Gaza –Islamic University
Faculty of Information Technology
Computer Science Department

الجامعة الاسلامية - غزة
كلية تكنولوجيا المعلومات
قسم علم الحاسوب

QAR - Attendance Management System using NFC technology

A Graduation Project Presented to the
Faculty of Information Technology
The Islamic University of Gaza

By
Abd Alazez M. Alswaisi 120140980
Ahmed R. Alhessi 120141465
Hazem H. Hussain 120141867

Supervisor
Dr. Motaz saad

26 – Jun – 2018

Table of Contents

Table of Figures

Table of Tables

List of Abbreviations

QAR	Quick Attendance Registration
NFC	Near Field Communication
ER	Entity Relationship
API	Application Programming Interface
FK	Foreign Key
PK	Primary Key
JSON	JavaScript Object Notation
AEBAS	Aadhar Enabled Biometric Attendance System
SDK	Software Development Kit
IDE	Integrated Development Environment
HTML	Hypertext Markup Language
PHP	Hypertext Preprocessor
URL	Uniform Resource Locator

Abstract

As mobile usage keep growing up, the demand on new software is increasing as well, and in different fields. One of these fields is attendance registration. This project aims to develop an application to monitor students' attendance during exams and provides synchronization with a backend; our application helps to facilitate the track of exams for controllers (supervisors) to track student attendance. We use Agile methodology to develop this application. This application saves time, applies authentication and prevents forgery, and it can be extended for other areas rather than educational institutions. Near field communication technology is using magnetic fields to send and receive data between two NFC cards; we use NFC technology as it's now supported by many of smartphones and no need for a specific reader to do the job, and all students have NFC card attached to their ID and we are going to use that card. QAR is a very useful, easy to use and helpful application for controlling and managing exams.

Chapter One

Introduction

Chapter 1: Introduction

The main purpose of using phones was to make calls, but now mobiles can do so much more. The emergence of new touchscreen smartphones allows us to access the internet and social media, play music, video, playing games, navigation, travelling, education and much more.

Due to smartphone revolution in the world, the total number of mobile phone users for 2018 is forecast to reach 5.1 billion of users. And the number of mobile phone users in the world is expected to reach the 5.6 billion of users by 2019. [1]

One of the major uses of smart phone is the education utilities, such as grades, courses tables, exams and attendance applications. One of the ways that it can help teachers that it replaces all the paper work by easy, simple, fast, reliable and secured applications that makes all works easier and faster than the normal old paper way.

There are many applications developed to solve attendance automation problem such as “Fees & Attendance Registration”[2] which records attendance by entering data manually to the system, and “Time NFC Attendance”[3] which use NFC technology for registering attendance and “AEBAS Attendance”[4] which use fingerprint to register attendance.

There are many applications which are used for students or teachers attendance. Our application is different that we use the NFC technology and we specify the application for exams attendance only. The main idea of this application is that instead of the need of the supervisor action to check each student’s identification and mark him/her as attendant or absent, we store student identification in NFC card and the application will check student ID from the card and mark him/her as absent or attendant using the attached NFC reader in smartphone.

NFC (Near Field Communication): Close communication technology for transferring data between two close NFCs, one of them acts as reader and the other as writer, in a distance of 15 centimeters and 13.56 MHz frequency with about half megabyte

transferring speed, due to the short range of transmission, NFC based transactions are inherently secure.

Figure 1: NFC Card [5]

Why not Bluetooth? Bluetooth range is up to 30m and this range is enough to use another Bluetooth device from a distance to make fake registration, it also need up to 6 seconds set up time and there is configuration needed to connect it, not like NFC that connects automatically when it's in 10cm range.

NFC can be used for paying (Master Card & VISA), getting access, initiating service and for sharing contacts, photos, videos, files and more. [6]

	NFC	RFID	IrDa	Bluetooth
Setup Time	< 0.1ms	< 0.1ms	~ 0.5s	~ 0.6s
Range	Up to 10cm	Up to 3m	Up to 5m	Up to 30m
Usability	Human Centric, Easy, Intuitive, Fast	Item Centric, Easy	Data Centric, Easy	Data Centric, Medium
Selectivity	High, Given, Security	Partly Given	Line of Sight	Who are you?
Use Cases	Pay, Get Access, Share, initiate Service, Easy Setup	Item Tracking	Control & Exchange data	Network for data exchange, Headset
Consumer Experience	Touch, Wave, Simply Connect	Get Information	Easy	Configuration needed

Table 1: Comparing NFC to other range communication technologies [6]

1.1 Statement of the problem

The main problems in registering attendance during exams are time and efforts. Time is the most important factor during exams; we can't waste time during exams, and authentication which is required to confirm students' identity during exams, so there is a need for mobile application to make this process easy to save time and effort and prevents forgery.

1.2 Objectives

1.2.1 Main Objective

The main objective of this project is to develop an application that can replace paper system with automated system using NFC technology.

1.2.2 Specific Objectives

- Building Android application for reading NFC card.
- Make the application compatible with many different cards.
- Develop a simple program to write on NFC cards for testing.

1.3 The importance of the project

This project is important because it helps controlling exams in many ways, as it's simple and easy to use consume less time to check absent student, and is more secured as it helps controller check students identifications using simple technology that provide much time with less effort (NFC technology).

1.4 Scope and Limitations of the project

This application is developed to help controlling exams, this application is only be able to read NFC cards not to write on them and it is limited for attendance in exams and can be extended for other purposes later. Authentication is not automatically checked by our application, it only helps check personal photo and name inside the application based on information stored in NFC card. But not all mobiles have a NFC Reader, and application would not support all types of NFC cards.

Chapter Two

Related Works

Chapter 2: Related Works

1- **Fees & Attendance Registration**: The application does not use any distinctive technology as it registers the presence manually within the application but it is very easy to use the backup function and restore the data. Allows data to be saved from the application and transferred to another device to ensure continuity without any loss of data when changing devices.

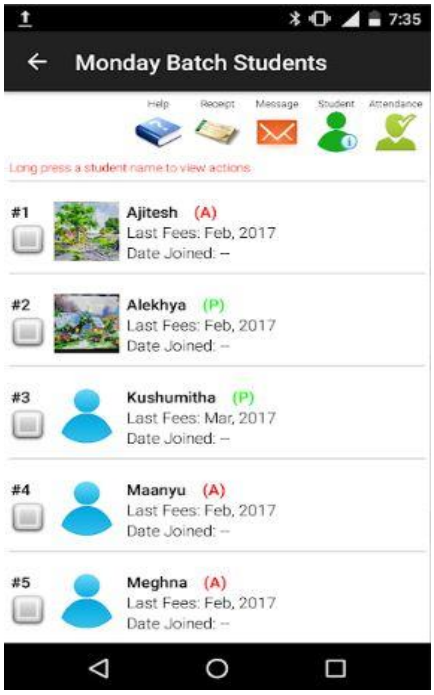
The Application has the following features:

Collect and manage student details, Group students into batches, Add new groups and the order in which groups will be displayed, Record Attendance of students, View the attendance history of a student, Data restore feature that allows data to be imported into the application from a previously backed up file in the same device or copied from a different device. [2]

Rating in Store: 4/5

Review:

Great app and very simple to use and very useful but this app does not have a good interface and not sync the database to cloud and there are problems sending SMS messages and not fast at work, because he's manually recording attendance.



2- **Time NFC Attendance**: Is an application that uses NFC technology in the presence and is characterized by the application in ease of use and works on the android environment and works without the internet and later synchronizes data when connected On the Internet. [3]

Rating in Store: 5/5

Review

3- AEBAS Attendance: This application that records attendance using a fingerprint, AEBAS - (Enabled Biometric Attendance System) app is for Android based mobiles. The purpose of this AEBAS application is to view own profile, attendance register, apply leave, view monitoring groups and there members & attendance. [4]

Rating in Store: 3/5

Review: The application is good and easy to use but there is slow response to the finger optics

Critical analysis:

In previous applications there was weakness in some aspects such as ease of use, design is weak and what we will improve in order to be our application is best and also in applications some previous applications were not here ensure the integrity of the data from the damage we will try to put several ways in order to ensure peace e data in all projected cases.

Chapter Three

Methodology

Chapter 3: Methodology

During the application life cycle, we follow Agile - extreme programming. Extreme programming of five phases as follows [7]:

Phase 1: Project Planning

During this phase we gathered user stories, communication feedback loop, story estimations, acceptance test criteria and scenarios for user stories

Phase 2: Analysis and Risk Management

Based on project planning phase, where an analyst gathered detailed requirement/user stories, UML diagrams, use cases and sequence diagram will be developed, so the requirement we needed to implement the system will be ready and some cases will return to gather more details form users and project planning.

Phase 3: Design

In this phase we developed system architecture, Entity-Relationship (ER) diagram for database and interfaces design for the application that it is changeable depending user needs and requirements so we developed prototype for interfaces then test it with user acceptance.

Phase 4: Implementation

Implementation constitutes the most important phase in the Extreme Programming life cycle. XP gives priority to the actual coding over all other tasks such as documentation to ensure that the customer receives something substantial in value at the end of the day.

Phase 5: Testing

Extreme program integrates testing with the implementation phase rather than at the end of the implementation phase. All codes have unit tests to eliminate bugs, and the code passes all such unit tests before release. Another key test is customer acceptance tests, based on the customer specifications. Acceptance test run at the completion of the coding, and the developers provide the customer with the results of the acceptance tests along with demonstrations.

Now we can summarize activities in this level:

1- Product Life Cycles
2- Releases
3- Iterations
4- Tasks
5- Development
6- Feedback

The following figure shows Agile methodology life cycle.

Figure 2: Agile Methodology

3.1 Tools and equipment and methods

3.1.1 System requirements

- Hardware
    1. NFC card.
    2. Mobile that supports NFC reader.
- Software
    1. Android Studio
    2. Java Language
    3. Photoshop
    4. Trello

3.2 Time Table

The following figure shows our time table based on agile methodology.

	Task Mod	Task Name	Duration	Start	Finish
1		**Project QAR**	**189 days**	**02/10/2017**	**21/06/2018**
2		**Iteration 1**	**90 days**	**02/10/2017**	**02/02/2018**
3		Learn about NFC technology	12 days	02/10/2017	17/10/2017
4		Collect Requirements	20 days	18/10/2017	14/11/2017
5		Review related works	11 days	15/11/2017	29/11/2017
6		Buy the "NFC" card from the AliExpress	6 days	30/11/2017	07/12/2017
7		Write Proposal	41 days	08/12/2017	02/02/2018
8		**Iteration 2**	**21 days**	**01/02/2018**	**01/03/2018**
9		Design Login screen	3 days	01/02/2018	05/02/2018
10		Design Attendance Registration Screen	3 days	06/02/2018	08/02/2018
11		Implement Login Screen	3 days	09/02/2018	13/02/2018
12		Implement Manually record attendees	3 days	14/02/2018	16/02/2018
13		Application settings	3 days	19/02/2018	21/02/2018
14		Desgin Application Settings Screen	3 days	22/02/2018	26/02/2018
15		Buy the "NFC" card from the AliExpress	1 day	27/02/2018	27/02/2018
16		Test the application	2 days	28/02/2018	01/03/2018
17		**Iteration 3**	**21 days**	**02/03/2018**	**30/03/2018**
18		Buy the "NFC" card from the AliExpress	3 days	02/03/2018	06/03/2018
19		Design time Table screen	4 days	02/03/2018	07/03/2018
20		Design Send requests screen	4 days	08/03/2018	13/03/2018
21		View next exam	4 days	14/03/2018	19/03/2018
22		Implement View controller requests	4 days	20/03/2018	23/03/2018
23		Implement View controller Table	4 days	26/03/2018	29/03/2018
24		Test the application	1 day	30/03/2018	30/03/2018
25		**Iteration 4**	**27 days**	**31/03/2018**	**07/05/2018**
26		Design all exams screen	3 days	02/04/2018	04/04/2018
27	✓	Design Contact Screen	2 days	05/04/2018	06/04/2018
28		Contact Course Teacher	4 days	09/04/2018	12/04/2018
29		Maintain Schedule	5 days	13/04/2018	19/04/2018
30		Handle Notification	3 days	20/04/2018	24/04/2018
31		Send requests for the control	4 days	25/04/2018	30/04/2018
32		Implement View All Exams	3 days	01/05/2018	03/05/2018
33		Test the application	2 days	04/05/2018	07/05/2018
34		**Iteration 5**	**30 days**	**08/05/2018**	**18/06/2018**
35		Inquire Faculty of Engineering about NFC	4 days	08/05/2018	11/05/2018
36		Design Receive notifications Screen	4 days	14/05/2018	17/05/2018
37		Design Student Display Screen	4 days	17/05/2018	22/05/2018
38		Implement Receive notification from Admin	4 days	23/05/2018	28/05/2018
39		Implement Register attendees using NFC	4 days	29/05/2018	01/06/2018
40		Implement View all students who attend the exam	4 days	04/06/2018	07/06/2018
41		Inform Controller	3 days	08/06/2018	12/06/2018
42		Test the application and get feedback from the user	3 days	13/06/2018	15/06/2018

Figure 3: Time Table

The following figures from 3 to 6 show the tasks for each iteration and the duration it needed to be done.

Figure 3 shows iteration 1 and its tasks.

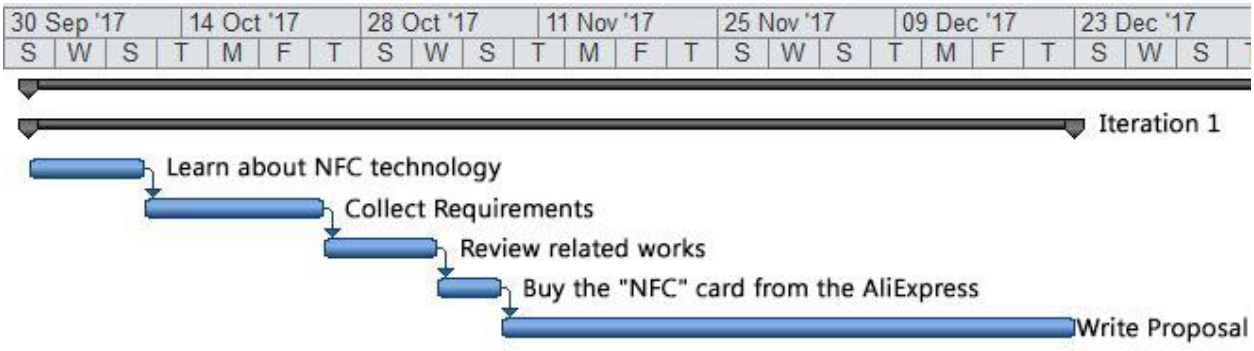


Figure 4: Iteration 1

Figure 4 shows iterations 2 and 3 and their tasks.

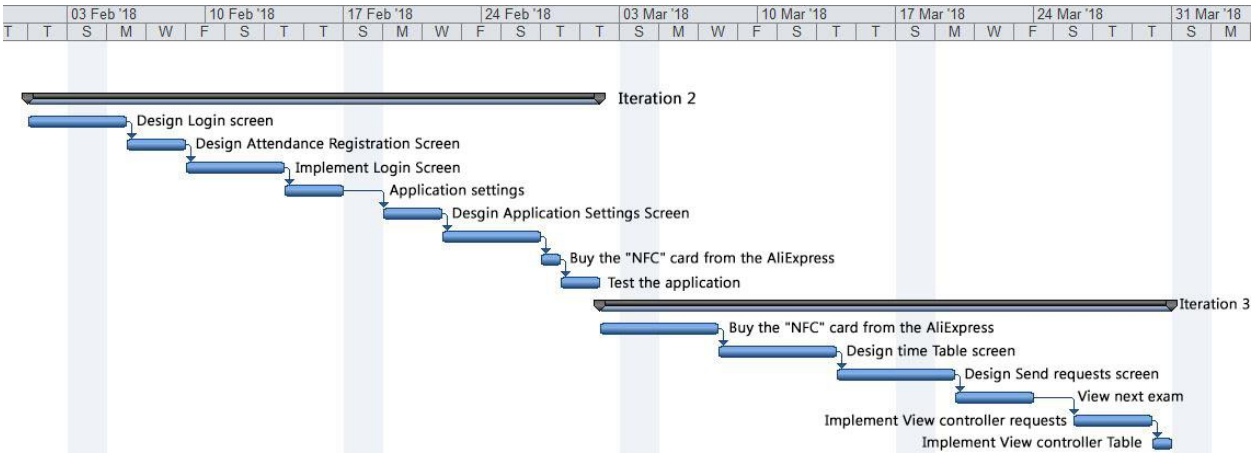


Figure 5: Iterations 2 & 3

Figure 5 shows iteration 4 and its tasks.

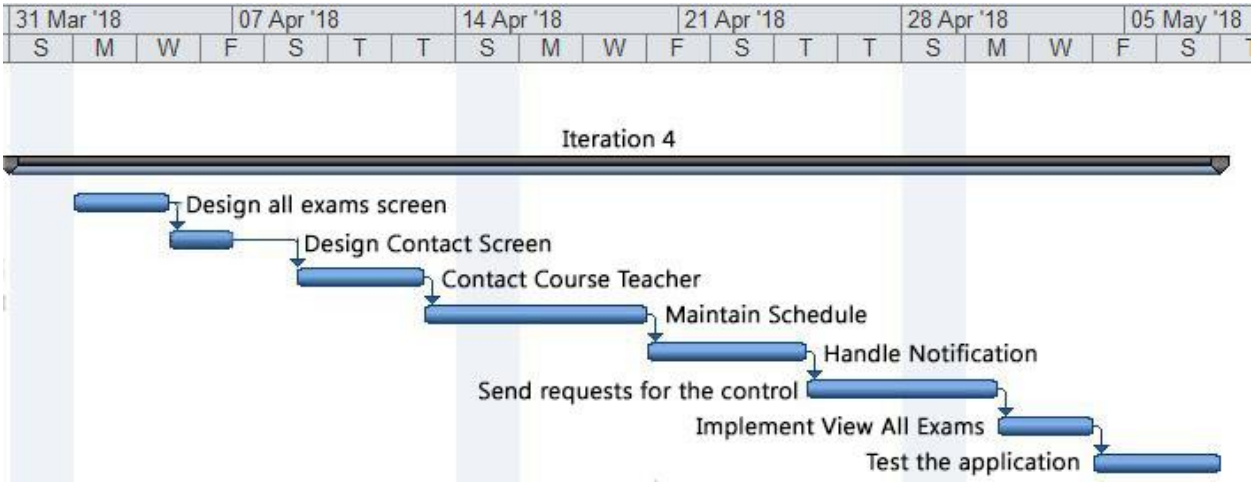


Figure 6: Iteration 4

Figure 6 shows iteration 5 and its tasks.

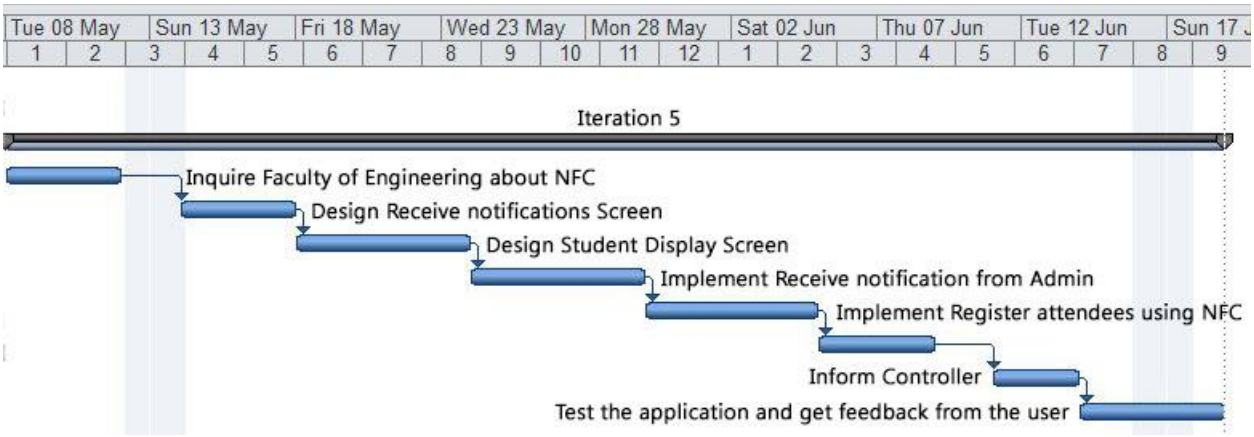


Figure 7: Iteration 5

Chapter Four

System Analysis

Chapter 4: System Analysis

In this section we present our system requirements; we present our system functionalities and then will explain our use case diagram.

