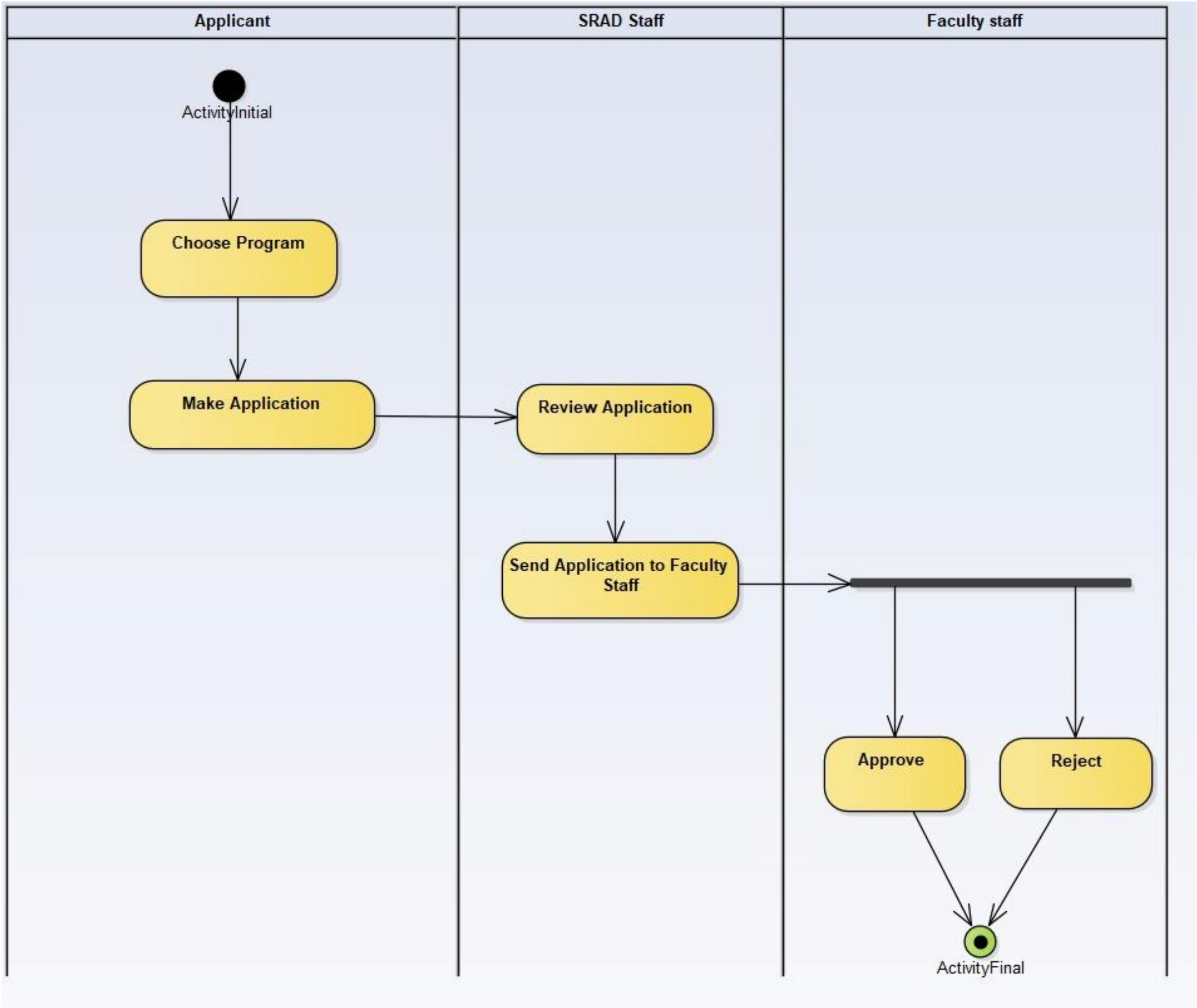


Figure: Flow based diagram implementation

8.2. State Machine Diagram

The new application will be prepared for the implementation of the state machine diagram pattern from the starting state, and then it will go on to the application review process where the application will be sent to the faculty for evaluation. From here, if the conditions are satisfied, the application would be accepted. However, if the conditions are not satisfied, the application would be rejected. Additionally, the new application procedure will be initiated appropriately for each of the final cases.

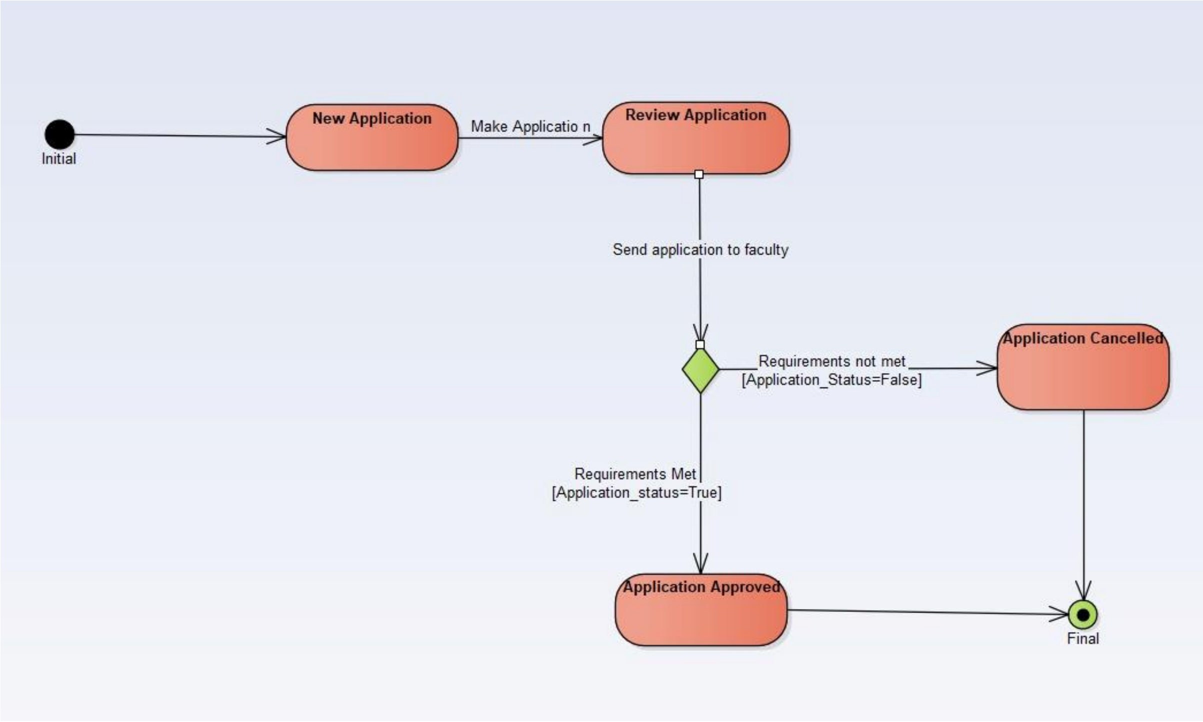


Figure: State machine diagram implementation