



Universiti  
Malaysia  
PAHANG

Engineering • Technology • Creativity

# SDD Document SEM II 20222023

## e-Munakahat System (EMUN)

### Group Name

1. Muhammad Amir Bin Mohamed Ali [CB21060]
2. Ahmad Suffian Bin Md Noor Suhaime [CB21137]
3. Chua Kian Pheng [CB21106]
4. Nik Alia Syafiqah Binti Nik Azuri [CB21035]
5. Shammene A/P Muneesvaran [CB21018]



Quantum Corp

# TABLE OF CONTENTS

1. INTRODUCTION	3
1.1 Purpose	3
1.2 System Identification	4
1.3 System Overview	6
1.4 Referenced Document	9
2. DATA DESIGN	10
2.1 Entity Relationship Diagram (ERD)	10
2.2 Data Dictionary	11
3. GENERAL ARCHITECTURE	16
3.1 Layered Architecture	16
3.1.1 Application Layer	16
3.1.2 Business Services Layer	21
3.1.3 Middleware Layer	25
3.2 MVC Package Relationship	26
4. DETAIL DESIGN	27
4.1 manageProfile [SDD-REQ-100]	27
4.2 auth [SDD-REQ-200]	30
4.3 Controller [SDD-REQ-1000]	36
4.4 Model [SDD-REQ-1100]	45
5. REQUIREMENT TRACEABILITY	47
5.1 manageProfile [UC-101-EMUN-001]	47
5.2 auth [UC-102-EMUN-001]	47
APPENDIX	48
APPENDIX B	48
APPENDIX C1	49
APPENDIX C2	51
APPENDIX C3	52
APPENDIX C4	53

# 1. INTRODUCTION

## 1.1 Purpose

This software design document (SDD) serves as a blueprint for the development of the e-Munakahat System (EMUN). It outlines the software design process, including the software's architecture, modules, interfaces, and data structures. The primary purpose of an SDD is to ensure that the software development team and stakeholders understand the design of the software application and how it will function.

The software design document (SDD) is a communication tool between the development team and stakeholders, ensuring everyone is on the same page regarding the software's design and implementation. It will also help the stakeholders to understand any limitations or assumptions that may impact the system's performance or usability, which will help the stakeholders to make informed decisions and better plan for the project. This will also help to ensure that the final product meets the needs of the users and the jurisdiction.

## 1.2 System Identification

The Software Design Document (SDD) belongs to the “e-Munakahat System” (EMUN).

Table 1.1 Document Identity.

System title	e-Munakahat System											
System abbreviation	EMUN											
System identification number	SDD_EMUN_QC_2023											
Subsystem Title	IkatanCinta											
Subsystem Abbreviation	IC											
Package ID	<div>SDD-REQ-100</div> <div>Meanings for terms use:</div> <table><tr><td>SDD</td><td>Software Design Document</td></tr><tr><td>REQ</td><td>Requirement</td></tr><tr><td>100</td><td>Package number</td></tr></table>		SDD	Software Design Document	REQ	Requirement	100	Package number				
SDD	Software Design Document											
REQ	Requirement											
100	Package number											
Use Case ID	<div>UC101-EMUN-001</div> <div>Meanings for terms used:</div> <table><tr><td>UC</td><td>Use Case</td></tr><tr><td>1</td><td>Number of the system module</td></tr><tr><td>01</td><td>Number of use case within a module in the subsystem</td></tr><tr><td>EMUN</td><td>e-Munakahat System (System name)</td></tr><tr><td>001</td><td>Document release number</td></tr></table>		UC	Use Case	1	Number of the system module	01	Number of use case within a module in the subsystem	EMUN	e-Munakahat System (System name)	001	Document release number
UC	Use Case											
1	Number of the system module											
01	Number of use case within a module in the subsystem											
EMUN	e-Munakahat System (System name)											
001	Document release number											
Requirement Traceability ID	<div>UC101-EMUN-001</div> <div>Meanings for terms used:</div> <table><tr><td>UC</td><td>Use Case</td></tr><tr><td>1</td><td>Number of the system module</td></tr><tr><td>01</td><td>Number of use case within a module in the system</td></tr><tr><td>EMUN</td><td>e-Munakahat System (System name)</td></tr><tr><td>001</td><td>Document release number</td></tr></table>		UC	Use Case	1	Number of the system module	01	Number of use case within a module in the system	EMUN	e-Munakahat System (System name)	001	Document release number
UC	Use Case											
1	Number of the system module											
01	Number of use case within a module in the system											
EMUN	e-Munakahat System (System name)											
001	Document release number											
Document Identification System	SDD_EMUN_IC_2023_V1.0											

Symbol/Number	Meaning
EMUN	It represents the system name which is e-Munakahat System (EMUN)
SDD	It represents the software design document (SDD)
QC	It represents the company name which is Quantum Corp SDN. BHD.
2023	It represents the year where the system has been released.
V1.0	V represent the word “Version” of the system while 1.0 is the general version of the system. The number will increase by 0.1 if there any updates or bugs fixed.

## 1.3 System Overview

Our system is a web-based system to support the Pahang Wedding management system. The purpose of the system is to provide a solution for managing wedding registration and operations efficiently. Users can use this system for marriage registration and information retrieval. It also eases the government to collect granular data on weddings in Pahang and store the information for future use. This system is handled by the Quantum Corp company. This system contains various functionalities such as user registration, marriage application, marriage registration, marriage preparation course, proof of payment, request for marriage, marriage card, marriage consultation, and special incentives for the bride and groom.

There are five modules in this system which are:

1. Registration and handle user profile

The initial stage of using the system is applicant registration, which is required for applicants to enter the system. It contains private information such as an Identity Card number, phone number, email address, and password. Since registration is based on the user's IC number, a unique number with no duplicates, each user has just one account for the system. Once the applicant's registration is successful, they can log in to access the system's contents after completing the registration process. The admin can register new staff through the admin dashboard. Once registration for staff is successful, they will be granted access to the system. The system also allows users (applicants & staff) to modify their profile details to update their information. In case the applicant has forgotten their password, they will be able to reset their password. Apart from that, the admin can update staff and applicant details through the admin dashboard.

2. Attend the marriage preparation course with proof of payment and to request a marriage.

After users register and login into the account, the users can proceed to the next step where they can apply for the marriage registration course. As a marriage registration course is an important step for the e-Munakahat system will update the name list of applicants. The system will show if the user has already paid for the course or not. So, then the staff can print the proof of payment and send the receipt to the applicants. The system shall allow the admin to manage the marriage preparation course details such as date, time, and where the course will be conducted. While the staff will handle the things that are related to the user or applicants. Approve the applications in the system and print the applicants' name list for the course for every session. If the applicant presents on the course day, the staff can approve their marriage course certificate. After the course ends, the applicants will be represented with the certificate. The staff can update the applicant's name from the system for obtaining the marriage course certificate. Next step for the applicant is request their marriage from E-Munakahat system.

3. An application to register a marriage within or outside the country, as well as a voluntary marriage, and to produce the marriage card or certificate with proof of payment.

Marriage registration is split into two processes which are voluntary and authorized registration. User needs to pass a pre-marriage course to be able to register their marriage. Voluntary registration is for the older generation who live far away from town and did not register when they married. For authorized registration, the user must prepare various forms and information in which the template is provided in the system. After staff from JAIP has approved the marriage registration, it will notify the user so that they know their marriage registration has been approved.

4. Register for a marriage consultation with a service advisor.

A consultation module is a module where the user can make an appointment to seek advice or to get a divorce. This module requires the users to enter their personal information and also their spouse information including their marriage information and upload related documents as proof. Staff can view the application made by the user and will decide whether to approve the appointment and responsible to assign consultant and slot to the consultation session. Staff are also responsible for managing the list of consultants and their information in this module. The staff can edit, add, and delete the consultants' information in the system.

## 5. Application for special incentive for the bride and groom

The marriage registration system features a special incentive module that enables users to apply for incentives, provided they meet the necessary prerequisites. This module allows individuals to input their personal information and upload necessary documents, making the process more streamlined. Additionally, the staff side of the platform allows for the review of applications and ensures that the process is efficient and user-friendly. The special incentive module in the marriage registration system also includes a notification feature that keeps applicants informed about the status of their applications. Once the staff reviews the application, the applicant will receive a notification, either confirming their approval or outlining the reasons for the denial. This ensures that applicants are aware of the status of their application in a timely and efficient manner."

