COLLEGE OF NATURAL AND COMPUTATIONAL SCIENCE

SCHOOL OF INFORMATION SCIENCE

Web-based Church Management System

In Partial Fulfillment of Requirements of B.SC Degree in Information Systems (Industrial Project-II)

Group Members

No.	Name		ID
1.	*Abas*	*Seid*	*NSR/9028/11*
2.	*Anteneh*	*Mulu*	*NSR/7775/11*
3.	*Birhanu*	*Kabito*	*NSR/4525/11*
4.	*Dawit*	*Mitiku*	*NSR/0893/11*
5.	*Menoralem*	*Bekalu*	*NSR/1778/11*
6.	*Nathnael*	*Yeshitila*	*NSR/0414/11*

Advisor: Mr. Betsegawu Dereje

Chapter 5
Object-Oriented Design

5.1 Introduction

Object-oriented design (OOD) is the process of planning a system of interacting objects for the purpose of solving a software problem. It is one approach to software design. The purpose of Object-Oriented Design (OOD) is to determine how to build the system and obtain the information needed to derive the actual implementation of the system. This is different from analysis, which focuses on understanding of what will be built, by focusing on how the system will be built (Horstmann, 2005)**.**

In this chapter, the team members are going to address the major Object-Oriented Design artifacts in order to bridge the gap between analysis and implementation, which are class type architecture, class modeling, collaboration modeling, component modeling, deployment modeling, user interface design, and other design artifact

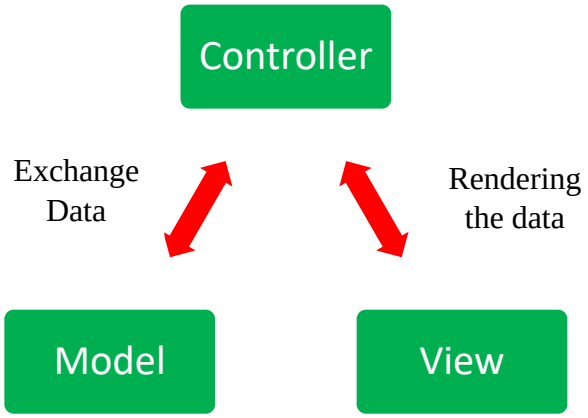
5.2 Class Type Architecture

Class type architecture provides a strategy for layering the class of system to distribute the functionality of our software among classes. Furthermore, class-type architecture provides guidance as to what other types of classes a given type of class will interact with, and how that interaction will occur. This increases the extensibility, maintainability, and portability of the system. (Ambler, 2001)

We selected the MVC architecture because the goal of MVC is to break down the whole system into three subsystems (modules). MVC is an application design pattern that separates the application data and business logic (model) from the presentation (view). MVC stands for Model, View & Controller. It is also called a component-based architectural style because each module can be implemented by software components: data components, presentation components, input controls, control dispatch, and business process components. Each module in the MVC architecture has its own responsibility. (Jackson, 2022)

Project team members with different expertise can work more efficiently in their own areas. For example, graphic professionals work on presentation of UI module, programmer professionals work on input processing such as authentication in the Controller module, and data processing and database professionals can focus on the Model module to provide all the data that the Web application needs.

Figure 5.1 MVC architecture



5.2.3 Model

The Model is that part of the Model-View-Controller design pattern that we can simply describe as the layer of the design pattern that handles the management of the data, which is received from the corresponding layers and then sent back to those layers. One thing to note here is that the Model does not know where the data comes from and how it is received.

List of models

- Account
- Member
- Admin
- Educ_Material
- Event
- Message
- Notification
- Church_Profile

- Museum_Record
- Tithe
- Offering
- Promise
- Service_Payment
- Family

5.2.3 View

The View consists of presentation logic aspects such as templates and caching and the code that involves presentation. Precisely, the View defines exactly what is presented to the user. Usually, Controllers pass data to each View to render in some format. This is where you are likely to find HTML markup in your MVC application.