We develop our application to be generic that works smoothly with any university requirements with the possibility to make customized versions of our system.

We used Passport plugin with our API for authorization that create and maintain a security token for all users to insure that no one can use a fake account.

A token is a piece of data created by server, and contains information to identify a particular user and token validity. The token contains the user's information, as well as a special token code that user can pass to the server with every method that supports authentication, instead of passing a username and password directly.

Token-based authentication is a security technique that authenticates the users who attempt to log in to a server, a network, or some other secure system, using a security token provided by the server.

An authentication is successful if a user can prove to a server that he or she is a valid user by passing a security token. The service validates the security token and processes the user request.

After the token is validated by the service, it is used to establish security context for the client, so the service can make authorization decisions or audit activity for successive user requests. [7]

4.1 System Functionalities

- Controller
  1. The Controller should be able to see time table (Show Time Table)
  2. The Controller should be able to see next exam (Show Next Exam)
  3. The Controller should be able to request absence from exam (Absence of Exam)
  4. The Controller should be able to send requests to Admin (Send request to Admin)
  5. The Controller should be able to check student's ID (Check student id)
  6. The Controller should be able to register student's attendance using NFC (Register student attendance)
  7. The Controller should be able to register student's attendance manually (Register student attendance.
- Admin
  1. The Admin should be able to assign new controller to exam (Assign new Controller)
  2. The Admin should be able to handle notifications from Controllers (Handle notifications)
  3. The Admin should be able to send notifications to Controllers (Send notifications to Controller)
  4. The Admin should be able to call course teacher (Call course teacher)

4.2 System Requirements: Use case diagram

The following figure shows the use case diagram for the system that includes all functions that our system provides.

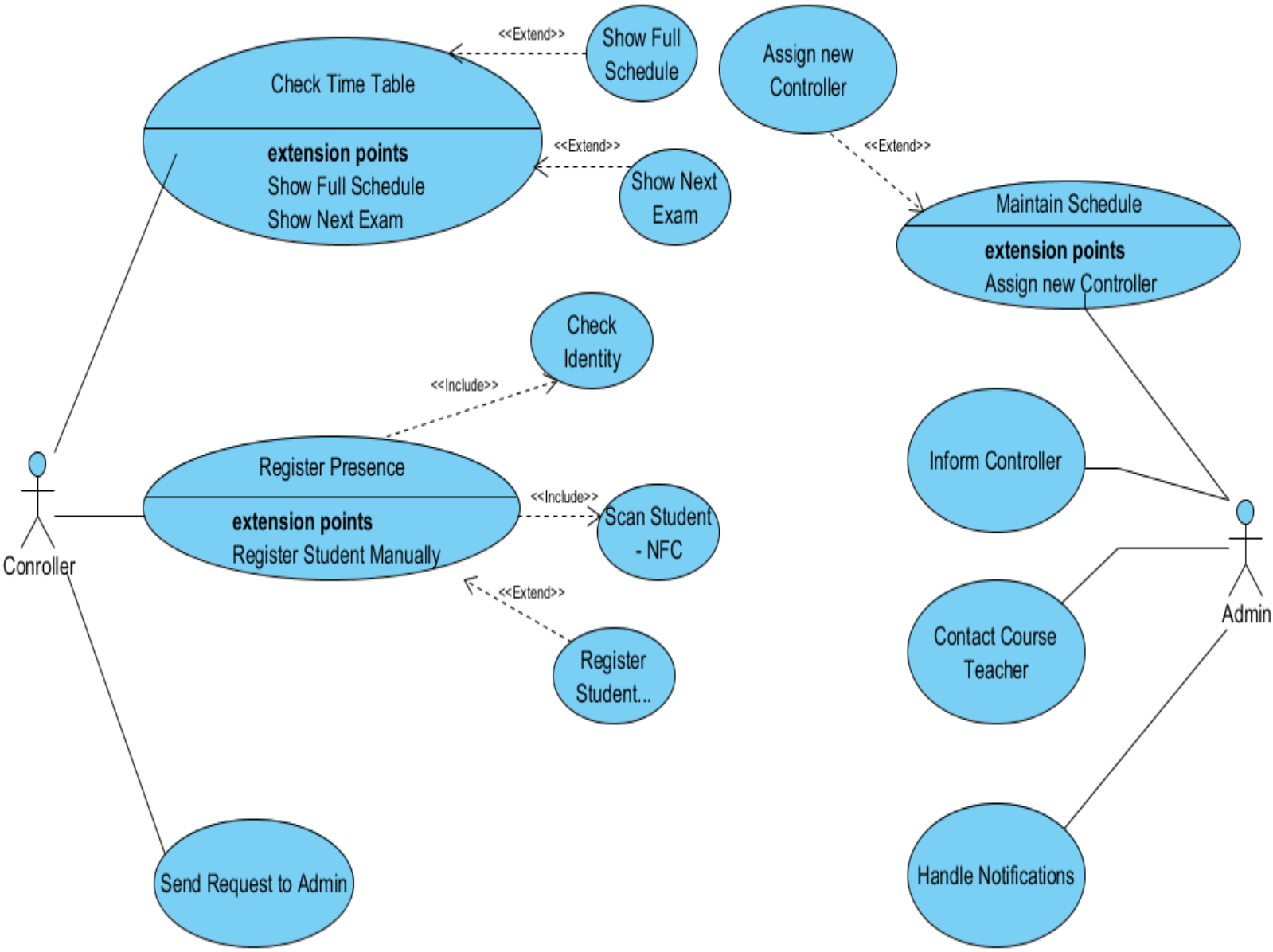


Figure 8: Use case diagram

The following table is a template table to explain the use case that will be filled on the next tables.

ID	Use case ID
Name	Use case Name
Description	A brief description about Use case
Priority	1,2,3 or removed
Status	High level description, Detailed description, Scenarios, Complete, Coded, Tested, Incomplete Pates, etc.
Actors	Actors names which are involved in Use Case.
Condition-Pre	Conditions required before the start of the use case.
Inputs	The inputs that enter to Use Case.
Normal Flow of Events	Steps to reach the Post-Conditions.
Otherwise Flow	The other flow if occur errors or exceptions
Condition-Post	Conditions that will be achieved after executing use case.
Output	The outputs that out from Use Case.
Constraints	List of constraints for this requirement.
Exceptions	Wrong behavior appears in normal flow.

Table 2: Use case details template

Tables from 2 to 7 explain use case and each function that our system provides.

ID	1
Name	Check Time Table
Description	Controller will be able to check his/her time table for the exams
Priority	1
Status	Completed, Coded, Tested
Actors	Controller
Condition-Pre	No pre-conditions
Inputs	No inputs
Normal Flow of Events	1. Controller will open time table page in the app 2. Application requests time table from database 3. Show time table on the screen
Otherwise Flow	No time table for this controller
Condition-Post	The system could find a result table.
Output	Display time table
Constraints	Constraints No
Exceptions	No exceptions

Table 3: Use case 1 - Check Time Table

ID	2
Name	Register Presence
Description	Controller will be able to register student's presence in a specific exam.
Priority	1
Status	dCompleted, Code, Tested
Actors	Controller
Condition-Pre	Use student's NFC card
Inputs	Information stored in the NFC card.
Normal Flow of Events	1. Controller will bring student's NFC close to mobile 2. Application will scan NFC card 3. Application will add student attendance to database
Otherwise Flow	This student is not registered in this exam.
Condition-Post	Add student's attendance into database
Output	Student's image, name and ID
Constraints	Valid student's NFC
Exceptions	Student forget NFC card, Controller will use manual registration.

Table 4: Use case 2 - Register Presence

ID	3
Name	Send request to Admin
Description	Controller will be able to send a request to Admin
Priority	1
Status	Completed, Coded, tested
Actors	Controller
onditionC-Pre	conditions-No pre
Inputs	Request type, text
Normal Flow of Events	Controller will open request page in the app. Controller will choose request type. Controller will enter a text that describes the request.
Otherwise Flow	No other possible flows
Condition-Post	Request will be sent to Admin.
Output	No output
Constraints	No Constraints
Exceptions	No exceptions

Table 5: Use case 3 - Send request to Admin

ID	4
Name	Maintain Schedule
Description	Admin will be able to maintain exams schedule.
Priority	2
Status	Completed, Coded, tested
Actors	Admin
Condition-Pre	condition-No pre
Inputs	First controller name & second controller name
Normal Flow of Events	Admin will open time maintain schedule screen in the app. Admin will choose first controller name. Admin will choose second controller name. Admin will select the exam. Assign the selected exam to a new controller.
Otherwise Flow	Controller has another exam in same time
Condition-Post	Assign exam to a new controller
Output	No output
Constraints	No Constraints
Exceptions	No Exceptions

Table 6: Use case 4 - Maintain Schedule

ID	5
Name	Inform Controller
Description	Admin sends a notification to controller with some instructions.
tyPriori	2
Status	Completed, Coded, Tested
Actors	Admin
Condition-Pre	condition-No pre
Inputs	Controller name
Normal Flow of Events	Admin will enter controller name Admin will enter a text then send
Otherwise Flow	Admin will enter exam name (course) Admin will enter a text and send to all controllers with same exam.
Condition-Post	The system will send notification to controller
Output	No output
Constraints	No Constraints
Exceptions	No exceptions

Table 7: Use case 5 - Inform Controller

ID	6
Name	Contact Course Teacher
Description	Admin will contact course teach on a phone call
Priority	2
Status	Completed, Coded, Tested
Actors	Admin
Condition-Pre	condition-No pre
Inputs	Course teacher name
Normal Flow of Events	Admin will enter course teacher name Application will call the teacher.
Otherwise Flow	No other flows
Condition-Post	conditions-No post
Output	Call teacher
Constraints	No Constraints
Exceptions	Teacher mobile is turned off

Table 8: Use case 6 - Contact Course Teacher

ID	7
Name	Handle Notifications
Description	Admin will be able to see notifications and handle them.
Priority	2
Status	Completed, Coded, Tested
Actors	Admin
Condition-Pre	Controllers send notifications to be handled
Inputs	No input
Normal Flow of Events	Admin will open notifications page in the app Application requests notifications from database Show notifications on the screen
Otherwise Flow	No notifications for this admin.
Condition-Post	nsAdmin handle notificatio
Output	Display notifications
Constraints	No Constraints
Exceptions	No exceptions

Table 9: Use case 7 - Handle Notifications

Chapter Five

Design

Chapter 5: Design

5.1 ER Diagram

The following figure presents the ER Diagram for our database.

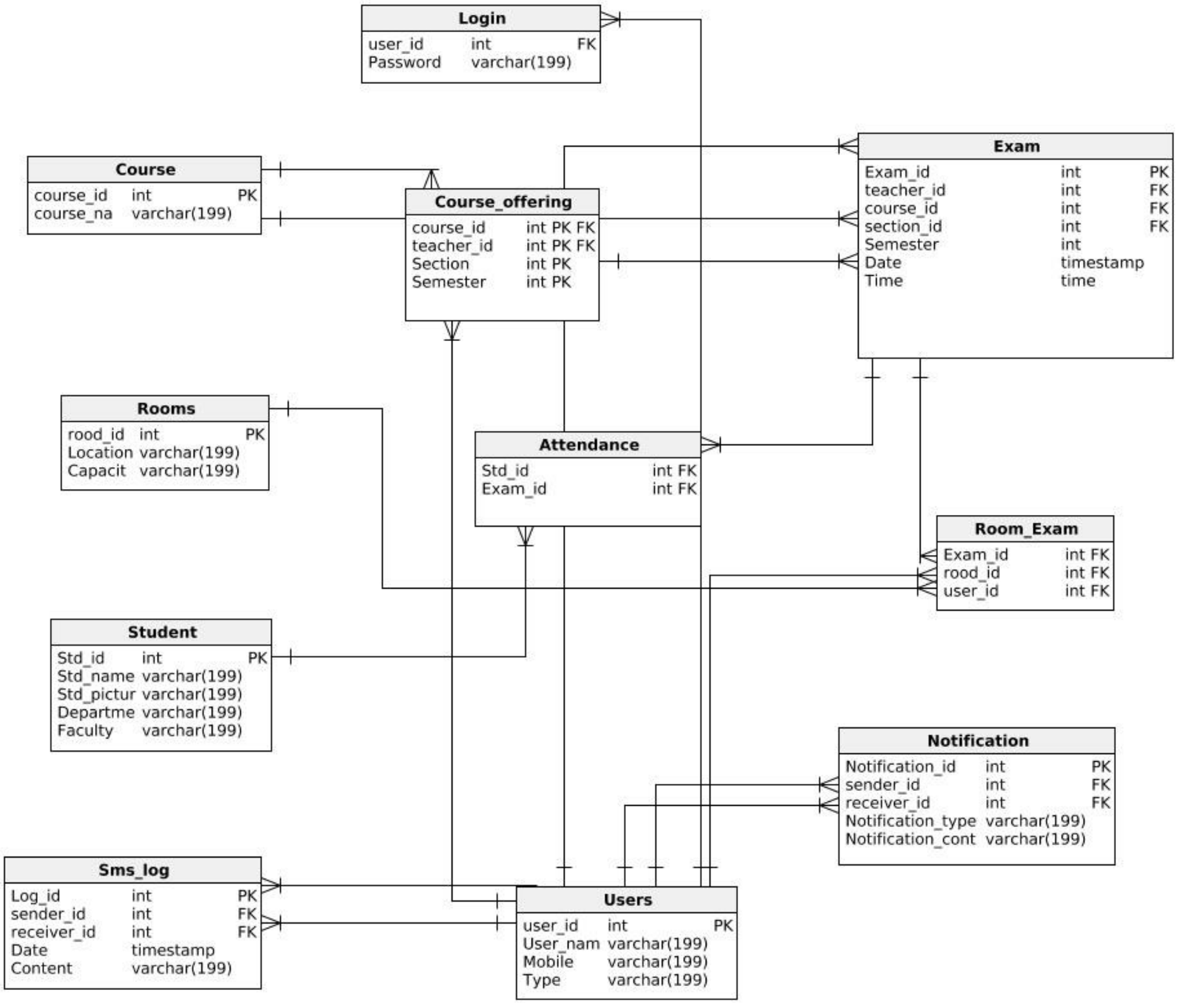


Figure 9: ER Diagram

5.2 Class Diagram

Figures from 5 to 18 describe Backend and mobile classes.

5.2.1 Backend Class Diagram

The following figure shows the backend class diagram for our Laravel classes.

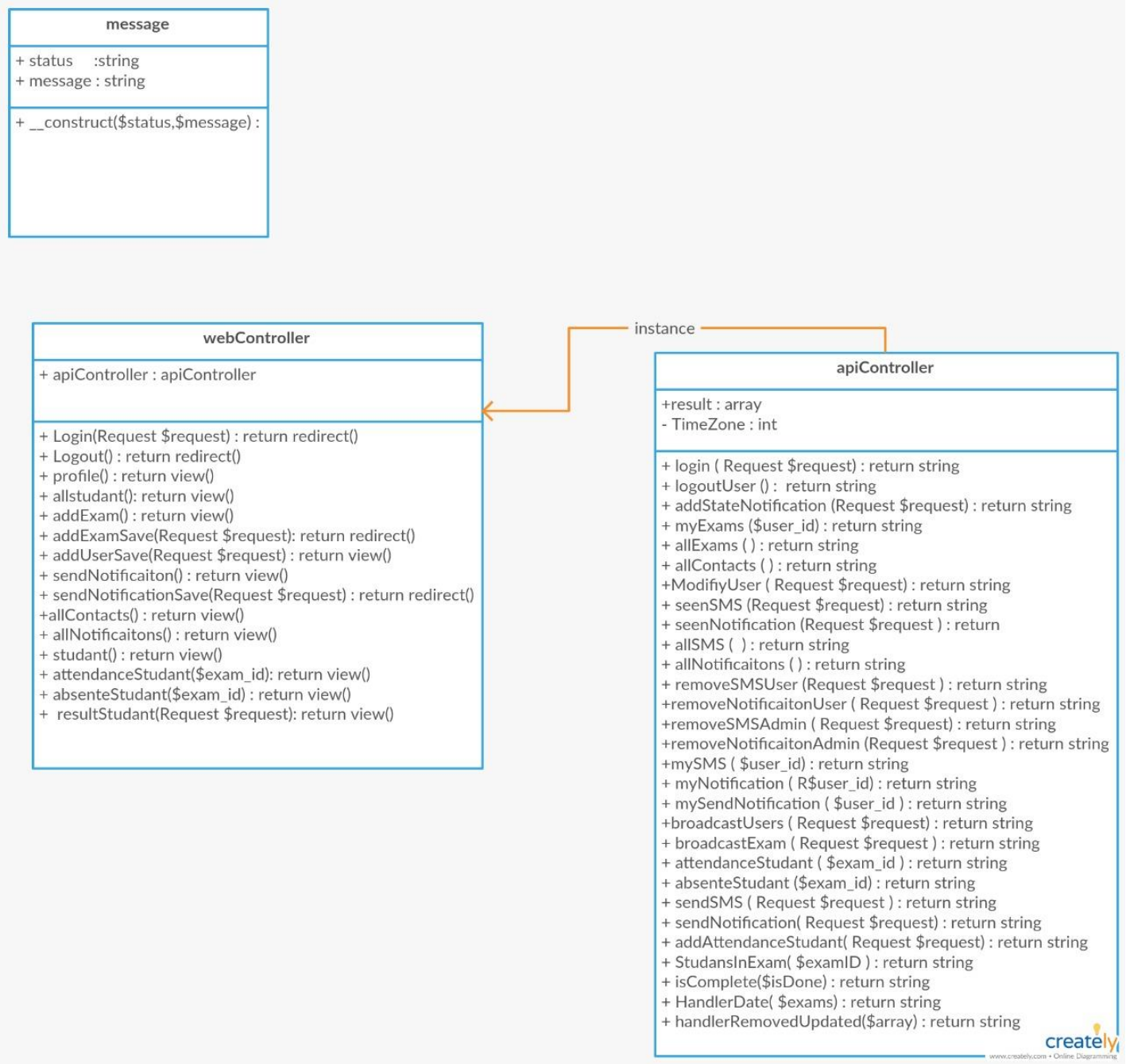


Figure 10: Laravel Class Diagram

5.2.2 Mobile Class Diagram

The following figures from 10 to 12 show Android class diagram that contain main functions for Admin.

