

FINAL REPORT

CSC 1103 OBJECT ORIENTED PROGRAMMING

**SECTION 04**

|  |  |
| --- | --- |
| Name | Matric Number |
| Abubakar Abubakar Yusif | 1821881 |
| Mohammad Mohammad (volunteer) | 1817733 |
| Barry Elhadj Mamadou Alpha | 1719211 |
| Abduoraeham |  |
| Maboyi |  |

PROJECT TITLE: D-Schedule

LECTURER: Dr, AMEER ATIF

Contents Table:

Content: page

Introduction.............................................................................................. 3

Project Diagrams ..................................................................................... 4

Activity .................................................................................................... 5

Sequence .................................................................................................. 6

Class UML ............................................................................................... 7

Interfaces .................................................................................................. 10

More Explanation ..................................................................................... 12

Task table .................................................................................................. 14

**Introduction:**

"D-Schedule" is a desktop based application written in java,javafx and Css. Currently tested On a Linux machine (who care's window)

The application is created to ease the life of Us (student) "No more procrastination in any task or due". the application has the following Features:

1. Allow anyone to create an account, all you need to-do is register your self with a valid information.

2. The application Can store data Offline (literally this thing is an offline application).

3. Your Password aren't stored in a plain text, all your credentials are encrypted(so you don't have to worry security)

4. Well documented application.

5. The code is free for any kind of distribution, (just don't claim you're the developer)

6. Great Project Structure, You don't need to be check every class when you're looking for something.

The application is Capable of doing many stuff at this date. and hoping we'll find contributors to this dummy application and make it reach more and more users.

To-do:

1. Enable Synchronization with Internet (need IIUM API so that our client won't have to enter Subject manually)

2. Notify via mail for an important important dates.

3. Allow Student to chat with each other.

"D-Schedule" came in to reality for the sake of the following heroes.

1. Material Ocean help with the color schema :: github.com/material-ocean.

2. Of-course never forget with github they helped us all the time. they do host our Code for free.

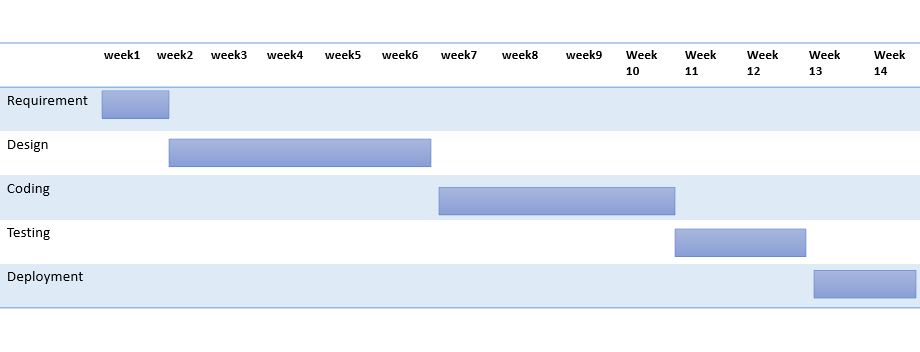
3. Credit goes to Google, Stack-over and many more online tutorials.