Lists of Views

- Home Page
- Login page
- Registration Page
- Contact Us Page
- About Us page
- Event Page
- Museum Page
- Edit Member Page
- Member List Page
- Add Museum Record Page
- Edit Museum Record Page
- Add Educ_Material Page
- Edit Educ_Material Page
- Educ_Material List Page
- Post News Page
- Edit News Page
- Edit Church Profile Page
- Send Notification Page
- Display Notification Page
- Display Message Page
- Add Tithe Page
- Display Tithe Records Page
- Edit Tithe Page
- Add Offering Page
- Edit Offering Page
- Display Offering Records Page
- Add Service Payment Page
- Edit Service payment page
- Display Service Payment Records Page
- Add Admin Page
- Edit Admin Page
- Admin List Page
- Update Profile Page

5.2.3 Controller

The Controller is a part of the Model-View-Controller (MVC) design pattern that we can simply describe as the logical layer of our application. It understands the requests that come from the other end, makes calls to the corresponding methods, performs primary checks, handles the logic of the request, and then returns the data to the corresponding View or redirects the end user to another route.

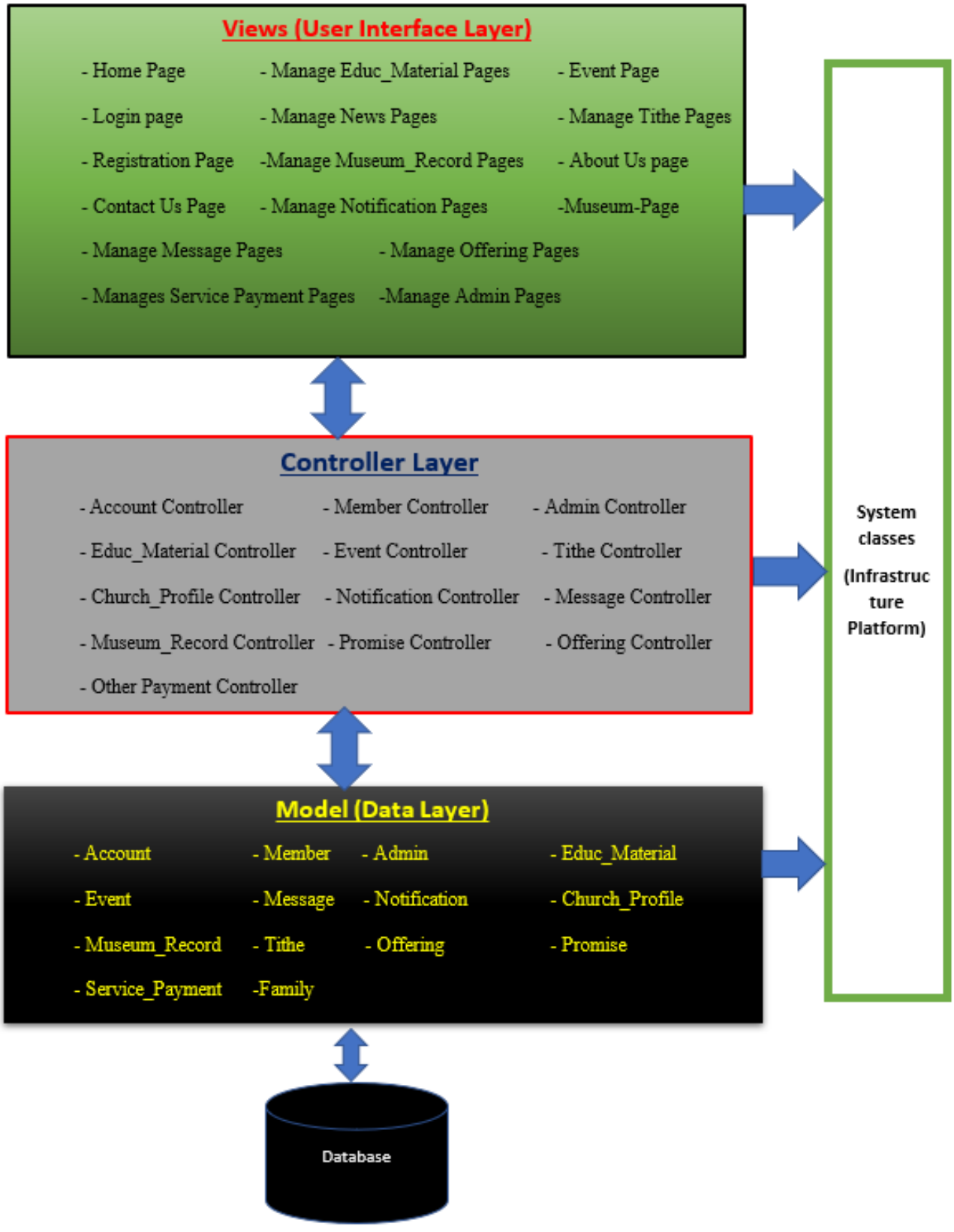
List of Controllers

- Account Controller
- Member Controller
- Admin Controller
- Educ_Material Controller
- Event Controller
- Museum_Record Controller
- Church_Profile Controller
- Notification Controller
- Message Controller
- Tithe Controller
- Promise Controller
- Offering Controller
- Service_Payment Controller