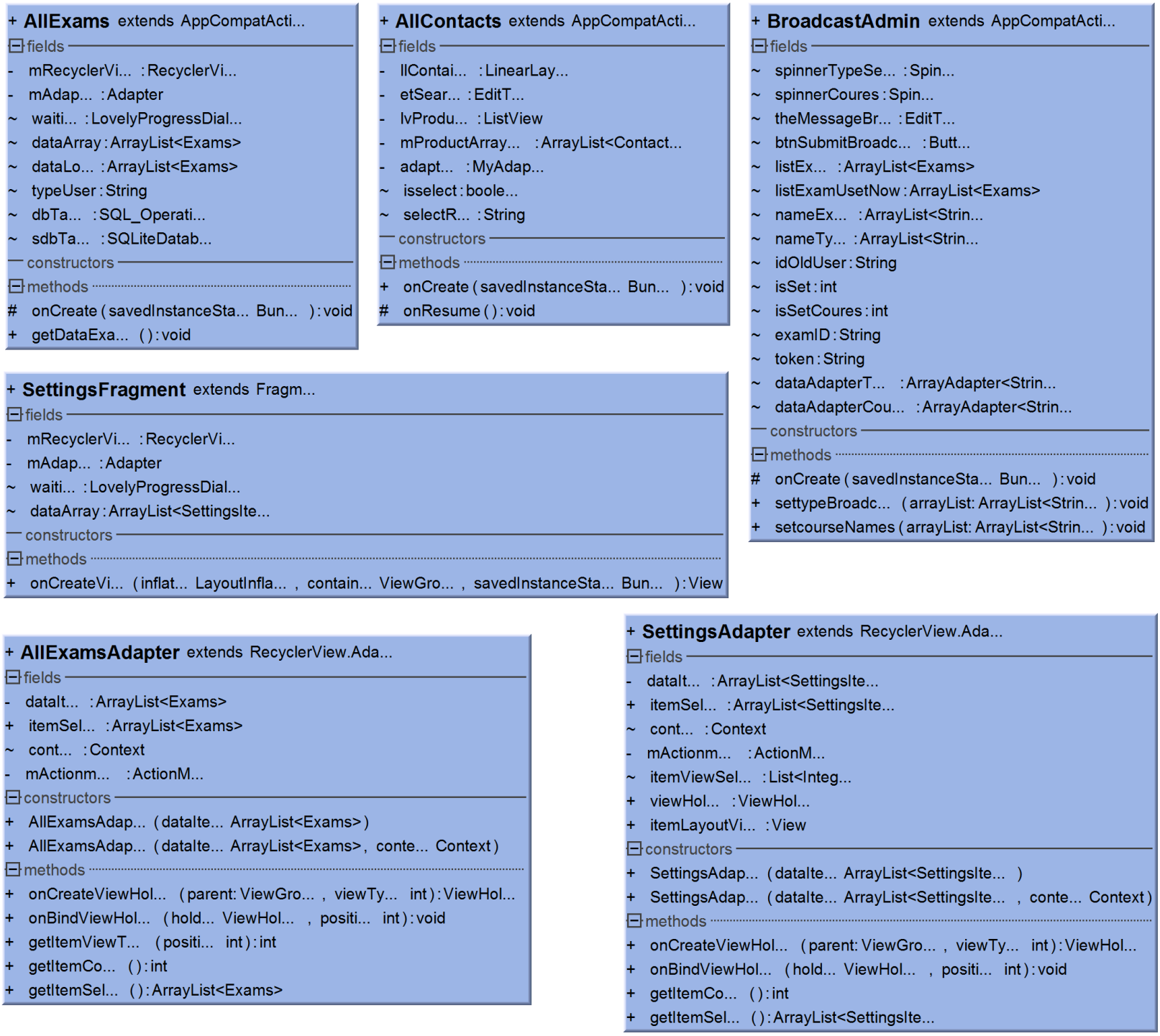


Figure 11: Package Admin – 1

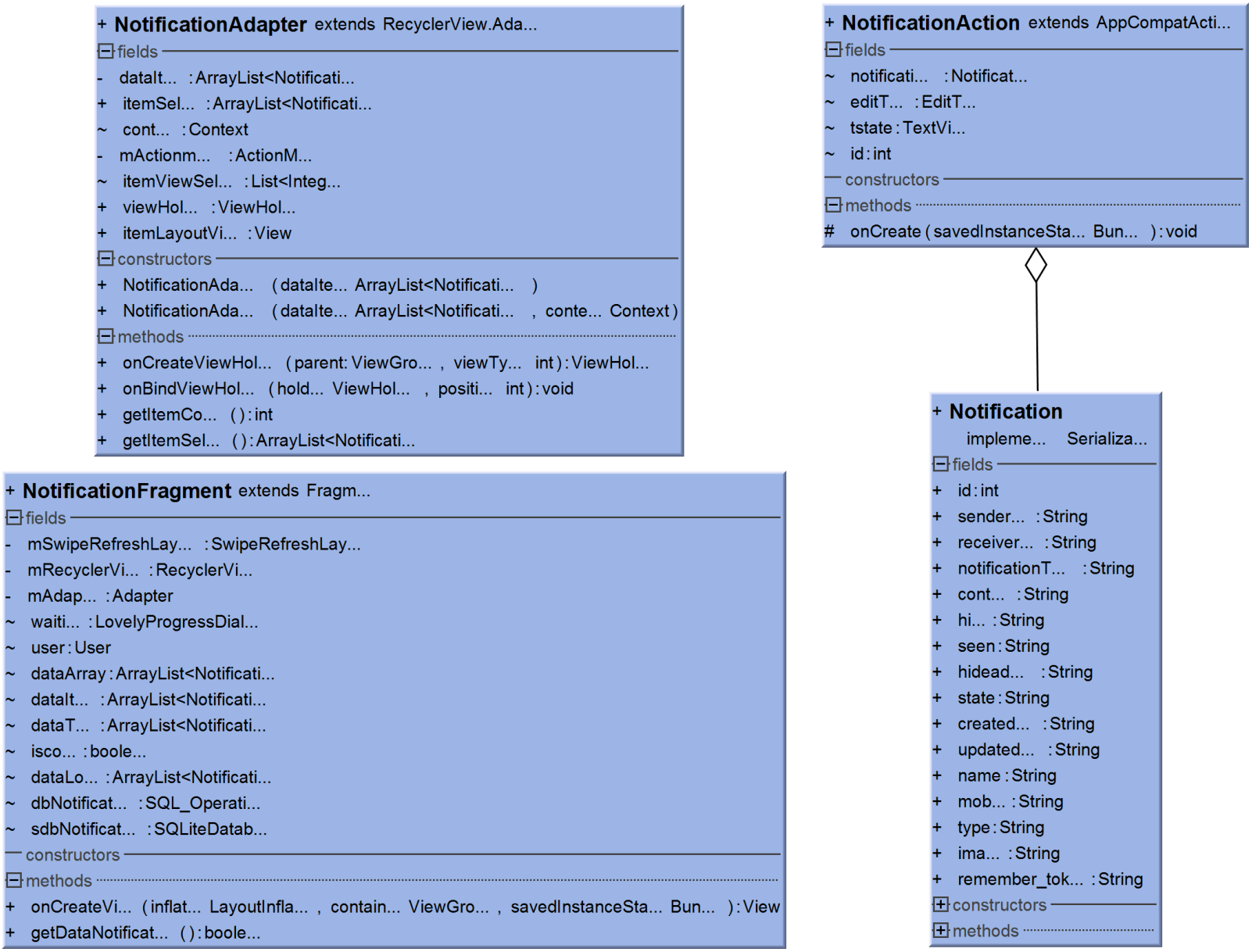


Figure 12: Package Admin – 2

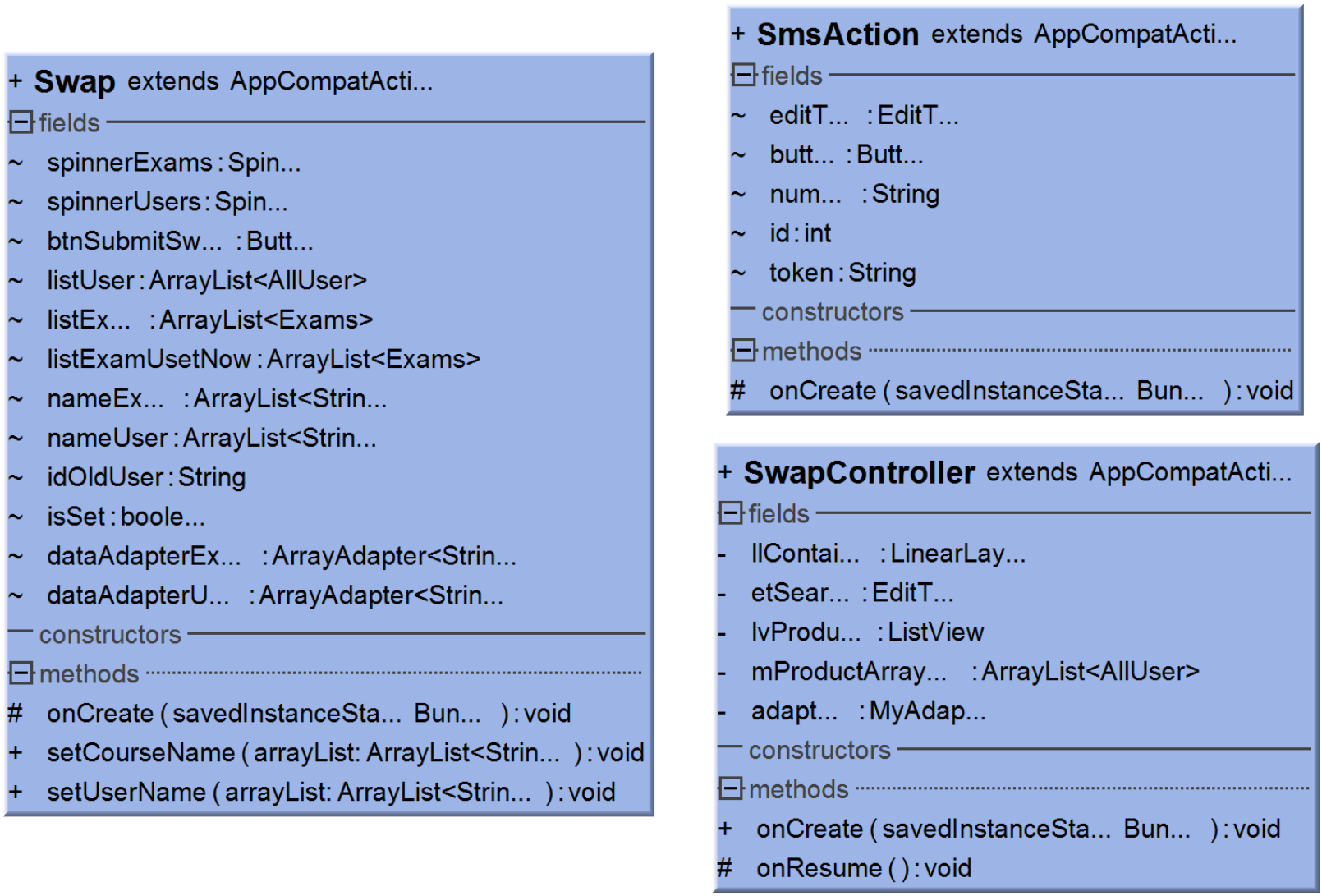


Figure 13: Package Admin – 3

Figure 13 shows Control package that contains login class and the main screen for the application.

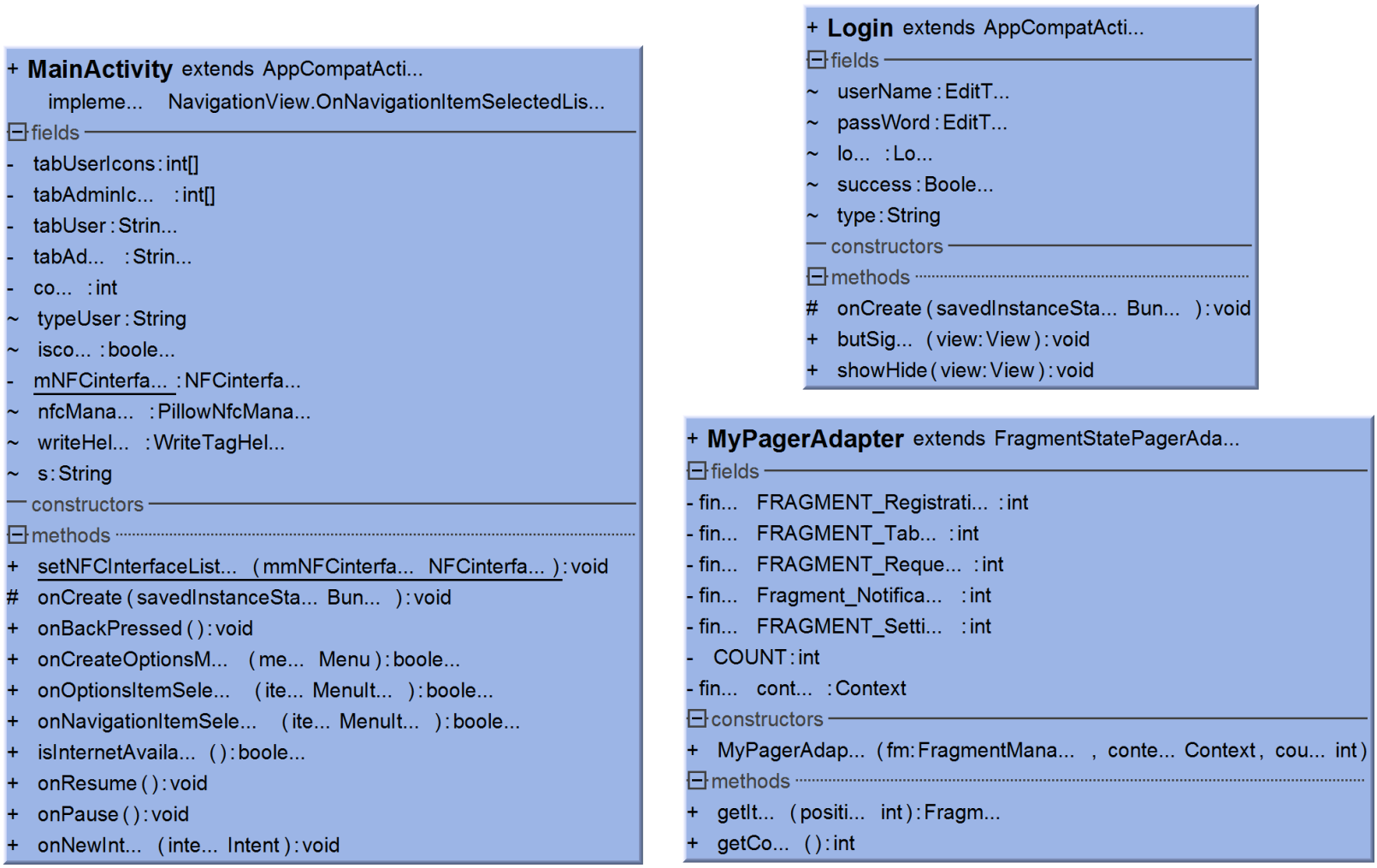


Figure 14: Package Control

Figure 14 shows a part of controller package that contain registration class and broadcast.

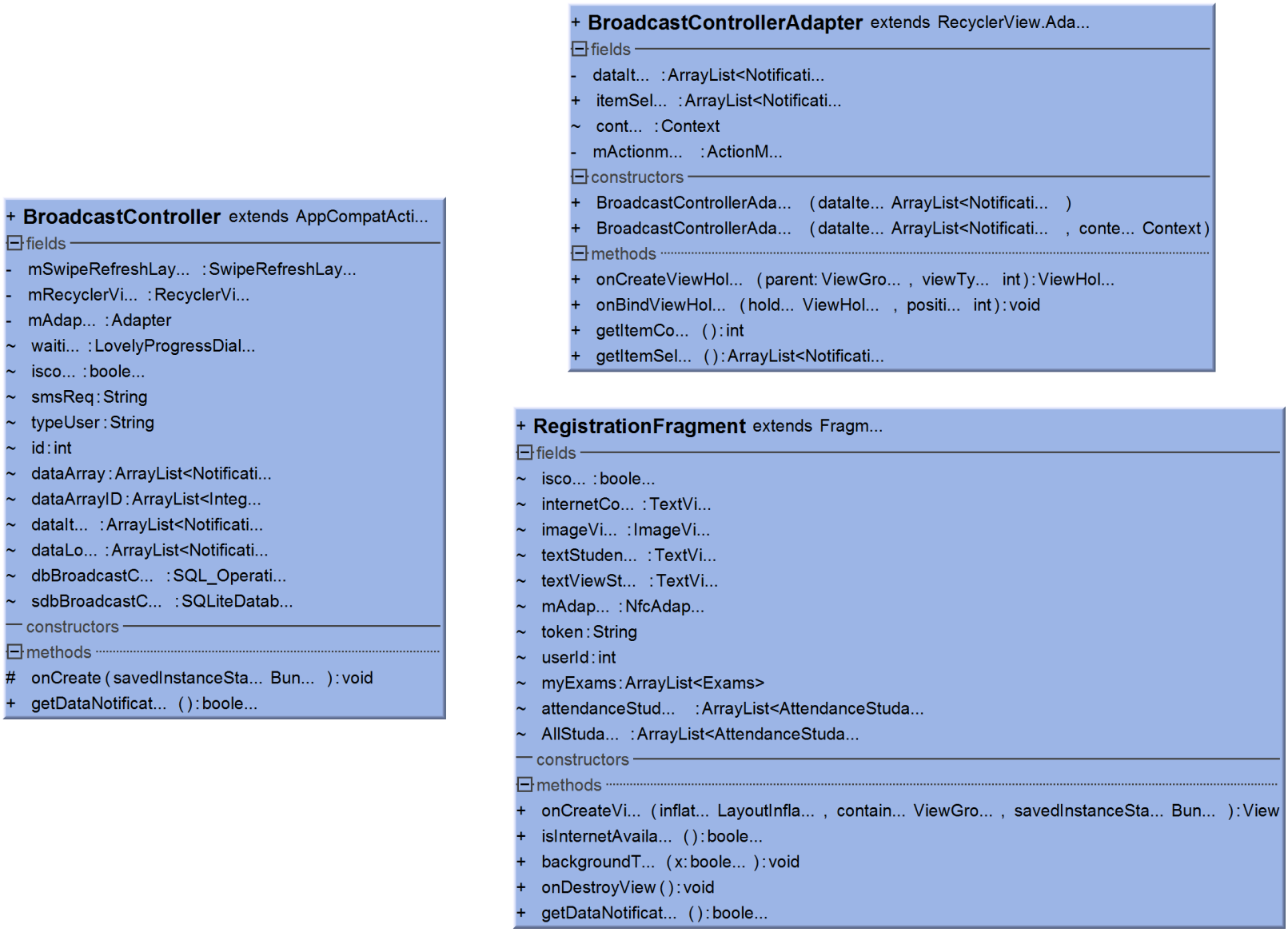


Figure 15: Package Controller - 1

The following figure shows the other part of controller package which contains tables fragment class, table adapter, request fragment class and request adapter.

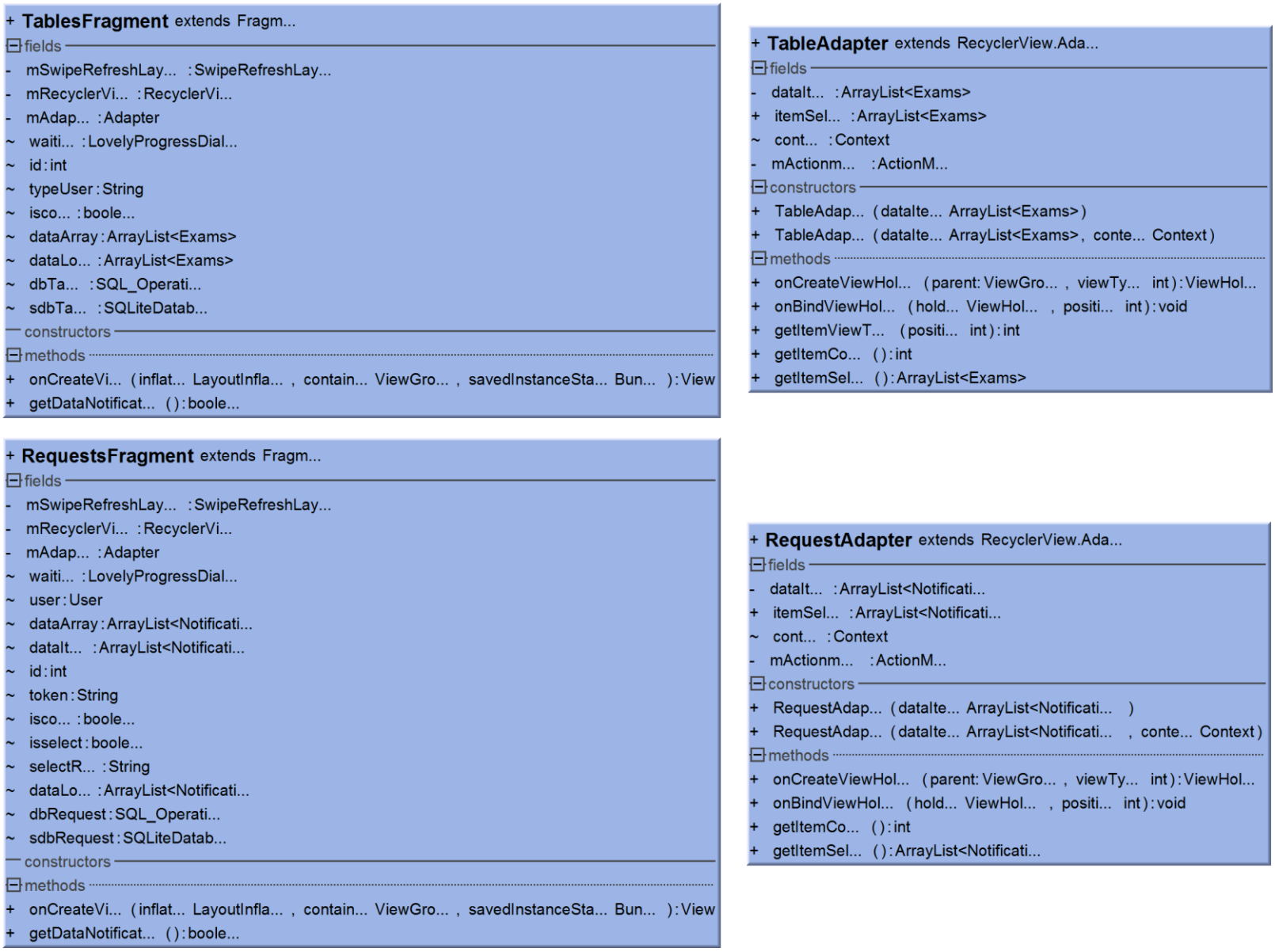


Figure 16: Package Controller – 2

The following figure shows the relationship between Login class and Model Login class.

