Group name: ALPHA

Group members: 1. Aktaruzzaman Ridoy (F16040112)

2. Fabiha Tafannum (F16040113)
3. Md. Nazim Uddin Nayim (F16040114)
4. Md. Jahidul Kabir (F16040117)
5. Golam Jilany Rayhan (F16040102)

Software Design Specification

Project name: Student Registration

Introduction: The software that we are working with is named 'Student Registration System'. In building and completing this whole project, this time our target is to make this assignment to accomplish the detailed architectural design and phased product delivery plan for our system. Some of the modules like packages/classes that we are using are 'about', 'login', 'addAdmin', 'addStudent', 'ManageCourse' etc. and so on. Here, in this document, we are going to enhance the design of our project with our planning components.

Design Document: Our document contains of 'Architecture Section' and 'A Process section' as required.

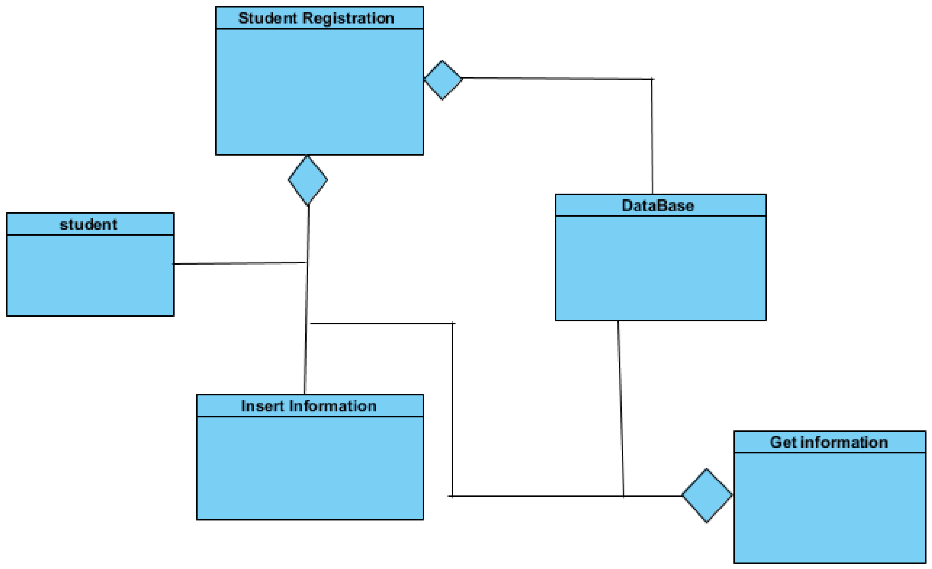
<u>System architecture:</u> In a nutshell, we are planning to comprise our software with ‘Student’s information’ with personal account. Here is our whole project plan epitomized below in two parts-

