DEPAETMENT OF COMPUTER SCIENCE AND ENGINEERING

University of Chittagong

Course Name : Software Engineering and Information System Lab

Course Code : CSE 516

Report 06

Detailed Design Document

Submitted to:

A.H.M Sajedul Haque

Assistant Professor,

Department of Computer Science & Engineering,

University of Chittagong.

Submitted by:

Marufa Sultana

ID: 18701008

Session: 2017-18

Report No-6
Detailed Design Documents
SchoolHub

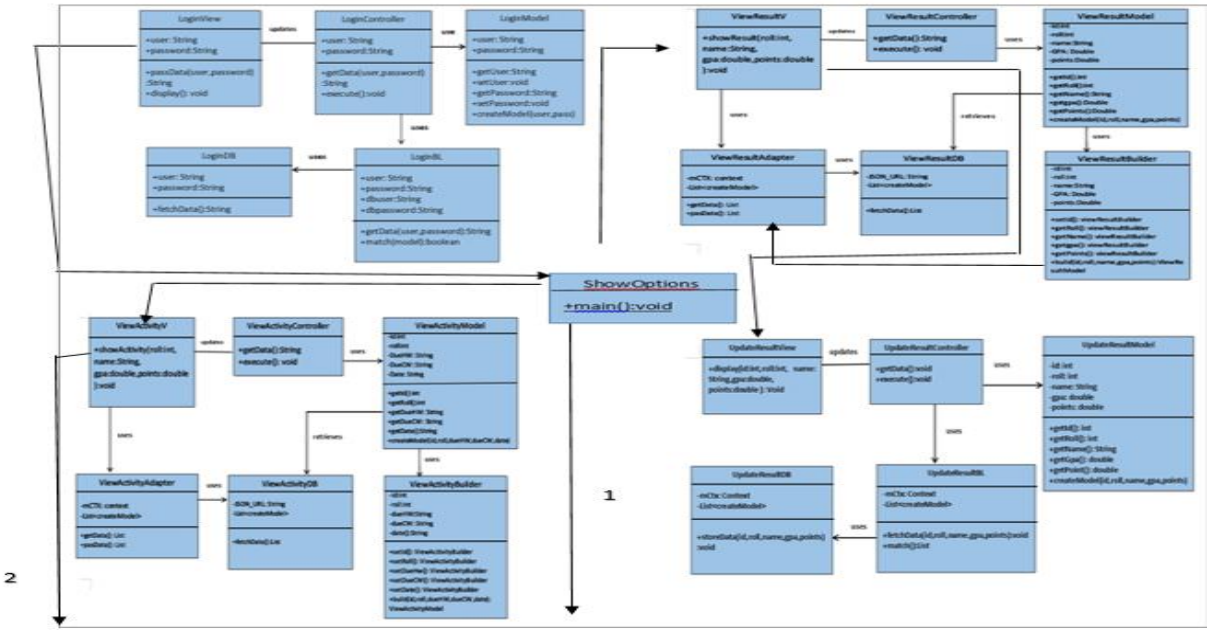
1. Introduction:

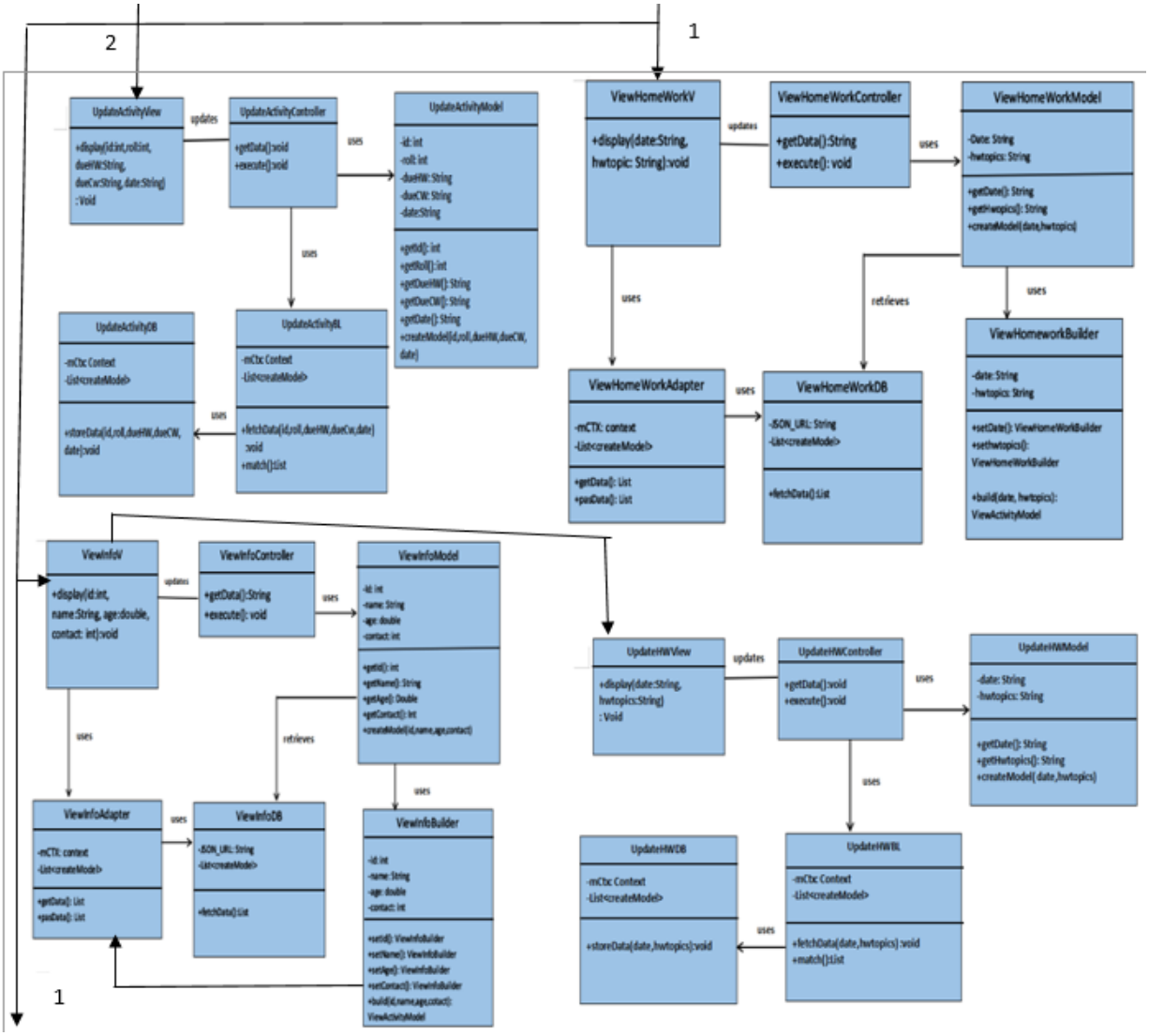
SchoolHub is a computerized system that is implemented to provide the guardians and the students with all the information of each students such as daily activities, exam results, notices and all other information of students and teachers, in order to reduce the workload of teachers and school authorities, also this system is developed to let guardian know about updates of their child's performance more efficiently and let them focus on improving their child's performance at ease. The purpose of this document is to illustrate the whole design details of the School App in detail. This document includes the conceptual class Diagrams of the whole system along with the implemented design patterns and the detailed class diagrams of each use case illustrating the structure of the whole system.

This document consists of 6 sections in total. The conceptual class diagrams of the previous document is shown in section 2, 3rd section describes the interfaces that are present in the system, the 4th section holds the description of the design patterns used to implement the system. After that, the detailed class diagrams of the used design patterns are described in the 5th section. And finally, the 7th section holds the conclusion of the document.

2. Architectural Design

Here, the architectural design of the whole system is illustrated in the following figure:




2
1
UpdateActivityView
+display(id:int,roll:int, dueHW:String, dueCw:String, date:String) :Void
updates
UpdateActivityController
+getData():void
+execute():void
uses
UpdateActivityModel
-id: int
-roll: int
-dueHW: String
-dueCW: String
-date:String
+getId(): int
+getRoll(): int
+getDueHW(): String
+getDueCW(): String
+getDate(): String
+createModel(id,roll,dueHW,dueCW, date)
uses
UpdateActivityDB
-mCtx: Context
-List<createModel>
+storeData(id,roll,dueHW,dueCW, date):void
uses
UpdateActivityBL
-mCtx: Context
-List<createModel>
+fetchData(id,roll,dueHW,dueCw,date) :void
+match():List
ViewHomeWorkV
+display(date:String, hwtopic: String):void
updates
ViewHomeWorkController
+getData():String
+execute(): void
uses
ViewHomeWorkModel
-Date: String
-hwtopics: String
+getDate(): String
+getHwopics(): String
+createModel(date,hwtopics)
uses
retrieves
uses
ViewHomeworkBuilder
-date: String
-hwtopics: String
+setDate(): ViewHomeWorkBuilder
+sethwtopics(): ViewHomeWorkBuilder
+build(date, hwtopics): ViewActivityModel
ViewHomeWorkAdapter
-mCTX: context
-List<createModel>
+getData(): List
+postData(): List
uses
ViewHomeWorkDB
-JSON_URL: String
-List<createModel>
+fetchData():List
ViewInfoV
+display(id:int, name:String, age:double, contact: int):void
updates
ViewInfoController
+getData():String
+execute(): void
uses
ViewInfoModel
-id: int
-name: String
-age: double
-contact: int
+getId(): int
+getName(): String
+getAge(): Double
+getContact(): int
+createModel(id,name,age,contact)
uses
retrieves
uses
ViewInfoAdapter
-mCTX: context
-List<createModel>
+getData(): List
+postData(): List
uses
ViewInfoDB
-JSON_URL: String
-List<createModel>
+fetchData():List
ViewInfoBuilder
-id: int
-name: String
-age: double
-contact: int
+setId(): ViewInfoBuilder
+setName(): ViewInfoBuilder
+setAge(): ViewInfoBuilder
+setContact(): ViewInfoBuilder
+build(id,name,age,cotact): ViewActivityModel
1
UpdateHWView
+display(date:String, hwtopics:String) : Void
updates
UpdateHWController
+getData():void
+execute():void
uses
UpdateHWModel
-date: String
-hwtopics: String
+getDate(): String
+getHwtopics(): String
+createModel(date,hwtopics)
uses
UpdateHWDB
-mCtx: Context
-List<createModel>
+storeData(date,hwtopics):void
uses
UpdateHWBL
-mCtx: Context
-List<createModel>
+fetchData(date,hwtopics) :void
+match():List