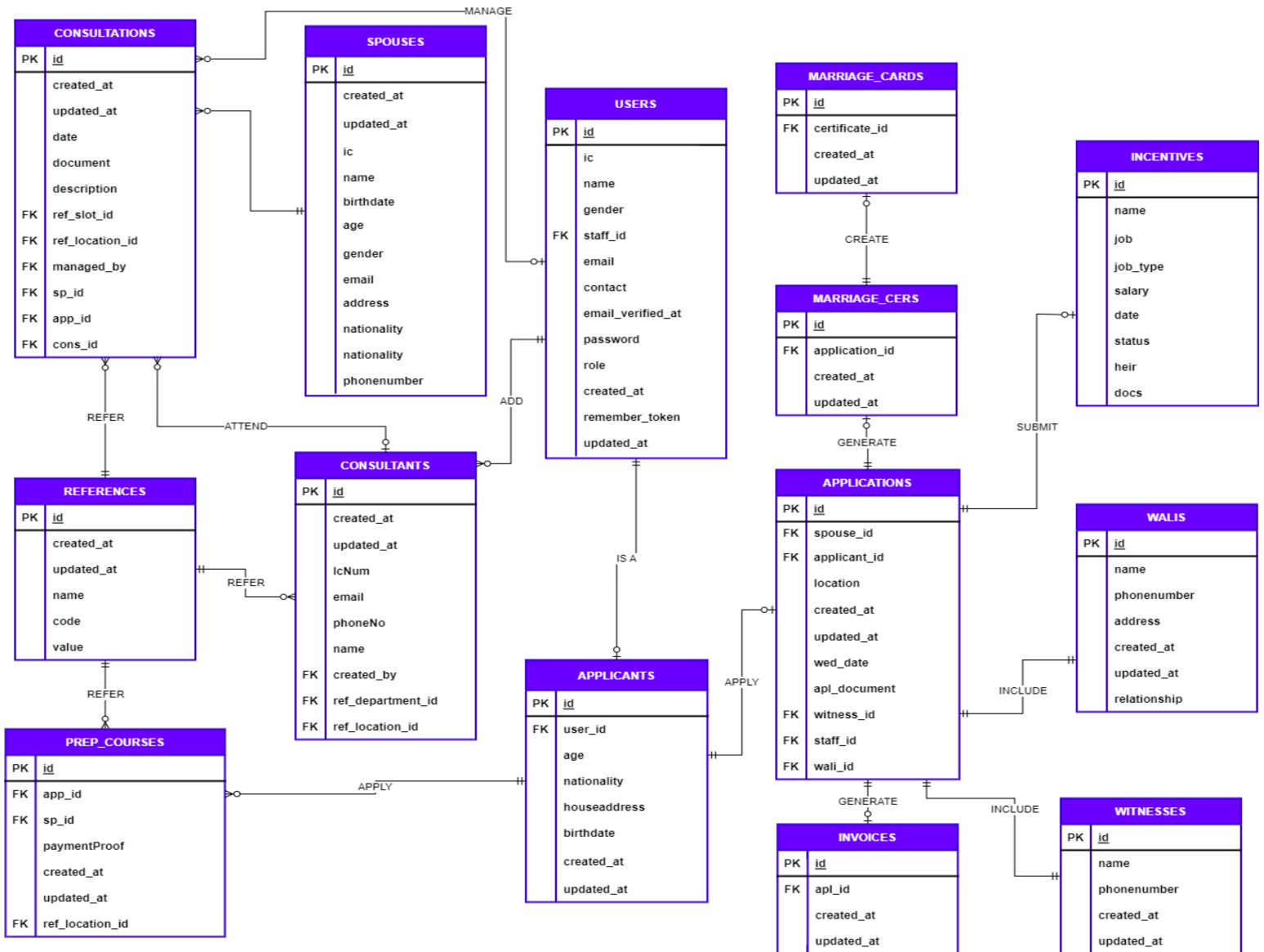


## 1.4 Referenced Document

1. Azma, B. A., et al (2023) BCS2243 Final Assessment Question Sem II 20222023.
2. Amir.A, et al (2023) SOFTWARE REQUIREMENT SPECIFICATION (SRS).
3. Layered Architecture. (2021, November 11). Retrieved May 2, 2023, from <https://www.baeldung.com/cs/layered-architecture>.
4. How to build a simple PHP MVC framework. (2021, May 30). Retrieved May 28, 2023, from <https://www.giuseppemaccario.com/how-to-build-a-simple-php-mvc-framework/>.
5. Simple PHP MVC Framework Example. (2023, May 26). Retrieved May 28, 2023, from <https://phpflow.com/php/simple-mvc-architecture-example-in-php/>.
6. What Is Middleware? Definition, Architecture, and Best Practices. (2022, February 18). Retrieved May 28, 2023, from <https://www.spiceworks.com/tech/cloud/articles/what-is-middleware/>.

## 2. DATA DESIGN

## 2.1 Entity Relationship Diagram (ERD)



## 2.2 Data Dictionary

### 2.2.1 USERS

Field Name	Description	Data Type	Constraint
id	User's ID	BIGINT	PK
ic	User's IC	VARCHAR (12)	NOT NULL, UNIQUE
name	User's name	VARCHAR (255)	NOT NULL
gender	User's gender	VARCHAR (8)	NOT NULL
staff_id	Staff's ID	VARCHAR (10)	
email	User's email	VARCHAR (255)	NOT NULL, UNIQUE
contact	User's contact	VARCHAR (15)	NOT NULL
email_verified_at	User's email verification dates	TIMESTAMP	NOT NULL
password	User's password	VARCHAR (255)	NOT NULL
role	User's role	TINYINT	NOT NULL
remember_token	Users remember token	VARCHAR (100)	
created_at	User's created date	TIMESTAMP	NOT NULL
updated_at	User's updated date	TIMESTAMP	

### 2.2.2 APPLICANTS

Field Name	Description	Data Type	Constraint
id	Applicant's ID	BIGINT(20)	PK
user_id	User ID	BIGINT(20)	Not null
age	Applicant's age	INT(10)	
nationality	Applicant's nationality	VARCHAR(255)	
houseaddress	Applicant's house address	VARCHAR (255)	
birthdate	Applicant's birthdate	DATE	
created_at	Applicant's created time	TIMESTAMP	Not null
updated_at	Applicant's updated time	TIMESTAMP	

### 2.2.3 CONSULTATIONS

Field Name	Description	Data Type	Constraint
id	Consultation ID number	BIGINT (20)	PK
created_id	Consultation created date	TIMESTAMP	NOT NULL
updated_at	Consultation update date	TIMESTAMP	
date	Consultation appointment date	DATE	NOT NULL
ref_slot_id	Consultation slot	BIGINT (20)	NOT NULL
description	Consultation description	TEXT	NOT NULL
document	Consultation document file path	VARCHAR (255)	NOT NULL
ref_location_id	Consultation location	BIGINT (20)	NOT NULL
managed_by	Staff who managed the consultation	BIGINT (20)	
sp_id	Spouse ID	BIGINT (20)	NOT NULL
app_id	Applicant ID	BIGINT (20)	NOT NULL
cons_id	Consultant ID	BIGINT (20)	

### 2.2.4 CONSULTANTS

Field Name	Description	Data Type	Constraint
id	Consultation's id	BIGINT (20)	PK
created_at	Consultation's created time	TIMESTAMP	NOT NULL
updated_at	Consultation 's updated time	TIMESTAMP	
lcNum	Consultant's IC number	VARCHAR (255)	NOT NULL
name	Consultant's name	VARCHAR (255)	NOT NULL
email	Consultant's email address	VARCHAR (255)	NOT NULL
ref_department_id	Consultant's departments	BIGINT (20)	FK, NOT NULL
ref_location_id	Consultant's working location	BIGINT (20)	FK, NOT NULL
phoneNo	Consultant's phone number	VARCHAR (255)	NOT NULL
created_by	Staff who register the consultant	BIGINT (20)	FK, NOT NULL

## 2.2.5 APPLICATIONS

Field Name	Description	Data Type	Constraint
id	Application's ID	INT	PK
spouse_id	Spouse ID	INT	FK
applicant_id	Applicant ID	INT	FK
location	Application location	VARCHAR (50)	NOT NULL
created_at	Application's created time	TIMESTAMP	NOT NULL
updated_at	Application 's updated time	TIMESTAMP	
wed_date	Wedding date	DATE (50)	NOT NULL
apl_document	Application's document	DOCUMENT	
witness_id	Witness ID	INT	FK
staff_id	Staff ID	INT	FK
wali_id	Wali ID	INT	FK

## 2.2.6 PREP\_COURSES

Field Name	Description	Data Type	Constraint
id	Course ID	INT	PK
app_id	Applicant ID	INT	FK
sp_id	Spouse ID	INT	FK
paymentProof	Course payment proof receipt	DOCUMENT	
created_at	Course's created time	TIMESTAMP	NOT NULL
updated_at	Course 's updated time	TIMESTAMP	
ref_location_id	Marriage preparation course location	BIGINT (20)	FK, NOT NULL

### 2.2.7 SPOUSES

Field Name	Description	Data Type	Constraint
id	Spouse's ID	BIGINT (255)	PK
created_at	Spouse's created time	TIMESTAMP	NOT NULL
updated_at	Spouse's updated times	TIMESTAMP	
ic	Spouse's IC	VARCHAR (12)	NOT NULL
name	Spouse's name	VARCHAR (255)	NOT NULL
birthdate	Spouse's birthdate	DATE	NOT NULL
age	Spouse's age	INT	NOT NULL
gender	Spouse's gender	VARCHAR (8)	NOT NULL
nationality	Spouse's nationality	VARCHAR (15)	NOT NULL
address	Spouse's address	VARCHAR (255)	NOT NULL
email	Spouse's email address	VARCHAR (255)	NOT NULL
phonenumber	Spouse's phone number	VARCHAR (15)	NOT NULL