1. According to class diagram
2. According to sequence diagram

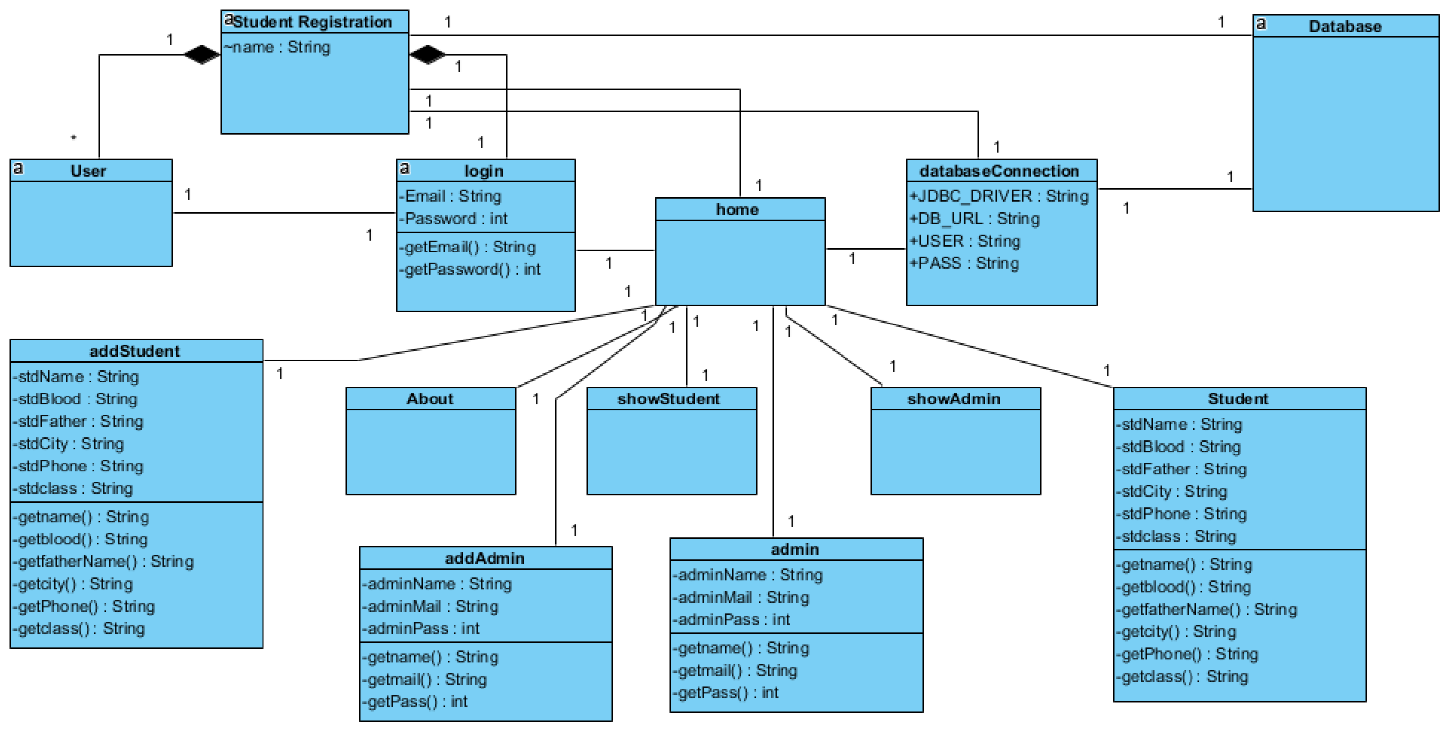
▪ According to class diagram

- The opening point of this software ‘Student Registration’, is a String data type.
- At the beginning, a login page will appear containing ‘mail id’ and ‘password’. And for this, we built a class named ‘login’. ‘String’ datatype for ‘email’ and ‘int’ datatype for ‘password’ were used to implement the login system.
- After logging in, a home page will be popped up that is performed in class module named as ‘home’.
- When a user will enter the home page he/she will see nine options that can be checked out. The classes ‘student’, ‘addStudent’, ‘showStudent’, ‘admin’, ’addAdmin’, ‘showAdmin’, ‘CourseDetails’, ‘ManageCourse’, ‘ShowCourse’ has been created for their functionality.
- Among the nine modules above, for ‘CourseDetails’, it has another package called ‘Add Course’. Some other sub modules are also there to describe the courses.