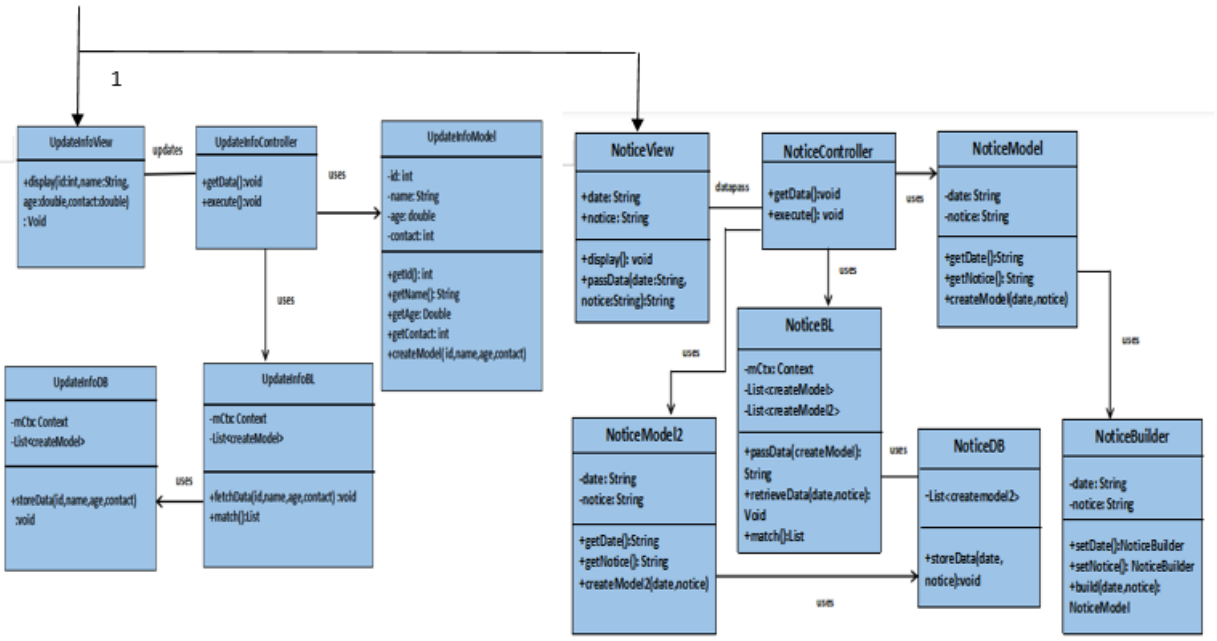


Figure 2.1: Architectural Design

3. Architectural Pattern

An architectural pattern is a general, reusable solution to a commonly occurring problem in software architecture within a given context. Architectural patterns are a method of arranging blocks of functionality to address a need.

Here, the architectural pattern of the whole system is illustrated in the following figure:

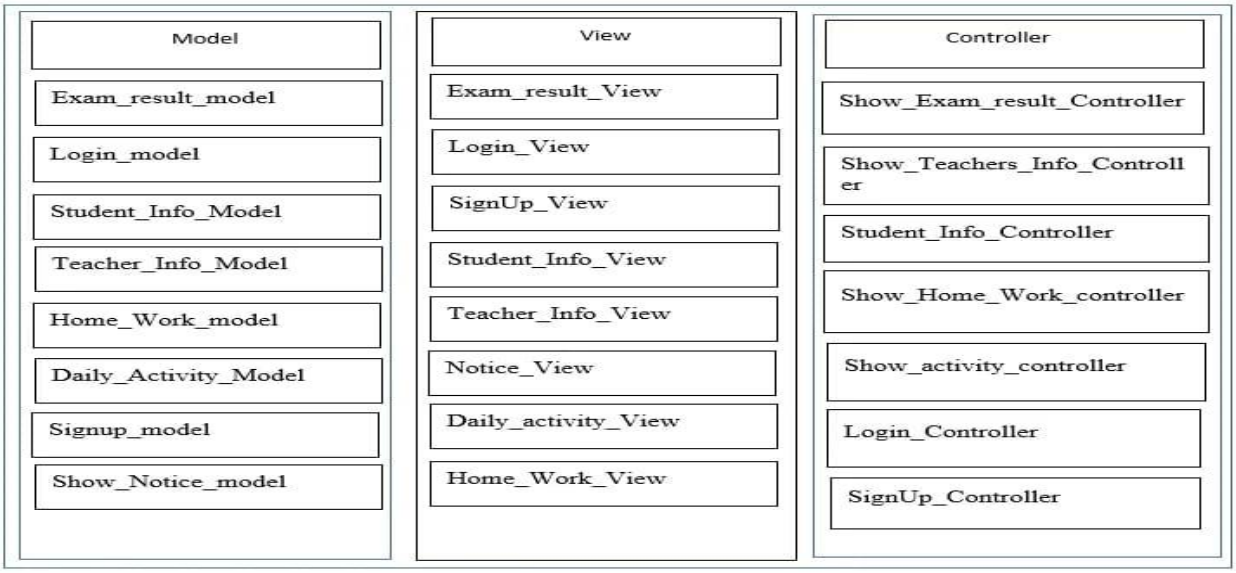


Figure 2.1: Architectural Pattern

4. Detailed Design

Here, we used more than 40 classes in our system. The pseudo code for the important classes are provided below bellow:

4.1 Login as teacher

Public void match(username, password, Teacher Code){

BEGIN

GET Username

GET Password

GET Teacher Code

IF (Username == EnteredUsername && Password == EnteredPassword && Teacher Code) THEN

Login Successful

Login Failed, Try Again!

END IF

END

}

4.2 Login as guardian

Public void match(username, password){

BEGIN

GET Username

GET Password

IF (Username == EnteredUsername && Password == EnteredPassword) THEN

Login Successful

Login Failed, Try Again!

END IF

END

}

4.3 Sign up

insertdatanewway2(){

BEGIN

Email field: input type = email, placeholder: “Your Email”

Name field: input type = username, placeholder: “Your Name”

Password field: input type = password, placeholder: “Your Password”

Signup submit: value: “Sign Up”, default state, disabled

IF email is blank

Error message: “please insert your email.”

IF name is blank

Error message: “please insert your Username.”

IF password is blank

Error message: “please insert your password.”

IF email && username && password confirmation all contain valid values

Enable Signup Submit

END}

4.4 Add Teacher info

Public void addData(){

BEGIN

Give Teachers information including Name, Gender, Phone Number, mail address Connect Database.

IF

Add Information after creating connection.

OUTPUT Inserted data successfully.

Else

OUTPUT ERROR

END

}

4.5 Add result

Public void insertdatanewway2(){

Give Students information including Name, Roll, GPA, Total marks.

Connect Database.

IF Connection successful

Add Information.

OUTPUT Inserted data successfully.

Else

OUTPUT ERROR

END

}

4.6 Show Exam result

Public void showdata(){

BEGIN

IF login as Student

Retrieve Students result information from database

Show Students Result

Else IF login as Teacher

Inert data into field

END

}

4.7 Show Students Information

Public void showdata(){

BEGIN

Retrieve Students information from database

Show Students Information

END

}

4.8 Show teachers info

Public void showdata(){

BEGIN

Retrieve information from database

Show Teachers detailed Information

END

}

5. System Demonstration

School App is an application for android based device with several functionalities. The functionalities of the system is shown using screen shots in bellow:

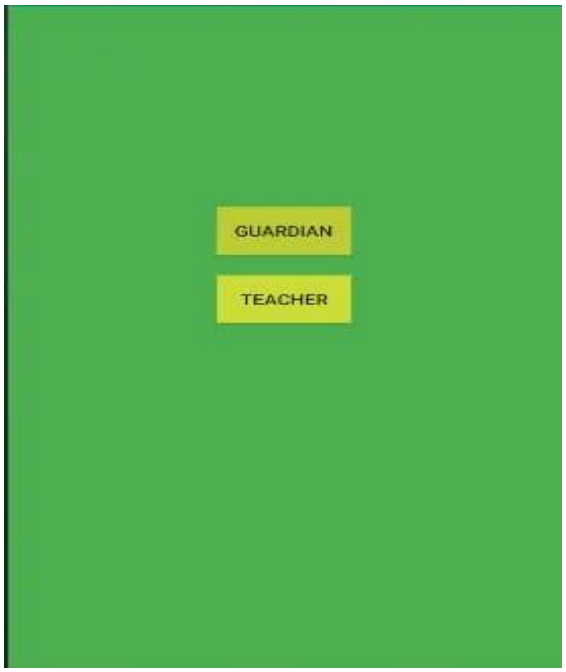


Figure 5.1 : Home page

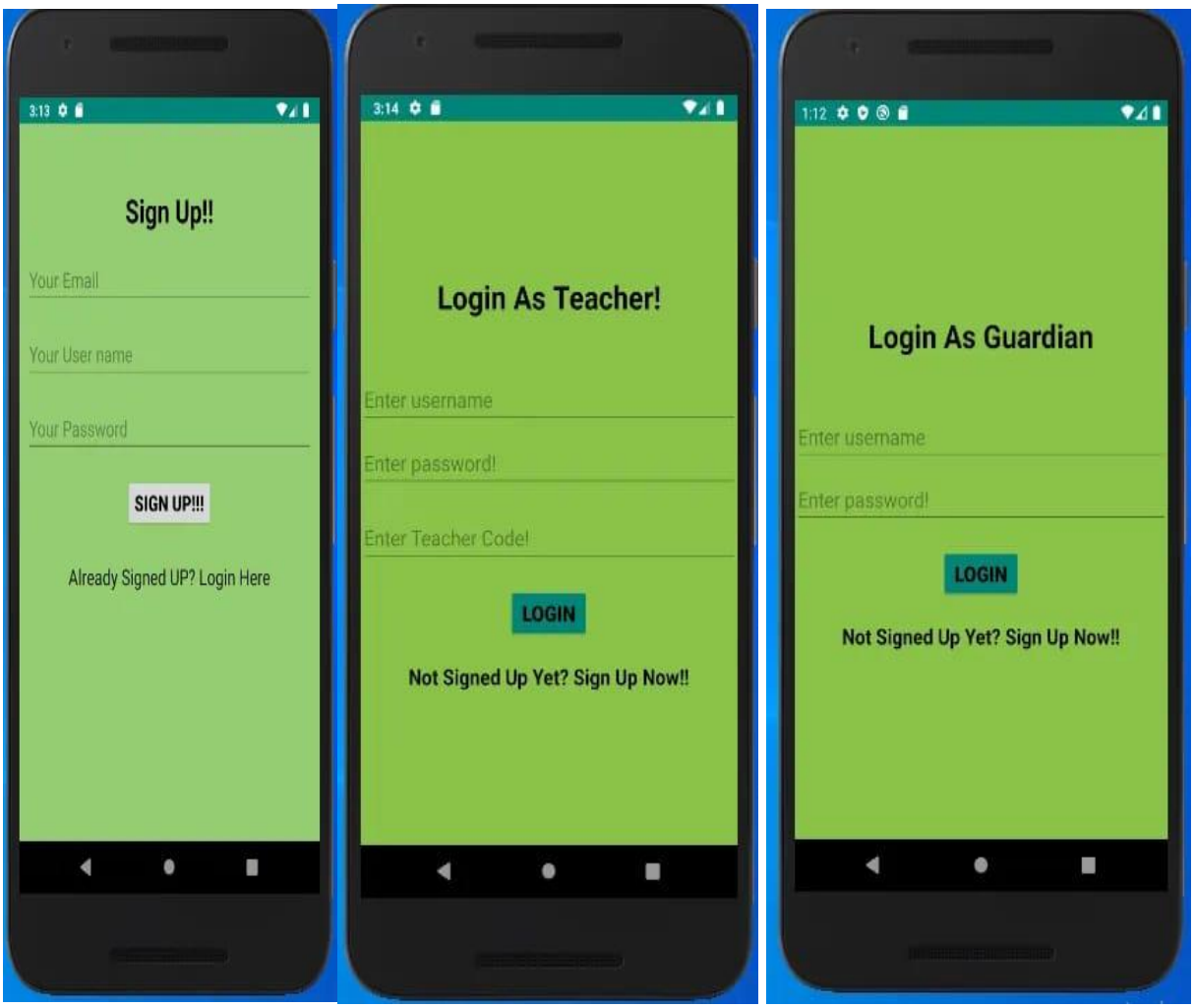