### 2.2.8 INVOICE

Field Name	Description	Data Type	Constraint
inv_Num	Invoice Number	INT	PK
apl_Num	Application Number	INT	FK
inv_date	Invoice Date	DATE (50)	

### 2.2.9 INCENTIVE

Field Name	Description	Data Type	Constraint
<u>id</u>	Incentive id	INT	PK
name	Incentive Applicant name	STRING	FK
job	Job	VARCHAR (255)	
job_type	Job Type	VARCHAR (255)	
salary	Salary	DOUBLE	
date	Incentive Application Date	DATE (50)	
status	Approve status	VARCHAR(255)	
heir	Heir	VARCHAR(255)	
docs	Incentive Documents	BLOB	
created_at	Incentive created time	TIMESTAMP	NOT NULL
updated_at	Incentive updated time	TIMESTAMP	

### 2.2.10 WALI

Field Name	Description	Data Type	Constraint
wali_icNuM	Wali's IC number	VARCHAR(60)	PK
wali_name	Wali's name	VARCHAR(60)	
wali_phonenumber	Wali's phone number	VARCHAR(60)	
wali_address	Wali's address	VARCHAR(200)	
wali_relationship	Wali's relationship	VARCHAR(60)	

### 2.2.11 WITNESS

Field Name	Description	Data Type	Constraint
wit_icNuM	Witness's IC number	VARCHAR(60)	PK
wit_name	Witness's name	VARCHAR(60)	
wit_phonenumber	Witness's phone number	VARCHAR(60)	
wit_address	Witness's address	VARCHAR(200)	

### 2.2.12 MARRIAGE\_CER

Field Name	Description	Data Type	Constraint
CER_Num	Certificate's number	INT	PK
apl_Num	Application's number	INT	FK
cer_issue_date	Certificate's issue date	DATE	

### 2.2.13 MARRIAGE\_CARD

Field Name	Description	Data Type	Constraint
card_Num	Card's number	INT	PK
cer_Num	Certificate's number	INT	FK
card_issue_date	Card's issue date	DATE	

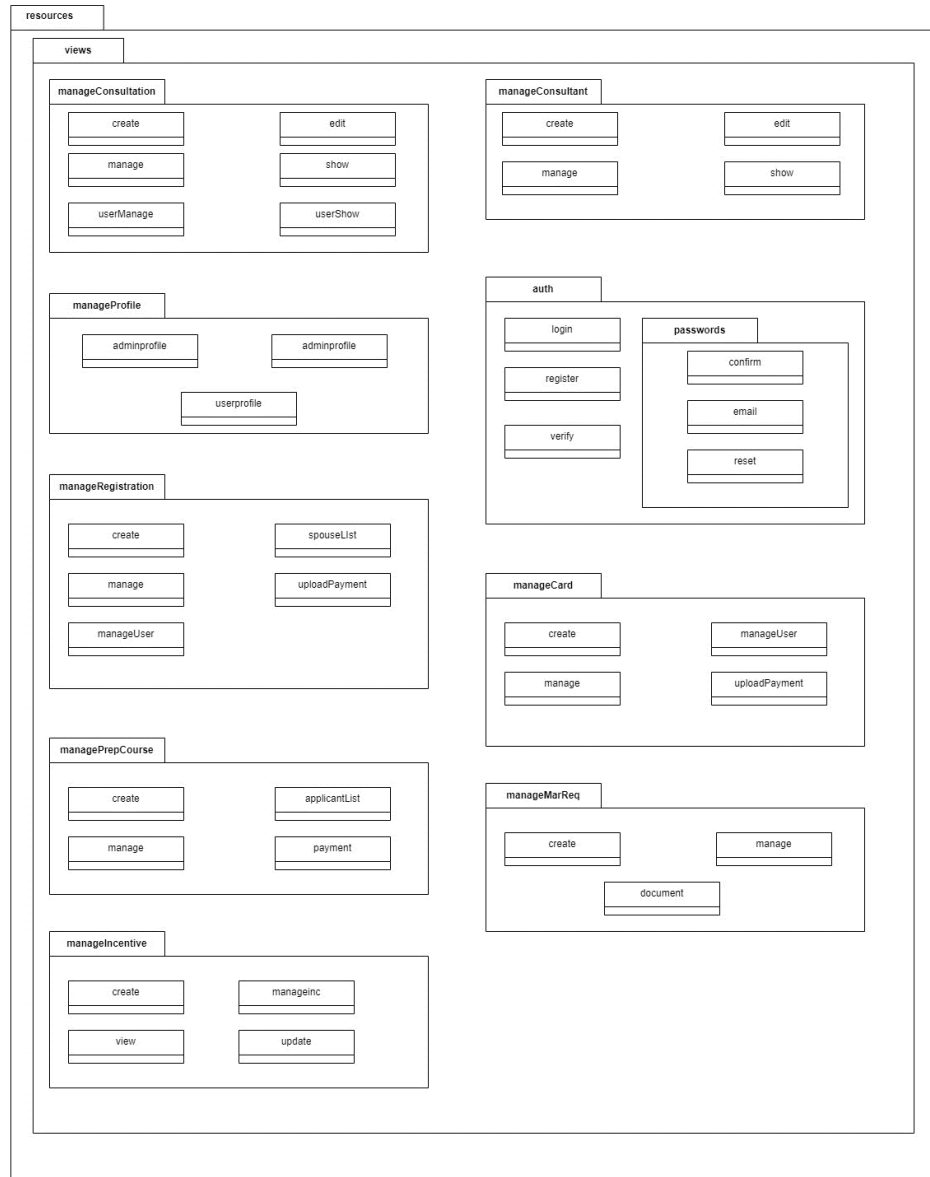
### 2.2.14 REFERENCES

Field Name	Description	Data Type	Constraint
id	References id	BIGINT (20)	PK
created_at	References created date	TIMESTAMP	
updated_at	References updated date	TIMESTAMP	
name	References name	VARCHAR (255)	NOT NULL
code	References code	VARCHAR (255)	NOT NULL
value	References value	VARCHAR (255)	NOT NULL

## 3. GENERAL ARCHITECTURE

### 3.1 Layered Architecture

#### 3.1.1 Application Layer





### 3.1.1.1 Manage Profile [SDD-REQ-100]

Class Name	Description
adminprofile	A view class for admin to register staff in the EMUN system
staffprofile	A view class for staff to manage their profile in the EMUN system
userprofile	A view class for user to manage their profile in the EMUN system

### 3.1.1.2 Auth [SDD-REQ-200]

Class Name	Description
login	A view class for user, staff and admin login into the EMUN system
register	A view class for user to register into the EMUN system
verify	A view class for user to verify the account in the EMUN system
confirm	A view class for user to confirm the account in the EMUN system
email	A view class for user to enter the email to reset password in the EMUN system
reset	A view class for user to reset password in the EMUN system.

### 3.1.1.3 Manage Preparation Course [SDD-REQ-300]

Class Name	Description
create	A view class for applicant to register the marriage preparation course by fill in the form
manage	A view class for applicant display the information regarding to marriage preparation course
applicantList	A view class for JAIP staff to see the list of applicants that apply marriage preparation course
payment	A view class for applicant to submit proof of payment

#### 3.1.1.4 Manage Marriage Request [SDD-REQ-400]

Class Name	Description
create	A view class for applicant to register the marriage application by fill in the form
manage	A view class for applicant edit their information
document	A view class for applicant to submit document

#### 3.1.1.5 Manage Registration [SDD-REQ-500]

Class Name	Description
spouseList	A view class that displays the spouse's information and allow the user to search, edit, delete, print and submit
create	A view class that allows the user to create a marriage registration application
manage	A view class that displays the user's marriage registration application and allow staff to accept or reject
manageUser	A view class that displays the marriage registration application
uploadPayment	A view class that displays payment receipt request and allow the user to upload their receipt

#### 3.1.1.6 Manage Card [SDD-REQ-600]

Class Name	Description
create	A view class allow the user to request for the marriage card
manage	A view class that displays the user's marriage card application and allow staff to accept or reject
manageUser	A view class that displays the marriage card application
uploadPayment	A view class that displays payment receipt request and allow the user to upload their receipt

### 3.1.1.7 Manage consultation [SDD-REQ-700]

Class Name	Description
create	A view class to display form to allow the user to fill information details
show	A view class to allow JAIP staff to view the information of the application
edit	A view class to allow JAIP staff to fill user appointment detail and approve their application
userManage	A view class to allow user to check the list of consultation applications made
userShow	A view class to allow user to view the information of the application made
manage	A view class to display the list of application and allow JAIP staff to manage the consultation application

### 3.1.1.8 Manage consultant [SDD-REQ-800]

Class Name	Description
edit	A view class to allow JAIP staff to update the consultant's information
manage	A view class to allow JAIP staff to manage the list of consultants
create	A view class to allow JAIP staff to enter information of new consultant
show	A class that handles the consultant information