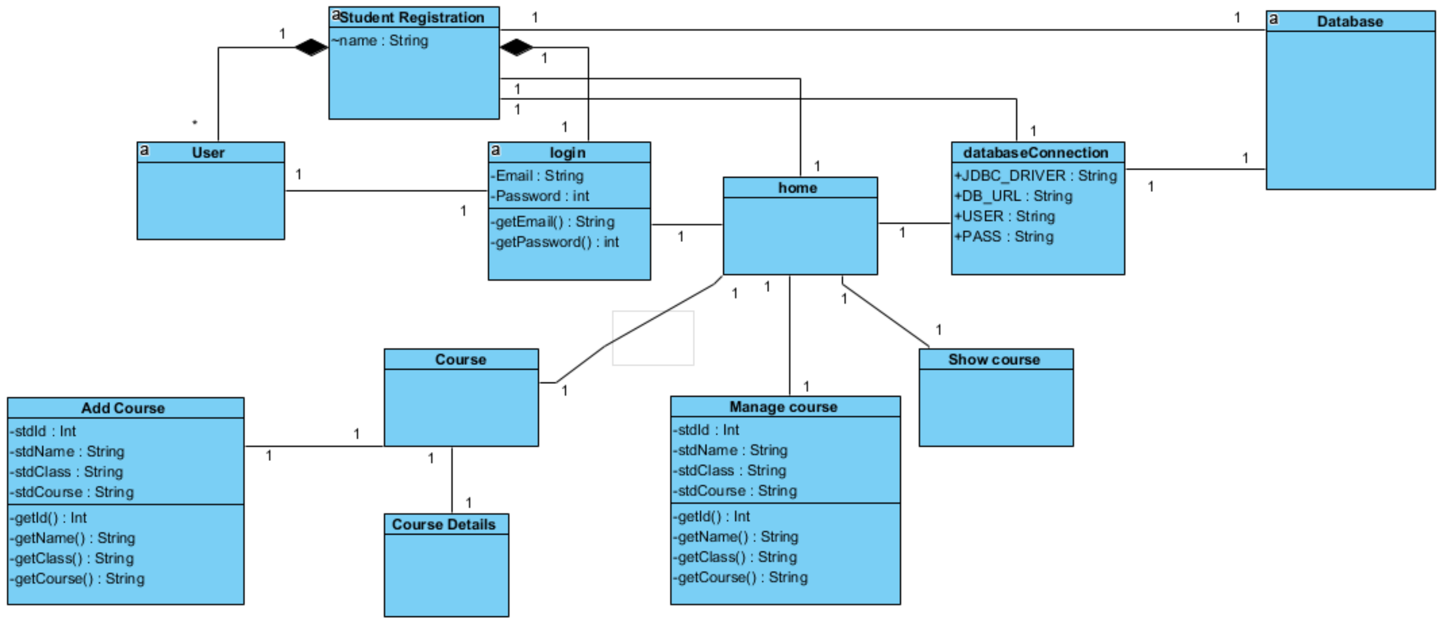
- Into the ‘Student’ and ‘addStudent’ classes, six variables ‘stdName’, ‘stdBlood’, ‘stdFather’, ‘stdCity’, ‘stdPhone’ and ‘stdclass’ were implemented with ‘string’ data type.
- For the admin panel ‘adminName’, ‘adminMail’ and ‘adminPass’ are the variables of both ‘admin’ and ‘addAdmin’ classes. Here also the variables are all ‘string’ data type.
- The class ‘databaseConnection’ has member functions JDBC_DRIVER, DB_URL, USER and PASS and all four of these are ‘string’ type.



