Addis Ababa Institute of Technology

SCHOOL OF INFORMATION AND TECHNOLOGY ENGINEERING

Department of IT/SW Eng.

AAiT INFORMATION MANAGEMENT SYSTEM

Software Design Specification

Team Members:

Name	ID
1. Tigist Wondimneh	UGR/2538/12
2. Tigist Zelalem	UGR/1852/12
3. Mekdes Kebede	UGR/3127/12
4. Abraham Shimekt	UGR/0129/12
5. Binyam Lemma	ATR/1341/10
6. Khansa Tahir	ATR/3185/11

Advisors: Ms. Nuniyat Kifle **Date:** Jan 4, 2022

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

DEFINITIONS, ACRONYMS, ABBREVIATIONS

AAiT-IMS……Addis Ababa Institute of Technology iformation management System

MVC ……Model View Controller

AMI…… Asynchronous Method Invocation

SDS…… Software Design Specification

HTML5……HyperText Markup Language version 5

CSS3……..Cascading Style Sheet version 3

MySQL…… "My", the name of co-founder Michael Widenius's daughter, and "SQL", the abbreviation for Structured Query Language

1. INTRODUCTION

1.1 PURPOSE

This System Design Specification is made with the purpose of outlining the system architecture and design of the Student Registration System in detail. It will provide insight during implementation of the system so as to make the software meet the requirements of the system.

1.2 GENERAL OVERVIEW

This SDS document has the Context diagram, Class diagram, Sequence diagram, Deployment diagram and detailed description about the classes in the Class diagram. It also has Architectural models of the system software.

The AAiT-IMS will be implemented using the three tier/ Model View Controller (MVC) system architecture. MVC is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller.

The Model component represents all data related logic in the system. The UI of the application is represented with the View component.Whereas the Controller component enables interconnection between the Model and the View. This architecture is preferred because, MVC:

- Supports Asynchronous Method Invocation (AMI)

  MVC supports AMI which will allow the team to build a faster loading web application systems.

- Easy Modification

  As different sections are independent of each other, changes in one section does not affect others.So if the need arises to modify a particular section, other parts will not be affected.

- Faster Implementation

  MVC model allows parallel development of the system, different members of the team will be working on different parts of the project making the implementation faster.

- Easy Maintenance

  Code duplication is minimized as this model provides an outline on how to arrange features into actual implementations.

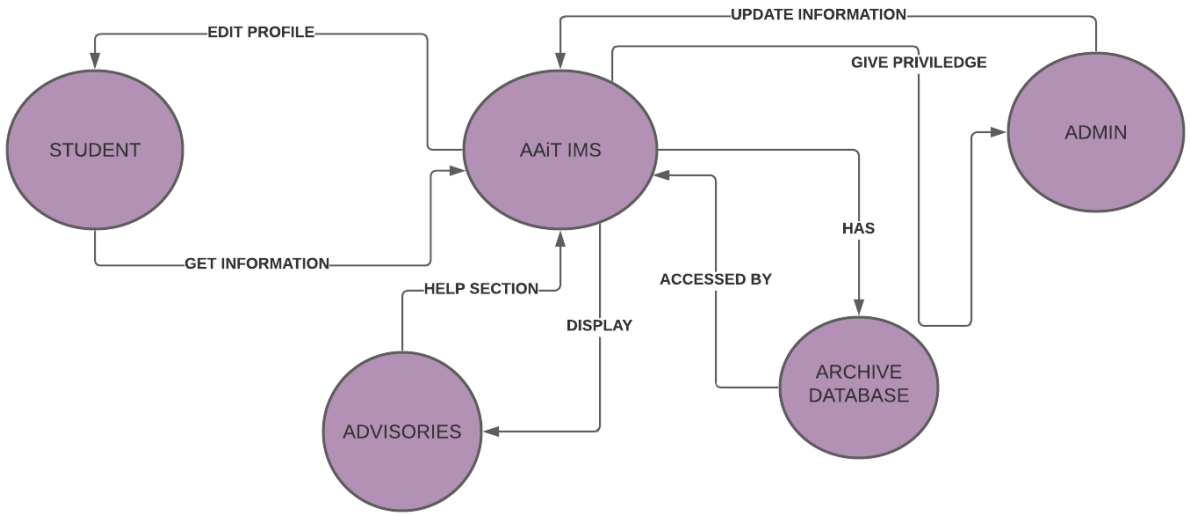

Figure 1: High Context Diagram

1.3 DEVELOPMENT METHODS & CONTINGENCIES

An Object Oriented Programming Approach with MVC will be used to develop our system software.

- HTML5 for content definition, CSS3 for layout and display, JavaScript and PHP for communication will be used to develop the front end of the system.
- The Database part of our system will be implemented with MySQL.
- The system will use Apache Web Server will be used to develop our system.
- The backend of the system will be developed by using Spring Boot framework.