Figure 5.2 Login Prompt Demonstrations

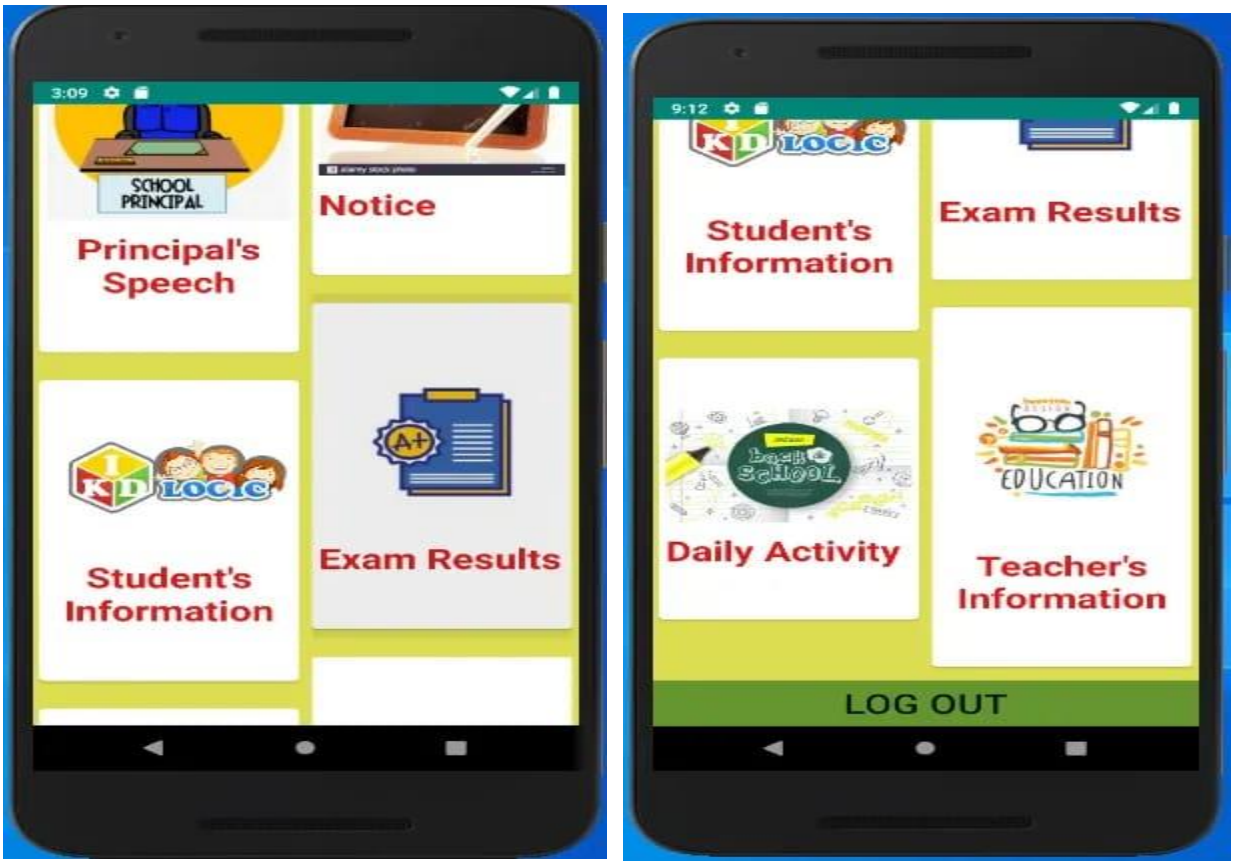
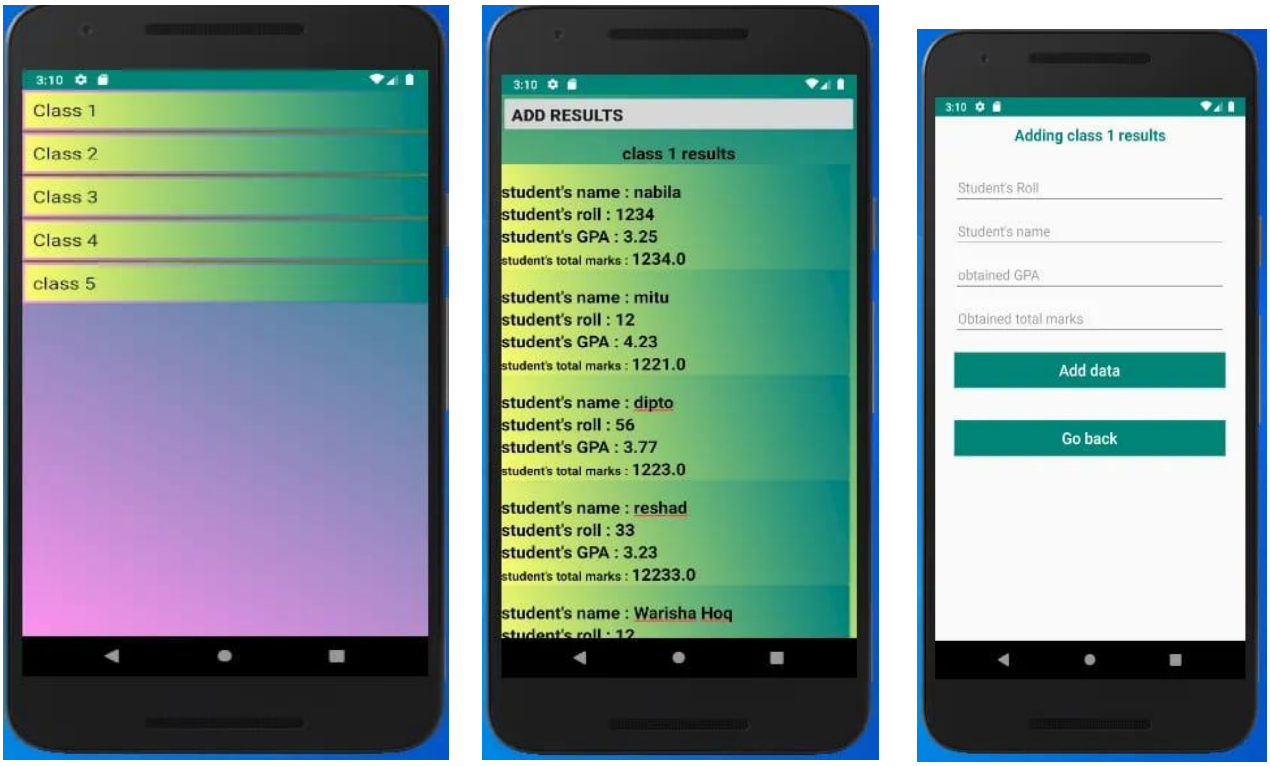