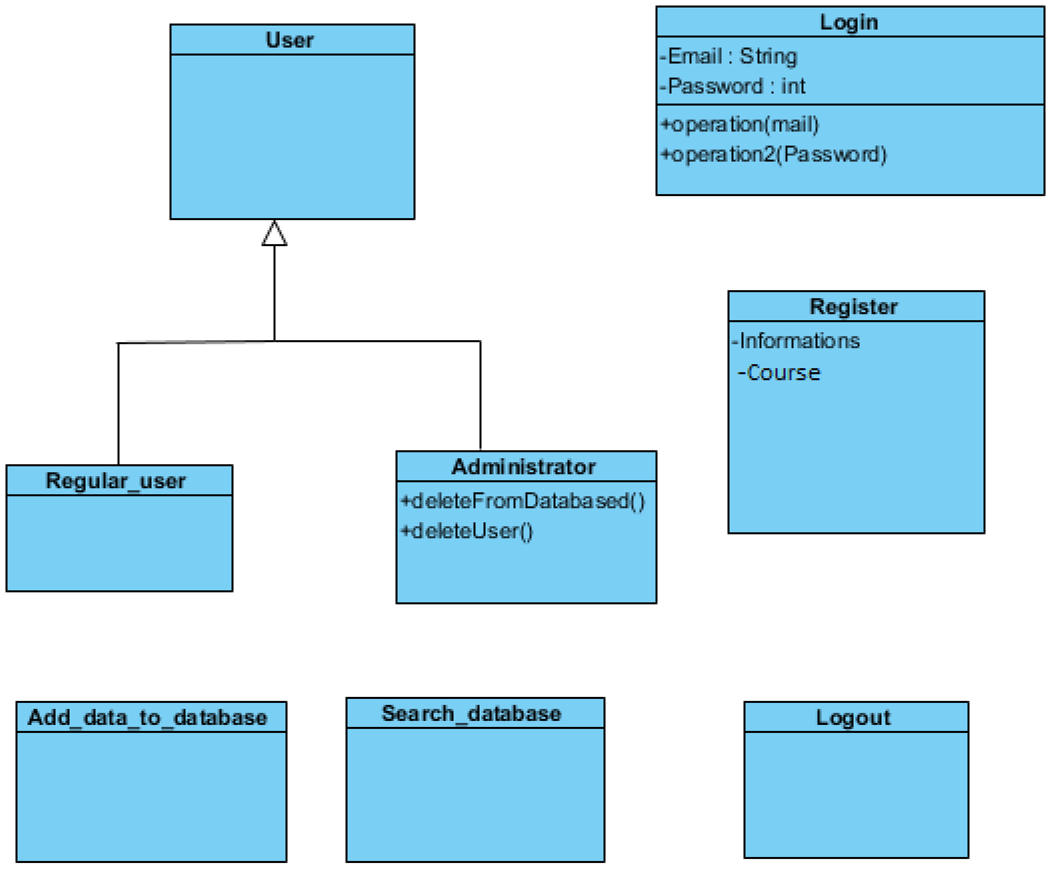
<u>UML Class Diagram 1.1:</u> Basic idea about our Project.



UML Class Diagram 1.2: About information Registration.



UML Class Diagram 1.3: About Course Registration.



UML Class Diagram 1.4: About How it works.