There are also system classes that implement software management and control functions that allow to operate and communicate in a computer environment, in addition to these three components (Model, View, Controller). System classes isolate applications from the operating system (OS) by covering OS-specific functionality and enhancing application portability. (Ambler, 2001)

The system's class type architecture is depicted in the diagram below.

Figure 5.2 Class type architecture



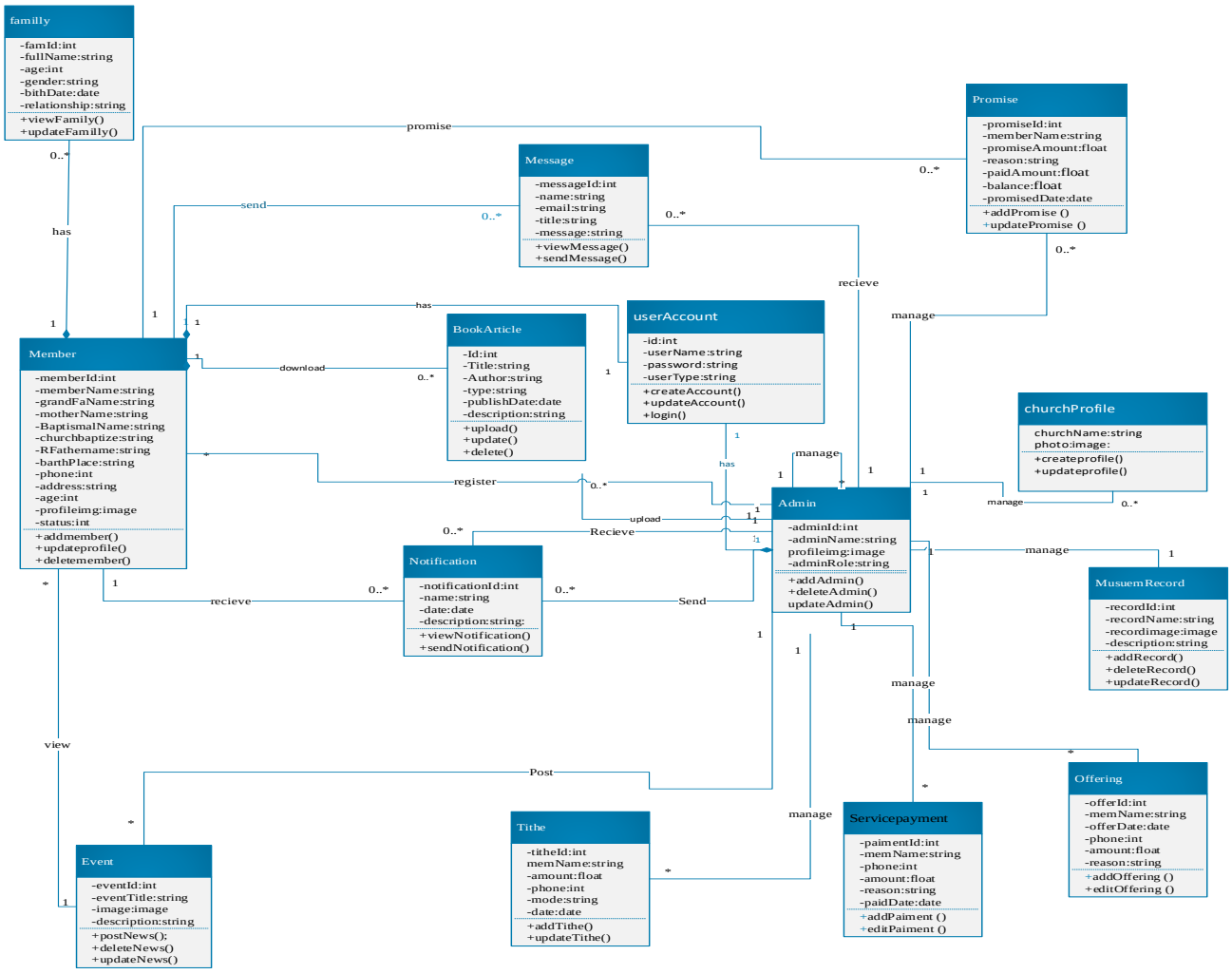
5.3 Design Class Modeling

5.3.1 Class Diagram

Class model shows the classes of the system, their interrelationships (including, inheritance, aggregation and association), and the operation and attributes of the classes. (Mayer, 1997)