4. Finally Our lecturer Dr, Ameer. The motivation behind this thing.

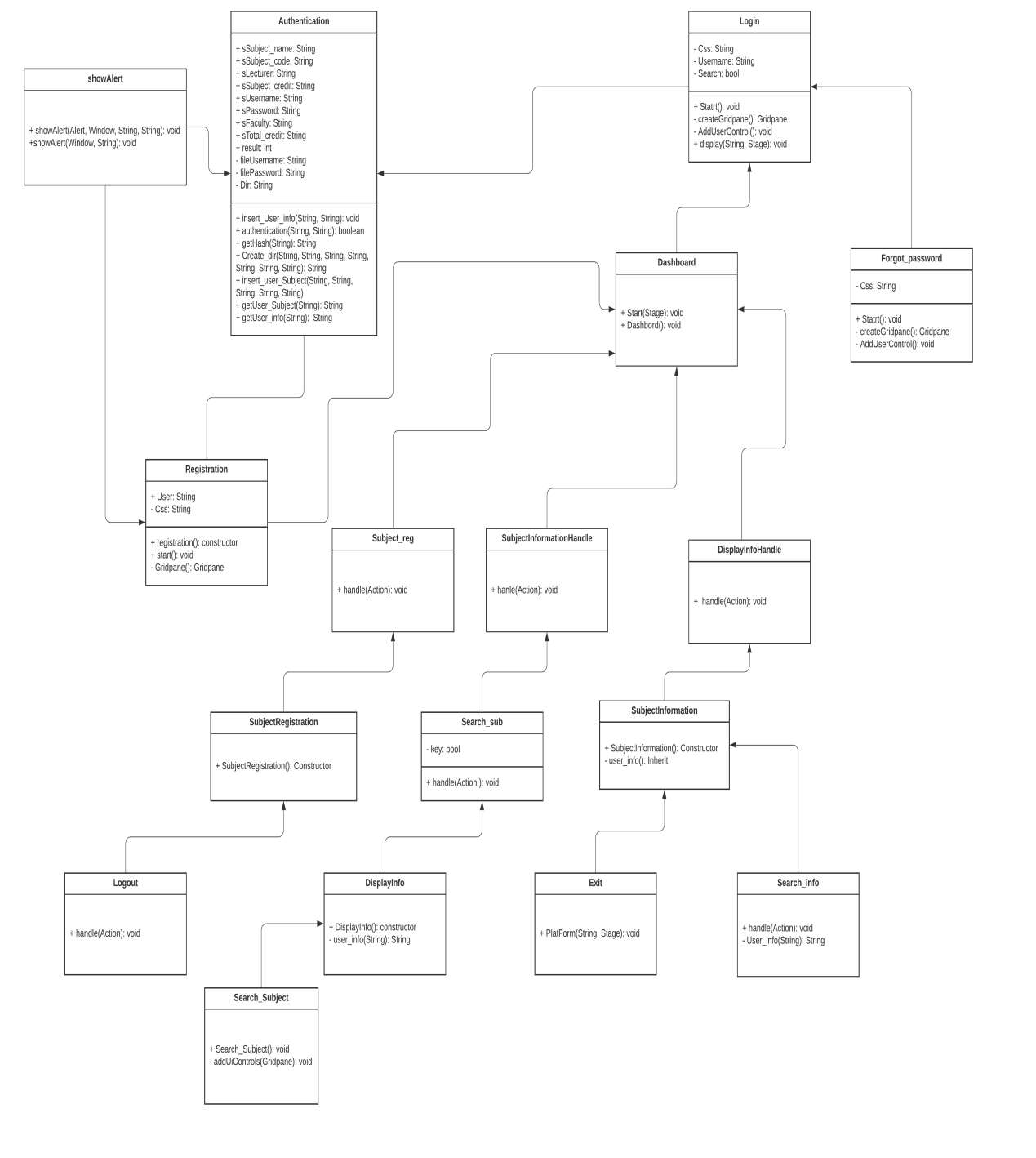
Written and documented by Abubakar Abubakakar Yusif.