Figure 5.3: User Dashboard Demonstration



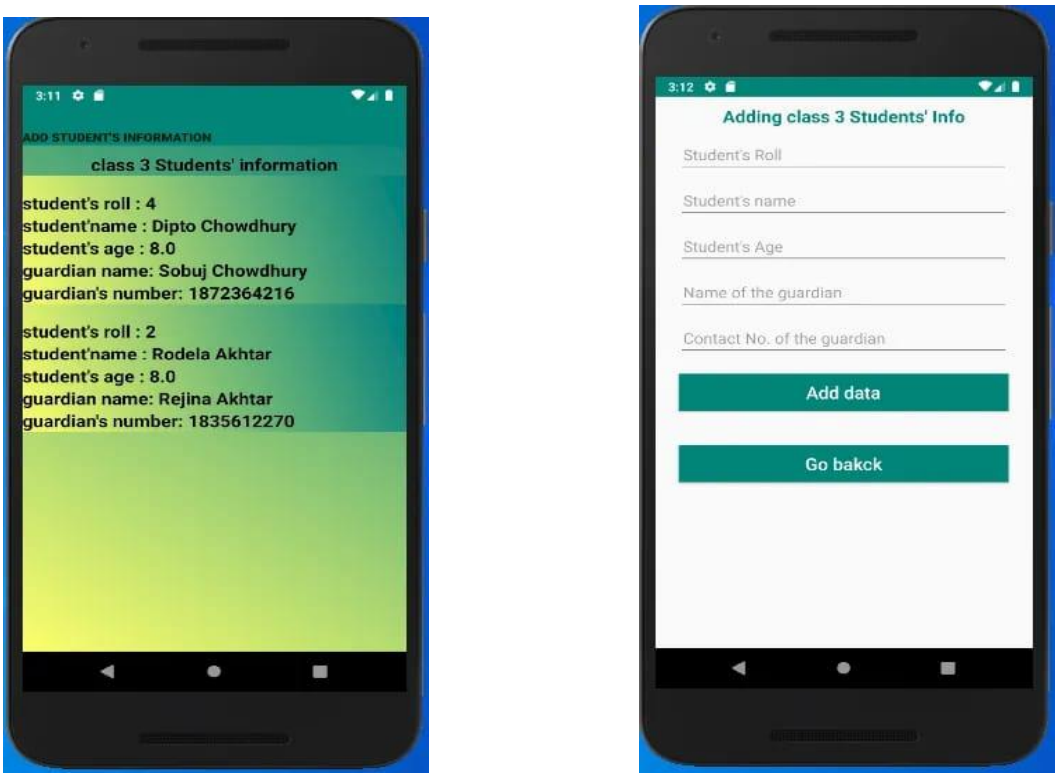


Figure 5.4: Manipulation of Demonstrate information (Students)