A class is a representation of an object. To describe a class, we define its attributes and methods. Attributes are the information stored about an object while methods are what the object or the class does. The purpose of design-class modeling is to model static structure of the software that will be built. The only difference with the analysis version of class modeling is the focus on the solution instead of on the problem domain.

Figure 5. 3 Class diagram



5.3.2 Class Description

In the following tables we will describe the class attributes with their datatype and visibilities and methods with their visibilities and parameters.

Table 5.1 Member class description

Class	**member**	It includes all registered members which are approved by the church management
Attribute	memberId	Type: int Visibility: private used to hold id of the member
	memberName	Type: string Visibility: private used hold the name of the member
	userName	Type: string Visibility: private used to hold the username of the member
	password	Type: string Visibility: private used to hold the password of the member
	grandFaName	Type: string Visibility:private used to hold the member's grand fathers name
	motherName	Type: string Visibility:private used to hold member's mother name
	BaptismalName	Type: string Visibility:private used to hold the members Baptismal Name
	churchbaptize	Type: string Visibility:private used to hold the name of the church where the member Baptized
	RFathername	Type: string Visibility:private used to hold the members Repentance father name
	barthPlace	Type: string Visibility:private used to hold members birthplace
	phone	Type: int Visibility: private used to hold members phone number
	address	Type: string Visibility:private used to hold members address
	age	Type: int Visibility: private used to hold member's age
	Profileimg:	Type: image Visibility: private used to hold members profile image
	status	Type: int Visibility: private used to hold the status whether the member approved or not
Method	addMember()	Visibility: public Allows to add members to the system
	UpdateMember()	Visibility: public Allows to Updates the data of member

	deleteMember()	Visibility: public Allows to Deactivates the member

Table 5.2 Book class description

Class	**BookArticle**	It includes all Ethiopian orthodox tedwahido church books and articles which can be uploaded
Attribute	Id	Type: int Visibility: private used to hold the id of the book/article
	Title	Type: string Visibility: private used to hold the title of the book /article
	publishDate	Type: date Visibility: private used to hold the published date of the book/ article
	Author	Type: string Visibility: private used to hold the name of the author of the book/ article
	description	Type: string Visibility: private used to describe the book/ article
	type	Type: string Visibility: private used to identify is it book or article
Method	upload()	Visibility: public Allows to upload books/articles
	update()	Visibility: public Allows to Updates the book's/article's data
	delete ()	Visibility: public Allows to Deletes the book/article from the website

Table 5.3 Admin class description

Class	**Admin**	It includes all registered admins Super Admin, Membership manager, Finance manager, education manager, PR manager
Attribute	adminId	Type: int Visibility: private used to hold the id of the admin
	adminName	Type: string Visibility: private used to hold the name of the admin
	userName	Type: string Visibility: private used to hold user name of the admin
	password	Type: string Visibility: private used to hold the password of the admin
	profileimg	Type: image Visibility: private used to hold image profile of the admin
	adminRole	Type: string

		Visibility: private used to hold the role of the admin
Method	addAdmin()	Visibility: public Allows to Adds admin to system
	editAdmin()	Visibility: public Allows to Updates the data of admins
	deleteAdmin()	Visibility: public Allows to deactivate admin

Table 5.4 MusuemRecord class description

Class	**MusuemRecord**	It includes some of the exhibits exist in the museum which are allowed to upload
Attribute	recordId	Type: int Visibility: private used to hold the id of the museum record
	recordName	Type: string Visibility: private used to hold the name of the museum record
	recordimage	Type: image Visibility: private used to hold the image of the record
	description	Type: string Visibility: private used to hold the description of the record
Method	addaeRecord ()	Visibility: public Allows to Adds record to the system
	updateRecord ()	Visibility: public Allows to Updates the data of the record
	deleteRecord ()	Visibility: public Allows to Deletes the record from system