### 3.1.1.9 Manage Incentive [SDD-REQ-900]

Class Name	Description
create	A view class that displays the apply incentive page upon request which includes an application form
view	A view class that displays the user's incentive application details, hence information in a specified format
update	A view class that displays the user's information and allows the user to edit their information
delete	A view class that displays the user's information and allows the user to delete their information

## 3.1.2 Business Services Layer

### 3.1.2.1 Controller [SDD-REQ-1000]



Class Name	Description
ApplicantController	Handles the logic and actions related to managing applicants in the application. Contains methods for creating, updating, and deleting applicant records, as well as retrieving applicant information.
ApplicationController	Manages the application-related functionality within the application. Contains methods for handling the creation, editing, deletion, and retrieval of application records.
ConsultantController	Responsible for handling the actions and operations related to managing consultants in the application. Contains methods for creating, updating, deleting, and retrieving consultant information.
ConsultationController	Handles the actions and operations associated with managing consultations in the application. Contains methods for scheduling consultations, updating consultation details, and retrieving consultation information.
IncentiveController	Manages the logic and actions related to incentives in the application. Contains methods for creating, updating, deleting, and retrieving incentive records.
InvoiceController	Handles the actions and operations related to managing invoices within the application. Contains methods for creating, updating, deleting, and retrieving invoice records.
Marriage_cardController	Manages the functionality associated with marriage cards within the application. Contains methods for creating, updating, deleting, and retrieving marriage card records.
Marriage_cerController	Handles the actions and operations related to managing marriage ceremonies in the application. Contains methods for creating, updating, deleting, and retrieving marriage ceremony records.
Prep_courseController	Responsible for managing the functionality related to preparation courses in the application. Contains methods for creating, updating, deleting, and retrieving preparation course records.

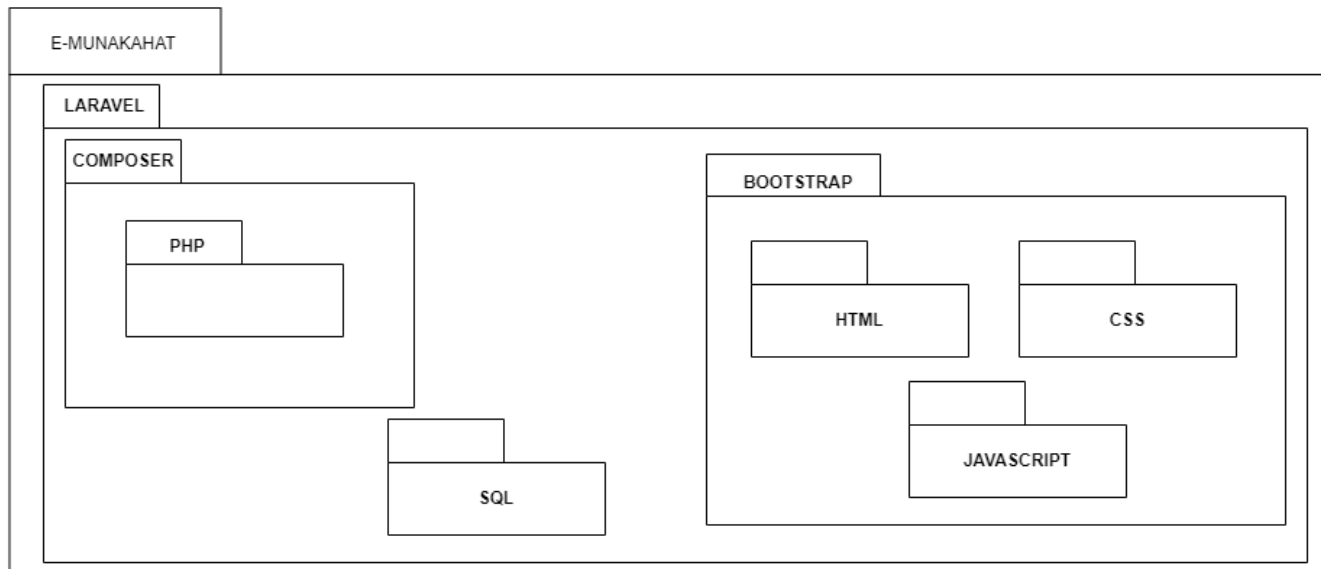
SpouseController	Manages the functionality and operations related to spouses within the application. Contains methods for creating, updating, deleting, and retrieving spouse records.
UserController	Handles the actions and operations related to managing users within the application. Contains methods for user registration, authentication, updating user information, and retrieving user details.
WaliController	Manages the logic and actions associated with managing walis (guardians) within the application. Contains methods for creating, updating, deleting, and retrieving wali records.
WitnessController	Handles the actions and operations related to managing witnesses within the application. Contains methods for creating, updating, deleting, and retrieving witness records.
HomeController	Handles the main functionality and actions related to the home page or main dashboard of the application. Contains methods for retrieving and displaying relevant information on the home page.
ConfirmPasswordController	This controller is responsible for handling password confirmations and uses a simple trait to include the behaviour
ForgotPasswordController	This controller is responsible for handling password reset emails and includes a trait that assists in sending these notifications from your application to your users.
LoginController	This controller handles authenticating users for the application and redirecting them to your home screen.
RegisterController	This controller handles the registration of new users as well as their validation and creation.
ResetPasswordController	This controller is responsible for handling password reset requests and uses a simple trait to include this behaviour.
VerificationController	This controller is responsible for handling email verification for any user that recently registered with the application.

### 3.1.2.2 Model [SDD-REQ-1100]

Class Name	Description
Applicant	This class retrieve and store applicant details
Application	This class retrieve and store application details
Consultant	This class retrieve and store consultant details
Consultation	This class retrieve and store consultation details
Incentive	This class retrieve and store incentive details
Invoice	This class retrieve and store invoice details
Marriage_card	This class retrieve and store marriage card details
Marriage_cer	This class retrieve and store marriage certificate details
Prep_course	This class retrieve and store preparation course details
Reference	This class retrieve and store reference details
Spouse	This class retrieve and store spouse details
User	This class retrieve and store user details
Wali	This class retrieve and store wali details
Witness	This class retrieve and store witness details