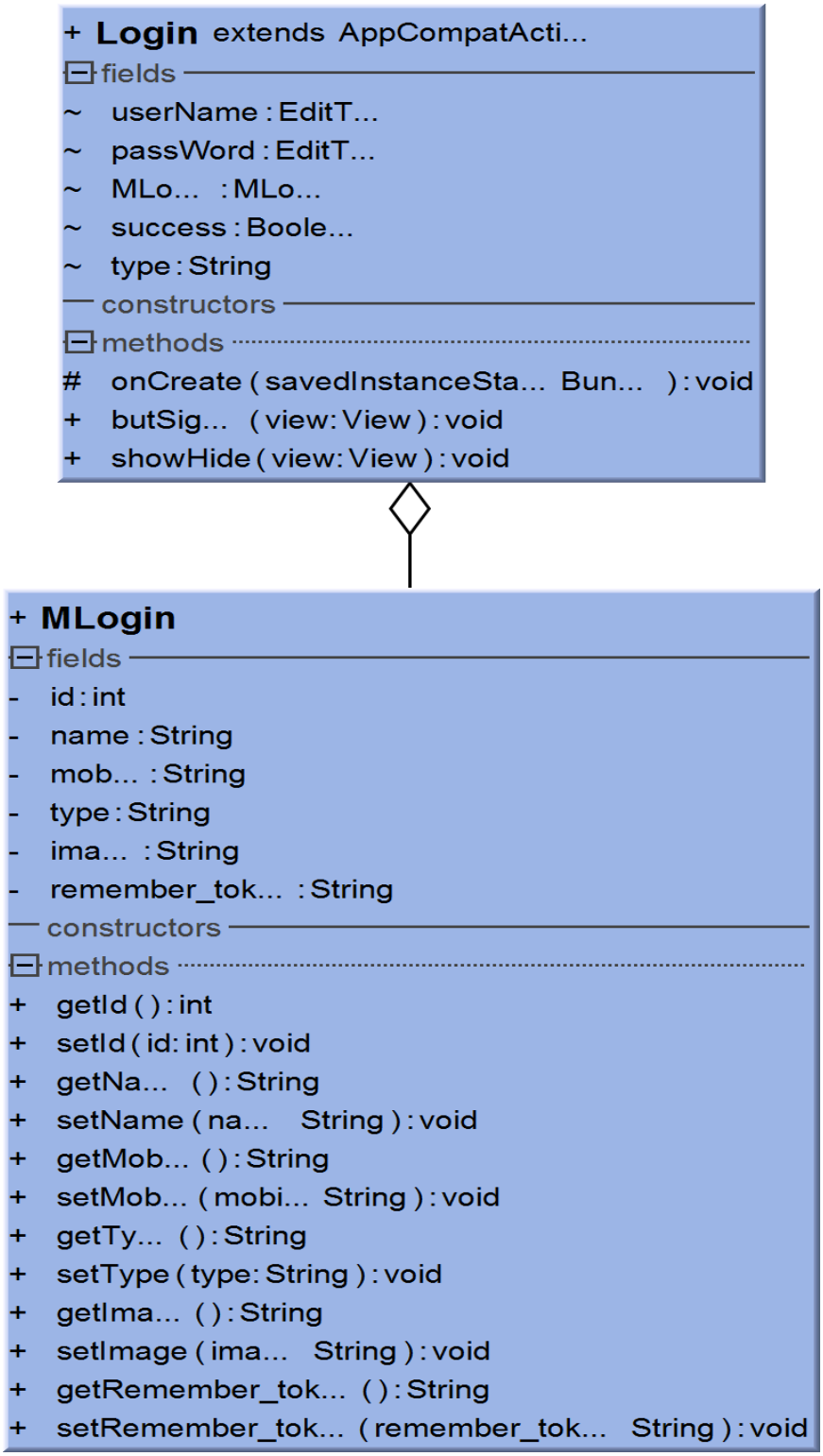
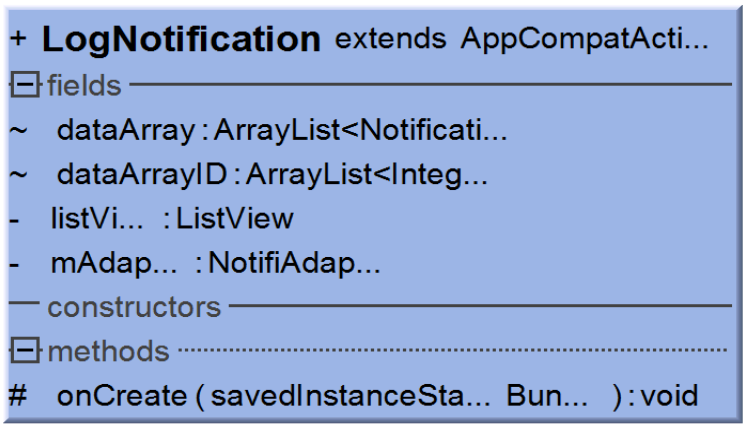
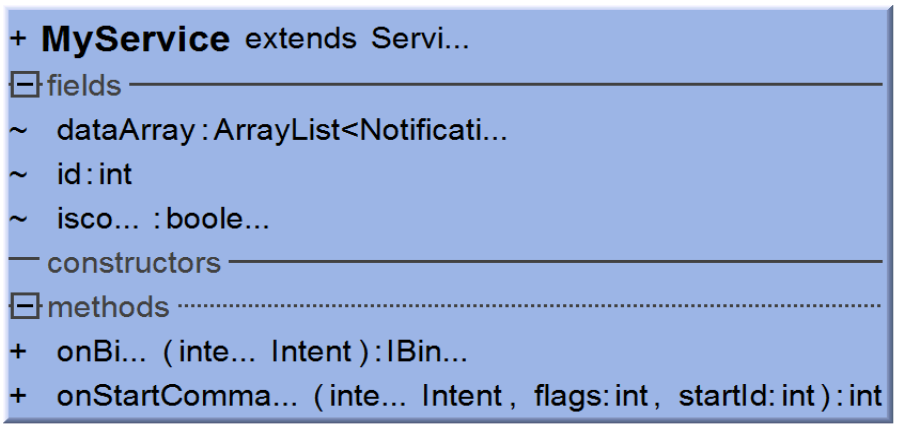


Figure 17: Package Login

The following figure shows Notification package that contain contains service and control it's notifications to display them as notifications.





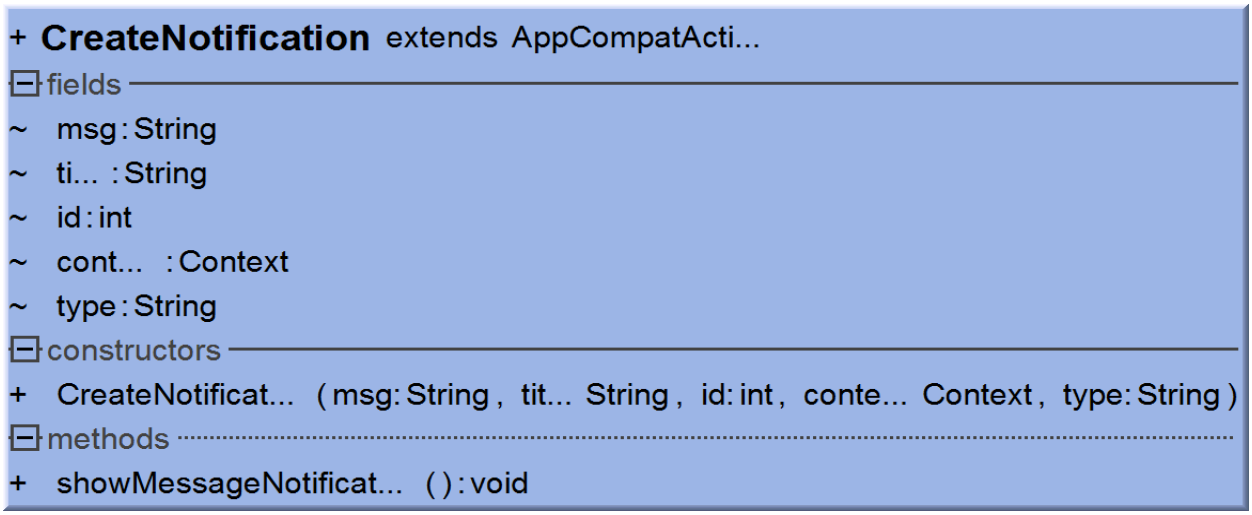


Figure 18: Package Notification

The following figure shows settings interface.

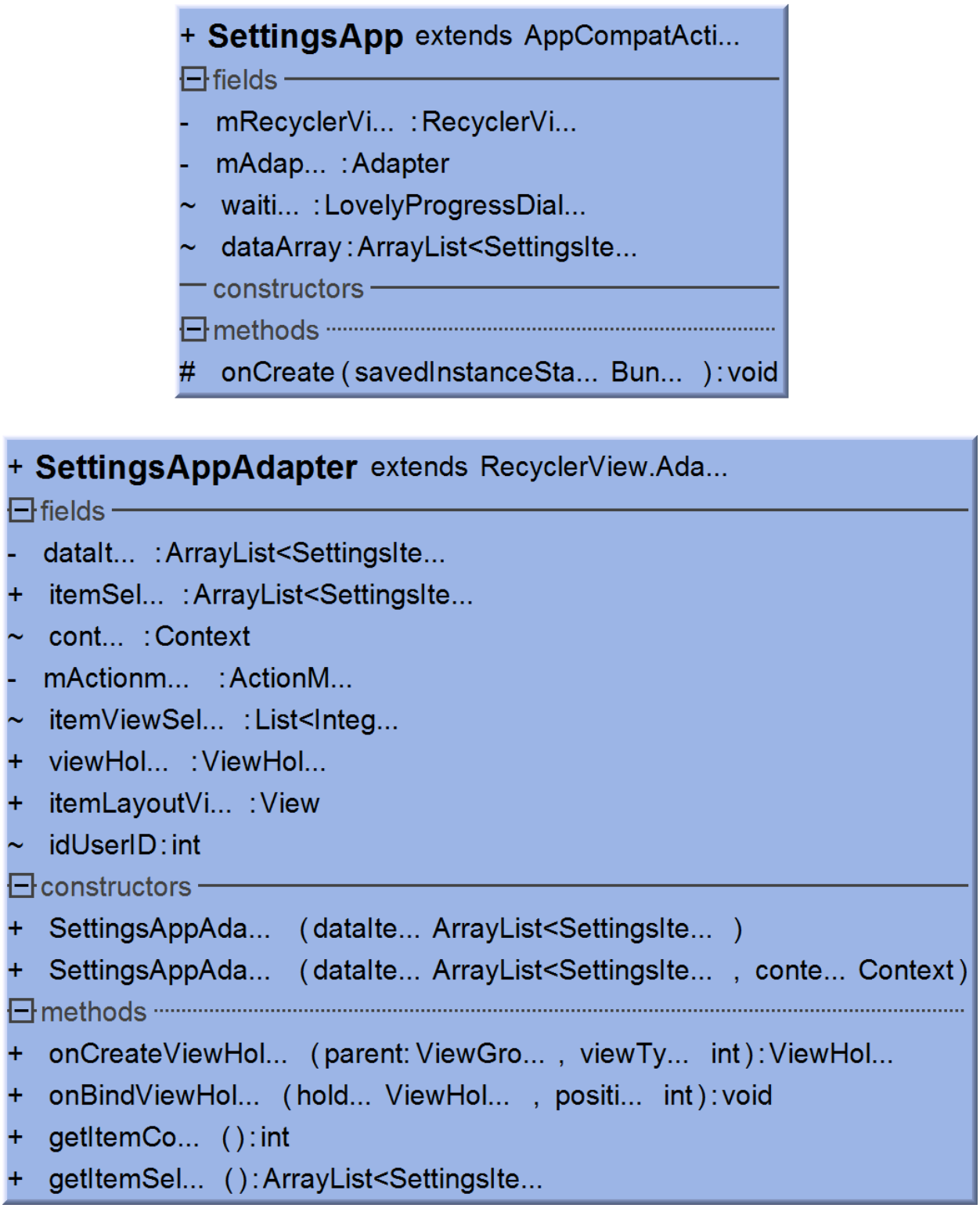


Figure 19: Package Settings

The following figure shows the classes that show students that are supposed to attend an exam

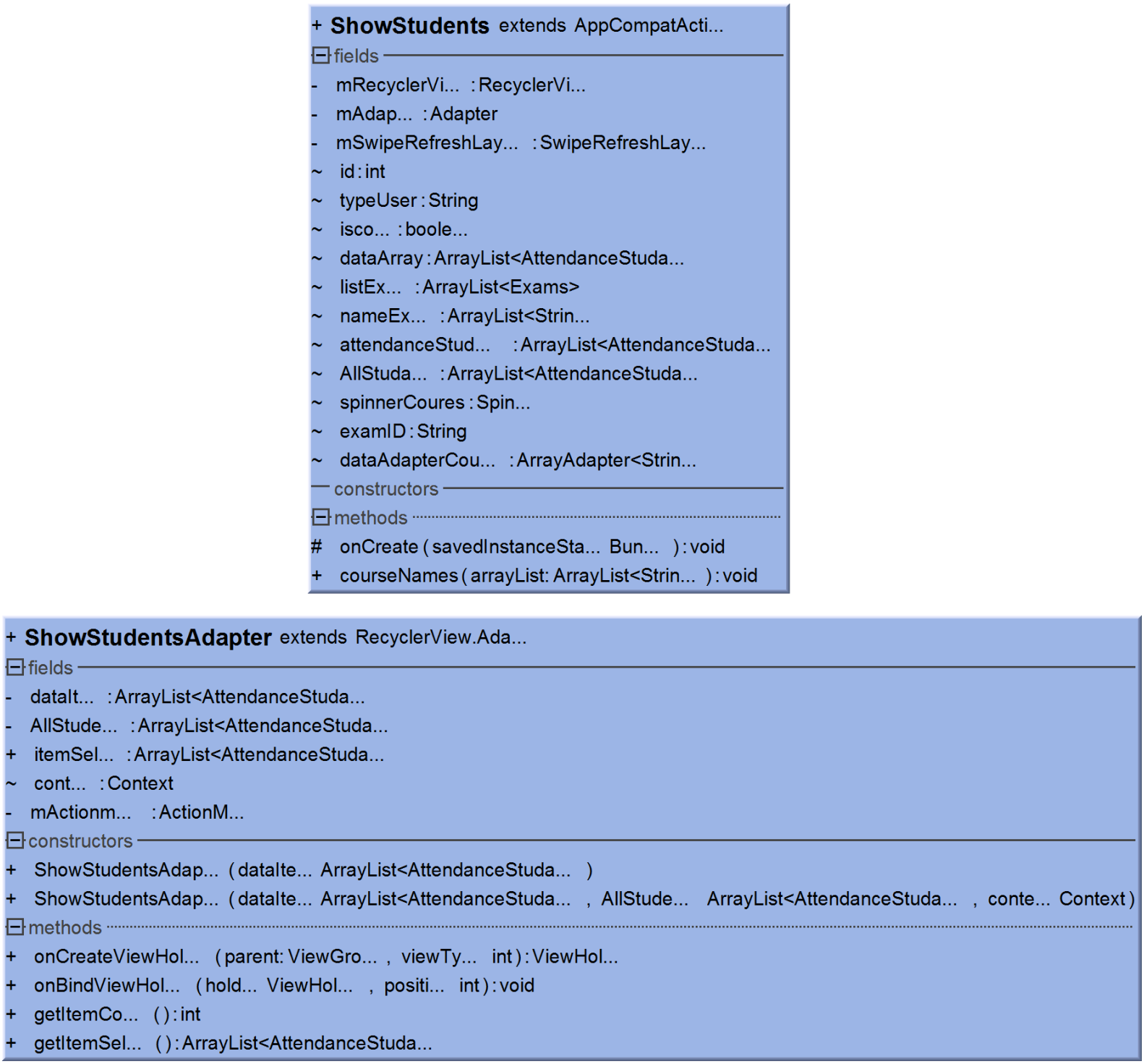


Figure 20: Show Student

The following figure shows the relationship between SQL_Operations class with the other classes.

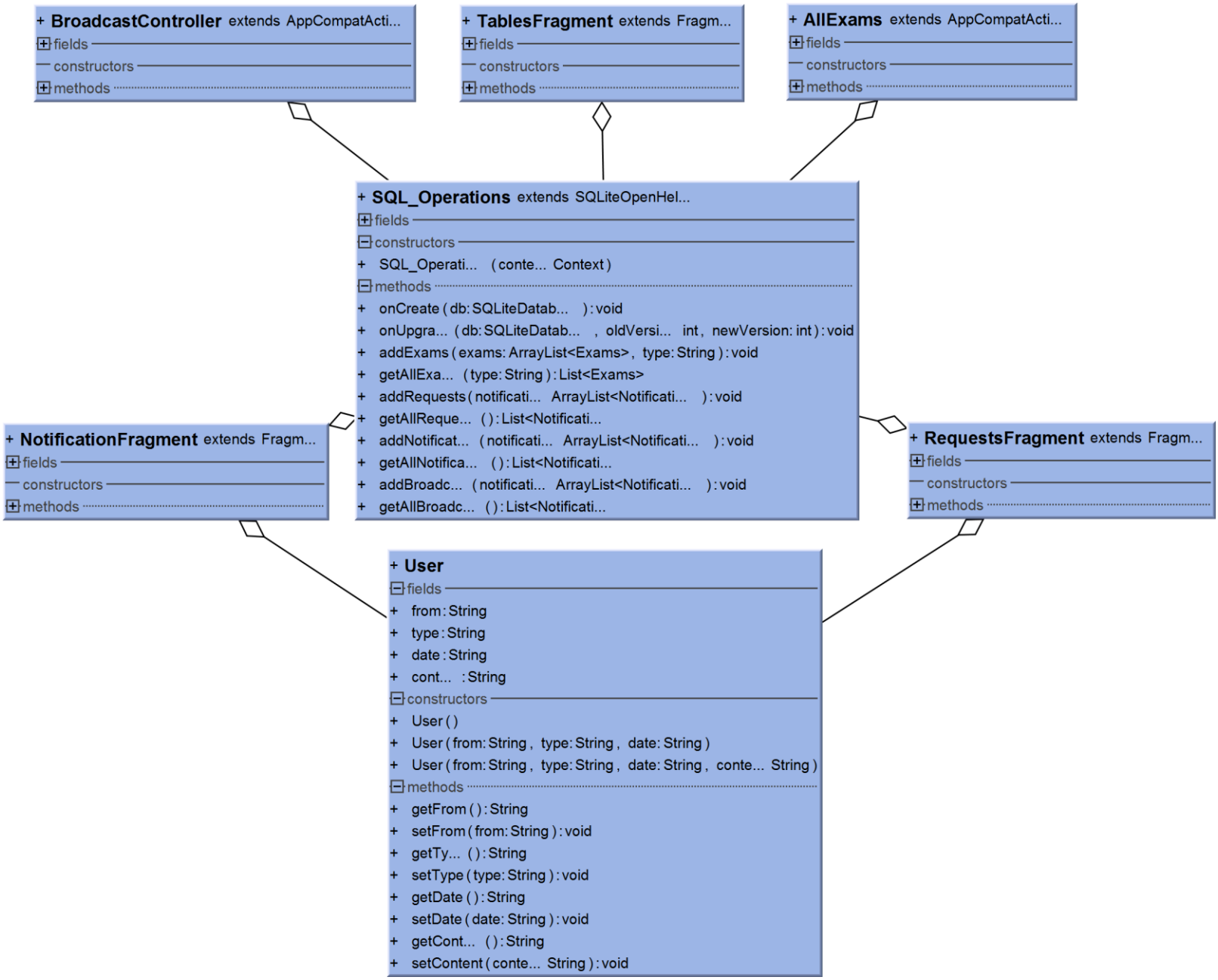


Figure 21: SQL Lite

The following figure shows the class that control all requests, it sends and receives data from and to server.