Table 5.5 Promise class description

Class	**Promise**	It includes all the types of incomes to the church from the member when he/she promise to pay an amount.
Attribute	promiseId	Type: int Visibility: private used to hold the id of the promise
	memberName	Type: string Visibility: private used to hold the name of the member who promise
	promiseAmount	Type: float Visibility: private used to hold the promised amount
	reason	Type: string Visibility: private used to hold the reason why he/she promise

	paidAmount	Type: float Visibility: private used to hold how much he paid
	balance	Type: float Visibility: private used to hold the remained unpaid
	promiseddate	Type: date Visibility: private used to hold the promised date to pay
Method	addPromise ()	Visibility: public Allows to adds promise by the member
	updatePromise ()	Visibility: public Allows to Updates the member promised

Table 5.6 Offering class description

Class	Offering	It includes all the types of incomes to the church from the member in voluntary for some purposes.
Attribute	offerId	Type: int Visibility: private used to hold the id of offering
	memName	Type: string Visibility: private used to hold the name of the member who offered
	reason	Type: string Visibility: private used to hold the reason why the member offered
	amount	Type: float Visibility: private used to hold the amount offered
	phone	Type: int Visibility: private used to hold the phone number of the who offered
	offerDate	Type: date Visibility: private used to hold dtae offered
Method	addOffering ()	Visibility: public Allows to Member adds offering
	editOffering ()	Visibility: public Allows to Updates the offering information

Table 5.7 Servicepayment class description

Class	Servicepayment	It includes all the types of incomes to the church from the member from weeding,kerestina, and Tezkar
Attribute	paimentId	Type: int Visibility: private used to hold id of the payment
	memName	Type: string Visibility: private used to hold the name of the member
	reason	Type: string Visibility: private used to hold reason why he/she pay

	amount	Type: float Visibility: private used to hold the amount paid
	phone	Type: int Visibility: private used to hold the phone number of the who paid
	paidDate	Type: date Visibility: private used to hold the date paid
Method	addPaiment ()	Visibility: public Allows to Adds the payment
	editpaiment()	Visibility: public Allows to Updates the payment data

5.8 Tithe class description

Class	Tithe	It includes all the types of incomes to the church from the member in the form of *Asratbekurat*
Attribute	titheId	Type: int Visibility: private used to hold the id of the tithes
	memName	Type: string Visibility: private used to hold the name of the member who paid
	mode	Type: string Visibility: private used to hold whither it is in check of other payment mode
	amount	Type: float Visibility: private used to hold amount paid
	phone	Type: int Visibility: private used to hold phone of the member paid
	paidDate	Type: date Visibility: private used to hold the date paid
Method	addTithe ()	Visibility: public Allows to Adds tithe data
	editTithe()	Visibility: public Allows to Updates the Tithes data

Table 5.9 Message class description

Class	**Message**	It includes all the messages sent from members or nonmembers to super admin
Attribute	messageId	Type: int Visibility: private used to hold the id of the message
	name	Type: string Visibility: private used to hold the name who send the message
	email	Type: string Visibility: private used to hold the email of the sender
	Title	Type:string

		Visibility:private used to hold the title of the message
	message	Type:string Visibility:private used to hold the content of the message
	+sendMessage()	Visibility: public Allows to Send message by members and non-members
	+viewMessage()	Visibility: public Allows to See the messages sent members and non-members

Table 5.10 Notification class description

Class	**Notification**	It includes all the notifications sent from super admin to members, Membership manager, Finance manager, education manager and PR manager
Attribute	notificationId	Type: int Visibility: private used to hold the id of the notification
	name	Type: string Visibility: private used to hold the name of the receiver
	Date	Type:date Visibility:private used to hold date sent
	description	Type:string Visibility:private used to hold the content of the text
	+sendNotification()	Visibility: public Allows to Sends the notifications
	+viewNotification()	Visibility: public Allows to Views the notifications

Table 5.11 Familly class description

Class	**Familly**	It includes all the family members which a member has including his wife
Attribute	famId	Type: int Visibility: private used to hold the id of the family
	fullName	Type: string Visibility: private used to hold the full name of the family member
	age	Type: int Visibility: private used to hold the age of the family member
	gender	Type:string

		Visibility:private used to hold the gender of the family member
	relationship	Type:string Visibility:private used to hold the relationship of the family member whither she is wife or daughter
	birthDate	Type:date Visibility:private used to hold the birthdate of the family member
	addFamilly()	Visibility: public Allows to Adds family member
	+viewFamilly()	Visibility: public Allows to See family members
	+updateFamilly()	Visibility: public Allows to Update family member's data