**Project Visual Diagrams:**

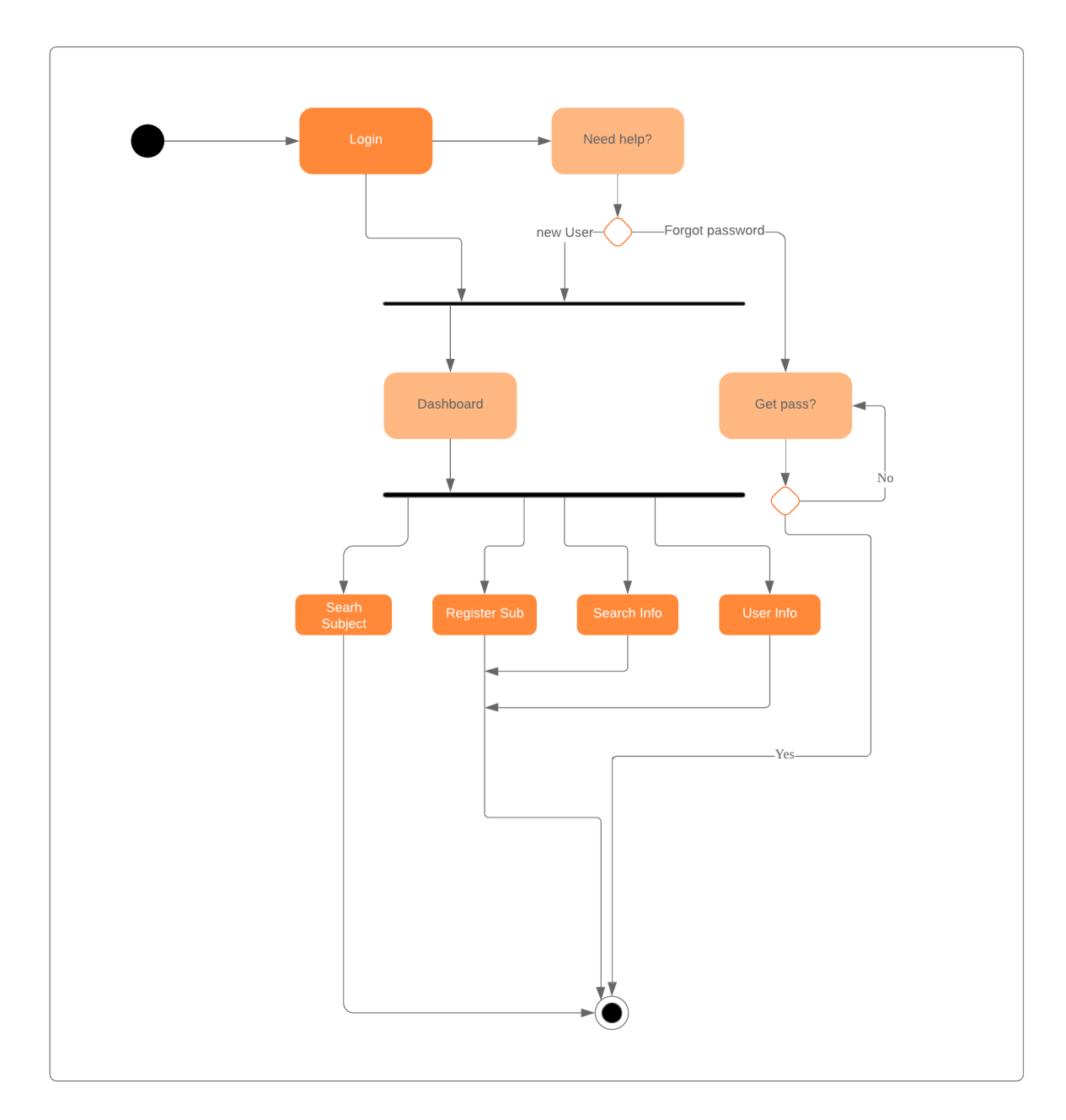
**Gantt chart**