Figure 22: Volley

The following figure shows the class that read NFC information stored in NFC card.

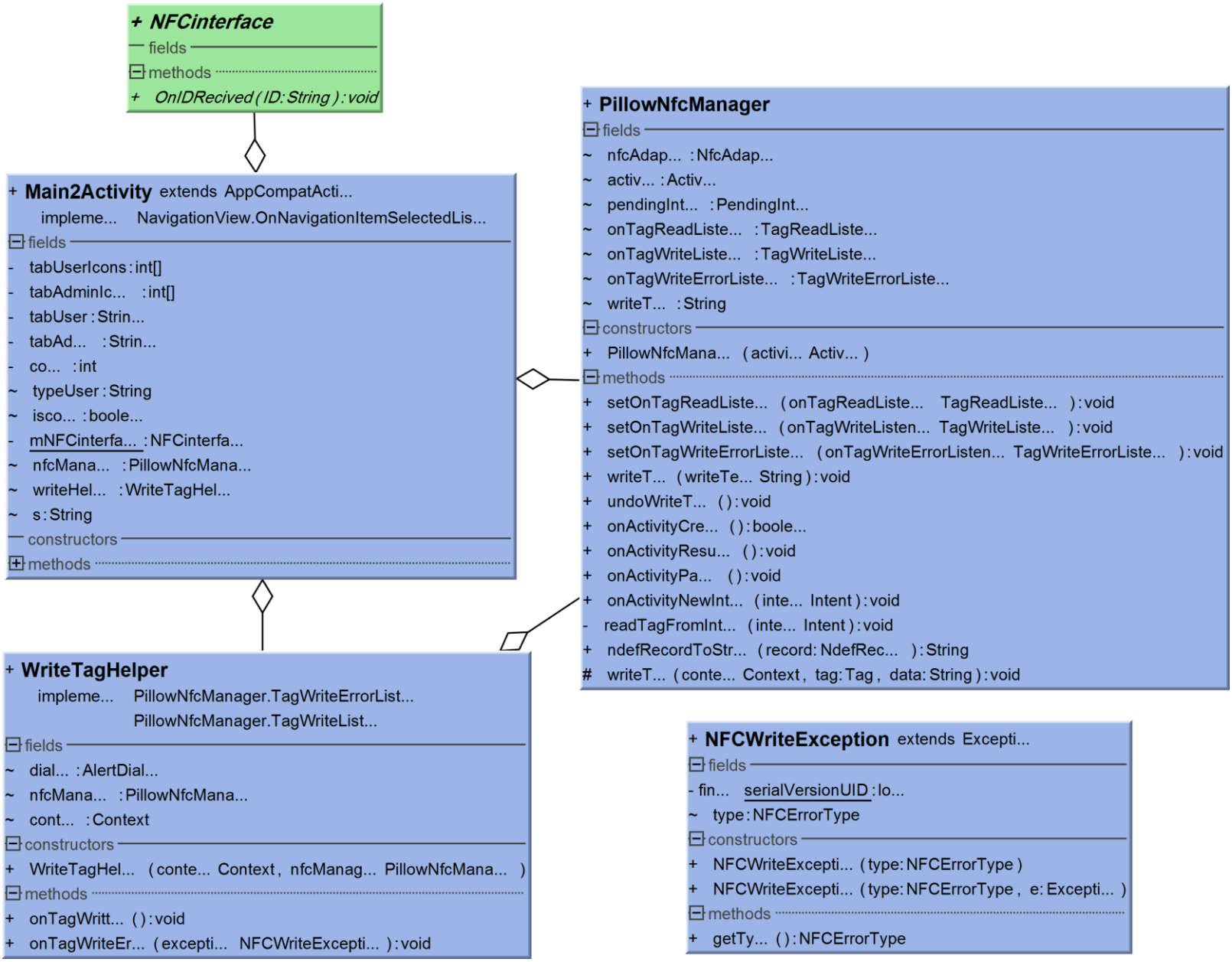


Figure 23: NFC Class

5.3 Sequence Diagram

Figures from 19 to 25 show the sequence diagrams, it explains the functionalities of our application for both Admin and Controller that are mentioned at system functions section.

Sequence Diagram 1: Controller will be able to check his/her time table for the exams.

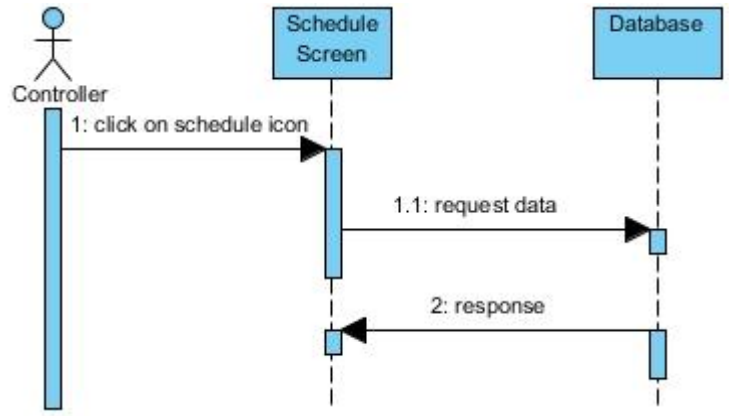


Figure 24: Check Time Table

Sequence Diagram 2: Admin will contact course teach on a phone call.

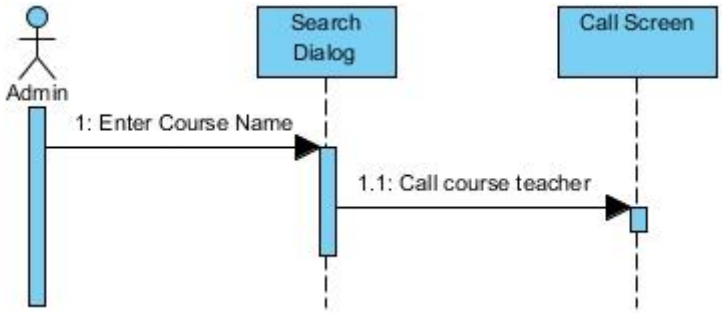


Figure 25: Contact Course Teacher

Sequence Diagram 3: Admin will be able to see notifications and handle them.

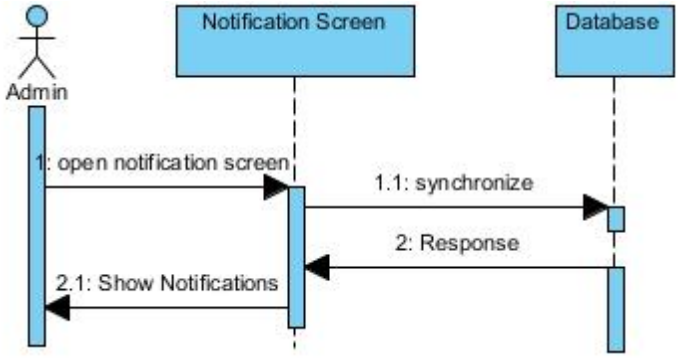


Figure 26: Handle Notifications

Sequence Diagram 4: Admin sends a notification to controller with some instructions.

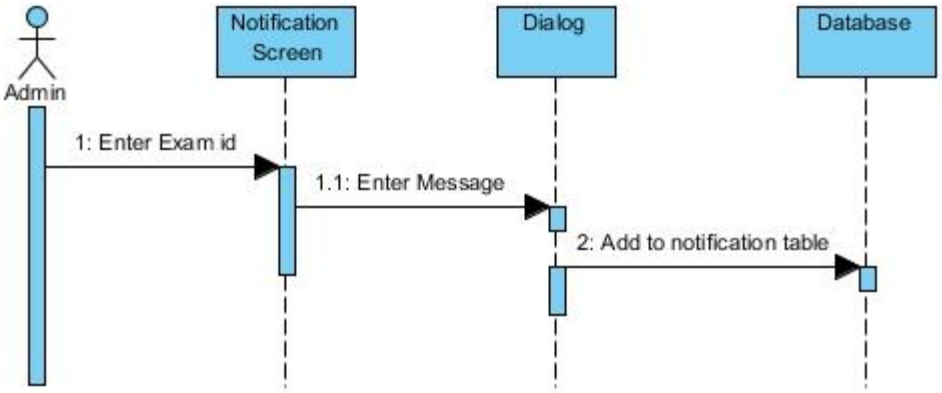


Figure 27: Inform Controller

Sequence Diagram 5: Admin will be able to maintain exams schedule.

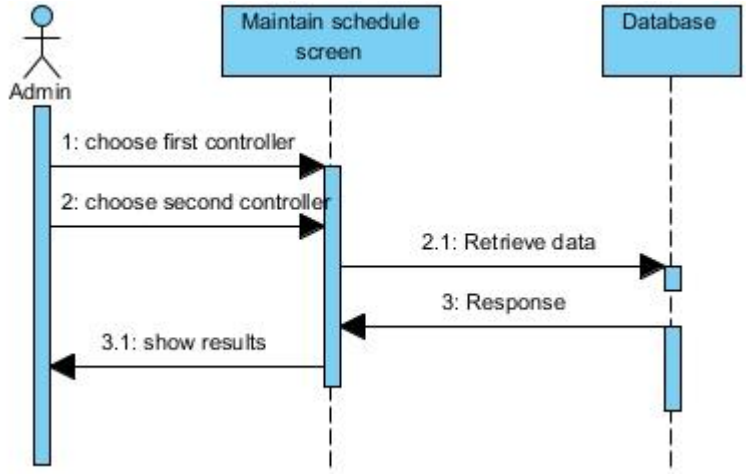


Figure 28: Maintain Schedule

Sequence Diagram 6: Controller will be able to register student's presence in a specific exam.

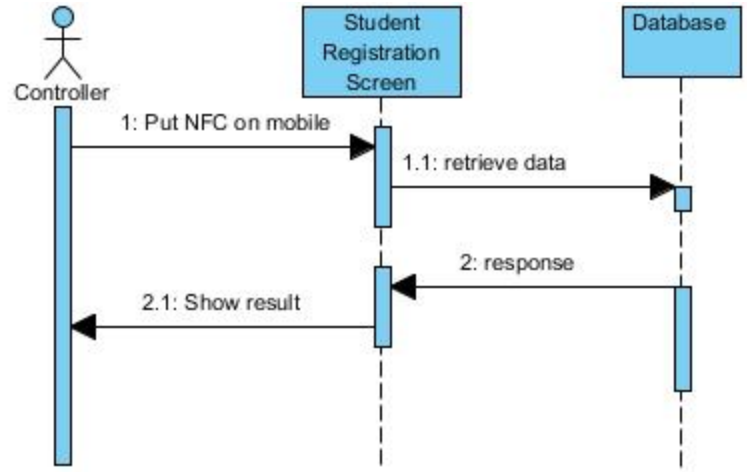


Figure 29: Register Presence

Sequence Diagram 7: Controller will be able to send a requests to Admin

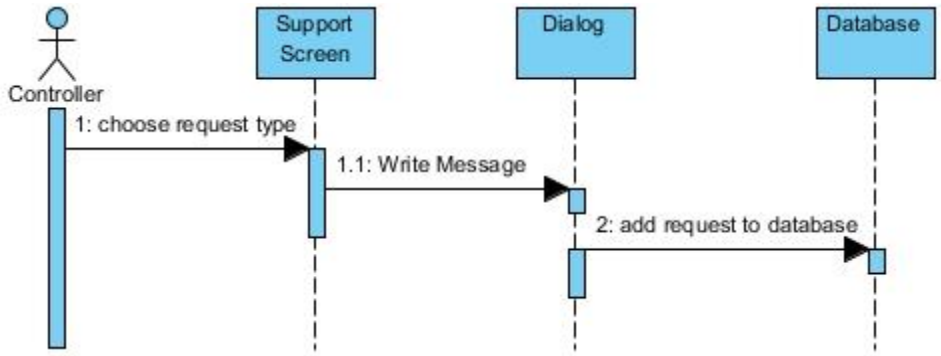


Figure 30: Send Request to Admin

5.4 Architecture Diagram

1- **Controller**: A controller is responsible for defining application (request handling) logic. Controllers are typically stored in app/Http/Controllers directory. Every controller must override base controller. The major advantage of using base controller is that it provides few convenience methods such as “middleware”, “validate” and “dispatch”.

2- **Model**: Models are responsible for handling database related methods. We can attach controllers with resource model it means the controller has their dependent database table.

3- **View**: Views are basically responsible for containing HTML (rendering objects). In Laravel is known as blade template.

4- **Routing in Laravel**: Routing plays the most important role in Laravel application rendering. Any url requested from user must be undergone through specific routes. All the routes of Laravel is defined under routes.php file. The location of routes.php is defined under app\Providers\RouteServiceProvider.php . By default, there are three routes that are automatically loaded by framework. [9]

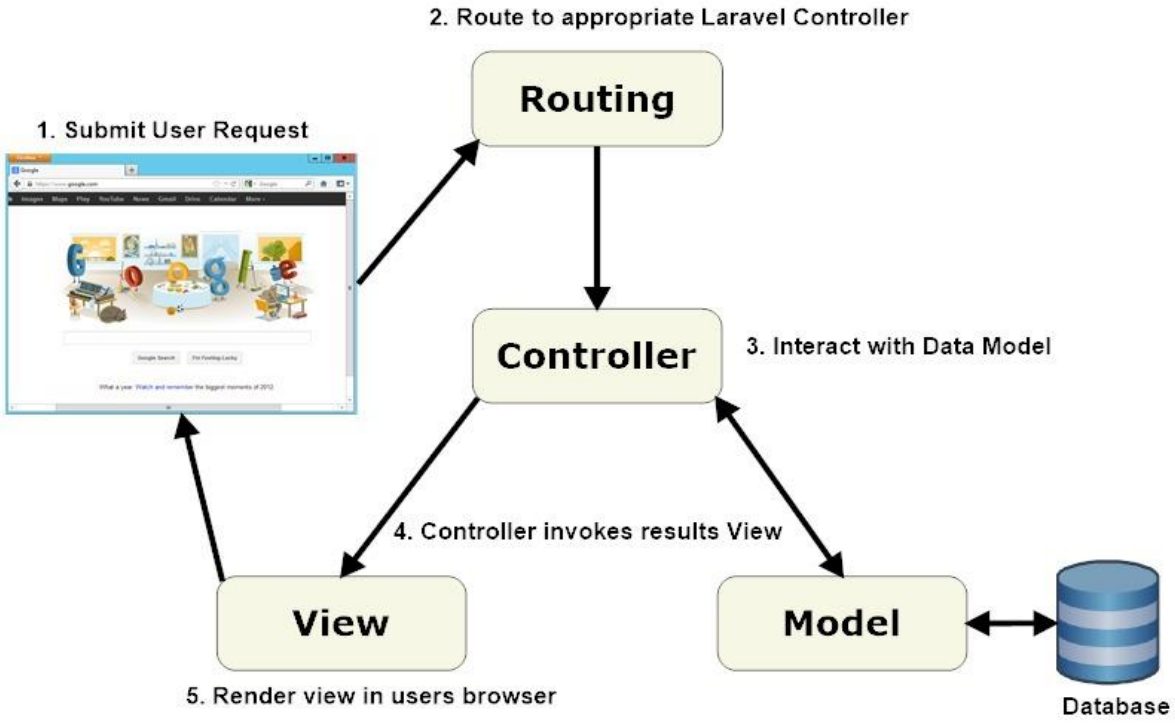


Figure 31: Architecture Diagram (Backend System)

The first step is building the required Web APIs. This is because from the android application, to communication with our web server we need an interface called API.

So our android device will send a request to our API, then our API will perform the requested task and it will give us the response related to the task. You can see the below diagram for more clarification. [10]

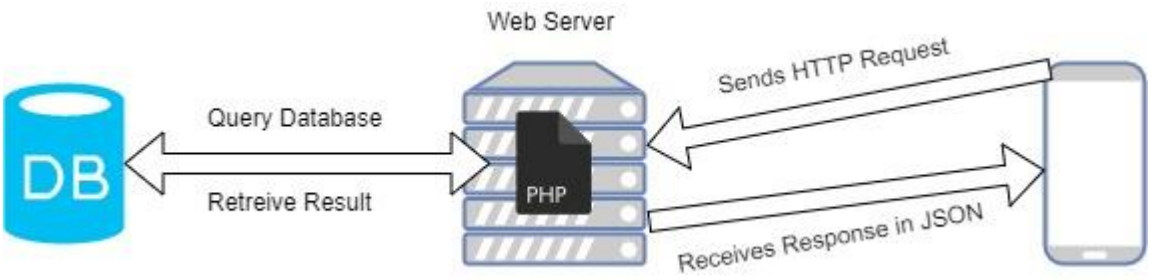


Figure 32: Architecture Diagram (DB, Web and Mobile platforms interaction)

5.4.1 Life cycle of request

All requests into the application are directed through the public/index.php script, when using Apache, the .htaccess file those ships with Laravel handles the passing of all requests to index.php from here.

The following figure describes how android side work, the client will request information to the web server and the web server which works with PHP and MySQL. When processed successfully, it will return the JSON value. [10]

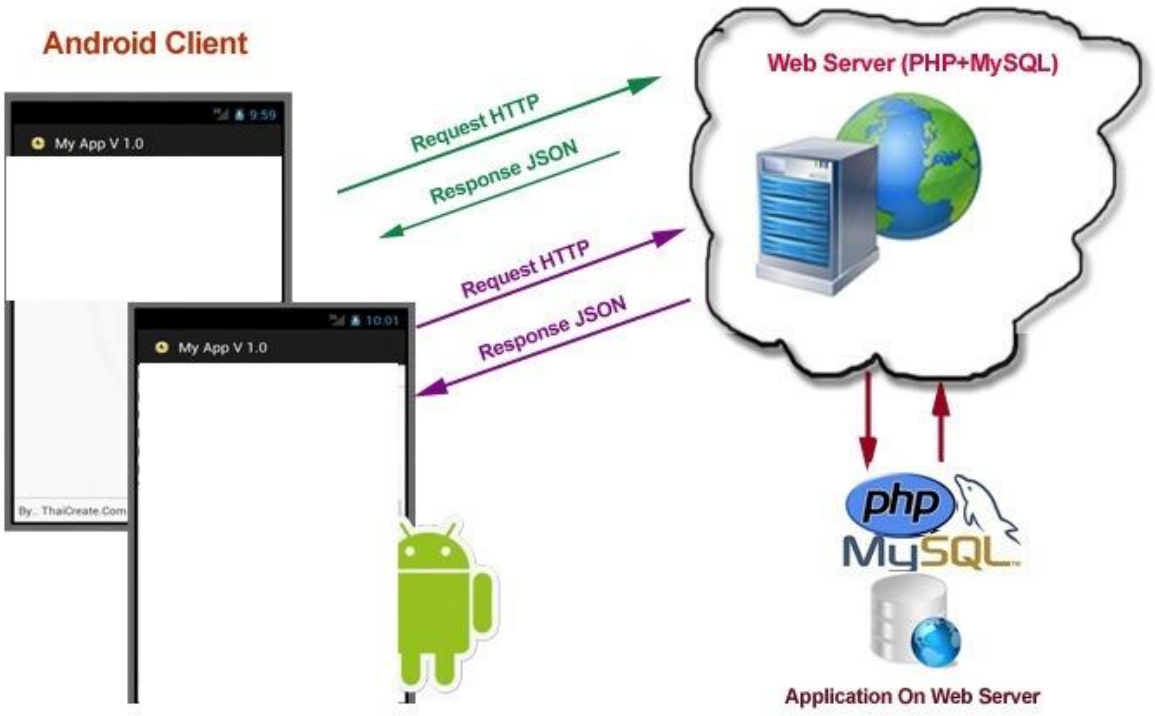


Figure 33: Architecture Diagram (DB, Web and Mobile platforms interaction)

Chapter Six

Implementation & Testing

Chapter 6: Implementation and Testing

We used Java programming language with Android platform, we used Android Studio IDE for coding, we used compile SDK minimum 19 and target 26, with API 23.