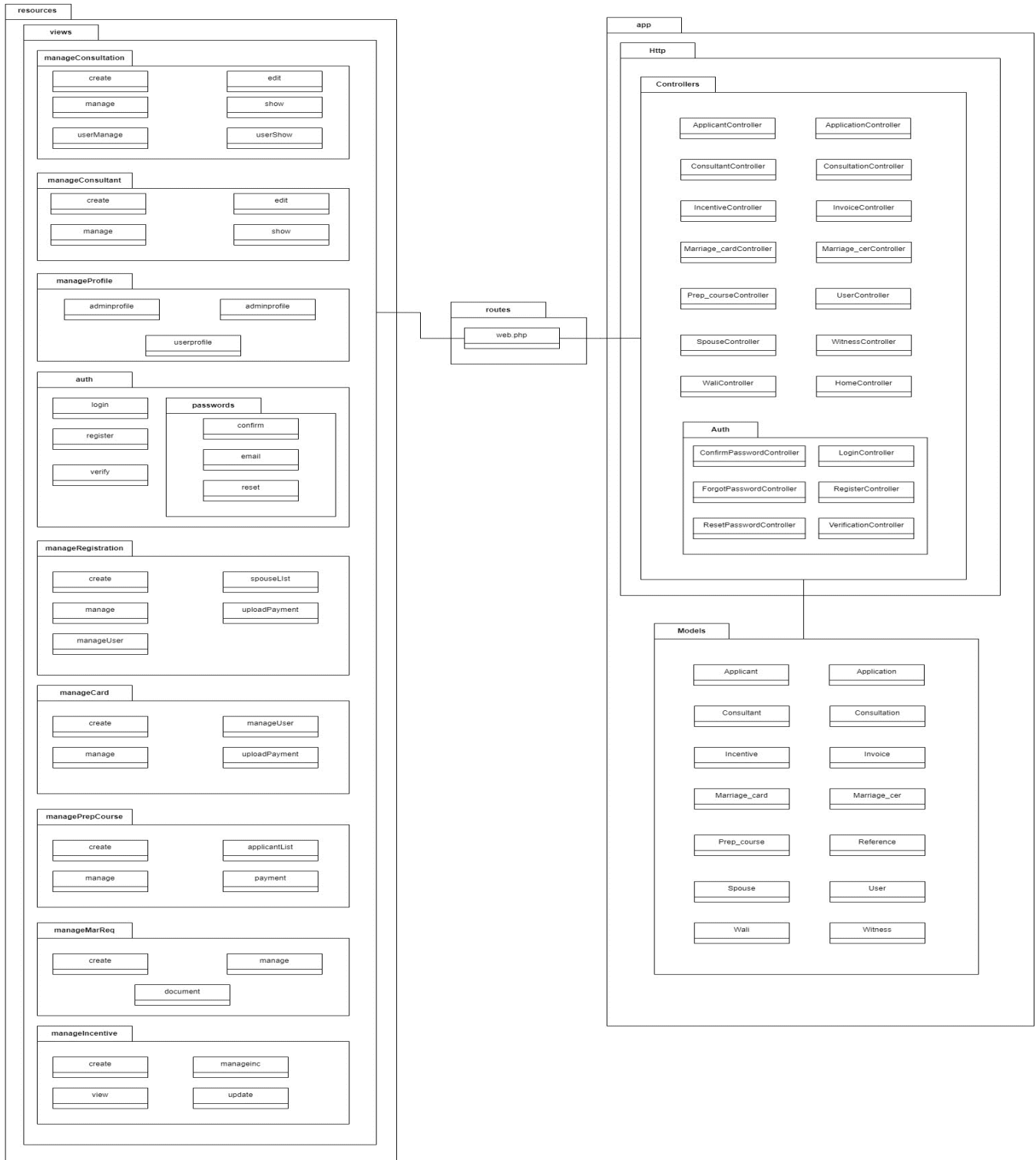


### 3.1.3 Middleware Layer



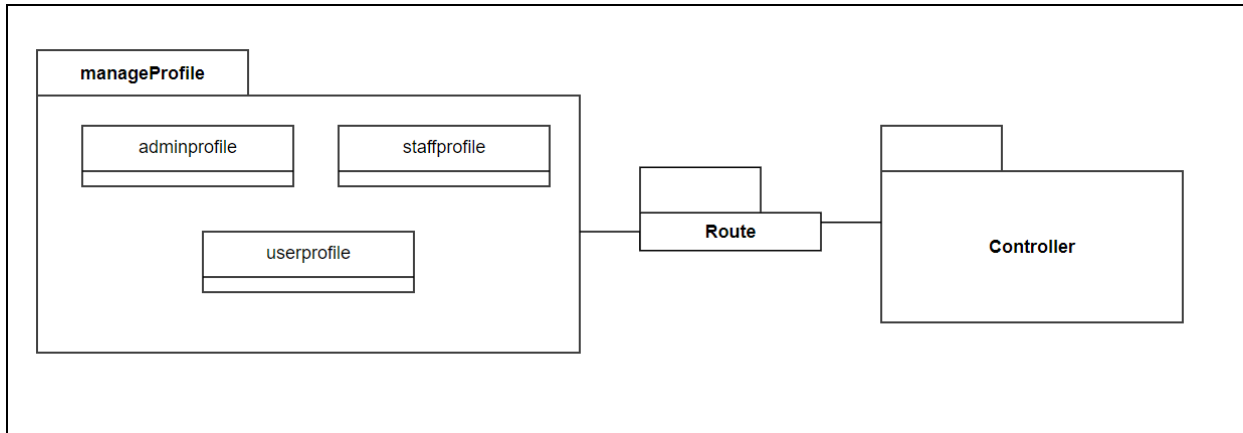
Package Name	Description
HTML	The package that contains of creating and design a website page
CSS	The package that contains of styling of the HTML package
JAVASCRIPT	The package that contains of enhancing the interactivity and functionality of website pages
LARAVEL	PHP web application framework. It contains various packages and classes for building web application.
PHP	The package that contains of code that builds dynamic web applications
SQL	The package that contains the database management

## 3.2 MVC Package Relationship



## 4. DETAIL DESIGN

### 4.1 manageProfile [SDD-REQ-100]



#### 4.1.1 adminprofile [SDD-REQ-101]

Class Type	Boundary class	
Responsibility	An interface to display the staff registration page that provides form for admin to register Staff.	
Attributes	Attributes Name	Attributes Type
	ic name gender contact email	Varchar Varchar Varchar Varchar Varchar
Methods	Method Name	Description
	registerstaff()	To register new account for the staff.
	viewregisterstaff()	To view the staff register page

Algorithm	BEGIN
	System display staff register page.
	Admin enter staff information.
	Admin click register staff button.
	END

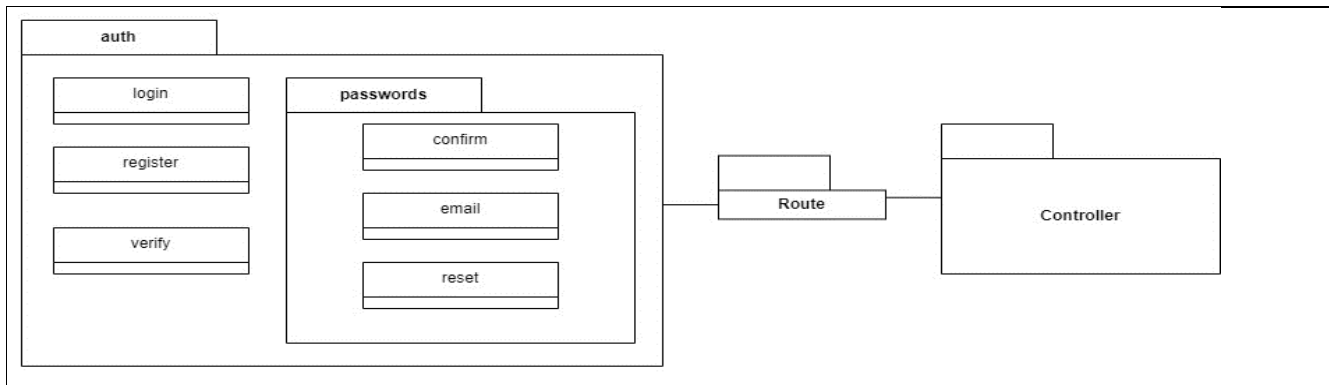
#### 4.1.2 staffprofile [SDD-REQ-102]

Class Type	Boundary class	
Responsibility	An interface to display staff user profile page that provides staff to update profile and change password.	
Attributes	Attributes Name	Attributes Type
	ic	Varchar
	name	Varchar
	gender	Varchar
	contact	Varchar
	email	Varchar
Methods	password	Varchar
	Method Name	Description
	viewprofile()	To view the user profile
	changepassword()	To change the password of the account.
Algorithm	updateprofile()	To update the detail of the profile
	BEGIN	
	System display user profile page	
Algorithm	IF staff make changes on user profile	
	Staff click "Update Profile" button	
	ELSE IF staff make changes on password	
	Staff click "Change Password" button	
	END IF	
	END	

#### 4.1.3 userprofile [SDD-REQ-103]

Class Type	Boundary class	
Responsibility	An interface to display staff user profile page that provides staff to update profile and change password.	
Attributes	Attributes Name	Attributes Type
	ic name gender contact email password	Varchar Varchar Varchar Varchar Varchar Varchar
Methods	Method Name	Description
	viewprofile()	To view the user profile
	changepassword()	To change the password of the account.
	updateprofile()	To update the detail of the profile
Algorithm	BEGIN  System display user profile page IF user make changes on user profile User click "Update Profile" button ELSE IF user make changes on password User click "Change Password" button END IF  END	

## 4.2 auth [SDD-REQ-200]



### 4.2.1 login [SDD-REQ-201]

Class Type	Boundary class	
Responsibility	An interface to display the login page that provides form for user, staff and admin login into the system.	
Attributes	Attributes Name	Attributes Type
	ic password	Varchar Varchar
Methods	Method Name	Description
	reset_pass_controller()	To enable user reset their password by entering email address
	authentication_controller()	To receive the user input in userLogin
Algorithm	<p>BEGIN</p> <p>    Display a login form with input fields for IC number and password.              if there is an error message then                  Display the error message.              else if the user is already registered then                  Show an error message indicating that the IC number is already registered.              end if              Provide an option to remember the user's login.              Allow the user to submit the form to the login route.              if the user is new then                  Provide a link for user registration.              else if the user forgot the password then                  Provide a link to the password reset page.              end if</p> <p>END</p>	

#### 4.2.2 register [SDD-REQ-202]

Class Type	Boundary class	
Responsibility	An interface to display the register page that provides a form for user to register into the system.	
Attributes	Attributes Name	Attributes Type
	ic name gender contact email password	Varchar Varchar Varchar Varchar Varchar Varchar
Methods	Method Name	Description
	registeruser()	To enable user to register themselves
Algorithm	<p>BEGIN</p> <p>    Display a registration form.</p> <p>    Capture user inputs for IC number, name, gender, contact, email, password, and password confirmation.</p> <p>    Validate user inputs:</p> <p>        if any error occurs, display the corresponding error message next to the input field.</p> <p>    Allow the user to submit the form:</p> <p>        if the form is submitted then</p> <p>            Register the user by processing the submitted data.</p> <p>            Redirect the user to a success page or perform any necessary actions.</p> <p>        else if the user inputs are incorrect or incomplete then</p> <p>            Display appropriate error messages.</p> <p>        end if</p> <p>END</p>	

#### 4.2.3 verify [SDD-REQ-203]

Class Type	Boundary class	
Responsibility	An interface to display the verify page that provides instructions for user to verify the account.	
Attributes	Attributes Name	Attributes Type
	email	Varchar
Methods	Method Name	Description
	verifyaccount()	To verify new account where user can click on the link.
Algorithm	<p>BEGIN</p> <p>    Display a verification card with a message to verify the user's email address.</p> <p>    if a fresh verification link has been sent then</p> <p>        Display a success alert indicating that a fresh verification link has been sent to the user's email address.</p> <p>    end if</p> <p>    Display a message instructing the user to check their email for a verification link.</p> <p>    Provide an option for the user to request another verification link:</p> <p>        if the user wants to request another verification link then</p> <p>            Process the request by submitting a form.</p> <p>            Redirect the user or perform any necessary actions.</p> <p>        end if</p> <p>END</p>	