2. SYSTEM ARCHITECTURE

2.1 SUBSYSTEM DECOMPOSITION

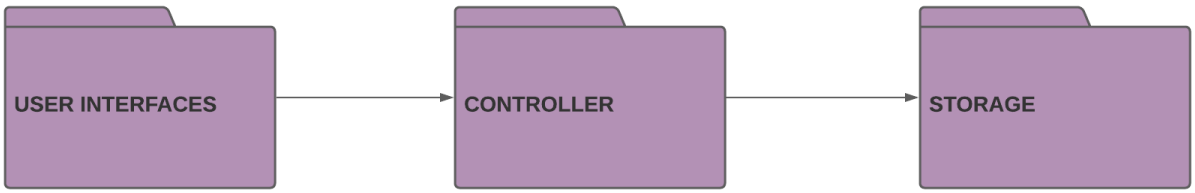

Figure 2: Component Diagram Layer 1

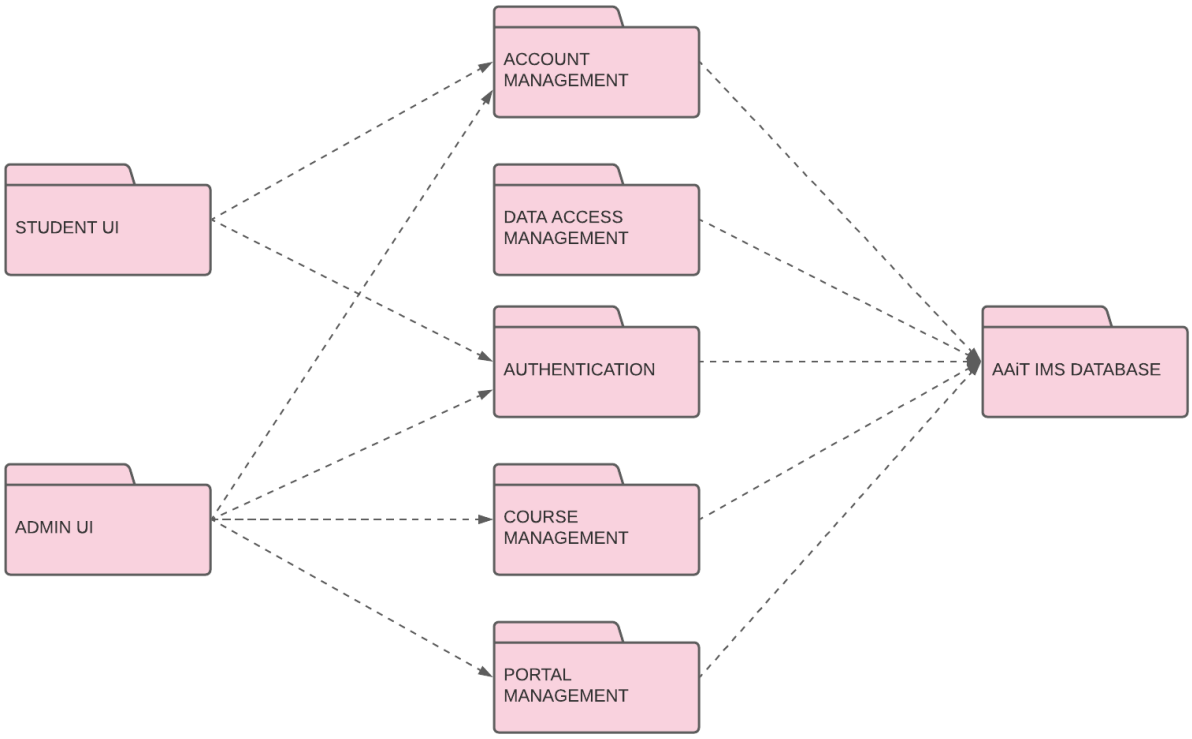

Figure 3: Component Diagram Layer 2

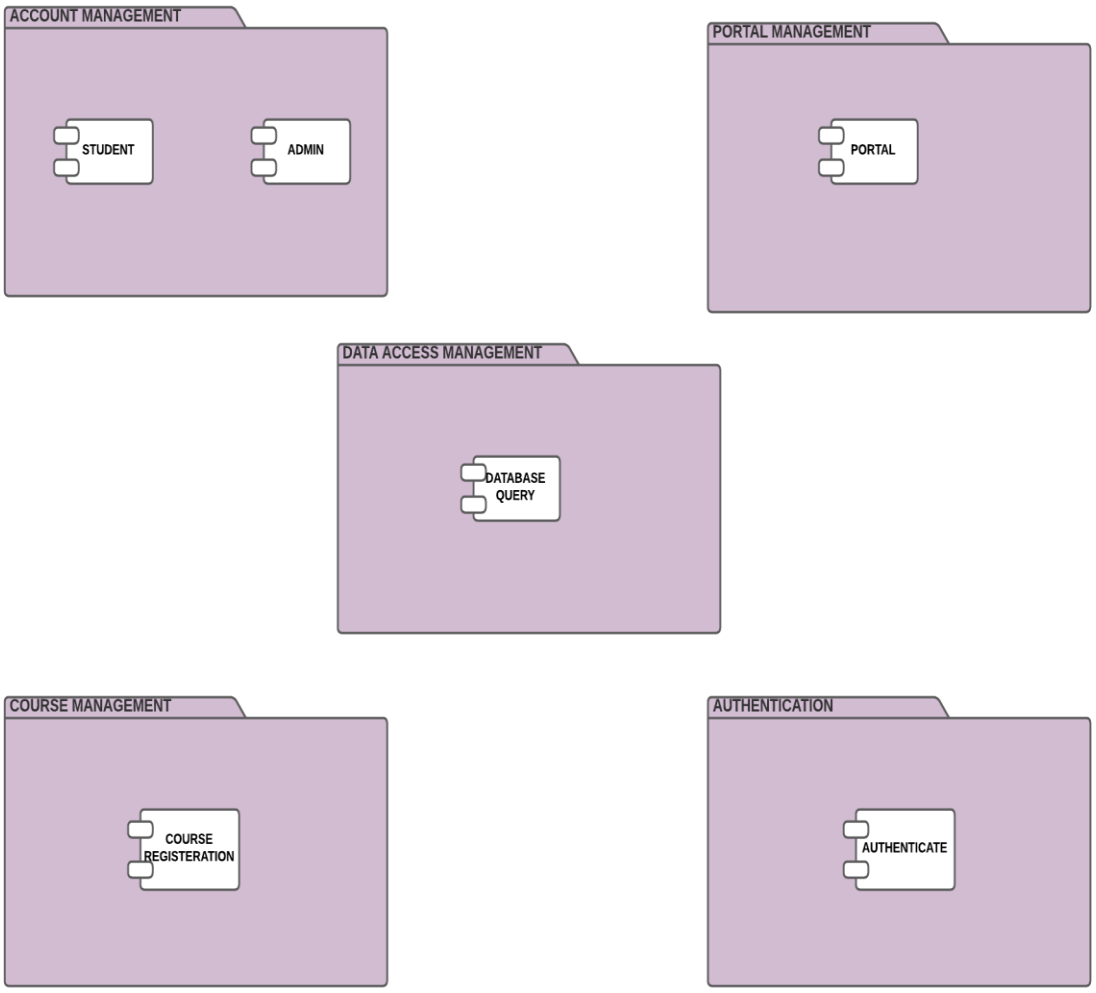

Figure 4: Component Diagram Layer 3

2.2 HARDWARE/SOFTWARE MAPPING

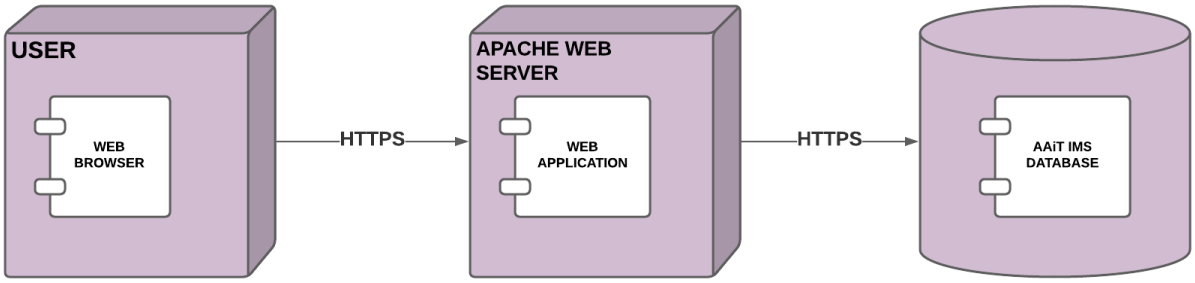

Figure 5: Deployment Diagram

3. OBJECT MODEL

3.1 CLASS DIAGRAM

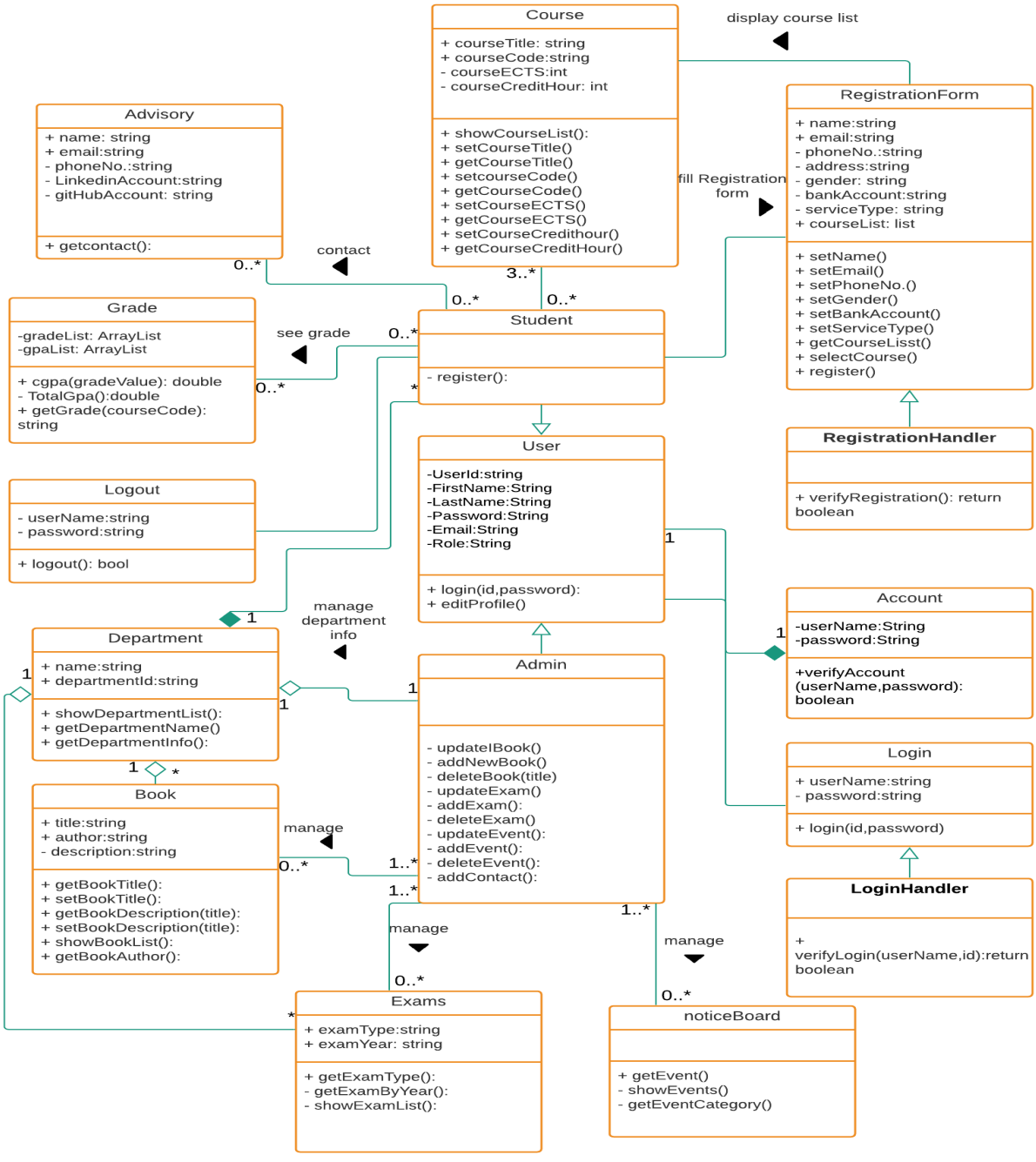

Figure 6: Class Diagram

3.2 SEQUENCE DIAGRAM

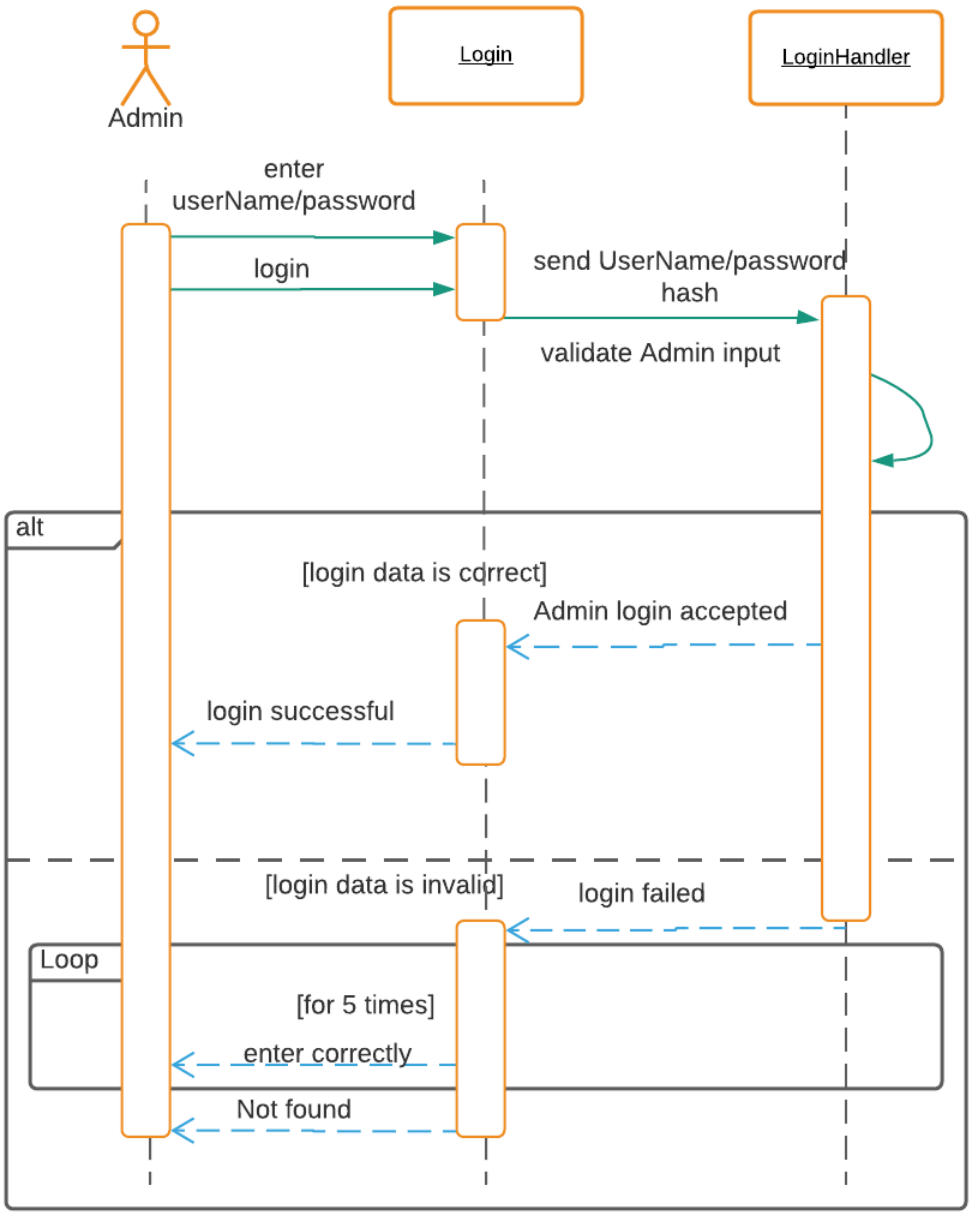

Figure 7: Sequence Diagram for admin login

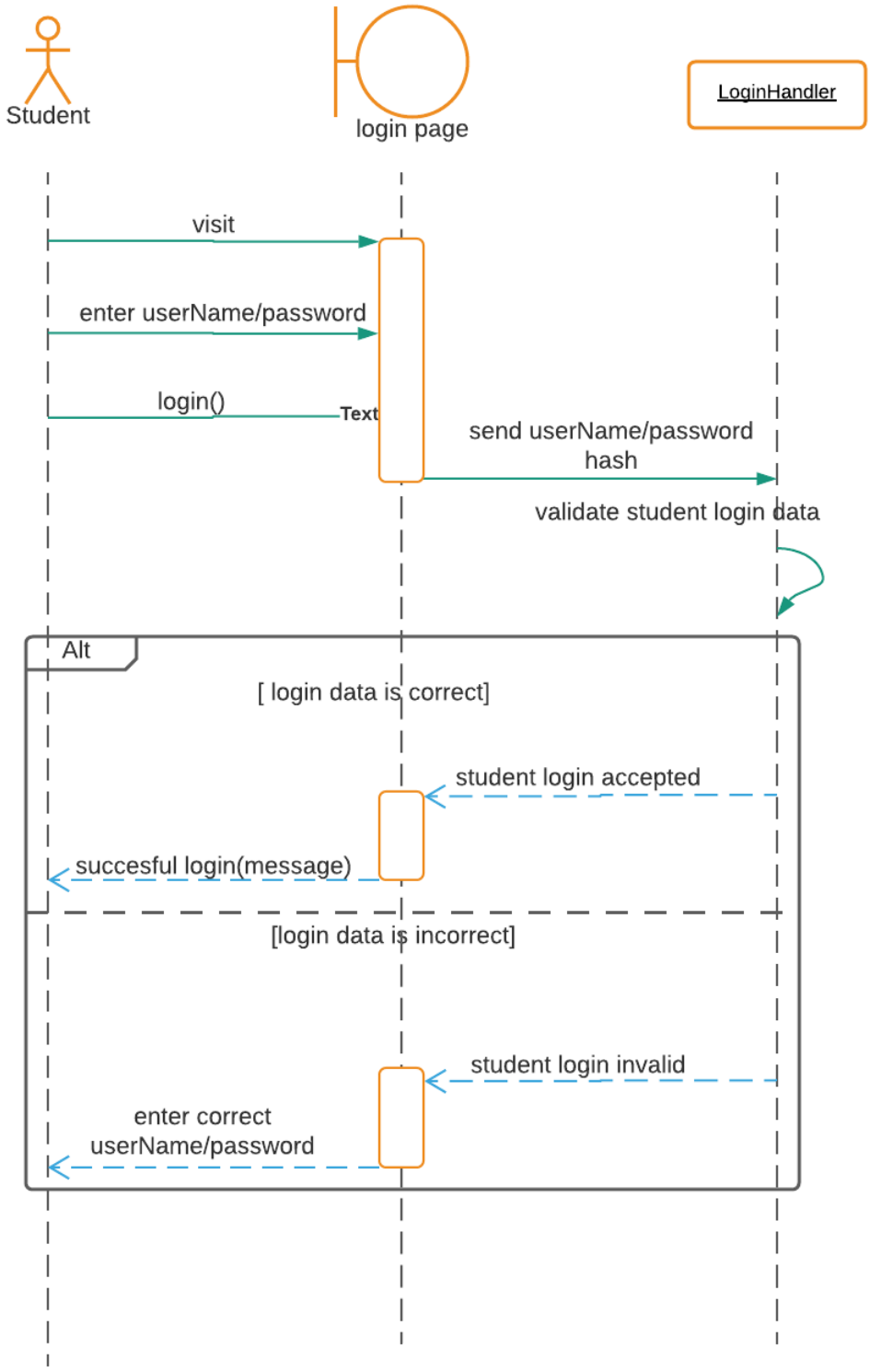

Figure 8: Sequence Diagram for student login

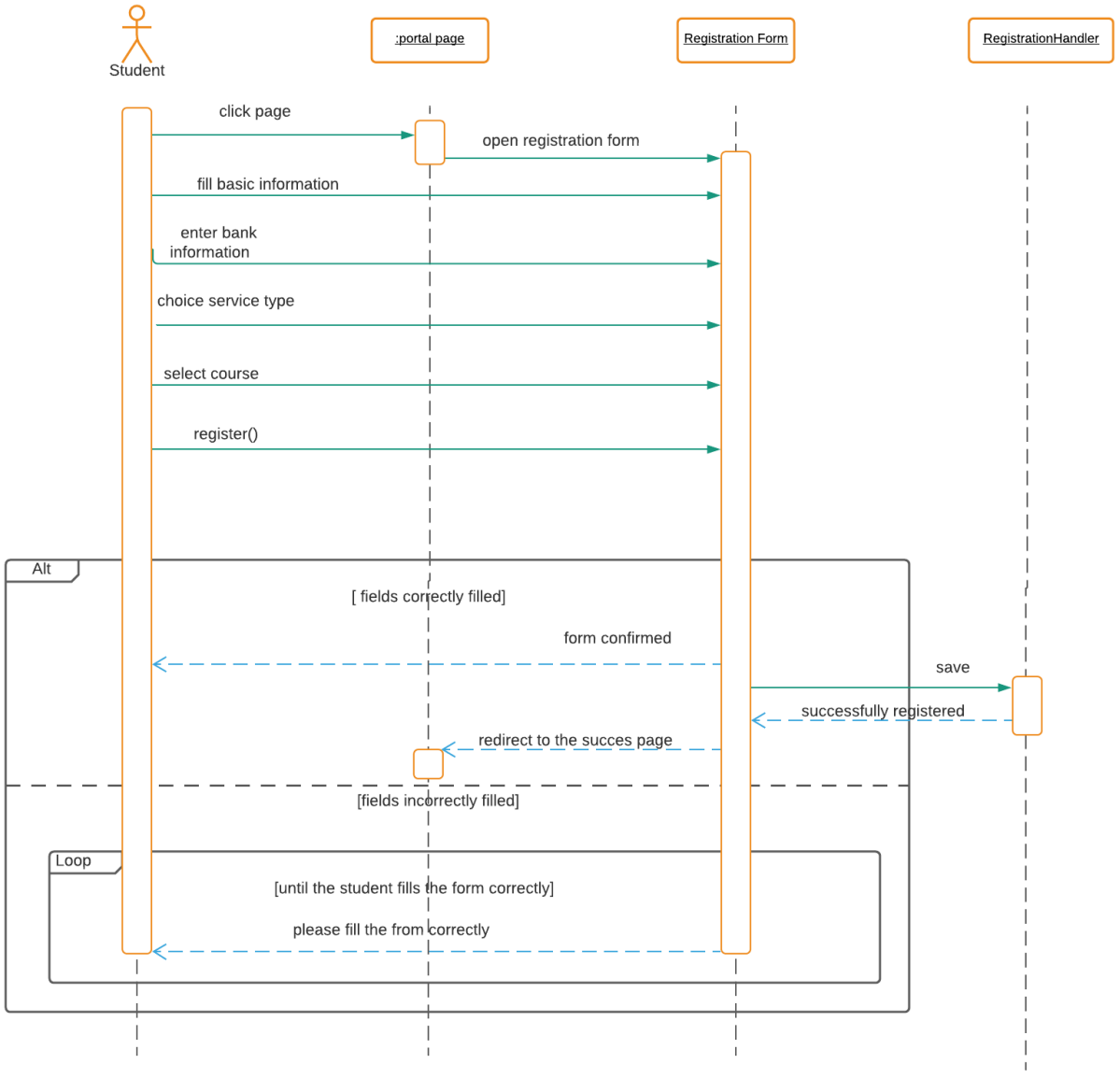

Figure 9: Sequence Diagram for student portal access

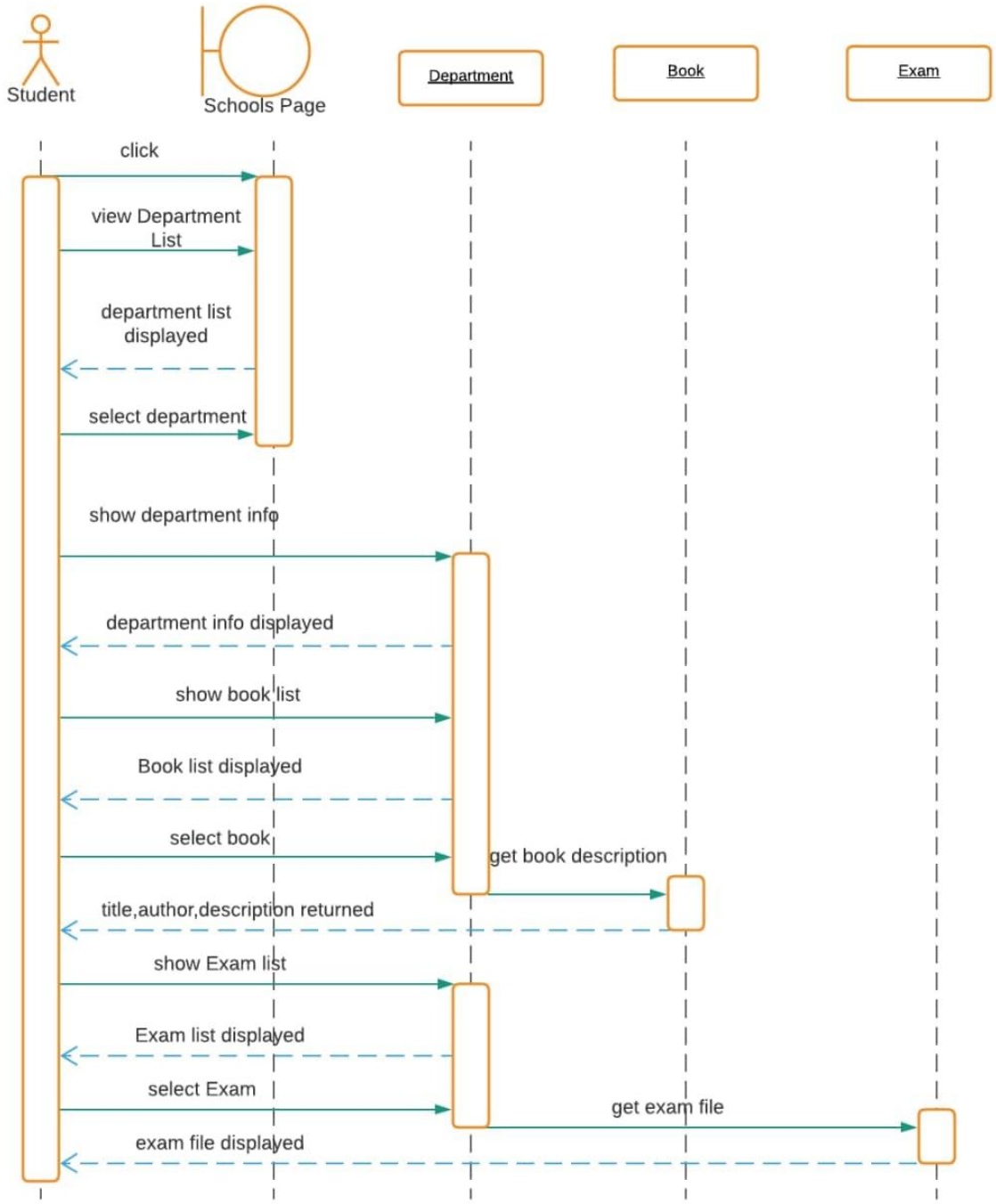

Figure 10: Sequence Diagram for student department information access

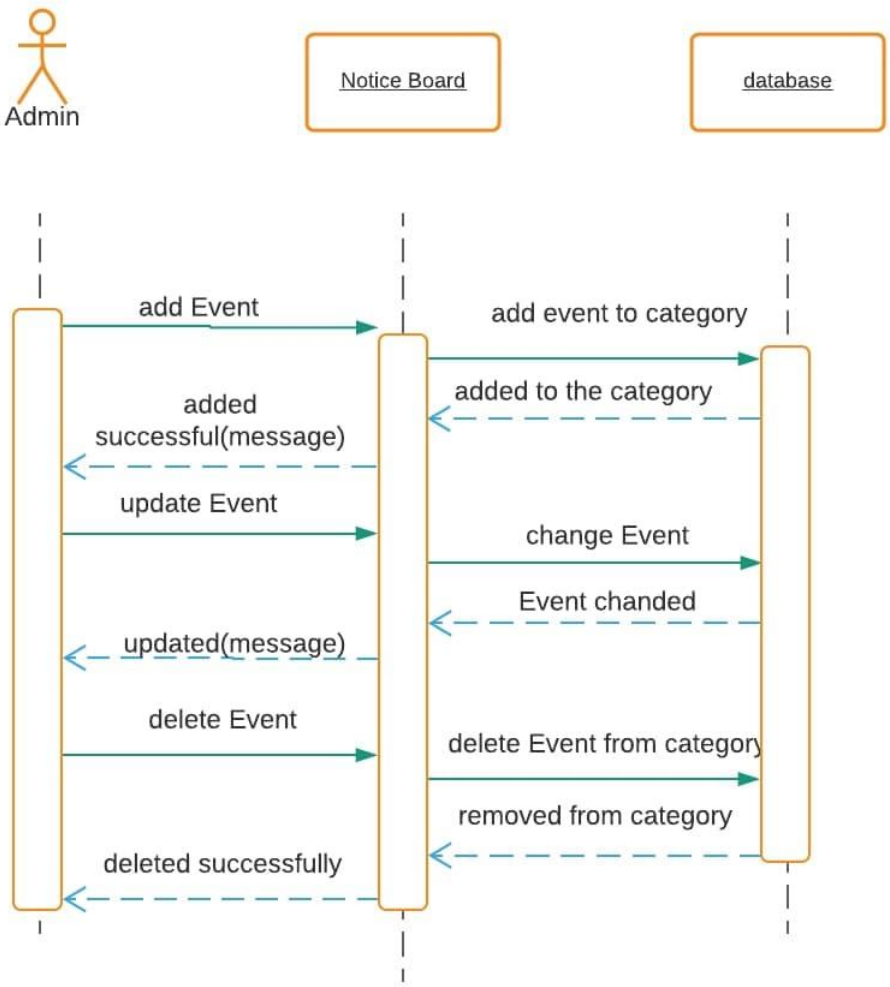

Figure 11: Sequence Diagram for student Notice Board Access

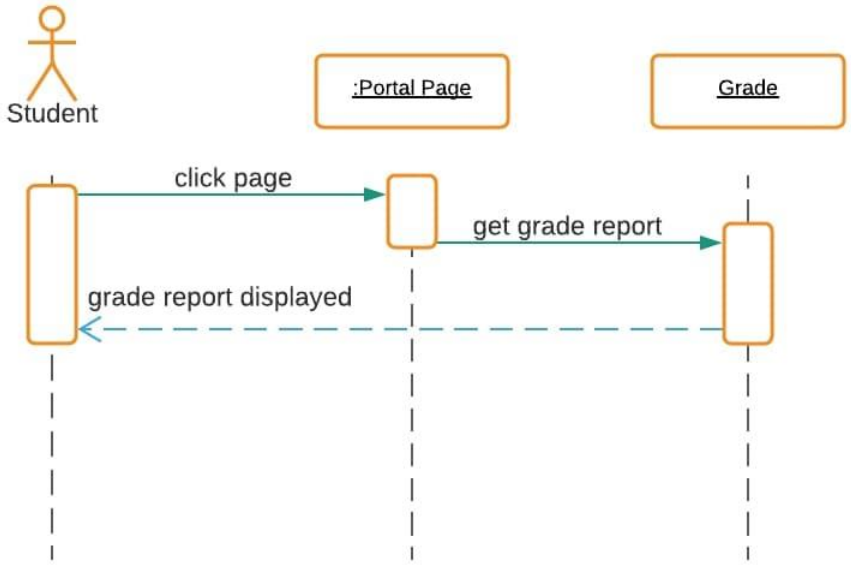

Figure 12: Sequence Diagram for viewing grade

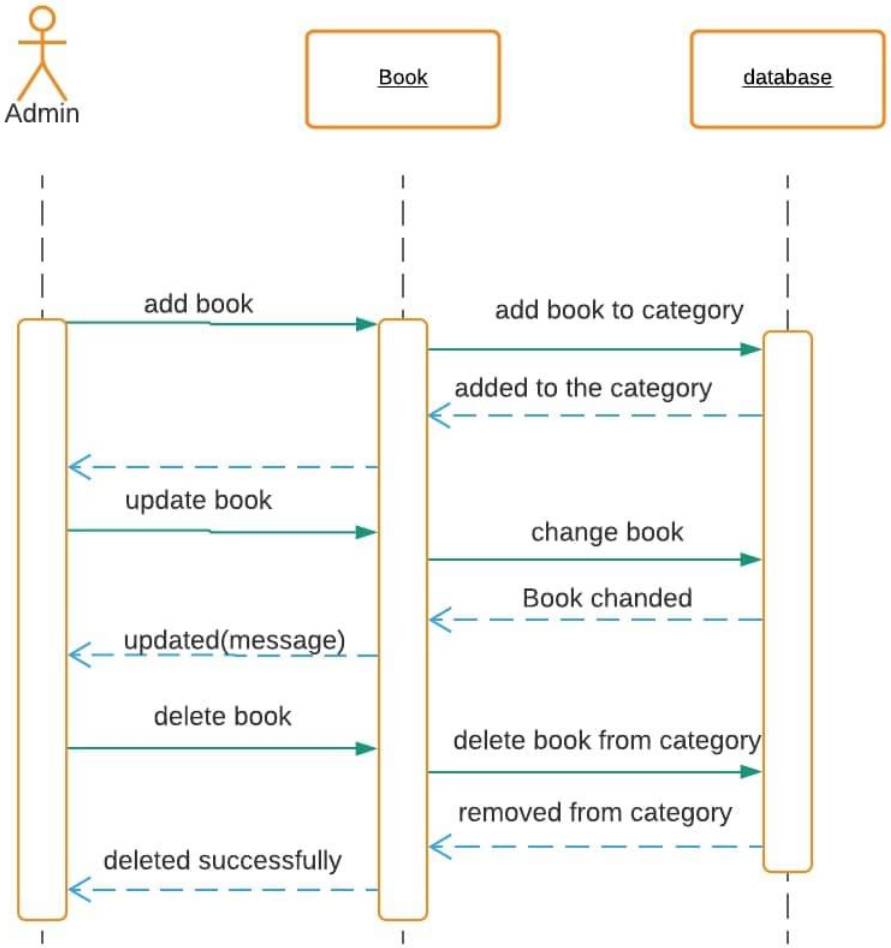