We used NFC tag type Ntag216(NT2H1611) with RF technology (ISO/IEC 14443 Type A) with a writable tag and Memory size of 924 Bytes, to test the application.[11]

As our project is only supposed to read NFC card that already has information stored in it, we had to make a small program for testing purpose, that write some data on NFC to be able to test our application.

Our application works both online and offline, as it contains SQLite that store data and synchronizes them when there is internet connection, as it can work offline after receiving the data required from API as they are stored in SQLite and the application gets and store data offline temporarily until there is connection with the server.

We used Laravel framework for creating database and API, also we used it for designing user interfaces for Admin to maintain students, exams and show notifications.

We used our own server instead of Firebase because of 5 reasons as follow

1- Your data is not yours
    - Your data is hosted on servers that you don't own; it's not possible to export your user data. You can't export emails, and user accounts are not recoverable, you cannot export user accounts with passwords.
2- The problem of data migration
    - With Firebase, you can't deal easily with data-migration like you can do with a simple SQL database. Firebase uses JSON and there are almost no features SQL features, so you wouldn't be able to migrate from the existing database easily.
    - If you're managing large amounts of highly structured data. With a regular relational database you can write powerful queries in SQL to quickly get the data your looking for. There's no equivalent with Firebase. Also, you could use an ORM to further simplify data handling.

3- Limited querying

- Limited query abilities due to Firebase's data stream model. The Firebase Database queries are limited. Certain query options can't be combined. Because of this, you'll have to choose, either you order the documents by date or filter them with user's search query on the database side and perform the other action on the client side.

4- Very Oriented toward real-time sync

- Firebase seems very oriented toward real-time sync. If you type in a text field, the database gets updated automagically. If someone else types in a text field, your screen adjusts. The primary documentation examples and Angular integration seemed focused on this. It's not what I wanted, though. I'd be fine writing queries and using an event handler to update the database, or receive someone else's update.

5- Security rules are limited

- Security rules might be one of our biggest struggles with Firebase. The rules have been designed to be fast for Firebase's servers to execute at request time, but the downside of this is the rules are limited that it's very hard to build an enterprise platform on top of them.
- Security rules might be one of our biggest struggles with Firebase. The rules have been designed to be fast for Firebase's servers to execute at request time, but the downside of this is the rules are limited that it's very hard to build an enterprise platform on top of them.**[12]**

6.1 Code

The following figures show a part of the code for mobile application.

```
package Control :

public class MainActivity extends AppCompatActivity {
    new VolleyRequests().setIReceiveData(new VolleyRequests.IReceiveData() {
        @Override
        public void onDataReceived(Object o) {
            if (o != null) {
                login = (Login) o;
                type = login.getType();
                }
            }
        }).doRequestForLogin(userName.getText().toString(),passWord.getText().toString());
    }
}
```

Figure 34: Login

```
package Fragment.Admin;

public class AllExams extends AppCompatActivity {

    new VolleyRequests().setIReceiveDataA(new VolleyRequests.IReceiveDataA() {
        @Override
        public void onDataReceivedA(ArrayList o) {
            dataArray = o;
            if (o != null) {
                mRecyclerView.setLayoutManager(new LinearLayoutManager(AllExams.this));
                mAdapter = new AllExamsAdapter(dataArray,AllExams.this);
                mRecyclerView.setAdapter(mAdapter);
                mRecyclerView.setItemAnimator(new DefaultItemAnimator());
            }
        }
    }).doRequestForTable("",typeUser);
}
```

Figure 35: Get all Exams

```
package Control :

public class Main2Activity extends AppCompatActivity
                           implements NavigationView.OnNavigationItemSelectedListener {

        nfcManager = new PillowNfcManager(Main2Activity.this);
        nfcManager.onActivityCreate();

        nfcManager.setOnTagReadListener(new PillowNfcManager.TagReadListener() {
            @Override
            public void onTagRead(String tagRead) {
            mNFCinterface.OnIDRecived(tagRead);
            Toast.makeText(Main2Activity.this, "tag read QAR:"+tagRead, Toast.LENGTH_LONG).show();
        }
    });
}
```

```
package Control.NFC :

public class PillowNfcManager {
private void readTagFromIntent(Intent intent) {
        String action = intent.getAction();
        if (NfcAdapter.ACTION_NDEF_DISCOVERED.equals(action)) {
            Tag myTag = (Tag) intent.getParcelableExtra(NfcAdapter.EXTRA_TAG);
            Parcelable[] rawMsgs = intent.getParcelableArrayExtra(NfcAdapter.EXTRA_NDEF_MESSAGES);
            if (rawMsgs != null) {
                NdefRecord[] records = ((NdefMessage) rawMsgs[0]).getRecords();
                String text = ndefRecordToString(records[0]);

                onTagReadListener.onTagRead(text);
            }
        }
    }
}
```

Figure 36: Read NFC

```
package Fragment.Admin;

public class NotificationFragment extends Fragment {

new VolleyRequests().setIReceiveDataA(new VolleyRequests.IReceiveDataA() {
        @Override
        public void onDataReceivedA(ArrayList o) {
            dataArray = o;
            if (o != null) {
                for (int i = 0; i < dataArray.size(); i++) {
                    if(!dataArray.get(i).getNotificationType().equalsIgnoreCase("public") &&
                    !dataArray.get(i).getHideadmin().equalsIgnoreCase("1") ) {
                        dataItem.add(dataArray.get(i));
                    }
                }
            mRecyclerView.setLayoutManager(new LinearLayoutManager(getActivity()));
            mAdapter = new NotificationAdapter(dataItem, getActivity());
            mRecyclerView.setAdapter(mAdapter);
            mRecyclerView.setItemAnimator(new DefaultItemAnimator());
        }
    })).doRequestForRequests("");
}
```

Figure 37: Get all Notifications

6.2 User Interface

6.2.1 Mobile Interface

Main screen for controller, it's used to register attendance of students by bringing NFC card close to a smartphone that includes NFC reader. It automatically registers student's attendance in the chosen exam.

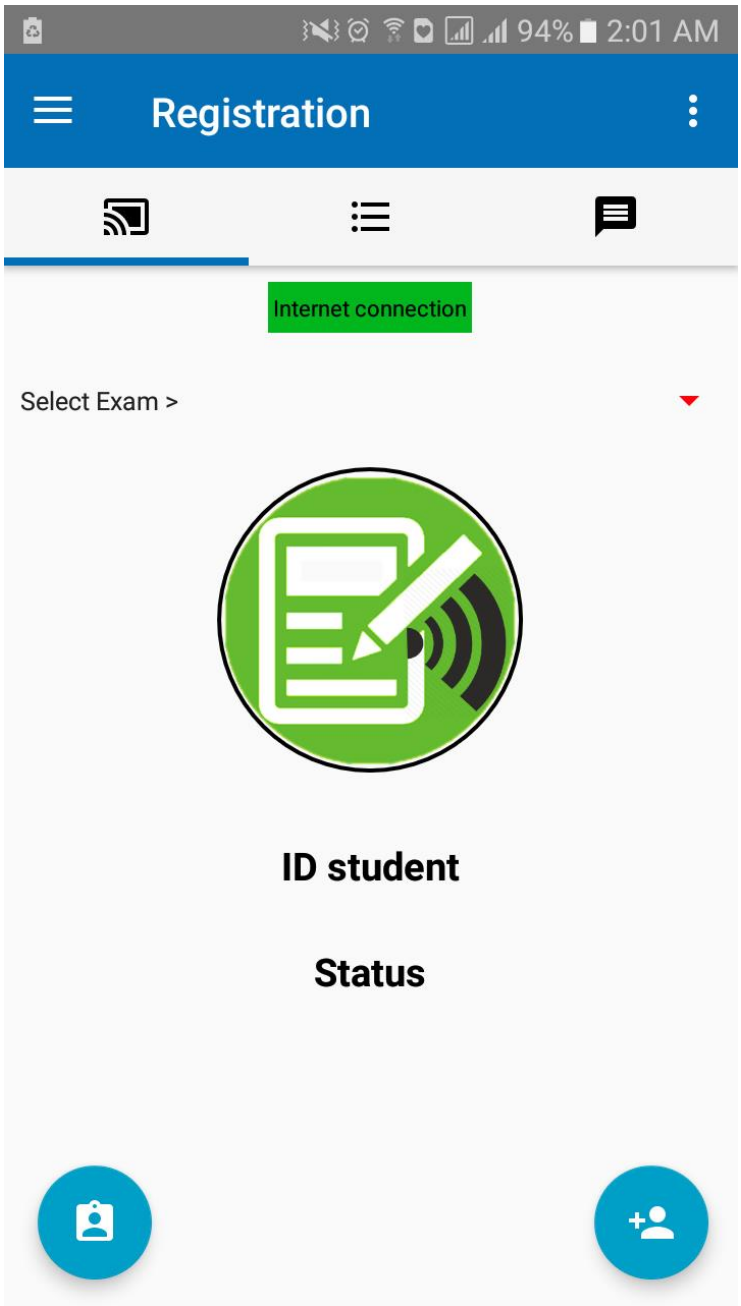


Figure 38: Register Attendance UI

The following figure shows the students that are registered in a chosen exam, if they attended the exam they will have green color on them, otherwise it will be red.

Figure 39: Show Students UI

The following screen show the time table for controller and it also mark the next exam.

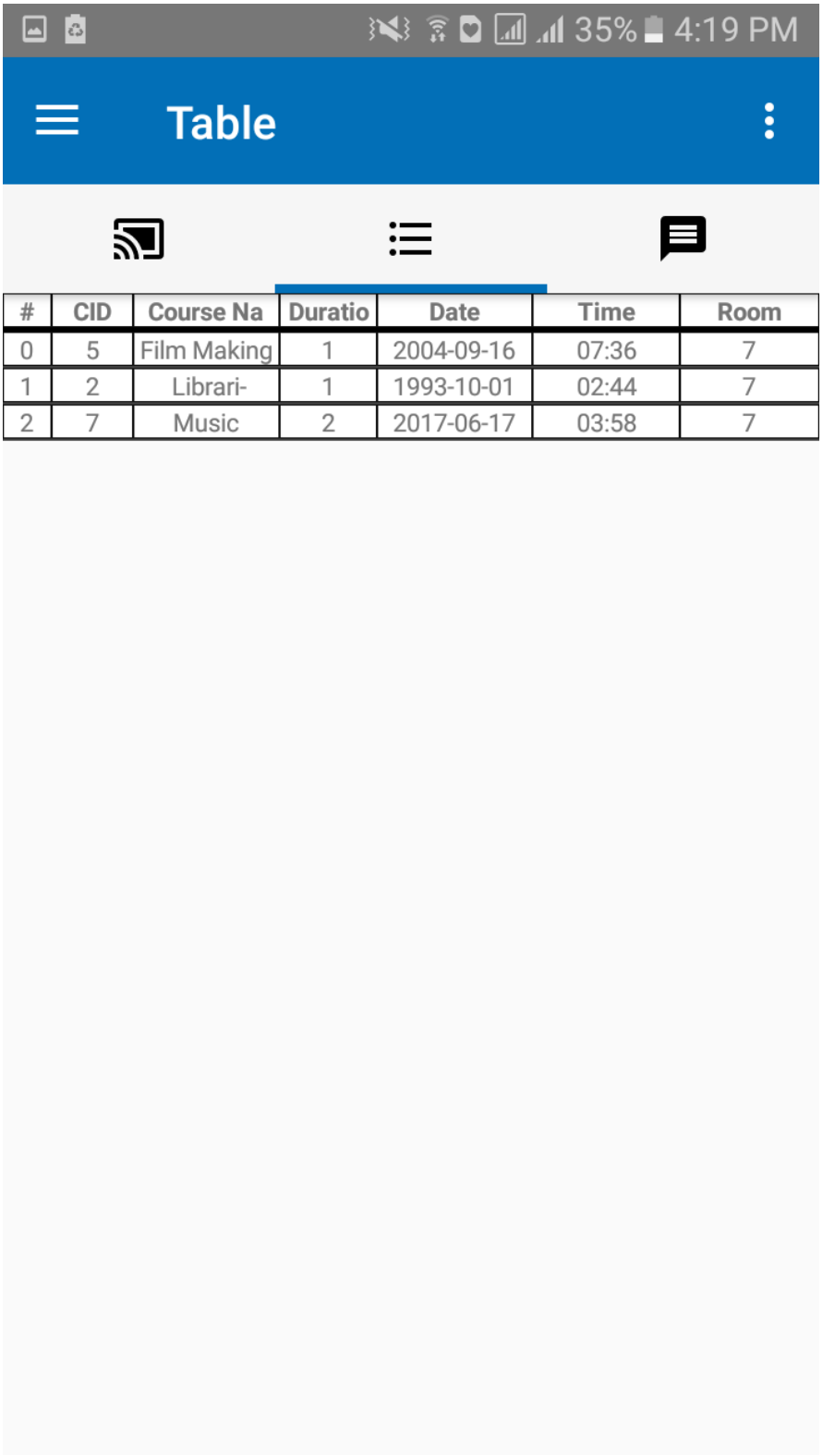


Figure 40: Show Time Table

The following figure shows All Exams that Admin can see and manage.

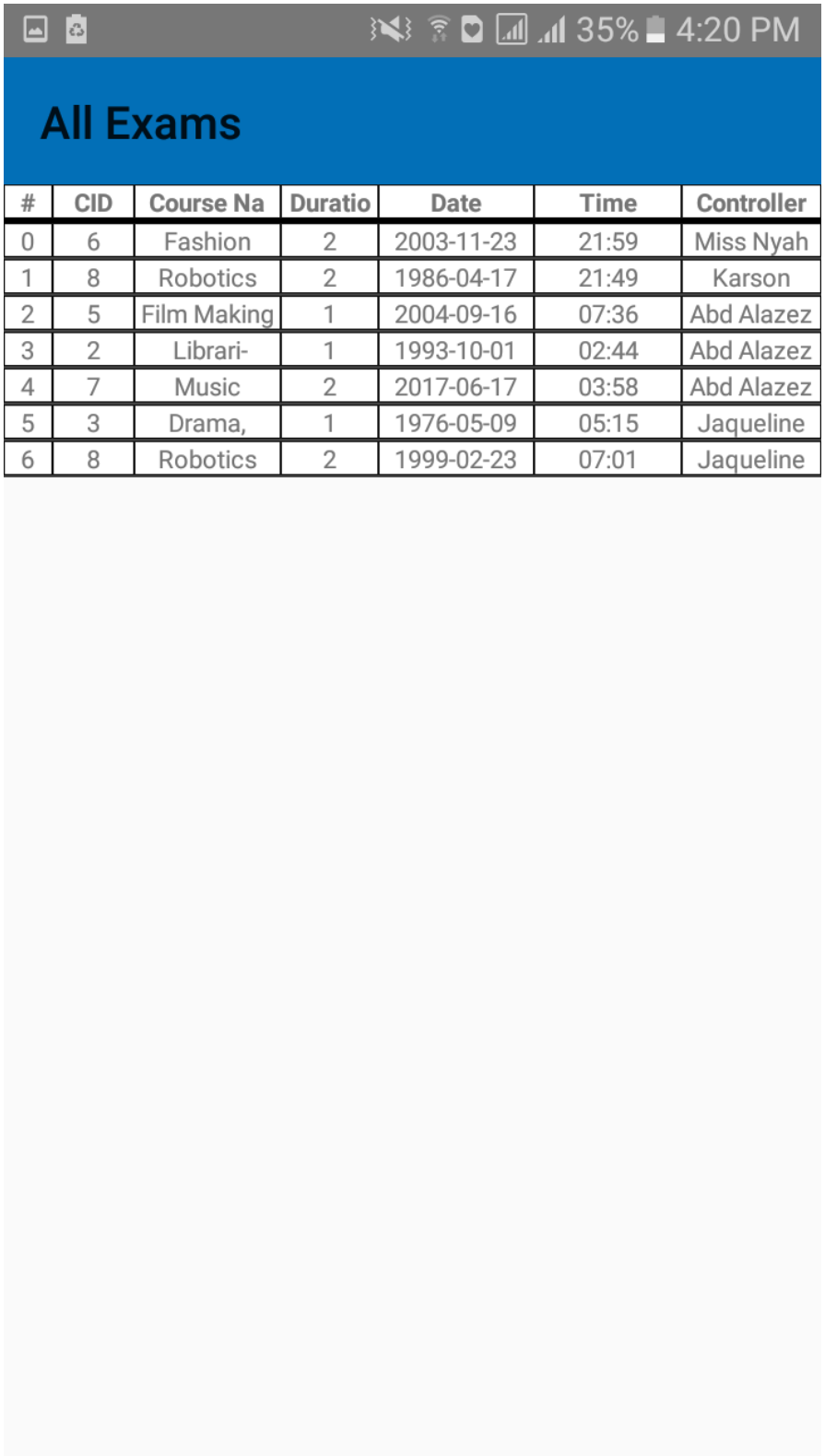


Figure 41: All Exams UI

The following figure show Admin's Maintain Schedule screen, Admin will be able to assign a new controller to a specific exam instead of another controller.

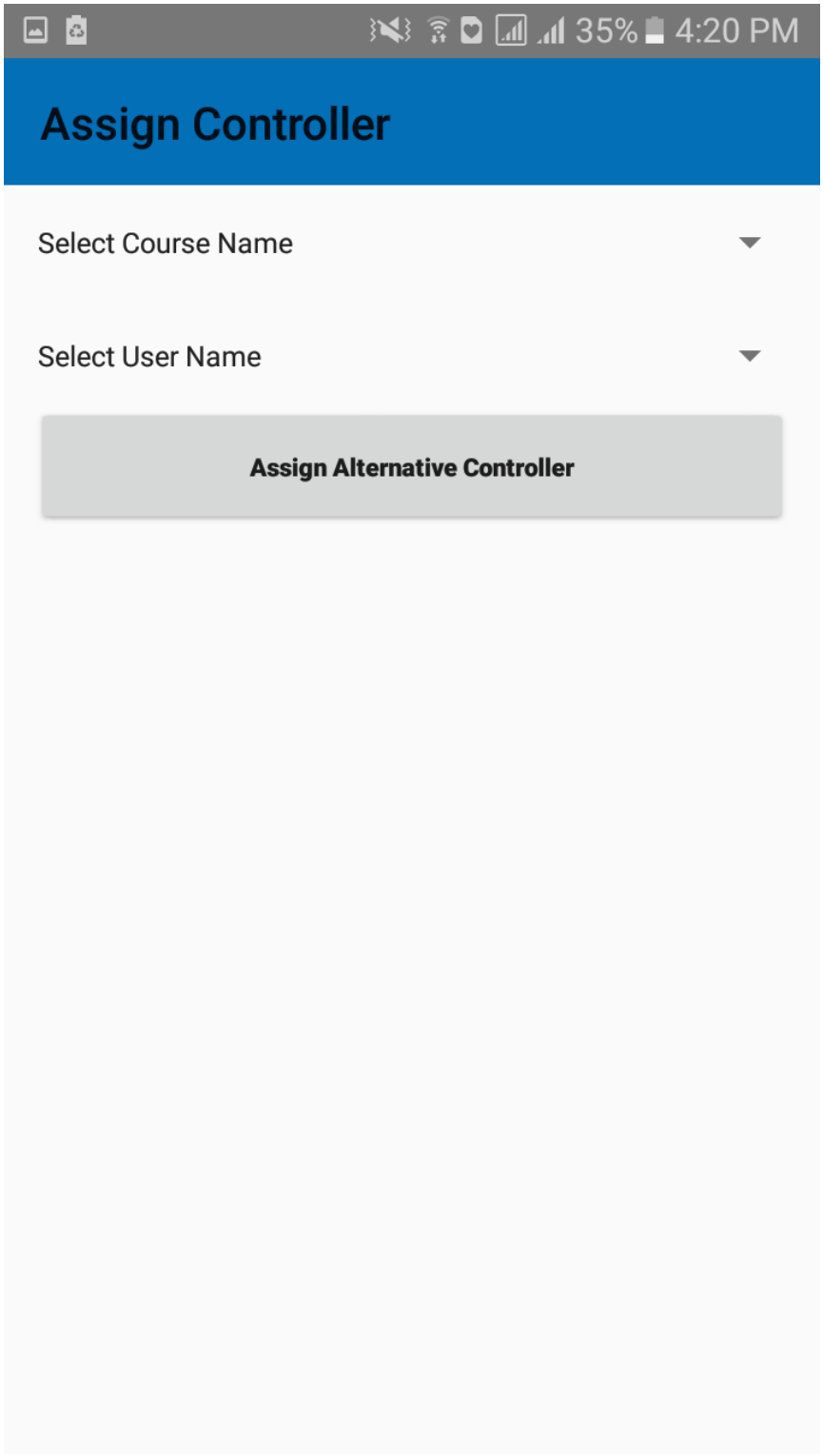


Figure 42: Maintain Schedule

The following figure shows Admin's Handle Notifications screen, Admin will be able to answer all notifications sent by controllers.