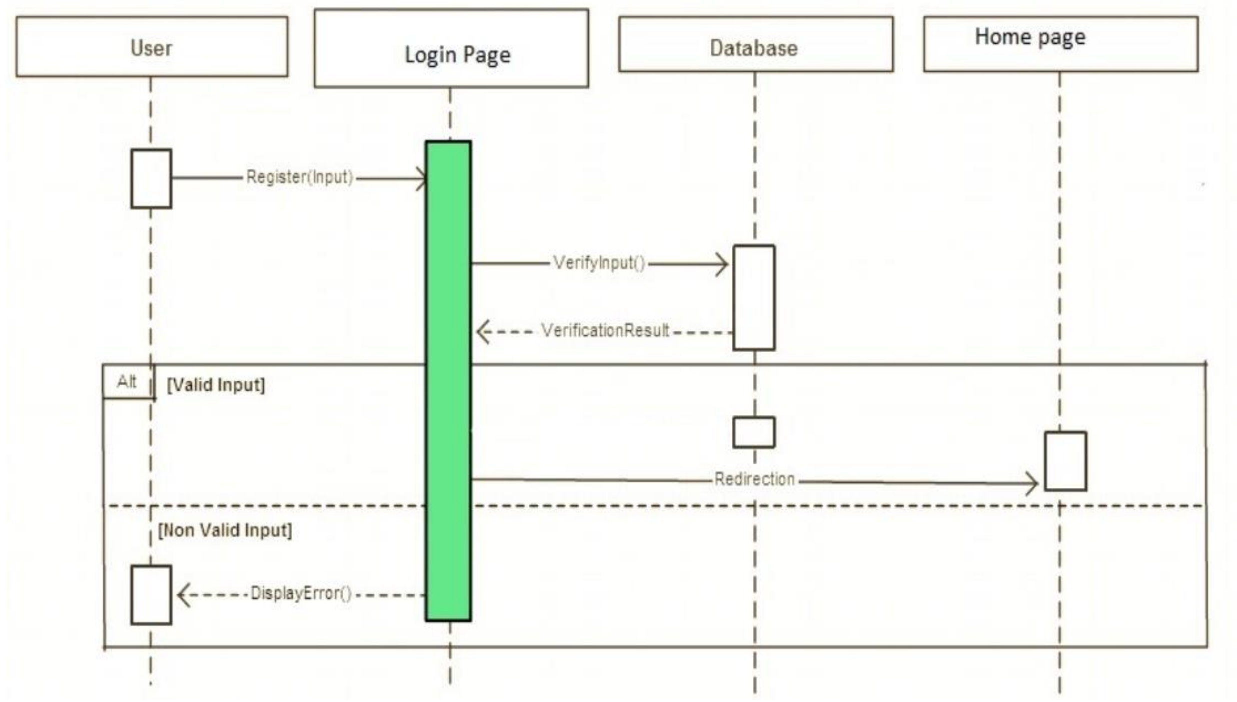
- **According to sequence diagram**

- At the commencement of starting work in our designed software, in the login page the user enters the mail and password and gets access to enter into next page.
- If entered details are valid, the user's account becomes available. If it's invalid then appropriate message for this displays to the user.
- After logging in, a home page appears that contains functions for students, admin and courses.

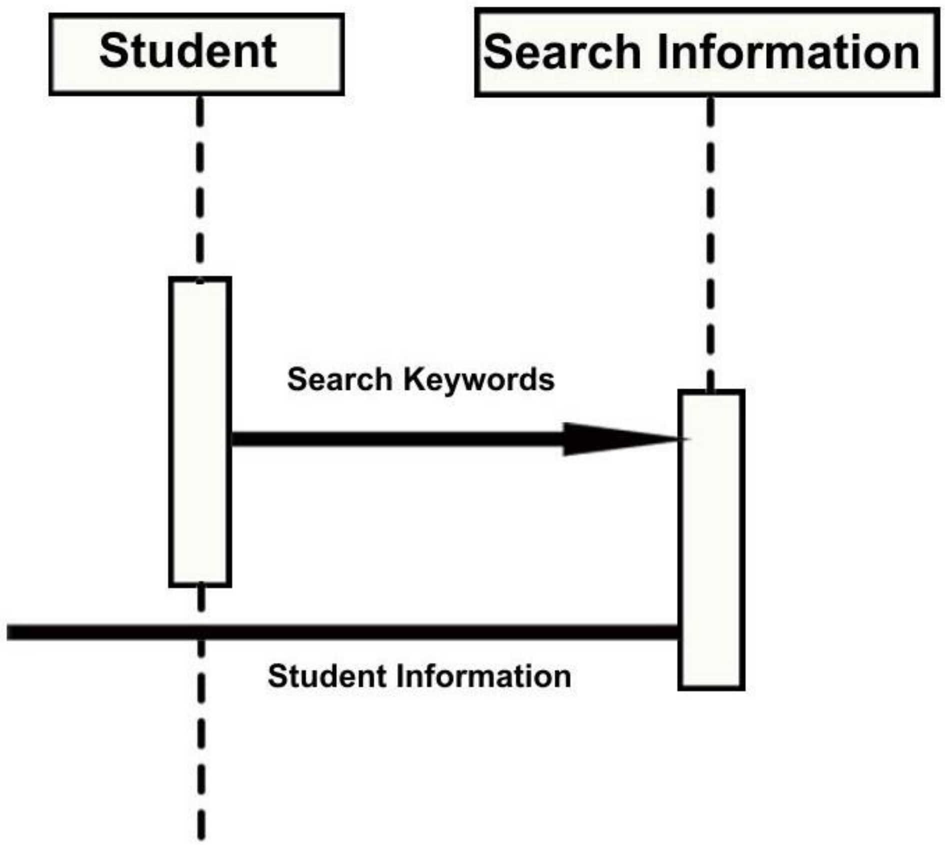
- Add Students, Operation and Show Students (for Students).
- Add Admin, Operation and Show Admin (for Admin).
- Course, Manage Course and Show Course.