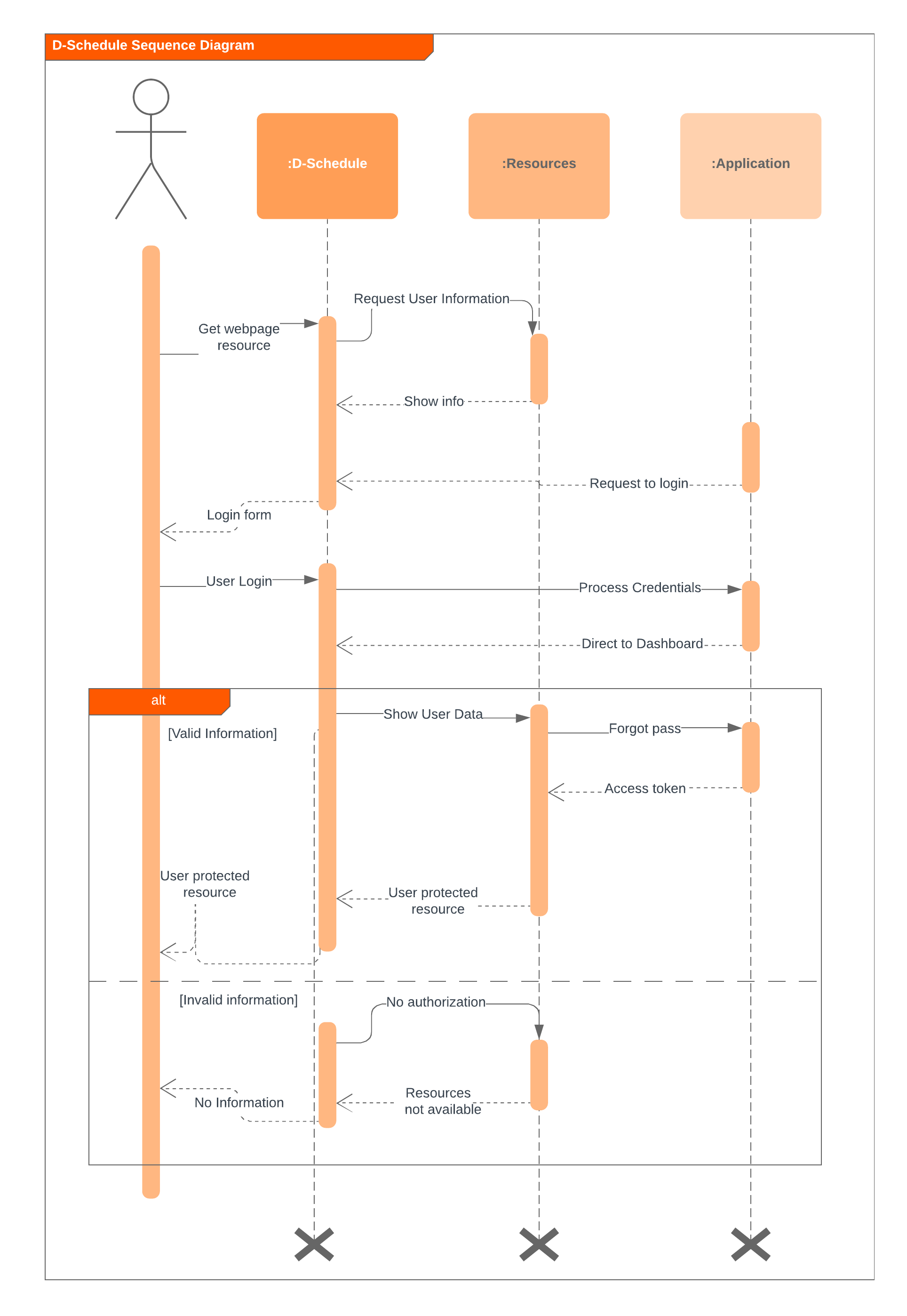
**Class UML diagram:**



Activity diagram:

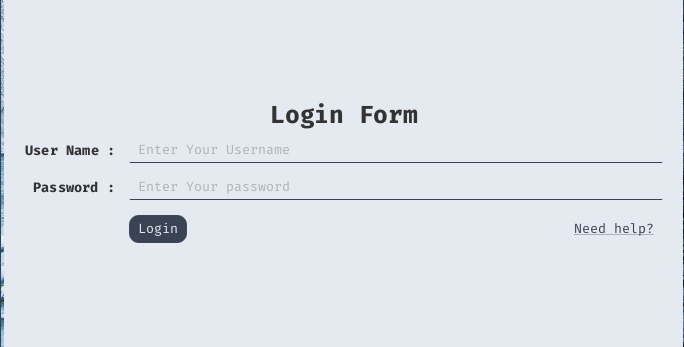


**Sequence diagram:**

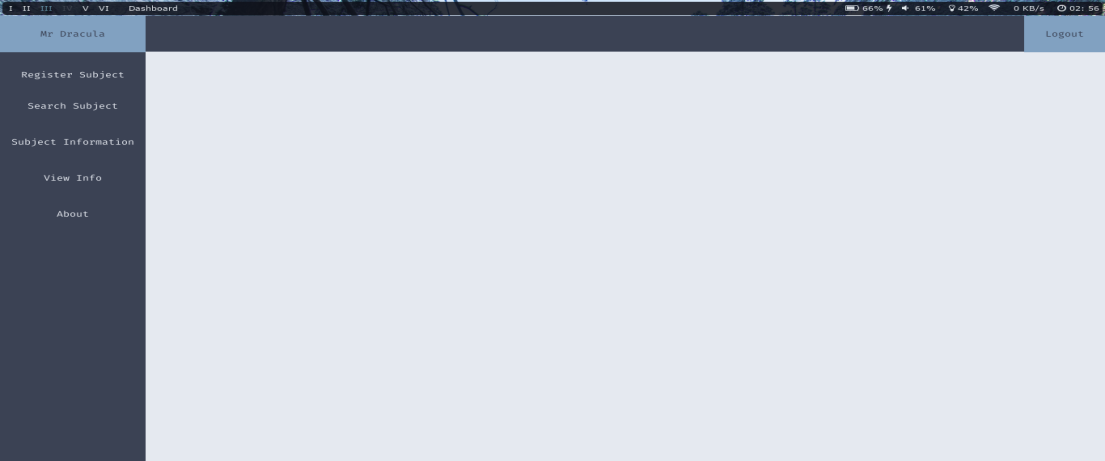


**INTERFACES:**

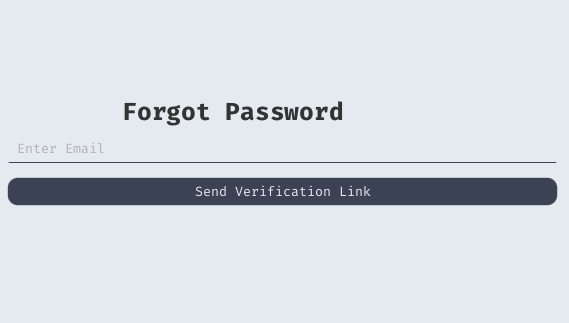
1. Login



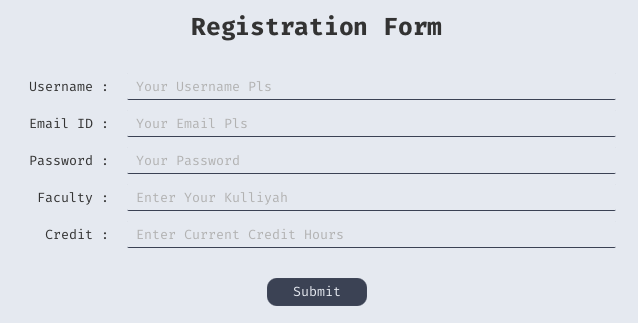
1. Dashboard



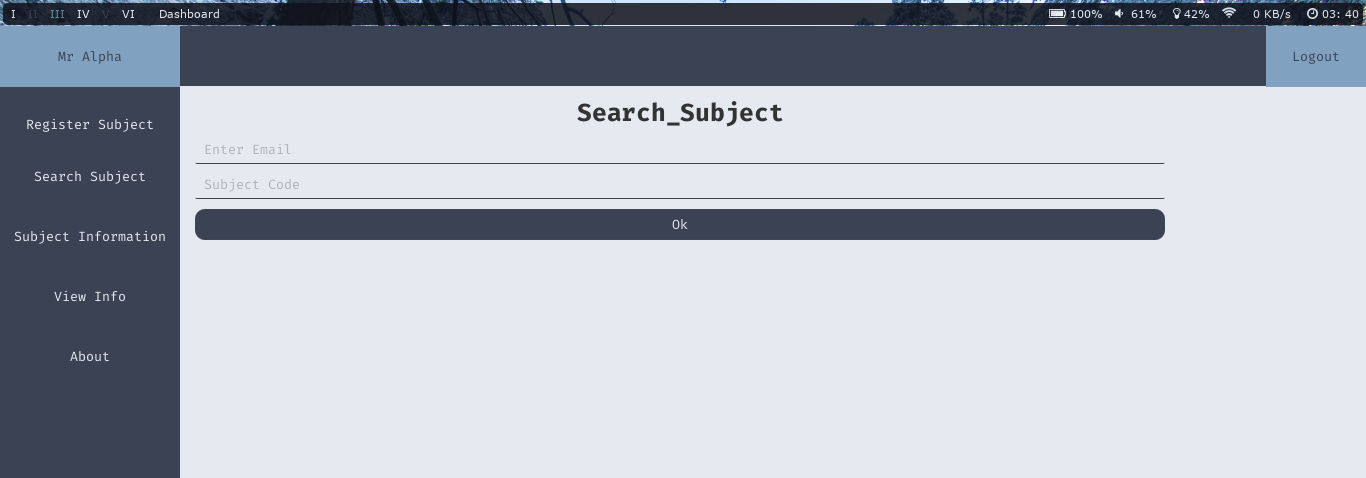
1. Forgot-password



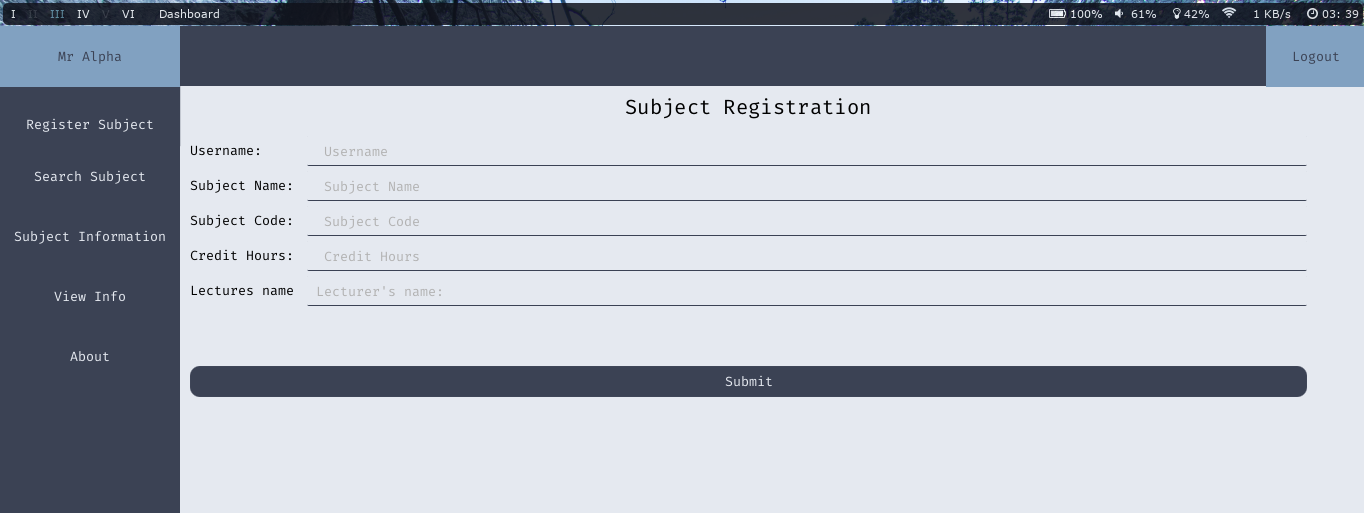
1. Registration



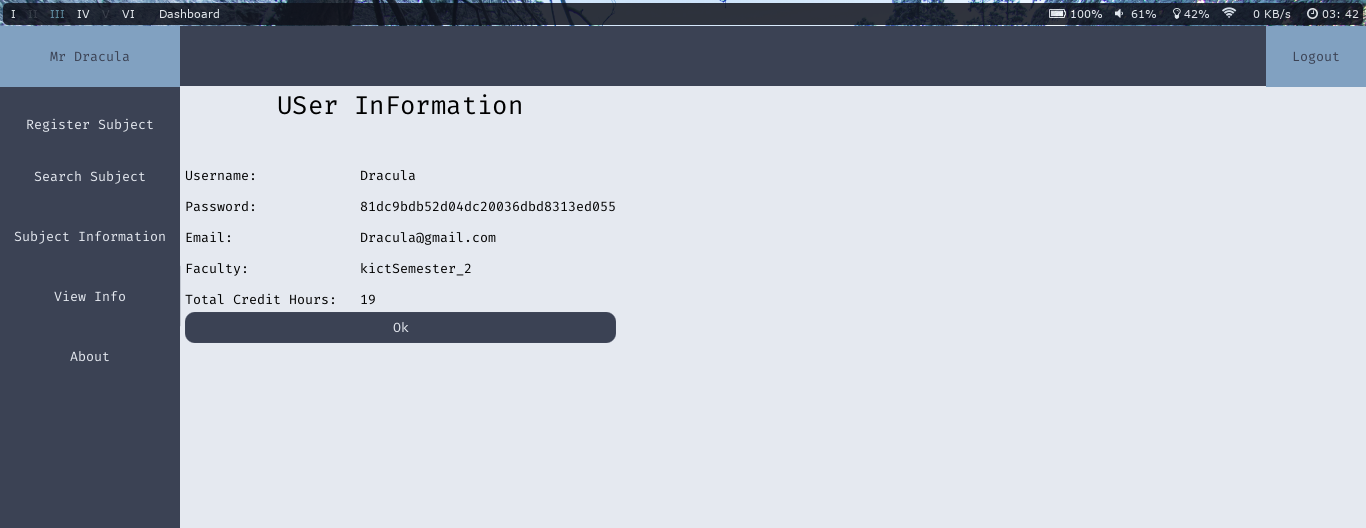
1. Search Subject



6 Add Subject



7 User-Info



8 Subject Info



\*\*The New-User class, also know as Registration page.

The class is design to allow new user to create a new account in Our system.

The class is connected with the follow Class:

1. Authentication

2. Dashboard

3. Login

::Authentication class is Used to Hash Both Username and password. Using Md5 and then create User Folder with the generated Md5. (for User Information Storage)

::Dashboard will pulled off when the User completed Registration.

::Login The class that will redirect user to Registration Class

FIELDS AND BUTTONS:

:::Username: take Username, compare it with all Username in database and return true if its not found. (means no duplicate Username)

:::Email: hold the value of Email before we create File to store it.

:::Password: take password, Create a hash of it Using Md5 and return the hash value.(One more thing. We'll never Store a plain text of either username or password in our database)

:::Faculty: also know as Kulliyah, Lol, doing same thing as username

::: Credit: literally Still not sure should i remove it or leave it. but i think I'll leave it. (depends on my mood next time when debugging)

:::Submit: Instantiate an object of Dashboard and Authentication.

\*\*Login Page. Probably the hardest Page.

Before we even go future i would like to take tour to the Project Structure.

The Login page Consist of Two clickable buttons, Which are Login and Need help.

The login button will redirect you to Dashboard. while Need Help button will prompt the following prompts to the Users. 1. Forgot\_password, 2. New User, and exit.