Figure 43: Handle Notifications UI

6.2.2 Web Interface

In this section we present web interface for our backend.

The following figure shows web interface that is used to add new exams to exam table by Admin.

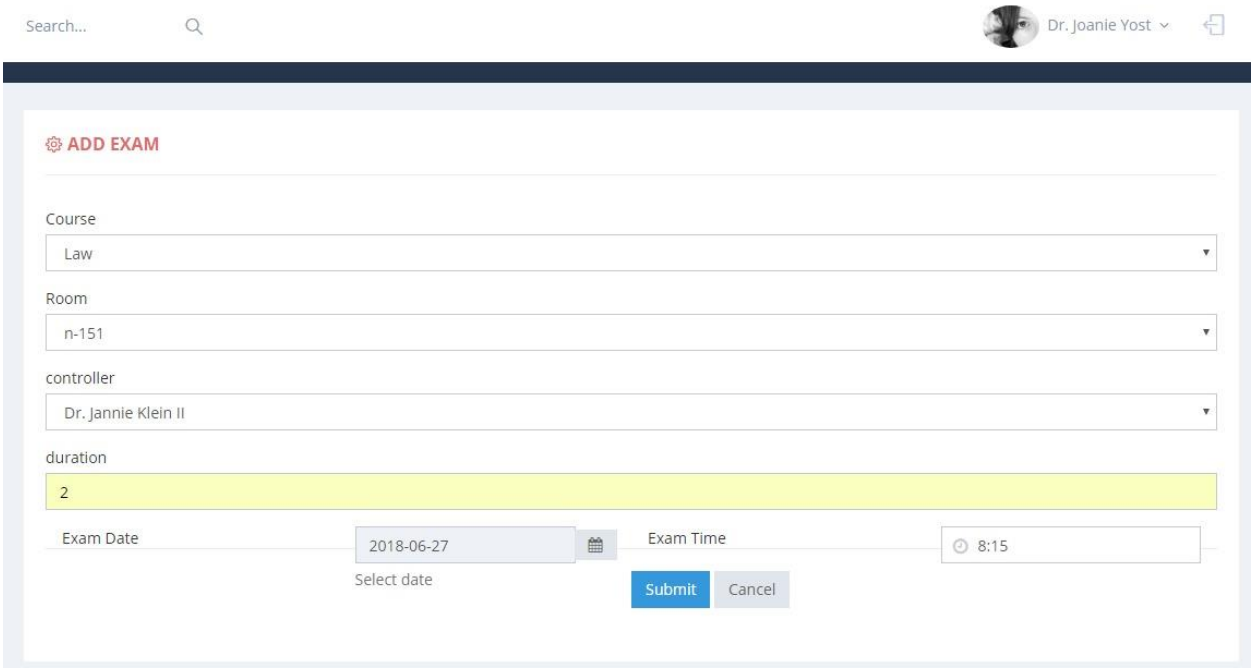


Figure 44: Laravel Add Exam Screen

The following figure shows users table in database, it shows each user and their types.

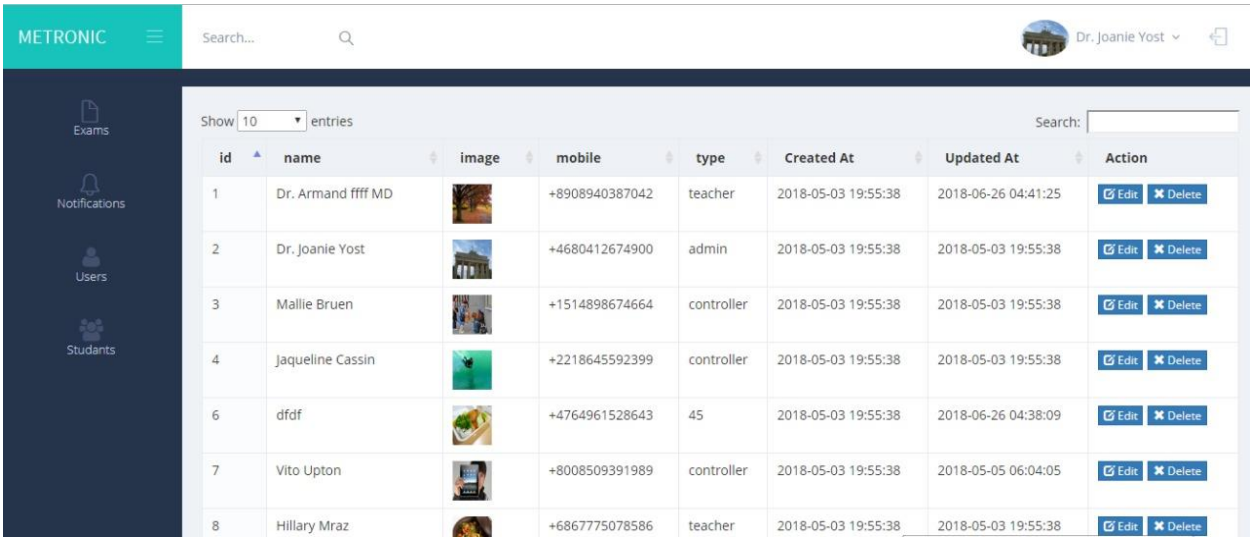


Figure 45: Laravel Show all users

The following figure shows notifications table in database.

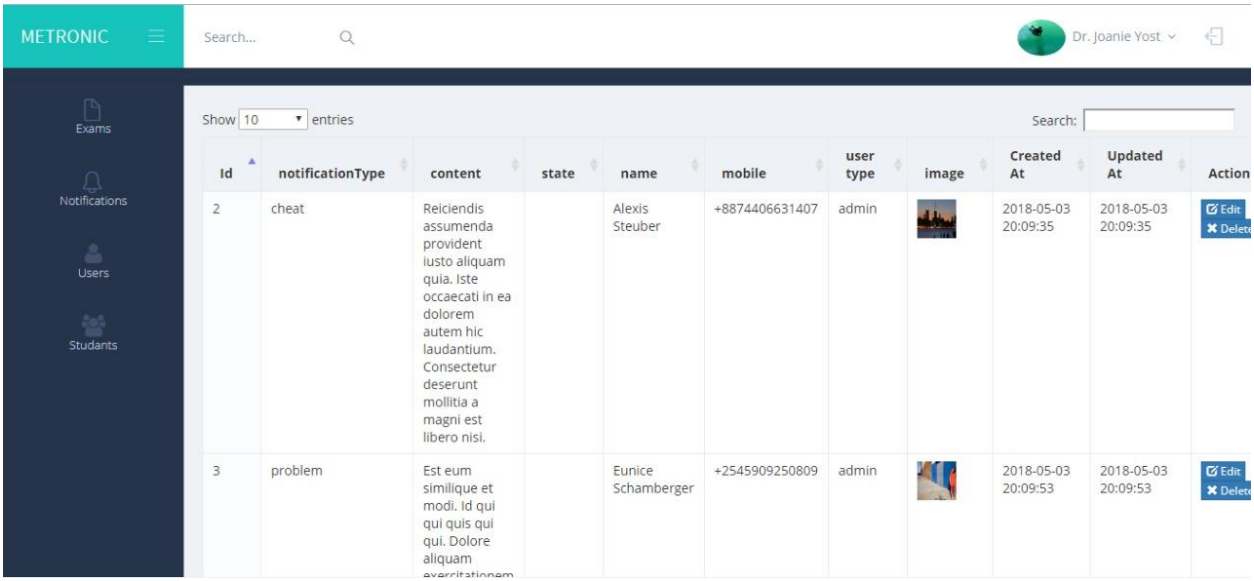
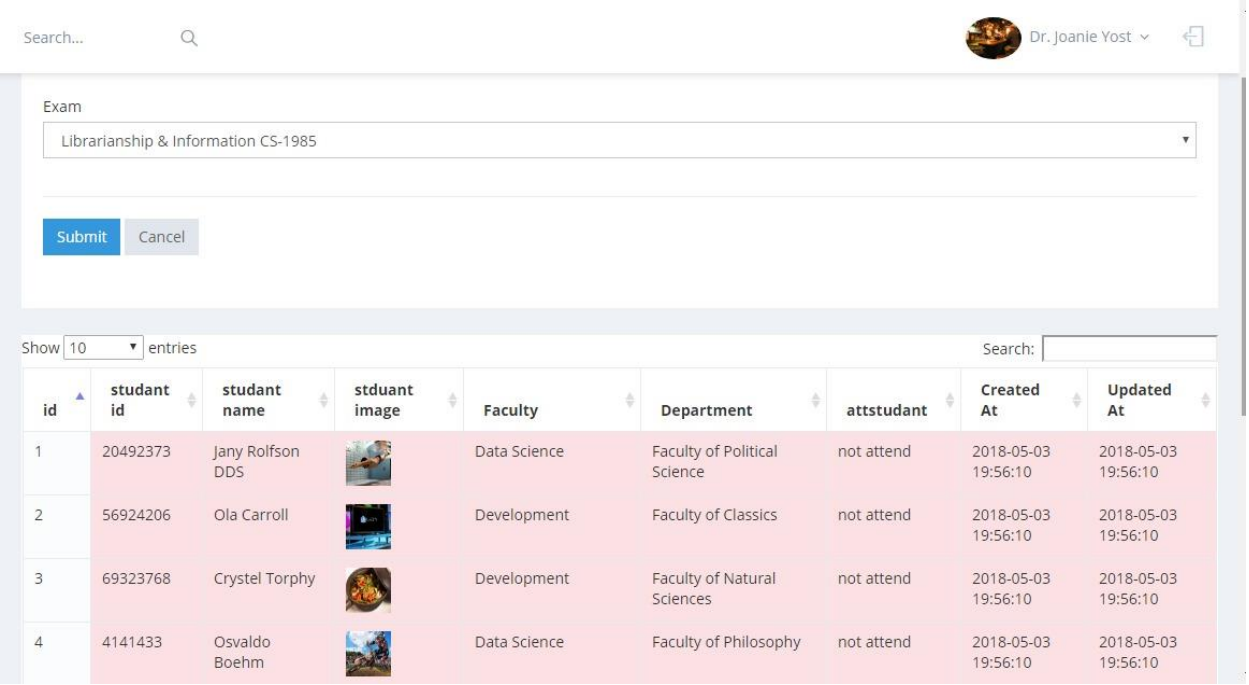


Figure 46: Laravel Show Notifications

The following figure shows students that are registered in a specific exam and it will mark students with green if they attended and red if they didn't.



id	studant id	studant name	stduant image	Faculty	Department	attstudant	Created At	Updated At
1	20492373	Jany Rolfson DDS		Data Science	Faculty of Political Science	not attend	2018-05-03 19:56:10	2018-05-03 19:56:10
2	56924206	Ola Carroll		Development	Faculty of Classics	not attend	2018-05-03 19:56:10	2018-05-03 19:56:10
3	69323768	Crystel Torphy		Development	Faculty of Natural Sciences	not attend	2018-05-03 19:56:10	2018-05-03 19:56:10
4	4141433	Osvaldo Boehm		Data Science	Faculty of Philosophy	not attend	2018-05-03 19:56:10	2018-05-03 19:56:10

Figure 47: Laravel Students Attendance in Exam

The following figure shows all students table and some of their details like student picture, department, faculty and others.

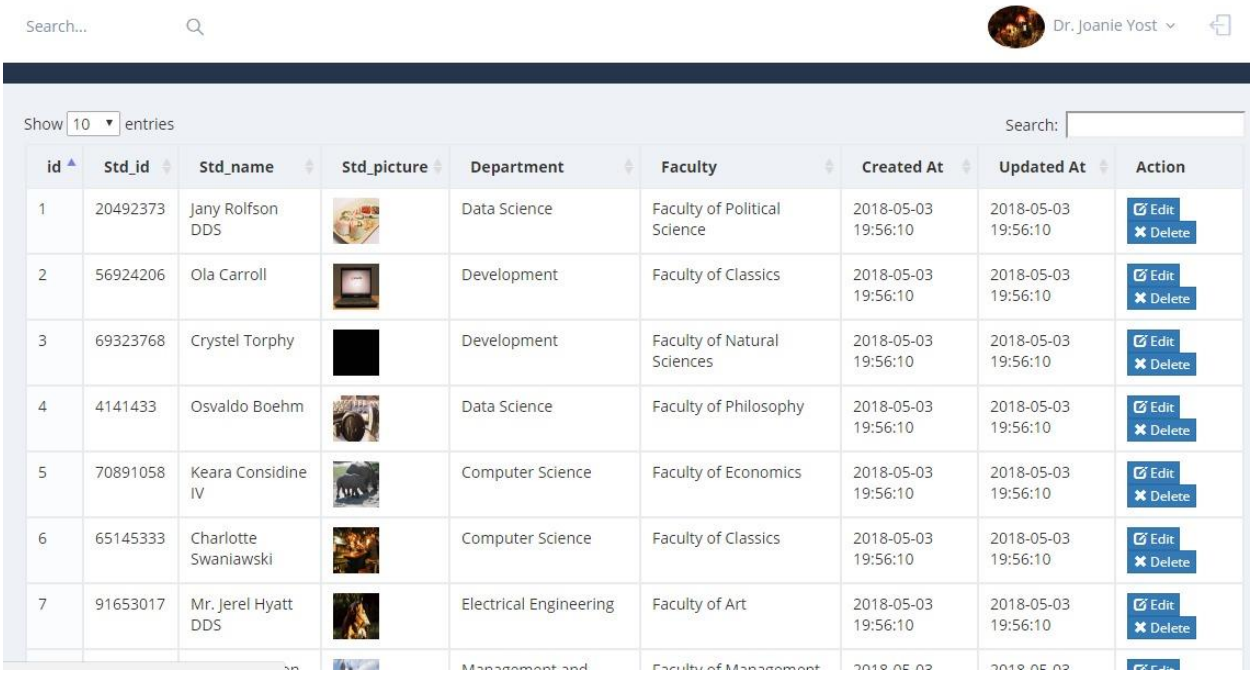


Figure 48: Show all Students

6.3 Testing

In this section we present our tests that we made on the system to insure that all functions are working correctly.

As mentioned in the beginning of this chapter, we developed a small application (Pillow NFC) to write information on NFC to test it on our main application.

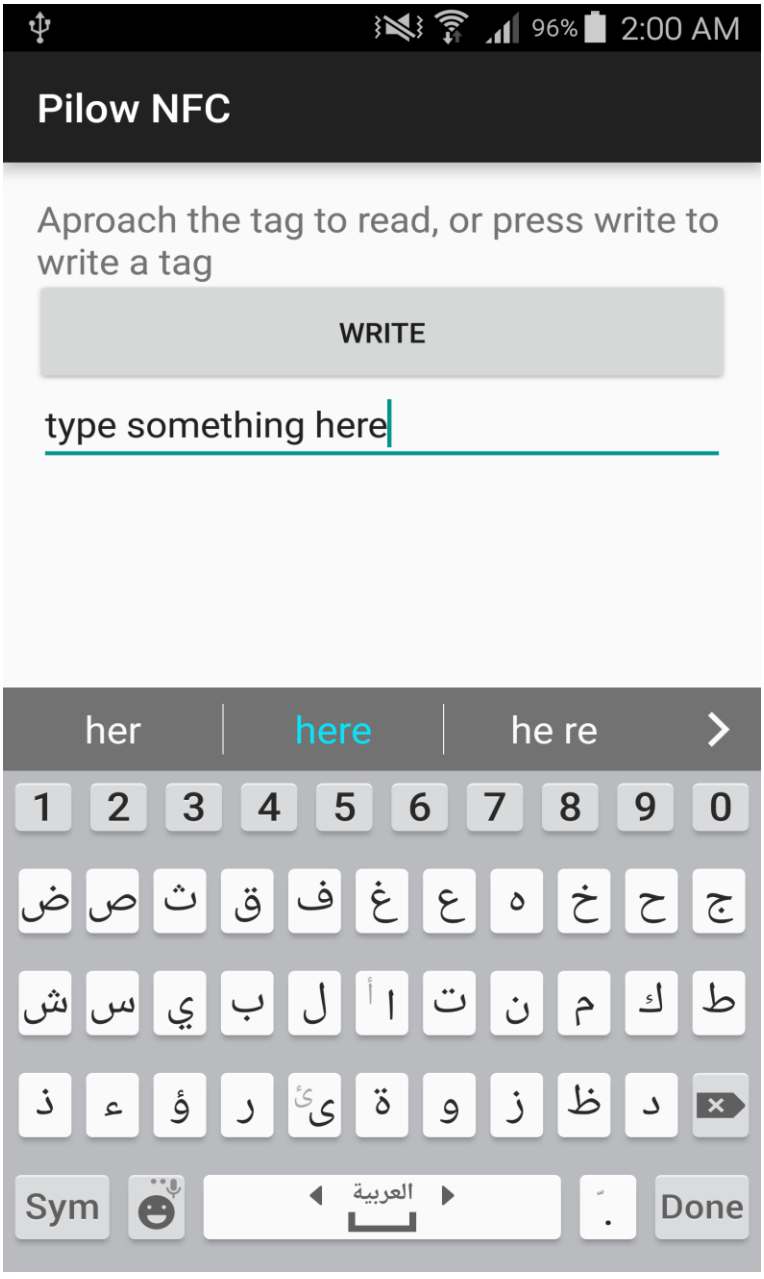


Figure 49: Pilow NFC UI

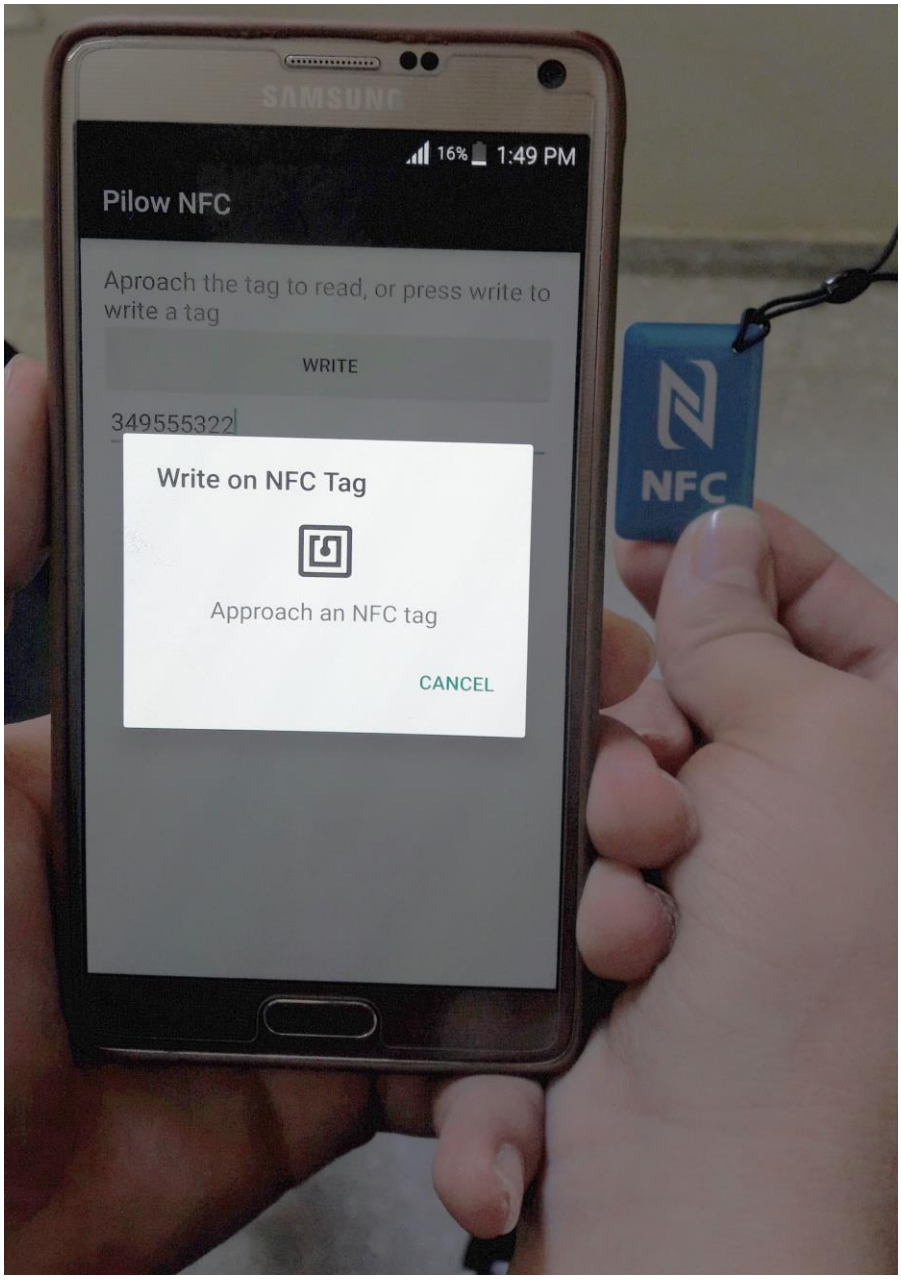


Figure 50: Writing on NFC

And then we used our main application to read student id store in NFC to do the main functionality that our application provide.