## 4.2.4 confirm [SDD-REQ-204]

Class Type	Boundary class	
Responsibility	An interface to display the confirm page that provides a link for user to verify the account and a form to confirm the password.	
Attributes	Attributes Name	Attributes Type
	email password	Varchar Varchar
Methods	Method Name	Description
	viewconfirm()	Show the status of the account.
Algorithm	<p>BEGIN</p> <p>    Display a card with a heading "Confirm Password".</p> <p>    Display a message instructing the user to confirm their password before proceeding.</p> <p>    Begin a form with a method of "POST" and an action of "password.confirm".</p> <p>    Validate the password entered by the user:          if the password is invalid then              Display an error message indicating the validation error.          end if</p> <p>    Provide an option for the user to confirm the password:          if the user confirms the password then              Process the request by submitting the form.              Redirect the user or perform any necessary actions.          end if</p> <p>    Provide an option for the user to reset their password if forgotten:          if the "password.request" route exists then              Display a button or link labeled "Forgot Your Password?" that links to the "password.request" route.          end if</p> <p>END</p>	

## 4.2.5 email [SDD-REQ-205]

Class Type	Boundary class	
Responsibility	An interface to display the password reset page that provides a form for user to enter the email.	
Attributes	Attributes Name	Attributes Type
	email	Varchar
Methods	Method Name	Description
	resetpassword()	To let the user enter their email to reset the password.
Algorithm	<p>BEGIN</p> <p>    Display a card with the heading "Reset Password".</p> <p>    Check if a session status message exists:              if a session status message exists then                  Display a success alert with the session status message.              end if</p> <p>    Begin a form with a method of "POST" and an action of "password.email".</p> <p>    Retrieve the user's email address:              if the user's email address is provided and valid then                  Submit the form to send a password reset link to the user's email address.              Redirect the user or perform any necessary actions.              end if</p> <p>    Validate the email address entered by the user:              if the email address is invalid then                  Display an error message indicating the validation error.              end if</p> <p>    Provide an option for the user to send a password reset link:              if the user clicks the "Send Password Reset Link" button then                  Process the request by submitting the form.                  Redirect the user or perform any necessary actions.              end if</p> <p>END</p>	

#### 4.2.6 reset [SDD-REQ-206]

Class Type	Boundary class	
Responsibility	An interface to display the update password reset page that provides a form for user to enter the new password.	
Attributes	Attributes Name	Attributes Type
	email password	Varchar Varchar
Methods	Method Name	Description
	passwordtoken()	To reset the password.
Algorithm	<p>BEGIN</p> <p>    Display a card with a heading "Reset Password".</p> <p>    Begin a form with a method of "POST" and an action of "password.update".</p> <p>    Retrieve the reset password token:          if the reset password token is provided then              Include the token in a hidden input field.          end if</p> <p>    Retrieve the user's email address:          if the user's email address is provided and valid then              Retrieve and display the user's email address in an input field.          Display an error message if the email address is already registered.          end if</p> <p>    Retrieve the new password:          if a new password is provided and meets the required criteria then              Display an input field to enter the new password.              Display an error message if the password does not meet the criteria.          end if</p> <p>    Retrieve the confirmation password:          if a confirmation password is provided and matches the new password then              Display an input field to confirm the new password.</p>	

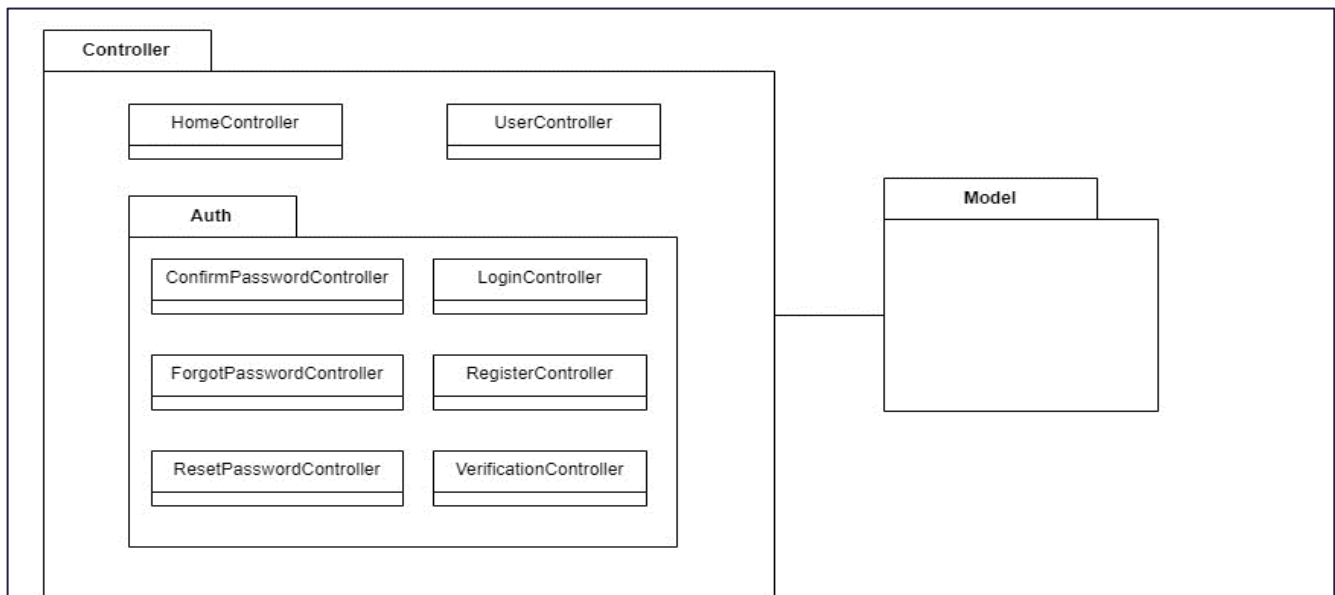
```

end if

Provide an option for the user to reset the password:
if the user clicks the "Reset Password" button then
    Process the request by submitting the form.
    Redirect the user or perform any necessary actions.
end if
END

```

### 4.3 Controller [SDD-REQ-1000]



#### 4.3.1 HomeController [SDD-REQ-1014]

Class Type	Controller Class	
Responsibility	The HomeController is responsible for managing user, staff and admin profiles and updating passwords in the application.	
Attributes	Attributes Name	Attributes Type
	None	None
Methods	Method Name	Description
	construct ()	Constructor method that sets up the middleware 'auth' for the HomeController class.
	indexUser()	Shows the user profile dashboard by returning the view 'manageProfile.userprofile'.

	indexStaff()	Shows the staff profile dashboard by returning the view 'manageProfile.staffprofile'.
	indexAdmin()	Shows the admin profile dashboard by returning the view 'manageProfile.adminprofile'.
	updatePassword()	Handles the request for updating the password. Validates the old_password and new_password, checks if the old password matches the authenticated user's password, and updates the password if successful.
Algorithm	<p>BEGIN</p> <p>    Create a new instance of the HomeController.</p> <p>    Define a constructor method that adds the 'auth' middleware to the controller.</p> <p>    Define a method called indexUser:          if the method is called then              Display the 'userprofile' view.          end if</p> <p>    Define a method called indexStaff:          if the method is called then              Display the 'staffprofile' view.          end if</p> <p>    Define a method called indexAdmin:          if the method is called then              Display the 'adminprofile' view.          end if</p> <p>    Define a method called updatePassword that accepts a Request object:          begin              Validate the request:              if the 'old_password' and 'new_password' fields are required and the 'new_password' is confirmed then                  Continue to the next step.              else                  Redirect back with an error message.          end if</p>	

```

        Check if the provided 'old_password' matches the authenticated
        user's password:
        if the passwords do not match then
            Redirect back with an error message.
        end if

        Update the authenticated user's password with the provided
        'new_password':
        if the password is successfully updated then
            Redirect back with a success message.
        end if
    end
END

```

#### 4.3.2 UserController [SDD-REQ-1011]

<b>Class Type</b>	Controller Class	
<b>Responsibility</b>	The update method in UserController updates user information and redirects to the appropriate home route based on the user's role.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	None	None
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	update(Request \$request, \$id)	Update the specified resource in storage. Retrieve the user, update their information, and redirect to the appropriate home route based on their role.
<b>Algorithm</b>	<p>BEGIN</p> <p>Create a new instance of the UserController.</p> <p>Define a method called update that accepts a Request object and an ID:</p> <pre> begin \$user = User::find(\$id); \$user-&gt;update(\$request-&gt;all());  if the user's role is 'staff' then </pre>	

```

        Set the $homeRoute variable to 'staff.home'.
    else if the user's role is 'admin' then
        Set the $homeRoute variable to 'admin.home'.
    else
        Set the $homeRoute variable to 'user.home'.
    end if

    Define a method called storeStaff that accepts a Request
    object:
    if the method is called then
    begin
        $request->merge([
            'password' => Hash::make('1234'),
            'role' => 1,
            'staff_id' => 'stf' . rand(100, 999),
        ]);
        $user = User::create($request->all());
        Redirect back with a success message.
    end
end
END