Table 5.12 Event class description

Class	**Event**	It includes all the upcoming and past events of the church
Attribute	eventId	Type: int Visibility: private used to hold the id of the event
	eventTittle	Type: string Visibility: private used to hold the title of the event
	image	Type:image Visibility:private used to hold the image of the event
	description	Type:string Visibility:private used to hold the description of the event
	+postEvent ()	Visibility: public Allows to Post new events
	+updateevent ()	Visibility: public Allows to Update posted events
	deleteevent()	Visibility: public Allows to Cancel events

Table 5.13 churchprofile class description

Class	**churchProfile**	It is the profile of the church
Attribute	churchName	Type: string Visibility: private used to hold the name the church
	photo	Type: image Visibility: private used to hold the image the church

method	Createprofile()	visibity: pulic Allows to create the church's profile
	Updateprofile()	visibity: pulic Allows to update church's profile

Table 5.14 userAccount class description

Class	**userAccount**	It is the account of the user
Attribute	id	Type: string Visibility: private used to id of the account
	userName	Type: string Visibility: private used to hold the user name of the user
	password	Type: string Visibility: private used to hold the password of the user
	userType	Type: string Visibility: private used to hold the type of the user is he admin or member
method	createAccount()	visibity: pulic Allows to create user account
	updateAccount()	visibity: pulic Allows to updates the user account
	Login()	visibity: pulic Allows the user login to the system.

5.4 Collaboration Diagram

Collaboration diagrams focus on the structural aspects of objects interaction. Collaboration diagrams show the message flow between objects in an Object-Oriented application, and also imply the basic associations (relationships) between classes. Both the sequence and the collaboration diagrams represent the same information but differently.

Instead of showing the flow of messages, it depicts the architecture of the object residing in the system as it is based on object-oriented programming.