Forgot\_password will triggered Forgot\_password page, new user will triggered the Forgot\_password page, and exit will Simply exit from the application

The 2 Fields are Username and Password (literally everyone in this world user it);

\*\*Forgot Password.

The Simplest Class in the whole Application.

Literally one consist of a single button, and one Text field.

the Email fields: Take an email value, and return true if it exist in our database, if it returns then the submit button will trigger an email verification to the email. else, the user will get an error message with a suggestion of its solution.

\*\*Dash-Board:::

As first time i thought this thing gonna be easy, but when i dig in to it i found it complicated. The Class is hard-coded in javafx. literally all Dashboard is doing is bringing all the splits class into something useful.

The Dashboard consists of the following buttons:

1. Homepage

2. Logout

3. User Information

4. Subject registration

5. Register a new Subject into our database

6. Search Subject

7. About

whereby all of the above buttons override a class. then call the class either by instantiating a new object or using a static values. ( check the code for more details). all those classes got a validation checking to ensure the data given by the user is accurate. And any error encounter the user will get a little Alert box showing the error and the possibilities in solving the problem.

\*\*User-Info::Authentication:::

This is the core input and output to the application, All the data given by the user are checked in this class before they passed to any method. This class is also used to manage user information Stuff. The class is enriched with Polymorphisms and encapsulation, whereby all the data aren't visible to any other person except the current user object with a valid credentials. Below are the methods in the class.

1. Insert\_user\_info(String, String) void

2. authentication (String, String) bool

3. get\_hash(String) String

4. Create\_dir (String, String, String, String, String, String, String) void

5. Insert\_user\_subject(String, String, String, String, String, String, String) void

6. getUser\_subject(String) void

7. getUser\_info(String) void

\*\*The-Show-Alert Class::

This class is meant only for the alerts which needed to show to the User. This class doesn't have any method, but i overload the Constructor with two different signature.(mean you can use the messages in different places, and this makes it easy not to always copy and paste the Alert class, now we have one single class to refer when ever we need an alert box) the constructors are as follows:

Alert(Window,String)

Alert (alert-type, Window, String, String)

\*\*The Subject-Registration::

By the name is easy to guess what does this class did. but lets pretend you cant understand it by its name. The class is design to take a stream of String from the User. and Store them into a directory created with an encrypted Username, its Quite surprising asking user to enter his username again when registering. But here as we tend to have as much safety as we can, the important of this is to ensure that the given username by the object are the same as the real-time username.

The class takes not much arguments and Simply writes them into a file within the given directory.

\*\*The Subject-Search:::

This is made to make it easy for the user to search a submit with a given Username and Subject Code. this Class Simply returns true if the Subject is found else false will be returned, and the user will get a notified information showing that the given task in not successful.

\*\*The Subject-Information::

A very Simple Class that will print all the found subject in the database,which are associate the the current user. This class is connected with authentication Class, because we need to find the real-time hash of the username to ensure everything is going as intended.

\*\*The eXit-Class:::

This class does prompt the user when ever User tries to quit the application, this class is meant to reduce the mistake many Users make when using application. Any button that will triggered quitting, will first trigger this class. and the modality of the class is set to true. (means you can continue using the application until you respond to the quitting message)

\*\*The About-Class::

This class to have a quick show to the user about the developers of this great application. More about look at the About page, which is located in the main-Page or Dashboard.

**TASK DISTRIBUTION TABLE**

|  |  |  |
| --- | --- | --- |
| No: | Names: | Task |
| 1 | BARRY ELHADJ MAMADOU ALPHA | IDEAS AND 2 CLASSES |
| 2 | MABOYI | INTERFACE AND DEBUGGING |
| 3 | ABDURAHMAN | 2 INTERFACES AND CODING |
| 4 | ABUBAKAR ABUBAKAR | 2 INTERFACES AND CODING |

LIST OF THE CONSTRAINT AND CHALLENGES FACE

1. Lack of good javafx knowledge
2. Making the dashboard