****

**Examination System**

Table of Contents

[Chapter 1 6](#_Toc313887676)

[Introduction: 6](#_Toc313887678)

[Problem defination examination: 7](#_Toc313887678)

[Problem defination examination: 7](#_Toc313887678)

[Problem defination examination: 7](#_Toc313887678)

[Problem solution: 8](#_Toc313887678)

[Objectives and concentrations: 8](#_Toc313887679)

[Scope and limitations: 9](#_Toc313887680)

[Project Organization (The team): 10](#_Toc313887681)

[Monitoring and reporting mechanisms: 10](#_Toc313887686)

[Project management approach: 11](#_Toc313887687)

[Chapter 2 12](#_Toc313887688)

[Software Requirement Specification 12](#_Toc313887689)

[(1) Preface: 13](#_Toc313887690)

[(2) Introduction: 13](#_Toc313887691)

[(3) Glossary: 13](#_Toc313887692)

[(4) User Requirements Definition: 13](#_Toc313887693)

[(4.1)The products and process features: 14](#_Toc313887694)

[(6) System Requirement Specification: 14](#_Toc313887696)

[(6.1) Functional System Requirement: 14](#_Toc313887697)

[(6.2) Non-Functional System Requirements: 16](#_Toc313887698)

[6.4) Software Quality Attributes 16](#_Toc313887699)

[(6.3) System Interfaces: 17](#_Toc313887700)

[(7) System Models: 17](#_Toc313887701)

[(8) System Evolution: 17](#_Toc313887702)

[Chapter (3) 18](#_Toc313887704)

[System Design 18](#_Toc313887705)

[Introduction: 19](#_Toc313887706)

[Project Analysis and Design: 19](#_Toc313887706)

[Context Diagram: 20](#_Toc313887707)

[Models: 21](#_Toc313887708)

[3.1) Interaction model: 21](#_Toc313887709)

[System Architecture: 22](#_Toc313887710)

[Data flow Diagram: 23](#_Toc313887714)

[Data Base Diagram: 24](#_Toc313887714)

[ER Diagram: 25](#_Toc313887706)

[Chapter (4) 26](#_Toc313887704)

[Software Implementation 26](#_Toc313887705)

[Technology used: 27](#_Toc313887706)

[Home page for student: 28](#_Toc313887706)

[Home page for admin: 29](#_Toc313887706)

[Admin Login: 30](#_Toc313887706)

[Admin Task: 31](#_Toc313887706)

[Manage Exam: 32](#_Toc313887706)

[Question Paper: 33](#_Toc313887706)

[Manage course: 34](#_Toc313887706)

Manage Student[: 35](#_Toc313887706)

[Student Regestration: 36](#_Toc313887706)

[Student Login: 37](#_Toc313887706)

[Exam: 38](#_Toc313887706)

[Setting: 39](#_Toc313887706)

[Chapter (5) 40](#_Toc313887704)

[System test 40](#_Toc313887705)

[Home page for student: 41](#_Toc313887706)

[Exam: 42](#_Toc313887706)

[result exam Question: 46](#_Toc313887706)

[setting: 47](#_Toc313887706)

[Admin Login: 48](#_Toc313887706)

[Home page for Admin: 49](#_Toc313887706)

[Add new student: 50](#_Toc313887706)

[Question paper: 51](#_Toc313887706)

[Edit course: 52](#_Toc313887706)

[Manage student: 53](#_Toc313887706)

[Manage Exam: 54](#_Toc313887706)

[Managee course: 55](#_Toc313887706)

[The code of examination system for Web: 56](#_Toc313887706)

[examination system for Android : 75](#_Toc313887706)

# 

# Chapter 1

**Introduction**

## 1.1) Introduction:

This document will propose all features and procedures to develop the system.

This document specially containing details about objectives, scope limitation, process model, primary requirements, team development, possible project risks, project schedule, and finally monitoring and reporting mechanisms.