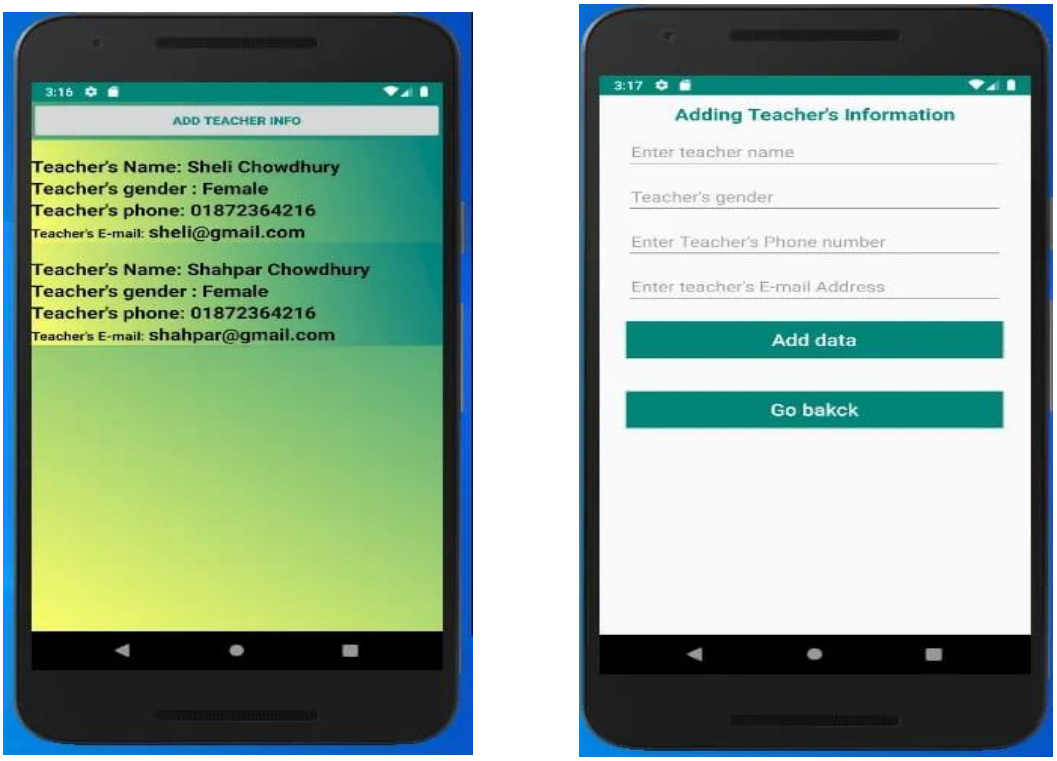


Figure 5.4 : Manipulation of Demonstrate information (Teacher)

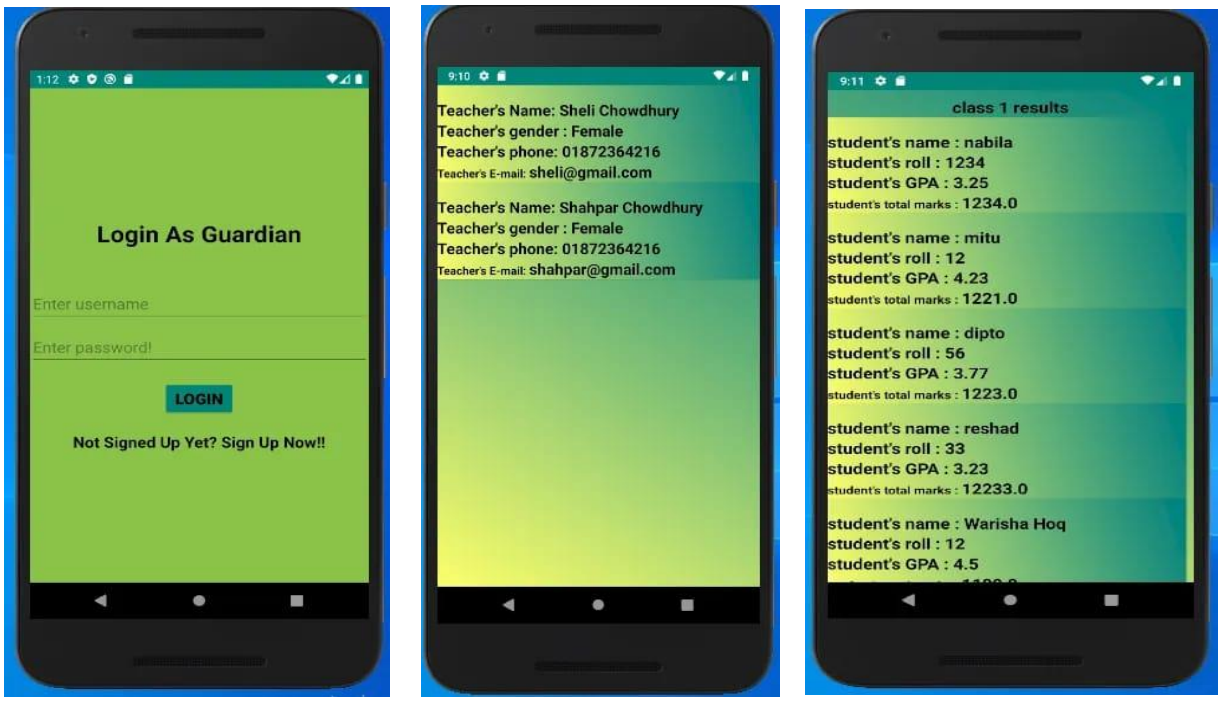


Figure 5.5 : Guardian Demonstration

6. System Limitation

There exit no system developed by man that is perfect and complete. School App system has also some limitation on its functions. The limitation of the system is given bellow:

1. Inputs to the system has to be given manually by keyboard.
2. There are no option for notifying users when necessary.
3. The system is available only for android based devices.
4. Any type of data can be taken in all the input. Data type is not selected instantly.

7. Future Work

Technology is ever changing. To cope with the change and the satisfaction of a user, a system need to be kept constantly updated. To overcome the limitation of the system as well as to satisfy the need of its user, following changes are planned to make in future versions of the School App system.

1. Notification system will be added to notify the user.
2. Similar system will be developed for PC platform.
3. We will fixed data type for all the input.

8. Conclusion

We have tried our best to solve the problem that we have been aiming since the beginning, yet we have still got some limitations, we are prune to overcome those limitations in no time and serve a working computerized system accordingly.