Figure 13: Sequence Diagram for accessing Books

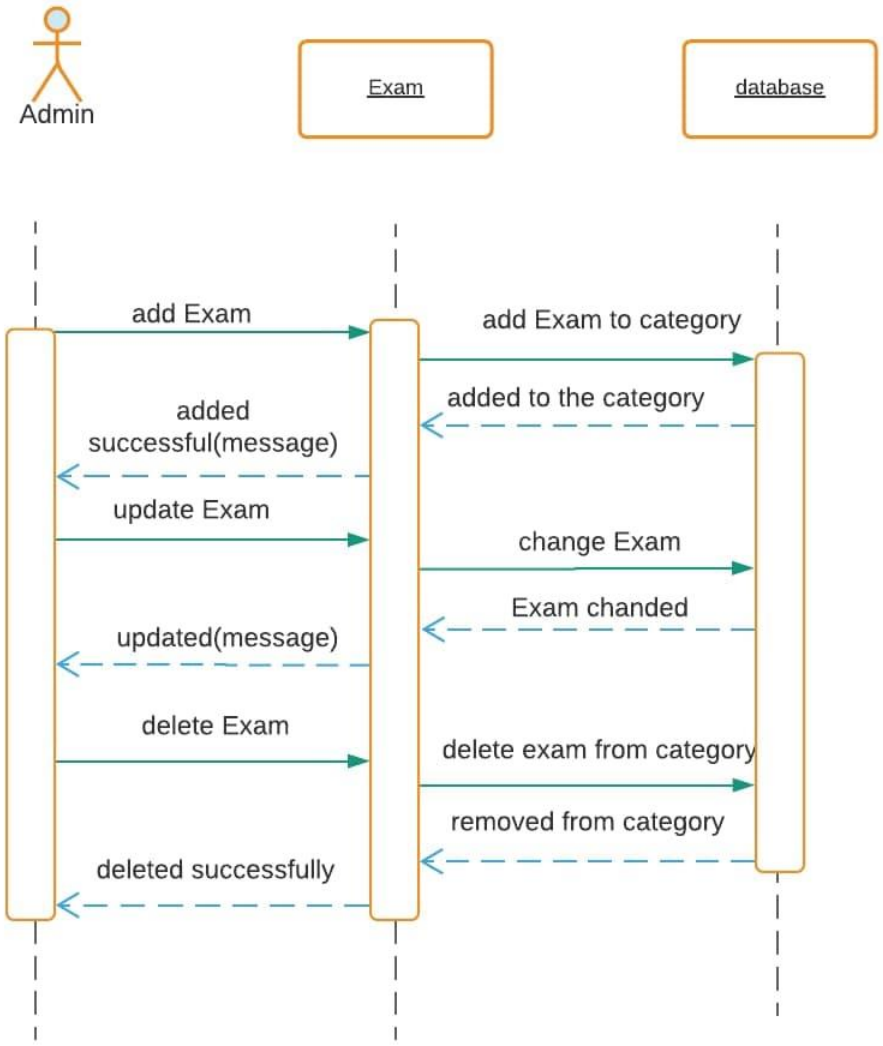

Figure 14: Sequence Diagram for accessing Exams

4. DETAILED DESIGN

Table: 1 login class

login
- userName: string **-password: string**
+ login(id, password): bool

Table: 2 Attributes description for Login class

Attribute	Type	Visibility	Invariant
Username	String	Private	Username <>NULL and must not contain special characters and integers.
Password	string	Private	password<> NULL must be at least 6 digits

Table: 3 Operations description for login class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
Login(id, password)	Public	Boolean		The application features shouldn't be available	After verification if the return is true the user will be able to access information and enable to do some tasks. If not error message will be displayed.

Table: 4 Student class

Student class
- fname : string **- lname: string** **+ UserName:string** **+ StudnetId: string** - password: string - email: string
+ editProfile(): **-register():** **+ login():**

Table: 5 Attributes description for Student class

Attribute	Type	Visibility	Invariant