Exam System is very useful for Educational Institute to prepare an exam, safe the time that will take to check the paper and prepare mark sheets. It will help the Institute to testing of students and develop their skills. But the disadvantages for this system, it takes a lot of times when you prepare the exam at the first time for usage. And we are needs number of computers with the same number of students.

The effective use of "Examination System", any Educational Institute or training centers can be use it to develop their strategy for putting the exams, and for getting better results in less time.

## **Problem definition Examination**

# is very helpful to users. The aim of this project is to provide quick, immediate and easy way to appear the exam. It can provide special advantages to the students/applicants. The examination system can automatically add the marks allocated in each question to determine the total mark for the questions. A time limit can be set for the questions. Examination System allows jumping to specific questions based on the previous answer. The examination system limits the number of times a student can write a question. Login module helps the user to login to the site. For that he/she must type the username and password correctly. The login provision in this page helps the already registered user to directly access the site and there is a link for registration to a user who is new to this site. Student module is mainly for the students. This helps the students to register for the exam and answer the exam. For registration name, address, phone no, role id, password should be entered and other are filled. This will provide result after the exam according to correct and wrong answer. Administrator module is mainly for the administrator. This will contain the creation of question paper, preview of already created question paper, and the report of the administrator. For creating the question paper he/she must enter Exam ID, No: of questions, total time (in min), Marks per Question, Option type. For showing the preview of already created question paper he/she must enter the correct Exam id.

# General objective

# General objective of our project is to change the current manual system into computerized one. This project would be very useful for educational institutes where regular evaluation of students’ is required.

# Specific objective

# examination project assesses student by conducting objective tests.

# Responses by the candidates will be checked automatically.

# It reduces time consuming. Being an integrated examination system reduce paper work. Questions can have multiple options, multiple answers or can be text answers.

# To allow department to create tests and answers

# The result will be shown after some time to the participating students.

# Can generate various report for evaluation purpose when and where required.

# This project will enable educational institutes to conduct test and have automated checking of answers based on the response by the candidates.

# It would enable educational status to perform testes quiz and create feedback forms.

### The problems are solved

# • Students need to wait for their results.

# • Examiner are not able to maintain the student record in large quantity.

# • Manipulation of student record is difficult for examiners .

# • And another problem of sending question paper from one place to another place.

# • And have a problem of change the choice of option which is not possible in paper .

### Problem Solution

# • We introduce our project to conduct exam easilycomputer system and reducing paper work.

# • We provide user id and password. The id password is unique so its data will be secured.

# • Admin can upload the question paper .

# • Admin can change the profile of student.

### Objectives and concentrations:

* Corporate between the data stored in the server of the Institution and our On-line Exam system. To deal with On-line System in an easy way and an efficient mannered. (connection process)
* Create strong and secrete data base that allow for any connection in a secret way, to prevent any outside or inside attacks.
* Specify a privilege for each person to allow each person use this system to create his own exam. And have a complete control on his exam.
* Allow each person to create more than one exam with different way to create variant questions.

### Scope and limitations:

* Examination System is designed for Educational Institutes (like schools, universities, training centers).
* The system handles all the operations, and generates reports as soon as the test is finish, that includes name, mark, time spent to solve the exam.
* Allow students to see or display his answers after the exam is finish.
* The type of questions is only multiple choice or true and false.

## 1.2) Project Organization (The team):

|  |  |  |
| --- | --- | --- |
|  | **Job Title** | **Description** |
| **1** | Project Manager | * To manage all processes in the project |
| **2** | SW Designer | * To design the models and diagrams that helps the programmer in implementation phase. |
| **3** | Two Testers | * One from outside the team and the other from the inside the project team. |
| **4** | Two programmers | * Professional in ASP.NET and SQL * To programming the processes of the project. |
| **5** | SW Analyst | * To analyze the requirements of Examination System. |
| **6** | Writer | * Collects drafts from each member. * Rewrite and reformate the documents come from each member. * Have good print skills. * Have a good skill to correct grammars of statements. |

## 1.4) Monitoring and reporting mechanisms:

The manager should monitor all activities in the project via minimize, avoid the risks or via management control as follows:

* + - 1. Put a table for all SW requirements and print in percentage how much finish.
      2. Using software programming to monitor programmer's progress.
      3. Using spyware profile to monitor the team.
      4. Using software that calculate how many lines written per hour.
      5. monitoring the risks as follows:
         1. Change the probability and effect.
         2. Delete risks or add a new one depends on the working on project.