Figure 5.4 Registration Collaboration Diagram

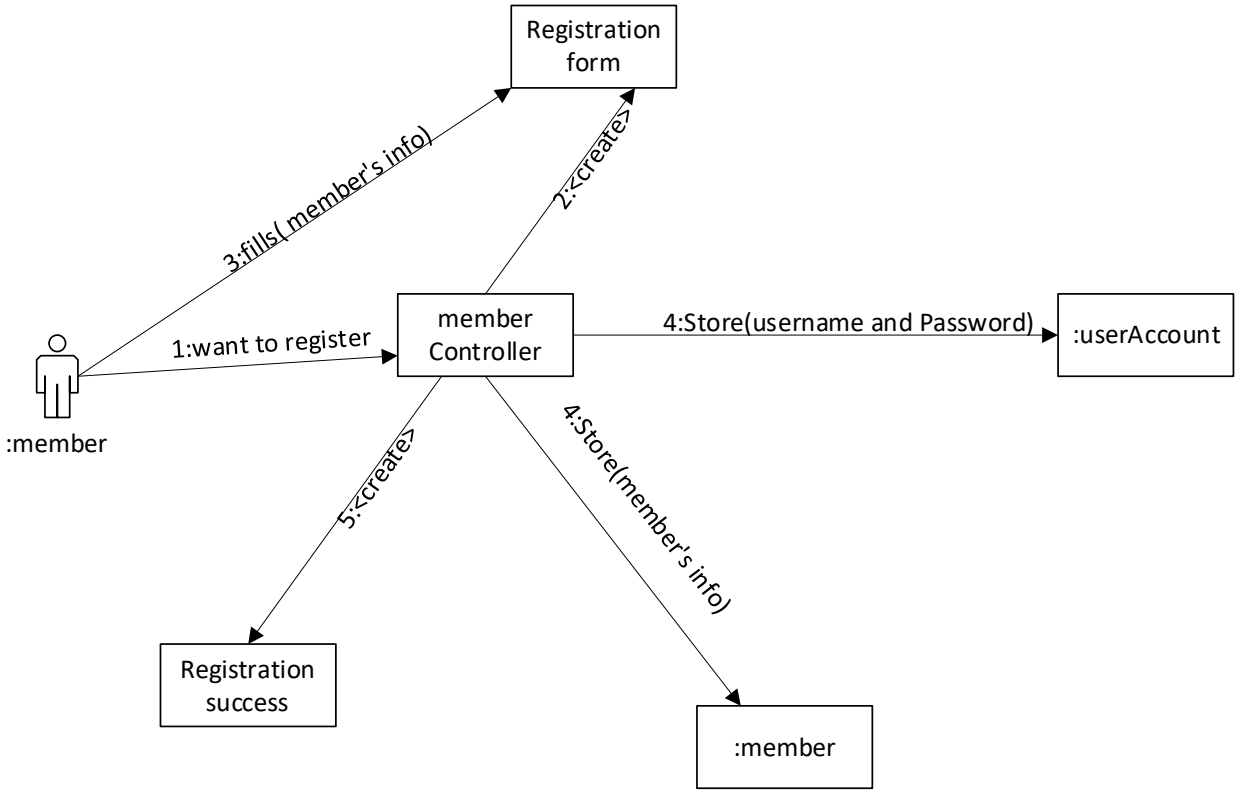


Figure 5.5 Notification Collaboration Diagram

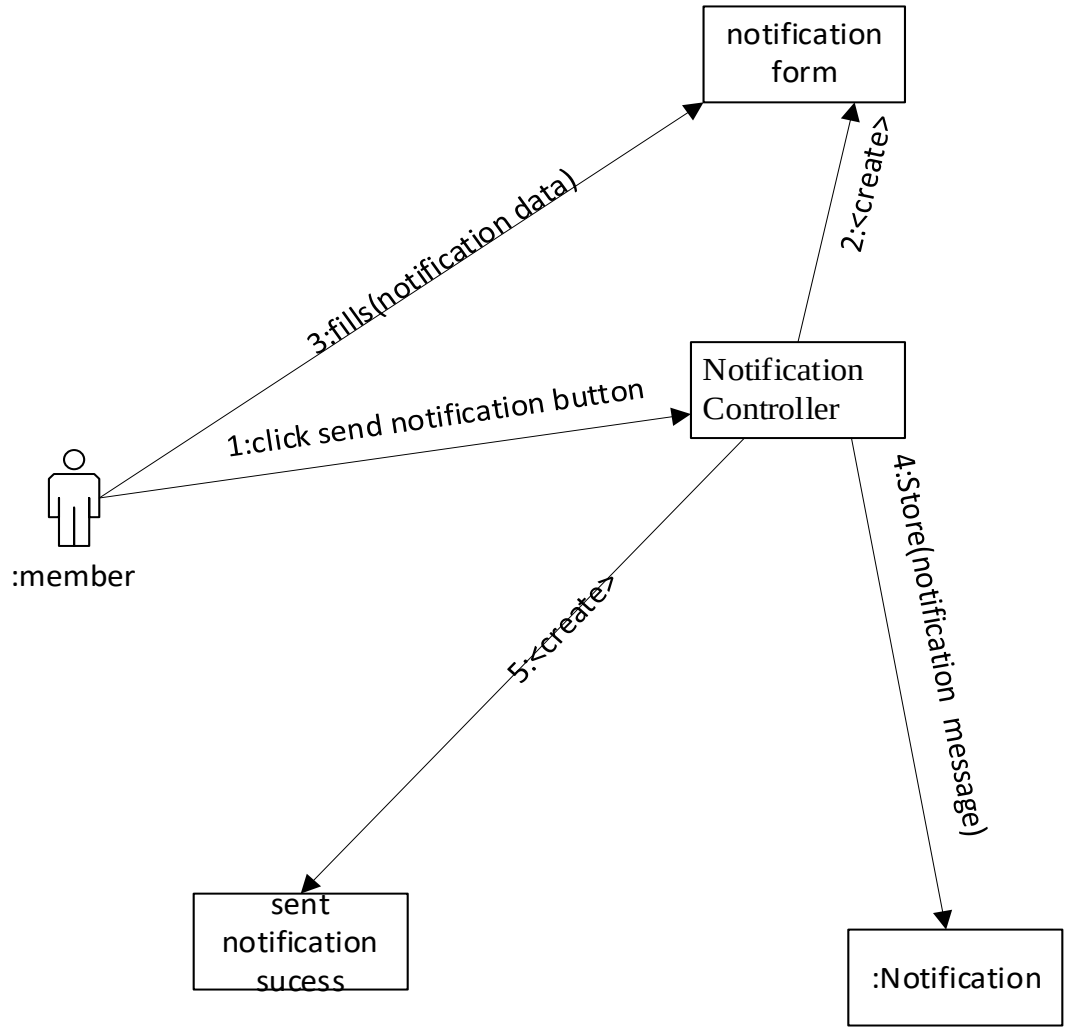
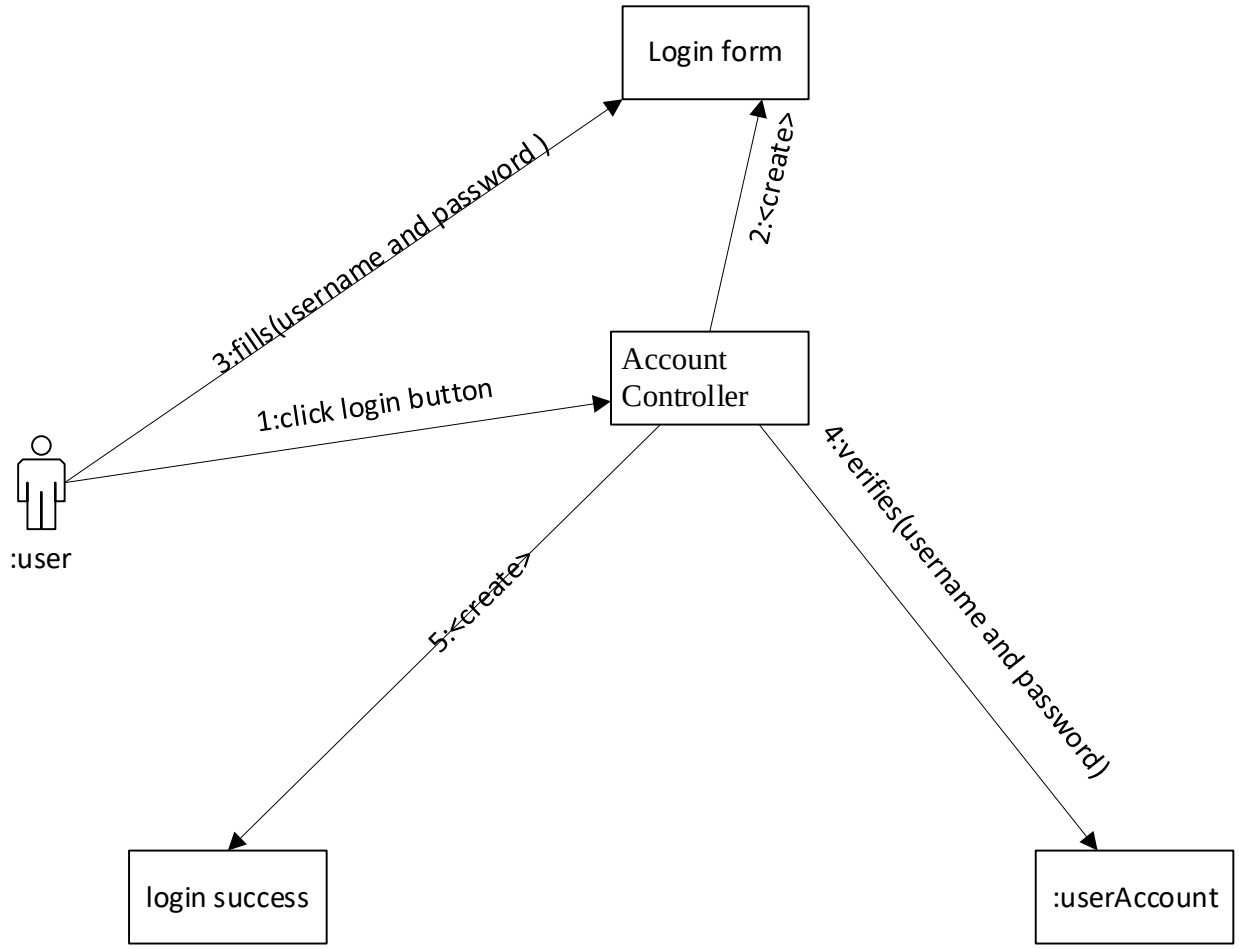


Figure 5.6 Login Collaboration Diagram



References

Ambler, S. W. (2001). Determining How to Build Your System: Object-Oriented Design. In S. W. Ambler, *The Object Primer Second Edition* (p. 254). Cambridge: Cambridge University Press.

Horstmann, C. S. (2005). *Object-Oriented Design and Patterns, 2nd Edition.* United States of America: Bruce Spatz .

Jackson, P. (2022, January 8). *PHP MVC Framework Tutorial: CodeIgniter Example*. Retrieved from Guru99: https://www.guru99.com/php-mvc-frameworks.html