lname	String	Public	Must not contain special characters and integers.
fname	string	Public	Must not contain special characters and integers.
UserName	string	private	UserName<> NULL Must not contain special character. must be unique
StudentId	string	private	Id <> NULL must be unique

password	string	private	password<> NULL must be at least 6 digits
email	string	public	Must end with @gmail.com or @live.com

Table: 6 Operation descriptions for Student class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
editProfile():	public	void	.	The user must have an account	View profile info
register():	private	bool			
login ():	public	string		The user must have an account	The user will access portal and main page

Table: 7 admin class

admin
- fname : string
- lname: string
+ UserName:string
+ AdminId: string

- password: string - email: string
- updateIBook() **- addNewBook()** **- deleteBook(title)** **- updateExam()** **- addExam():** **- deleteExam()** **- updateEvent():** **- addEvent():** **- deleteEvent():** **- addContact():** **+ addAdvisor(fname,lname,email** **,phoneNo,socialMediaAccount):Void** **+removeAdvisor(advisorId):Void**

Table: 8 attribute descriptions for admin class

Attribute	Type	Visibility	Invariant
lname	String	Public	Must not contain special characters and integers.
fname	string	Public	Must not contain special characters and integers.

UserName	string	private	UserName<> NULL Must not contain special character. must be unique
AdmintId	string	private	Id <> NULL must be unique
password	string	private	password<> NULL must be at least 6 digits
email	string	public	Must end with @gmail.com or @live.com

Table: 9 Operation descriptions for admin class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
updateIBook()				The user must be an admin to update	Book Will be updated
addNewBook()	private	void		The usser must be an admin to add	Book will be added to the database
deleteBook(title)	private	bool	Title	The user must be an admin delete	Deleted book will be lost permanently
updateExam()	private	void		The user must be an admin to update	Book Will be updated
addExam():	private	void		The usser must be an admin to add Exams	Exam will be added to the database

deleteExam()	private	bool	title	The user must be an admin delete Exam	Deleted exam will be lost permanently