## 1.5) Project management approach:

**Software Process Model:**

To solve an actual problems in an industry, software developer or a team of developers must integrate with a development strategy that include the process, methods and tools layer and generic phases. This strategy is often referred to a process model or a software developing paradigm.

Our project follows the **waterfall model.**

**The steps of waterfall model are:**

* Requirement Definition
* System and Software Design
* Implementation
* Integration and System Testing
* Operation and Maintenance

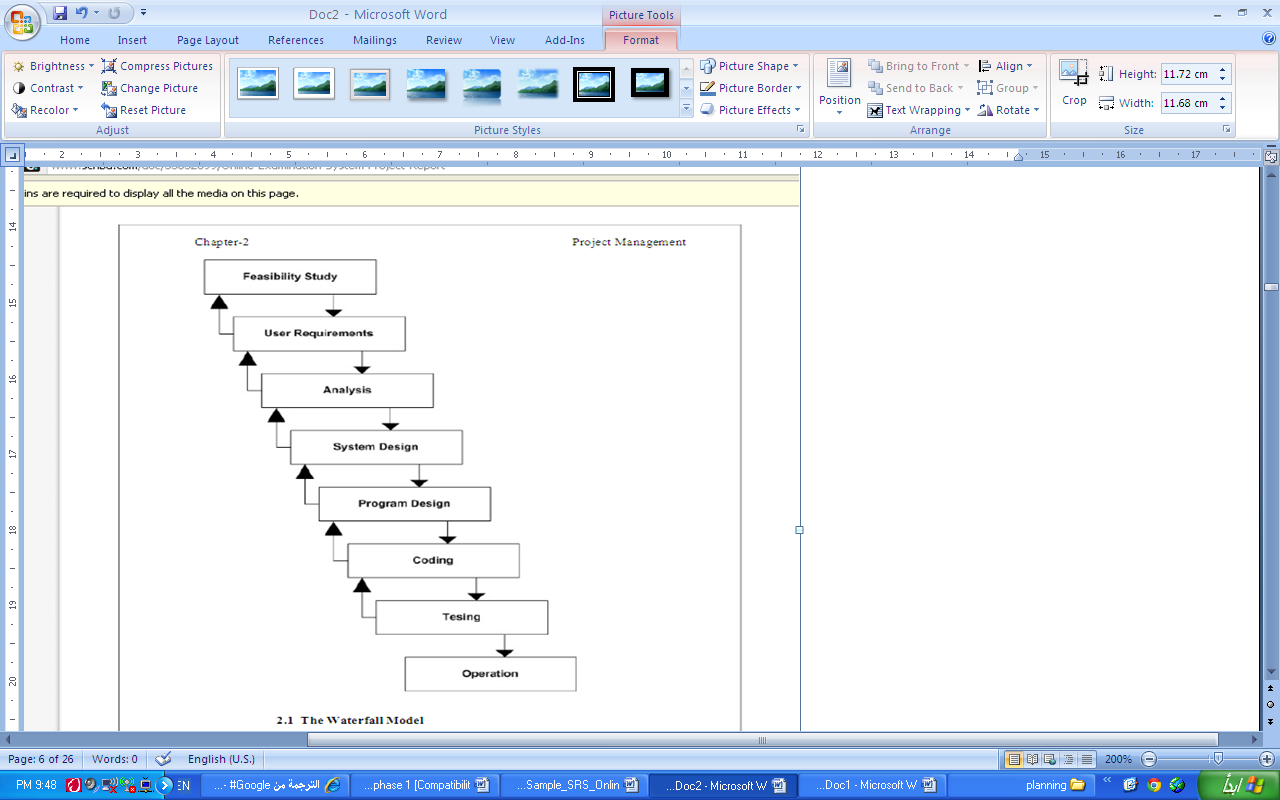


Figure (1.1): Waterfall model

# 

# Chapter 2

# Software Requirement Specification

## (1) Preface:

This is for ***Examination System***.