```

#### 4.3.3 ConfirmPasswordController [SDD-REQ-1015]

<b>Class Type</b>	Controller Class	
<b>Responsibility</b>	This controller is responsible for handling password confirmations and uses a simple trait to include the behaviour.	
<b>Attributes</b>	<b>Attributes Name</b>	<b>Attributes Type</b>
	None	None
<b>Methods</b>	<b>Method Name</b>	<b>Description</b>
	construct()	Constructor method that is called when the ConfirmPasswordController class is instantiated. It sets up the middleware 'auth' to ensure the user is authenticated.
	showConfirmForm()	Display the password confirmation form.

	confirm()	Confirm the user's password.
Algorithm	<p>BEGIN</p> <p>Create a new instance of the ConfirmPasswordController.</p> <p>Use the ConfirmsPasswords trait to include the password confirmation behavior.</p> <p>Set the redirectTo variable to the RouteServiceProvider's HOME constant.</p> <p>Define a method called __construct: if the method is called then end if</p> <p>END</p>	

#### 4.3.4 LoginController [SDD-REQ-1017]

Class Type	Controller Class	
Responsibility	This controller handles authenticating users for the system and redirecting them to the home screen	
Attributes	Attributes Name	Attributes Type
	None	None
Methods	Method Name	Description
	login(Request \$request)	Handles the login request. Validates the input fields (ic and password), attempts to authenticate the user, and redirects based on the user's role. If authentication fails, it redirects back to the login page with an error message.
Algorithm	<p>BEGIN</p> <p>Create a new instance of the LoginController.</p> <p>Use the AuthenticatesUsers trait for convenient authentication functionality.</p> <p>Set the redirectTo variable to the RouteServiceProvider's HOME constant.</p>	



```

Define a method called __construct:
if the method is called then
end if

Define a method called login that accepts a Request object:
if the method is called then
    $input = $request->all();

    Validate the request data: 'ic' field is required and must
    be 12 digits, 'password' field is required.

    if the authentication attempt is successful then
        if the authenticated user's role is 'admin' then
            Redirect to the 'admin.home' route.
        else if the authenticated user's role is 'staff' then
            Redirect to the 'staff.home' route.
        else if the authenticated user's role is 'user' then
            Redirect to the 'user.home' route.
        end if
    else
        Redirect to the 'login' route with an error message
        indicating invalid credentials.
    end if
end
END

```

#### 4.3.5 ForgotPasswordController [SDD-REQ-1016]

Class Type	Controller Class	
Responsibility	This controller is responsible for handling password reset emails and includes a trait that assists in sending these notifications from the system to the users.	
Attributes	Attributes Name	Attributes Type
	None	None
Methods	Method Name	Description
	sendResetLinkEmail(Request \$request)	Sends the password reset email to the user. Handles the logic for sending the password reset email notification.

Algorithm	<p>BEGIN</p> <p>    Create a new instance of the ForgotPasswordController.</p> <p>    Define a method called sendResetLinkEmail:</p> <p>    if the method is called then</p> <p>        Send the password reset email to the user.</p> <p>    end if</p> <p>END</p>
-----------	---

#### 4.3.6 RegisterController [SDD-REQ-1018]

Class Type	Controller Class	
Responsibility	This controller handles the registration of new users as well as their validation and creation.	
Attributes	Attributes Name	Attributes Type
	None	None
Methods	Method Name	Description
	validator(array \$data)	Gets a validator for an incoming registration request. Validates the data based on the defined rules and returns the validator instance.
	create(array \$data)	Creates a new user instance after a valid registration. Creates a new user record in the database with the provided data and returns the created user instance.
Algorithm	<p>BEGIN</p> <p>    Create a new instance of the RegisterController.</p> <p>    Use the RegistersUsers trait to include the functionality for user registration.</p> <p>    Define a method called register:</p> <p>        Validate the incoming registration request data:</p> <p>            - Required fields: ic, name, gender, contact, email, password</p> <p>            - Additional validation rules for each field</p> <p>        if the validation fails then</p> <p>            Logic for handling failed validation</p> <p>        else</p>	

	<p>Create a new user instance with the provided data:</p> <ul style="list-style-type: none"> <li>- ic, name, gender, contact, email, hashed password</li> </ul> <p>Save the user instance to the database.</p> <p>Create a new record in the applicant table linked to the user:</p> <ul style="list-style-type: none"> <li>- user_id: the ID of the created user</li> </ul> <p>Redirect the user to the specified redirect path after successful registration.</p> <p>end if</p> <p>END</p>
--	--

#### 4.3.7 ResetPasswordController [SDD-REQ-1019]

Class Type	Controller Class	
Responsibility	This controller is responsible for handling password reset requests and uses a simple trait to include this behaviour.	
Attributes	Attributes Name	Attributes Type
	None	None
Methods	Method Name	Description
	resetPassword()	Reset the given user's password.
	validateEmail()	Validate the email for the given password reset request.
Algorithm	<p>BEGIN</p> <p>Create a new instance of the ResetPasswordController.</p> <p>Use the ResetsPasswords trait to include the functionality for handling password reset requests.</p> <p>Define a method called reset:</p> <p>Reset the user's password:</p> <ul style="list-style-type: none"> <li>- Use the token provided in the request to retrieve the user.</li> <li>- Validate and update the new password.</li> </ul> <p>Redirect the user to the specified redirect path after successfully resetting the password.</p>	

	END
--	-----

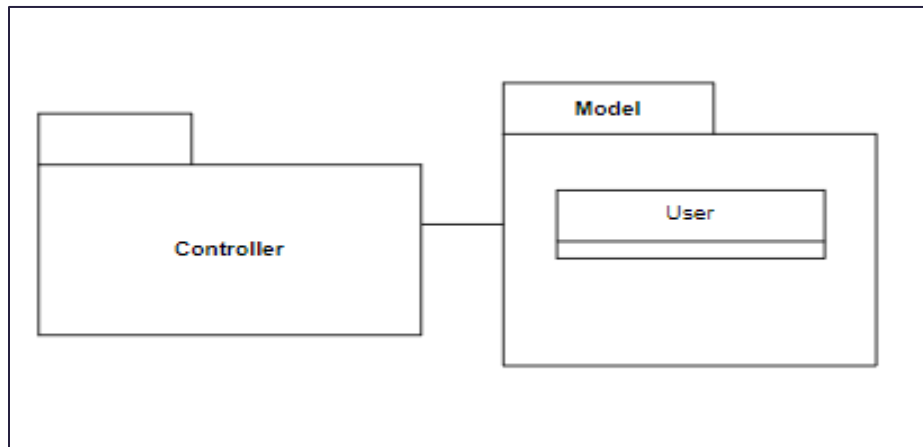
#### 4.3.8 VerificationController [SDD-REQ-1019]

Class Type	Controller Class	
Responsibility	This controller is responsible for handling email verification for any user that recently registered with the application.	
Attributes	Attributes Name	Attributes Type
	None	None
Methods	Method Name	Description
	verify()	Mark the authenticated user's email address as verified.
Algorithm	<p>BEGIN</p> <p>    Create a new instance of the VerificationController.</p> <p>    Use the VerifiesEmails trait to include the functionality for handling email verification.</p> <p>    Define a method called verify:</p> <p>        if the request has a signed URL:</p> <p>            Verify the user's email:</p> <p>                - Use the signed URL provided in the request to verify the user.</p> <p>                - Mark the user's email as verified in the database.</p> <p>        Redirect the user to the specified redirect path after successfully verifying the email.</p> <p>        end if</p> <p>    Define a method called resend:</p> <p>        if the user is eligible to resend the verification email (throttle):</p> <p>            Resend the email verification notification.</p> <p>        Redirect the user to the specified redirect path after resending the verification email.</p> <p>        end</p> <p>    Set the middleware:</p>	

- 'auth': Require authentication to access the verification routes.
- 'signed': Allow only signed URLs for the 'verify' method.
- 'throttle': Limit the rate of accessing the 'verify' and 'resend' methods.

END

#### 4.4 Model [SDD-REQ-1100]



##### 4.4.1 User [SDD-REQ-1112]

Class Type	Model Class	
Responsibility	This model manage the retrieving and storing data from users table and also manage its relationship with other tables.	
Attributes	Attributes Name	Attributes Type
	\$fillable	Array
	\$hidden	Array
	\$cast	Array
Methods	Method Name	Description
	role()	Method to returns an Attribute object representing the role attribute and provides a mapping functionality for converting input values to user roles from a predefined array. It ensures controlled access to the role attribute within the class and its subclasses.