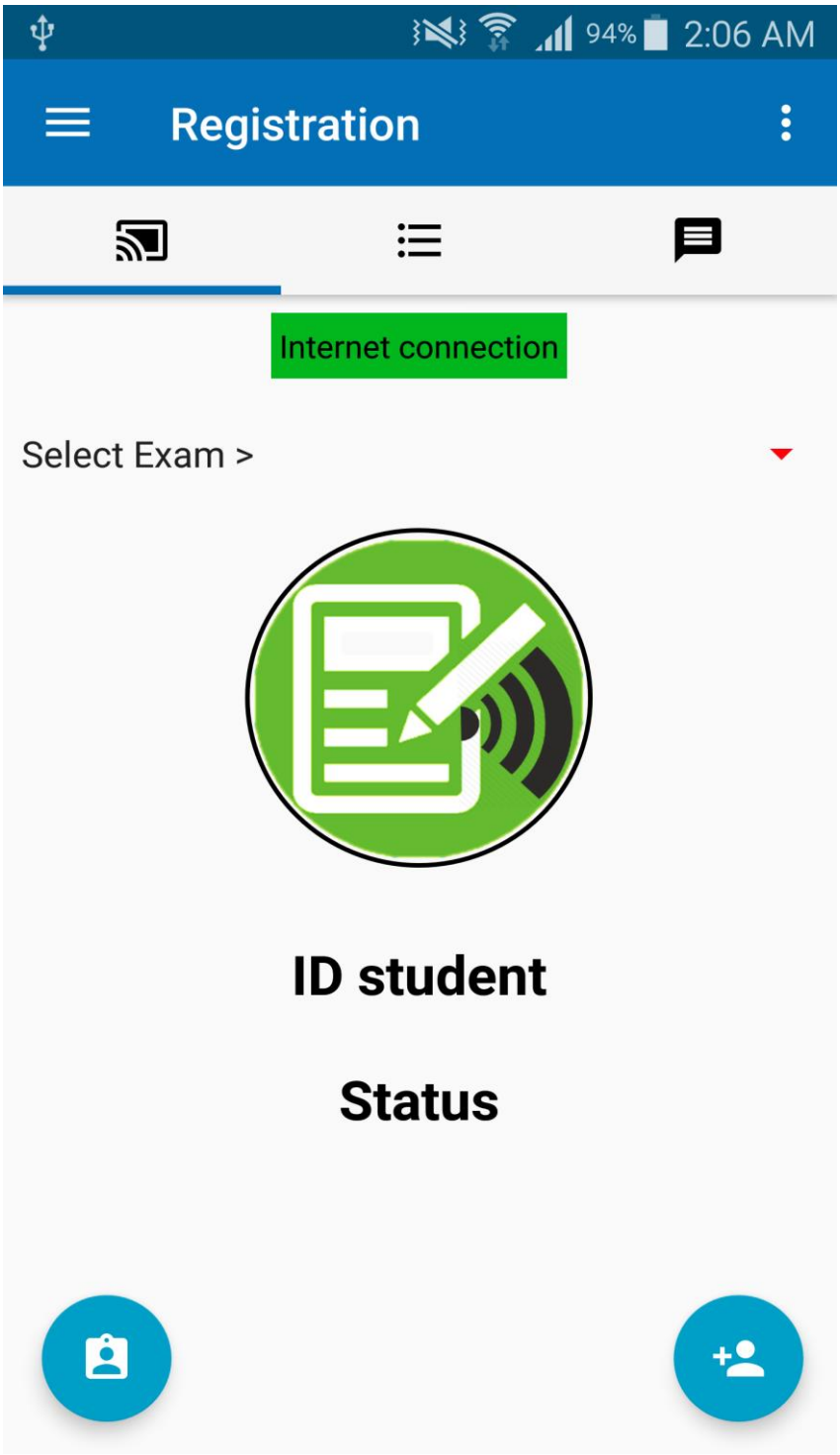


Figure 51: QAR UI

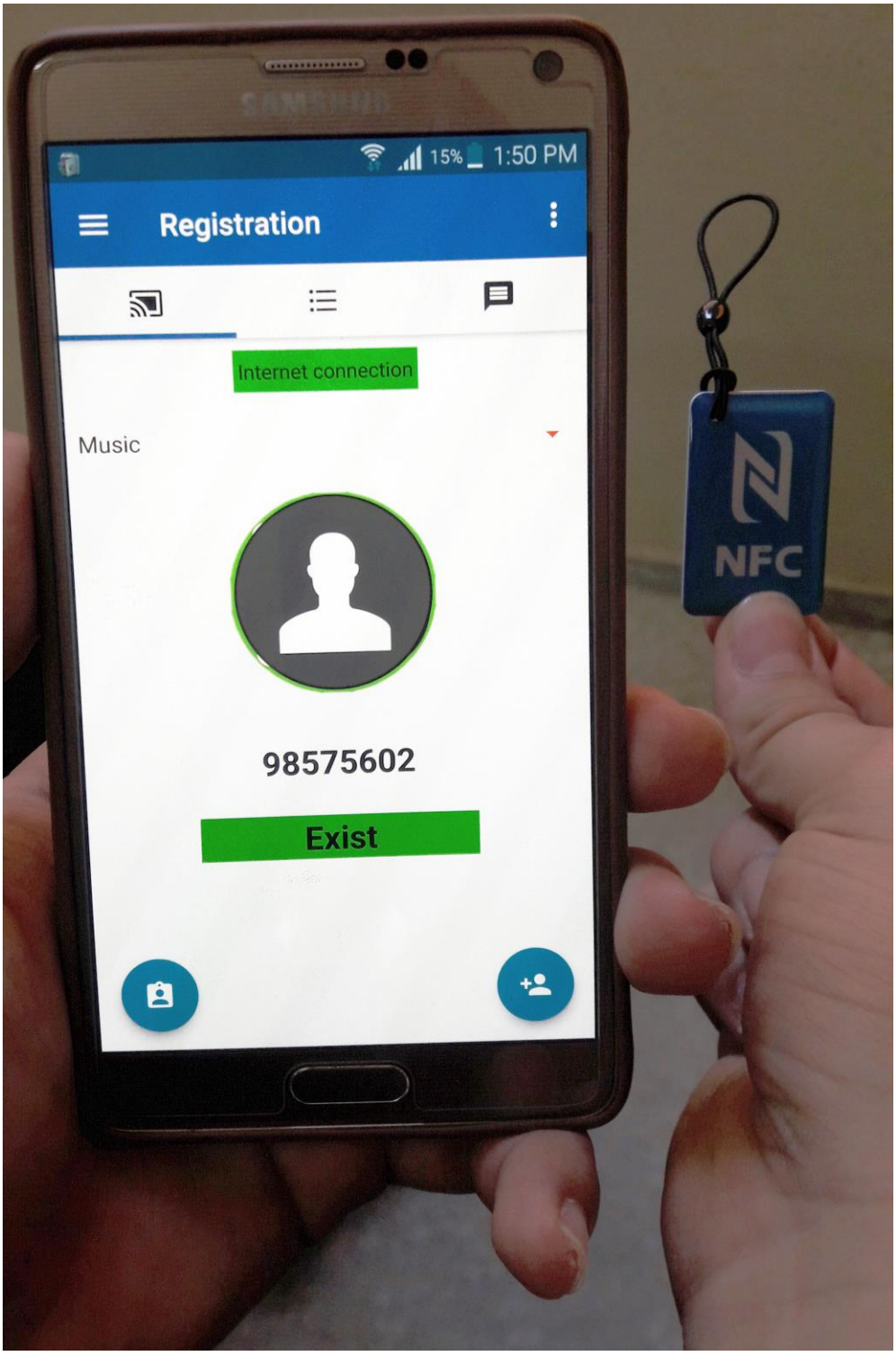


Figure 52: Reading NFC

6.3.1 JUNIT TEST

1- testNumExams() : number of exams that controller with id 13 should have which is right, 3 in and 3 added to database, pass.

2- testNumNotifications() : the actual number of notifications that admin with id 18 should receive is 15, and after test received 15 with 25ms, pass

3- testNumStudents() : the actual number of students who have exam with controller ID 13 are 5 students, after test they are 5 with 25ms, pass

```
17      */
18      @RunWith(AndroidJUnit4.class)
19      public class ExampleInstrumentedTest {
20          @Test
21          public void testNumExams() throws Exception {
22              Context appContext = InstrumentationRegistry.getTargetContext();
23
24              SharedPreferences pref = appContext.getSharedPreferences( s: "JunitTestQar", appContext.MODE_PRIVATE);
25              int valueExam = pref.getInt( s: "JunitTestQarNumExams",  i: -1);
26
27              assertEquals(valueExam,  actual: 3);
28          }
29          @Test
30          public void testNumNotification() throws Exception {
31              Context appContext = InstrumentationRegistry.getTargetContext();
32
33              SharedPreferences pref = appContext.getSharedPreferences( s: "JunitTestQar", appContext.MODE_PRIVATE);
34              int valueNotification = pref.getInt( s: "JunitTestQarNumNoti",  i: -1);
35              assertEquals(valueNotification,  actual: 15);
36          }
37          @Test
38          public void testNumStudents() throws Exception {
39              Context appContext = InstrumentationRegistry.getTargetContext();
40
41              SharedPreferences pref = appContext.getSharedPreferences( s: "JunitTestQar", appContext.MODE_PRIVATE);
42              int valueStudent = pref.getInt( s: "JunitTestQarNumStu",  i: -1);
43
44              assertEquals(valueStudent,  actual: 5);
45          }
46      }
47
```

Figure 53: JUNIT Class 1

As mentioned in figure 41, time elapsed to run JUNIT test is between 15 and 25ms

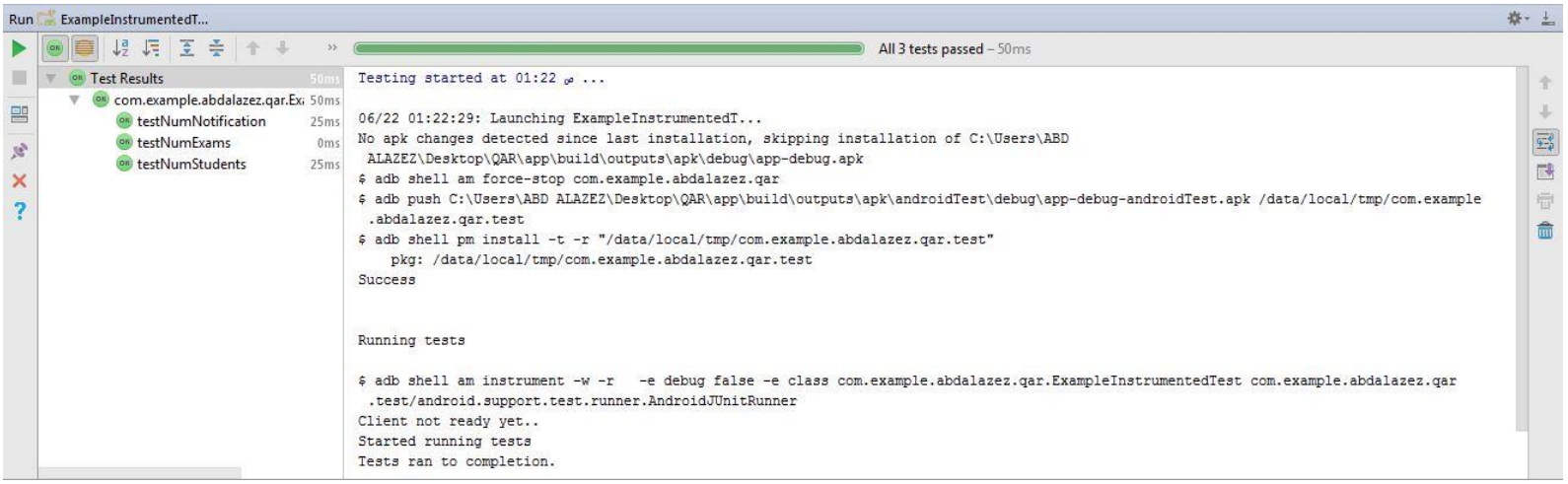


Figure 54: JUNIT Test

testNumStudentsIsAttendance() : the actual number of students that were registered in exam with id 8 are 3 students from 11, after test there are 3 students, pass.

```
@Test
public void testNumStudentsIsAttendance() throws Exception {
    Context appContext = InstrumentationRegistry.getTargetContext();

    SharedPreferences pref = appContext.getSharedPreferences( s: "JunitTestQar", appContext.MODE_PRIVATE);
    int valueStudent = pref.getInt( s: "JunitTestQarNumStudentsIsAttendance", i: -1);

    assertEquals(valueStudent, actual: 3);
}
}
```

Figure 55: JUNIT Class 2

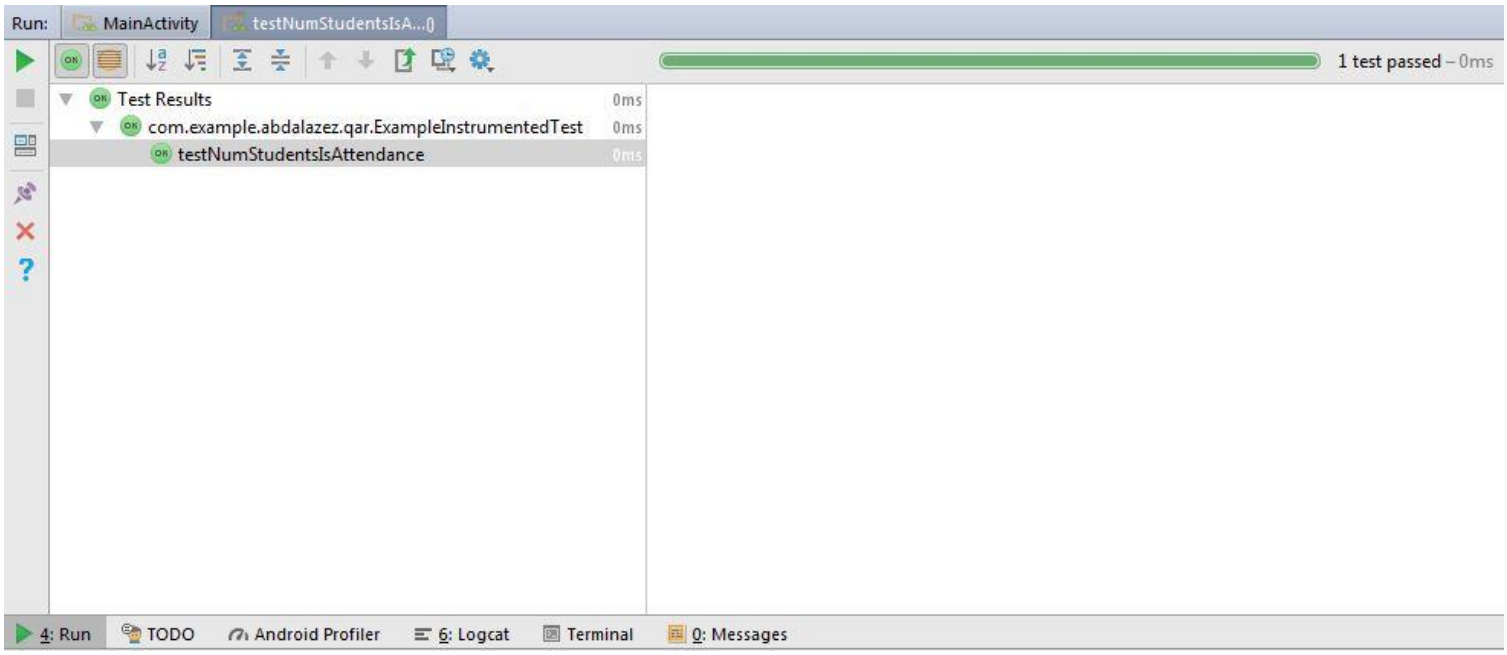


Figure 56: JUNIT Test 2

6.3.2 User Test

To test the system quality and usability from the user perspective, we asked users to test our application and evaluate the system based on their usage and we asked them for feedbacks.

Some users were not satisfied with the application; we took their feedback and made the updates on the system based on their feedback. Some users were not happy because they are not using android devices and the application is working on android operating systems.

After we made our updates on the system, most users were satisfied with the new functionalities that our system provide and they said that the system was useful, new and practical and they are looking forward to see the system to be used in organizations.

Chapter Seven

Conclusion & Future Work

Chapter 7: Conclusion & Future Work

7.1 Conclusion

QAR application is a helpful, easy to use and effective system to help controlling exams and registering attendance and it can be extended later for more uses such as registering attendance in festivals, in restaurants, in organizations and more.

If you're building something cool and needs a fast, reliable database, user authentication and usage tracking, then Firebase is a great way to go. Not recommend for complex project. Large application, very few people use serverless, if you do not want later to pay expensive to sit back.

7.2 Future Work

QAR can be extended to make a customized version to a specific institution or organization with the requirements specified by the institution.

We are also going to enhance some of our functions such as search function to search for teachers and students using name or id.

We are going to enhance notifications to be able to sort notifications based on type or date or sender.

Bibliography

[1] Number of mobile phone users worldwind from 2015 to 2020 (in billions) – (November 2016) retrieved from https://www.statista.com/statistics/274774/forecast-of-mobile-phone-users-worldwide/

[2] www.huesnshades.com. (2017). Fees & Attendance Register (2.5) [Mobile application software]. Retrieved from https://play.google.com/store/apps/details?id=com.hns.feeregister

[3] TnA. (2018). Time NFC Attendance (2.1.1) [Mobile application software]. Retrieved from https://play.google.com/store/apps/details?id=dlogic.net.nfc_tna&hl=ar

[4]. Logical Classes. (2018). My Attendance (AEBAS) (1.0) [Mobile application software]. Retrieved from https://play.google.com/store/apps/details?id=com.attendance.aebas

[5] NFC card - (Feb 25, 2017) Retrieved from https://ar.aliexpress.com/item/Waterproof-NFC-Tags-lable-Ntag216-13-56mhz-RFID-Smart-Card-for-All-NFC-enabled-phone-min/32814589897.html?spm=a2g0s.9042311.0.0.20494c4dw1sVuD

[6] An Introduction to Near-Field Communication and the Contactless Communication API (June 2006) retrieved from http://www.oracle.com/technetwork/articles/javame/nfc-140183.html

[7]. Ian Sommerville. Software Engineering (9 Edition). Pearson, 2010.

[8]. what is token based authentication - (Jul 2015) Retrieved from https://stackoverflow.com/questions/1592534/what-is-token-based-authentication

[9] General architecture of Laravel based application - (Jun 11, 2017) retrieved from http://www.cutehits.com/2017/06/general-architecture-of-laravel-based-application/

[10] android connect web server php mysql - (Mar 2017) retrieved from http://www.thaicreate.com/mobile/android-connect-web-server-php-mysql.html

[11]. Michael Roland. (2017). NFC TagInfo (1.12) [Mobile application software]. Retrieved from https://play.google.com/store/apps/details?id=at.mroland.android.apps.nfctaginfo

[12]. 5 reasons to not use Firebase for a big project - (Nov 28, 2017) Retrieved from https://medium.com/@reactsharing.com/5-reasons-to-not-use-firebase-for-a-big-project-81b543c77e8c