This document is the basic intended for any individual user, developer, tester, project manager or documentation writer that needs to understand the basic system architecture and its specifications.

## (2) Introduction:

The purpose of this document is to write the functional and non functional user or system requirements that represent the characteristics of ***Examination System***.

The scope and limitation of this system is:

* The Examination System design to educational institutes.
* Hold all operation and generate reports to student, teachers and administrator.
* Support multiple choices questions.
* Allow the student to prochoice the answer and to see his mark.
* Verify a security, authority and safty.

## (3) Glossary:

|  |  |  |
| --- | --- | --- |
|  | Short name | Description |
| 1 | ES | Examination System |
| 2 | Examination system | An exam written on a web site and solves the questions, also on the same web site from any place by entered user name and password. |
| 3 | Administrator | Who is responsible to create a new course, delete course, add member or delete it, i.e.: the person who control the system |
| 4 | Faculty member | A teacher in the faculty |

## (4) User Requirements Definition:

The user requirement for this system is to make the system fast, flexible, less prone to error, reduce expenses and save the time.

* Time can be saved by scheduling the exams, if it is available a question bank to store questions for different subjects.
* A system can be given a mark by checking the students answers, and give the result as soon as students finish his exam.
* A facility to generate a result chart as pre required without manual interface.
* The system should have records of students and faculty that can be access to the system which can be used only for the authorized person.
* The system should be more secure for management user records and more reliable to work at any conditions.

### (4.1)The products and process features:

This system must be designed as user required. So, the complete requirement must be found:

* **Quick scheduling:**

The system helps the faculty member to generate an automatic exam instead of using papers. Which save a time for writing, checking and for input marks. Also, student can see the exam when he login as an individual to the system.

* **Immediate results and solutions:**

When the student finishes his exam, the system checks her answers and compared with the correct answer. And the system saves the incorrect and correct answers and calculates the mark of correct answers. Then give the total mark. And send a report for student to see where he is fault.

* **Easy to store and retrieve information:**

Rather to save the information on a papers or in separate sheets. There are a data base management to store and retrieve the information needed by the administrator or Faculty member or student according a report generated by the system.

## (6) System Requirement Specification:

### (6.1) Functional System Requirement:

This section gives a functional requirement that applicable to the Examination System.

There are three sub modules in this phase.

* Candidate module.
* Examiner module.
* Administrator module.

**The functionality of each module is as follows:**

* **Candidate module**: The candidate will logon to the software and take his examination. He can also check his previous examinations marks and his details. The candidate will get result immediately after the completion of the examination.
* **Examiner module:** The database is prepared & loaded into the software. Selection for examination can be done language wise by the examiner. The results will be displayed immediately after completion of the examination.
* **Administrator module:** The administrator collects all the results after successful completion of the examination and sends to the head quarters as and when required

**The features that are available to the Administrator are:**

* The administrator has the full fledged rights over the ES.
* Can create/delete an account.
* Can view the accounts.
* Can change the password.
* Can hide any kind of features from the both of users.
* Insert/delete/edit the information of on the visible ES.
* Can access all the accounts of the faculty members/students.

**The features available to the Students are:**

* Can view the different categories of Test available in their account.
* Can change password.
* Can view their marks.
* Can view the various reading material.
* Can view and modify its profile but can modify it to some limited range.

**The features available to the Examiner are:**

* Can view the different categories of Test conducted by users.
* Can change password.
* Can view their marks.