Algorithm	<pre>BEGIN User Model Definition   Use HasApiTokens, HasFactory, Notifiable traits    Define fillable attributes    Define hidden attributes    Define attribute casting rules    Define a role() method for custom attribute casting   BEGIN Custom attribute casting logic     Map integer value to corresponding role string   END  END</pre>
-----------	--

## 5. REQUIREMENT TRACEABILITY

### 5.1 manageProfile [UC-101-EMUN-001]

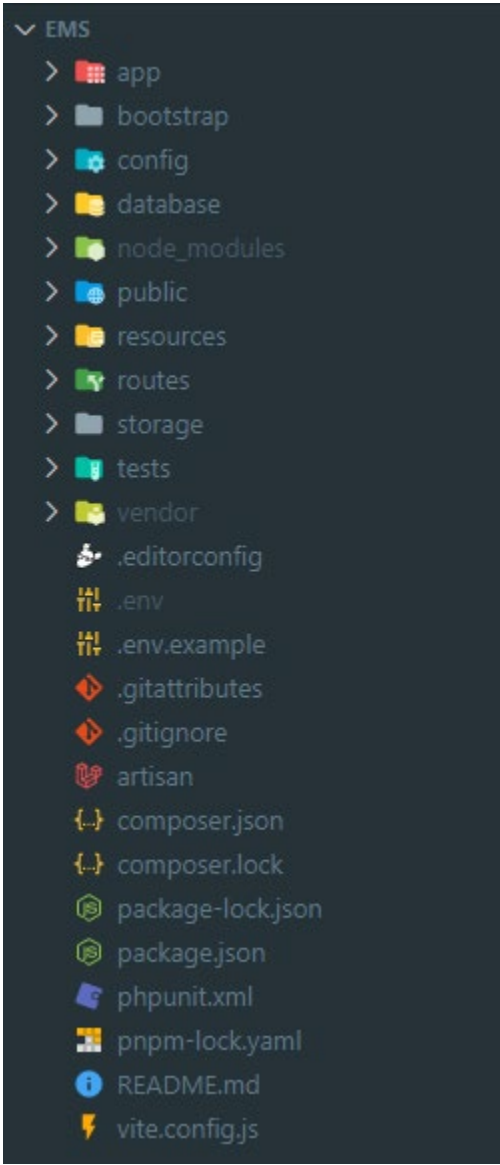
Use Case ID	Description	Package ID
UC-101-EMUN-001	Admin can register the staff	SDD-REQ-101
	Staff can view their profile. Staff can update their profile. Staff can change their account password.	SDD-REQ-102
	User can view their profile. User can update their profile. User can change their account password.	SDD-REQ-103

### 5.2 auth [UC-102-EMUN-001]

Use Case ID	Description	Package ID
UC-102-EMUN-001	User, staff and admin can login into the system	SDD-REQ-201
	User can register themselves to have an access into the system	SDD-REQ-202
	User can verify the account in the system	SDD-REQ-203
	User can confirm the account in the system	SDD-REQ-204
	User can enter the email to reset password in the system	SDD-REQ-205
	User can reset password system.	SDD-REQ-206

# APPENDIX

## APPENDIX B

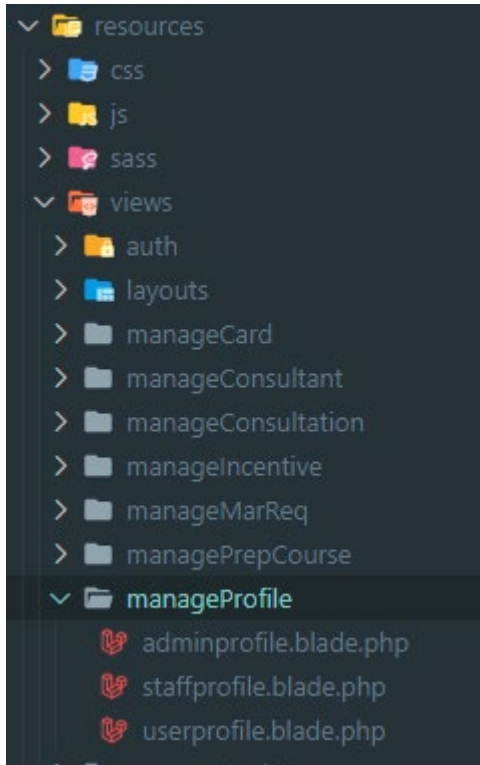


System Name	Layer		
EMS	resources	views	
	app	Http	Controllers
		Models	
	routes		



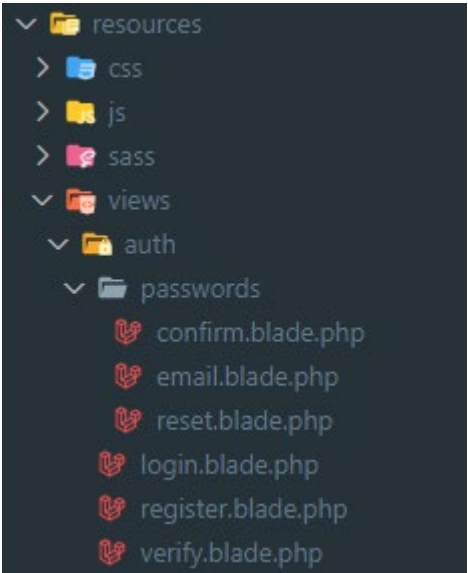
## APPENDIX C1

### manageProfile - Views Layer



Layer	Package	Class
views	manageProfile	adminprofile
		staffprofile
		userprofile

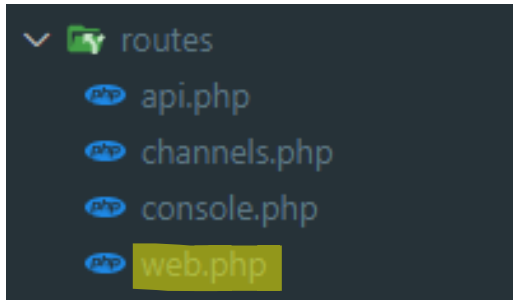
auth - Views Layer



Layer	Package		Class
views	auth	passwords	confirm
			email
			reset
			login
			register
			verify

## APPENDIX C2

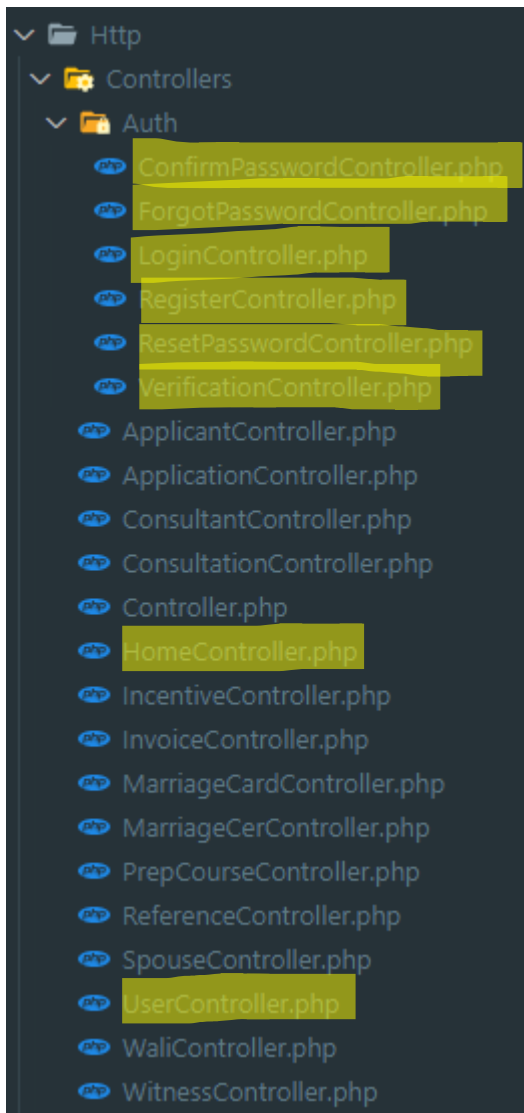
manageProfile & auth - Design Pattern Layer



Layer	Class
routes	web

## APPENDIX C3

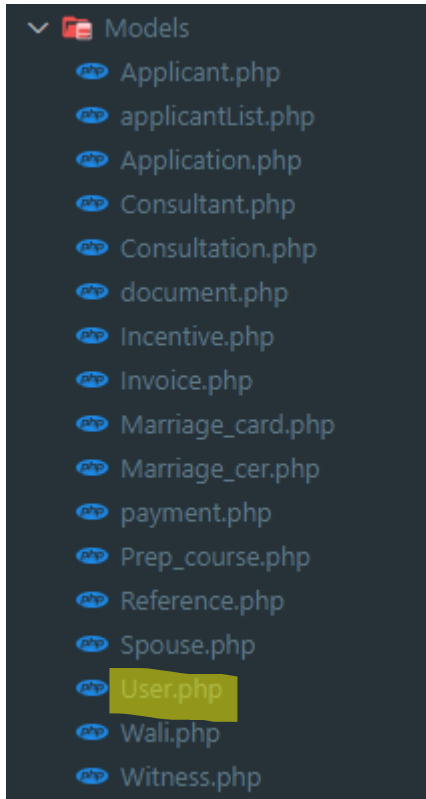
manageProfile & auth - Controller Layer



Layer	Class
Controllers	ConfirmPasswordController ForgotPasswordController LoginController RegisterController ResetPasswordController VerificationController HomeController UserController

## APPENDIX C4

manageProfile & auth - Model Layer



Layer	Class
Models	User