updateEvent():	private	void		The user must be an admin to update event	Event Will be updated
addEvent():	private	void		The usser must be an admin to add Events	Event will be added to the database
deleteEvent():	private	bool	title	The user must be an admin delete Events	Deleted Event will be lost permanently
addContact():	private	void		The usser must be an admin to add to contacts	Advisories Contact will be add to the database
addAdvisor(fname,lname,email ,phoneNo,socialMediaAccount):Void	public	void		The usser must be an admin to add Advisories	The advisory will be add to the addvisory list
removeAdvisor(advisorId):Void	public	void		The user must be an admin to remove Advisor	The user advisory will be deleted

Table: 10 noticeboard class

noticeboard
- title: string **- description: string**
+ getEvent() **- showEvents()** **- getEventCategory()**

Table: 11 attribute descriptions for noticeboard class

Attribute	Type	Visibility	Invariant
title	String	private	
description	string	private	

Table: 12 Operation descriptions for noticeboard class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
getEvent()	public	string	.		
showEvents()	public	list			
getEventCategory()	public	list			

Table: 13 department class

department
- name:string **- departmentList:ArrayList**
+ showDepartmentList(): **+ getDepartmentName()** **+ getDepartmentInfo():**

Table: 14 attribute descriptions for department class

Attribute	Type	Visibility	Invariant
name	String	Private	name <> NULL and must not contain special characters and integers.
departmentList	ArrayList	Private	departmentList<> NULL

Table: 15 Operation descriptions for department class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
showDepartmentList():	Public	object		The user must click on the schools button	Available for accessing information and enable to do some tasks
getDepartmentName()	public	string			

getDepartmentInfo():				The user must click on specific department	Gets the selected department info

Table: 16 RegistrationForm class

RegistrationForm
- name: string **email: string** **- phoneNo.: string** **- address: string** **- gender: string** **- bankAccount: string** **- serviceType: object**
+setUserData(*data) **+ setServiceType()** **+ getCourseList()** **+ selectCourse()** **+ register()**