- ➢ Next the user chooses whether he/she will be student or administrator and click on ‘Add Student’ or ‘Add Admin’ button according to that. A table is there to insert his/her information as requested and submit it then.
- ➢ If all the information are entered, a confirmation message show up to ensure the submission.
- ➢ Now the user needs to search own information. He/she has to enter into ‘Operation’ and put the correct id in search box there, all the information will then show up instantly.
- ➢ If user wants to update any data that has already been inserted, he/she just have to insert the new data in the correct box and click on update button. Here’s also the back option to go back further.
- ➢ All the information of the user that have registered will come up altogether in a sequenced table if user wants to visit ‘show students’ or ‘Show Admin’ options.
- ➢ Users also have that chance to select a course about which he/she wants to know the details.

- Course can be changed by entering into the ‘Manage Course’ option.
- Short keys are there on the top which will help users to go to home page or to logout or to know about our app in a shortcut way with the switches-
  - ctrl+A ➔ Home
  - ctrl+L ➔ Logout
  - ctrl+S ➔ About
- Last of all, the class called ‘databaseConnection’ has been created to connect our project with database with the help of the functions of this class.



<u>UML sequence diagram 2.1:</u> First user case of our SRS



UML sequence diagram 2.2: Second user case of our SRS

Risk Assessment: We need a solution that makes identifying and assessing risk simple. The most effective risk identification techniques focus on root cause. It's not enough to identify what happened. We need to understand why it happened in order to truly mitigate risk. If we want to find the risks in our project, first of all we must say about the programming side. If any code are written wrong even if a single comma/ semicolon/ dot are missing then the operation won't work out. We have used a database software named 'xampp'. If we don't start it and don't go to 'Admin' localhost site, the project will not run. That's mean internet connection is also necessary to work with our project.

Project Schedule and Team Structure:

Set specific day	We have set working day and time for each member in our group and the time is 6pm to 10pm.
Set tasks for team members	Every member got individual tasks for themselves. Such as, some worked with coding part, some had the task to find out the errors, some of us took care of the online side (downloading stuffs related to our project and solution of the errors) and some were to make the slide and document that we have to upload on GitHub.
Meeting time	We have scheduled a meeting time on every Sunday at 5pm.

Coding style guidelines

1-Formatting

1.1-Indentation

All indents are four spaces. All indenting is done with spaces, not tabs. Matching braces always line up vertically in the same column as their construct. All if, while and for statements must use braces even if they control just one statement.

1.2- Spacing

All method names should be immediately followed by a left parenthesis. All array dereferences should be immediately followed by a left square bracket Binary operators should have a space on either side. Unary operators should be immediately preceded or followed by their operand.
Commas and semicolons are always followed by whitespace.
All casts should be written with no spaces.
The keywords `if`, `while`, `for`, `switch`, and `catch` must be followed by a space.

1.3- Class Member Ordering

```
class Order
{
    // fields

    // constructors

    // methods
}
```

1.4- Maximum Line Length

Avoid making lines longer than 120 characters.

1.5- Parentheses

Parentheses should be used in expressions not only to specify order of precedence, but also to help simplify the expression. When in doubt, parenthesize.

2- Identifiers

All identifiers use letters ('A' through 'Z' and 'a' through 'z') and numbers ('0' through '9') only. No underscores, dollar signs or non-ascii characters.

2.1- Classes and Interfaces

All other identifiers, including (but not limited to) fields, local variables, methods and parameters, will use the following naming convention. This includes identifiers for constants.

The first letter of each word in the name will be uppercase, except for the first letter of the name. All other letters will be in lowercase, except in the case of an embedded acronym, which will be all uppercase. Leading acronyms are all lower case.

Test code is permitted to use underscores in identifiers for methods and fields.

3- Coding

3.1- Constructs to Avoid

Never use `do..while`.

Never use `return` in the middle of a method.

Never use `continue`.

Never use `break` other than in a switch statement.

3.2- Do Not Compound Increment or Decrement Operators

Use a separate line for an increment or decrement.

Never use pre-increment or pre-decrement.

3.3- Initialization

Declare variables as close as possible to where they are used.

3.4- Access

All fields must be private, except for some constants.

Thank You