Table: 17 Attributes description for RegistrationForm class

Attribute	Type	Visibility	Invariant
name	String	Private	Must not contain special characters and integers.

email	string	Private	Must not contain special characters and integers. Must contain **.** (dot) Must contain @
phoneNo.	string	Private	Must be less than 10 digit
address	string	private	Password<> NULL, must be at least 4 digits
gender	char	private	Must give an option of female or male
bankAccount	string	private	Must be a valid account number
serviceType	object	private	

Table: 18 Operation descriptions for registrationForm class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
register()	Public	void	.	The user must click on the register button	If the information are correct the user get registered
setServiceType(serviceType)	public	void	serviceType		Get all information of the selected school
setUserData(*data):	public	void	fname, lname,sex, bankAccount, address and		The user updates its profile

			email		
getCourseList()	public				
selectCourse()					

Table: 19 Exams class

ExamManager
+ examType:string **+ examYear: string**
+ getExamName(): **+ getExamByYear():** **+ showExamList():**

Table: 20 attribute descriptions for Exams class

Attribute	Type	Visibility	Invariant
examType	String	Private	type <> NULL
ExamTitle	string	private	Title<>NULL
ExamYear	string	Private	year <> NULL
ExamDescription	string	private	

Table: 21 Operation descriptions for Exams class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
getExamType():	public	void	.		
getExamByYear():	public	void			
showExamList():	public	void			
getExamDescription(title)	public	string			

Table: 22 Book class

Book
- title: string **- author: string** **- description: string**
+ getBookTitle(): **+ setBookTitle(t):** **+ getBookDescription(title):** **+ setBookDescription(title):** **+ showBookList():** **+ getBookAuthor():**

Table: 23 attribute descriptions for Books class

Attribute	Type	Visibility	Invariant
title	String	private	type <> NULL
author	string	private	year <> NULL and must contain the authors name and some information
Description	string	private	Description<> NULL

Table: 24 Operation descriptions for Books class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
getBookTitle():	public		.title		
setBookTitle(t):	public	void	title		
setBookDescription(title):	public	object	title		
getBookDescription(title)	public	string	title	The user must click on a specific book using the title	The user gets a short description of the book
showBookList():	public			The user must click the books sections	The user will get a least of books
getBookAuthor():	public				

Table: 25 Advisory Class

Advisory
- fname: string **- lname: string** **- advisorId:String** **- email:string** **- phoneNo.:string** **- socialMediaAccount: String**
+getName():String **+ getContact(): string**

Table: 26 attribute descriptions for Advisory class

Attribute	Type	Visibility	Invariant
Fname	String	Private	Must not contain special characters, underscore and integers.
Lname	String	Private	Must not contain special characters, underscore and integers.
AdvisorId	String	Private	AdvisorId<>NULL and can contain the mixture of special characters and integers. • Must be unique
Email	String	Private	it must be a link it must contain @,

			.com , and the total number of characters is at least 8.
PhoneNo	String	private	must be 10 digit integers
SocialMediaAccount	String	private	

Table: 27 Operation descriptions for Advisory class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
getName()	Public	String			The advisor's name will be displayed
getContact ()	public	string			The advisor's contact will be displayed

Table: 28 Grade class

grade
-gradeList: ArrayList **-gpaList: ArrayList**
+cgpa(gpaList): double **+gpa(gradeList): double**

+getGrade(): object

Table: 29 Attributes description grade class

Attribute	Type	Visibility	Invariant
gradeList	ArrayList	private	Type <> NULL
gpaList	ArrayList	private	Type <>NULL

Table: 30 Operation description gradecontroller class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
cgpa()	Public	Double	gpaList	Gpa must be passed as a parameter	CGPA will be displayed
gpaList()	Public	Double	gradeList	The grade kist must be fulfilled	GPA will be dispalyed
getGrade()	Public	Object			Grade will be dispalyed

Table: 31 Course class

Course

- courseTitle: string **- courseCode: string** **- courseECTS: int** **- courseCreditHour: int** **- courseList: ArrayList**
+ showCourseList(): **+ setCourseTitle()** **+ getCourseTitle()** **+ setcourseCode()** **+ getCourseCode()** **+ setCourseECTS()** **+ getCourseECTS()** **+ setCourseCredithour()** **+ getCourseCreditHour()**

Table: 32 Attributes description for Course class

Attribute	Type	Visibility	Invariant
courseTitle	String	Private	courseTitle<>NULL and must contain only characters.
courseCode	String	Private	courseCode<>NULL and must contain only integers and characters.
courseECTS	Integer	Private	courseECTS<>NULL and must contain only integers.
courseCreditHour	integer	Private	courseCreditHour<>NULL and must contain only

			integers.
courseList	Object	Private	

Table: 33 Operation descriptions for Course class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
showCourseList()	Public	ArrayList			
setCourseTitle()	Public	void	courseCode		
setCourseTitle()	Public	void	courseCode		
setcourseCode() getCourseCode()	public				
setCourseECTS()	**public**	**void**			
getCourseECTS()	**public**	**int**			
setCourseCredithour()	**public**	**void**			
getCourseCreditHour()	**public**	**string**			

Table: 34 logout class

Logout
- userName:string **- password:string**
+ logout(userName,password):Void

Table: 35 Attributes description for logout class

Attribute	Type	Visibility	Invariant
userName	String	Private	Must only contain characters
passWord	String	Private	Can be a mixture of integers and special characters and must be unique

Table: 36 operation descriptions for logout class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
logout()	Public	void	Username Password	The user must click on the logout button	The user will be out of the system

Table: 37 account class

Account
-userName:String -password:String
+ verifyAccount(userName,password): boolean

Table: 38 Attributes description for account class

Attribute	Type	Visibility	Invariant
userName	String	Private	Must only contain characters
password	String	Private	Can be a mixture of integers, characters and special characters and must be unique

Table: 39 operation descriptions for account class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
verifyAccount()		Boolean	Username Password		If the password and the username are correct return true else returns false

Table: 40 RegistrationHandler class

RegistrationHandler
+verifyRegistration():bool

Table: 41 operation descriptions for account class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
verifyRegistration()		Boolean	Username Password		If the form filled correctly return true else returns false

Table: 42 LoginHandler class

RegistrationHandler
+verifyLogin **():bool**

Table: 43 operation descriptions for account class

Operation	Visibility	Return type	Argument	Pre-Condition	Post Condition
verifyLogin ()		Boolean	Username, Password		If the password and the username are correct return true else returns false

REFERENCES

BIBLIOGRAPHY

Ian Sommerville (2007). Software Engineering, 239-389.

Degif Teka (2008 E.C). Student Management System. Addis Ababa University.

Software architecture patterns by mark richardslo

WEB RESOURCE

http://www.tutorialspoint.com/uml/index.htm at Dec 27, 2021.

https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/ at Dec 31, 2021.

ADDITIONALS

Previous projects done by senior students

Template given by our advisor Ms.Nuniyat Kifle