### (6.2) Non-Functional System Requirements:

#### 6.2.1) Performance Requirements

Some Performance requirements identified is listed below:

* The software shall support use of multiple users at a time.
* There are no other specific performance requirements that will affect development.

#### 6.2.2) Safety Requirements

The database may get crashed at any certain time due to virus or operating system failure. Therefore, it is required to take the database backup.

#### 6.2.3) Security Requirements

Some of the factors that are identified to protect the software from accidental or malicious access, use, modification, destruction, or disclosure are described below. Keep specific log or history data sets

* Assign certain functions to different modules
* Restrict communications between some areas of the program
* Check data integrity for critical variables
* Later version of the software will incorporate encryption techniques in the user/license authentication process.

1. Communication needs to be restricted when the application is validating the user or license. (i.e., using https).

### 6.4) Software Quality Attributes

The Quality of the System is maintained in such a way so that it can be very user friendly to all the users.

The software quality attributes are assumed as under:

* Accurate and hence reliable.
* Secured.
* Fast speed.
* Compatibility.

### (6.3) System Interfaces:

This section describes how the software interfaces with other software products or users for input or output.

#### 6.3.1) User Interface

Application will be accessed through a Browser Interface. The interface would be viewed best using 1024 x 768 and 800 x 600 pixels resolution setting. The software would be fully compatible with Microsoft Internet Explorer for version 6 and above. No user would be able to access any part of the application without logging on to the system.

## (7) System Models:

In this system we are use waterfall model to apply these ideas. Which is help us to separate each step and when we finish a one phase the output of it is the input to the next phase. Also, we can backwards if there is a new requirement or to apply any update.

## (8) System Evolution:

* **Including image support:**

Allow to adding students, faculty members and administrator images to the system. Which available for student to ensure that exam for his teacher. Also, the teacher can see his student's image.

* **Flags:**

Allow the student to put a symbol near the question that helps the student to return and review the questions and change them accordingly.

* **Enable and disable exam:**

Allow the faculty member to control for enable or disable the exam for his students.

* **Allow to transfer exam from one subject to another:**

So, that saves the time to rewrite the questions for future course.

* **Allow to upload the exam from word or excel file:**

So, that saves the time to enter a question in the on-line system, if the teacher needs not the direct answers.

# Chapter (3)

# System Design

## 3.1) Introduction:

Design is the abstraction of a solution; it is a general description of the solution to a problem without the details. Design is view patterns seen in the analysis phase to be a pattern in a design phase. After design phase we can reduce the time required to create the implementation.

In this chapter we are introduce context diagram, models, system architecture, principal system object, design model and object interface.

### 3.2) Project Analysis And Design:

This project will complete in two module which are as follow:-

**EXAMINATION SYSTEM**

**STUDENT**

**ADMIN**

### Admin Aspects:

* Taking back up of the database.
* Editing/deleting/creating the database.
* Changing the password.
* Logging into the system.
* Posting question in the above test.
* View a particular student and delete the student.
* Show result of examination.

### Student Aspects:

* Read the instruction of accessing this site.
* Request for registration.
* See the notification for Test.
* Logging into the system.
* Appear the Test or Exam.
* Find result at the end of exam

## 3.3) Context Diagram:

This diagram represents what are the bounders and scope of **Examination System** project. It describes the main objective of the system and its entities involved.

**Examination system**

**Administrator**

**Student**

**Faculty**

Figure (3.1.1): the context diagram of Examination System

**The Administrator can be done the following:**

* Create/delete accounts (add a list of faculty names and list of his student)
* Change password for Faculty/Student
* Create/ delete/update courses (subject).

**The Faculty can be done the following:**

* Change password.
* Insert questions.
* Specify the answers.
* Update mark of questions and answers.

**The Student can be done the following:**

* Change password.
* Choose exam.
* Review answers.
* See his exam mark.
* View other material.

## 3.4) Models:

### 3.4.1) Interaction model:

Is a dynamic model that shows how the system interacts with its environment. We use a data flow diagram.

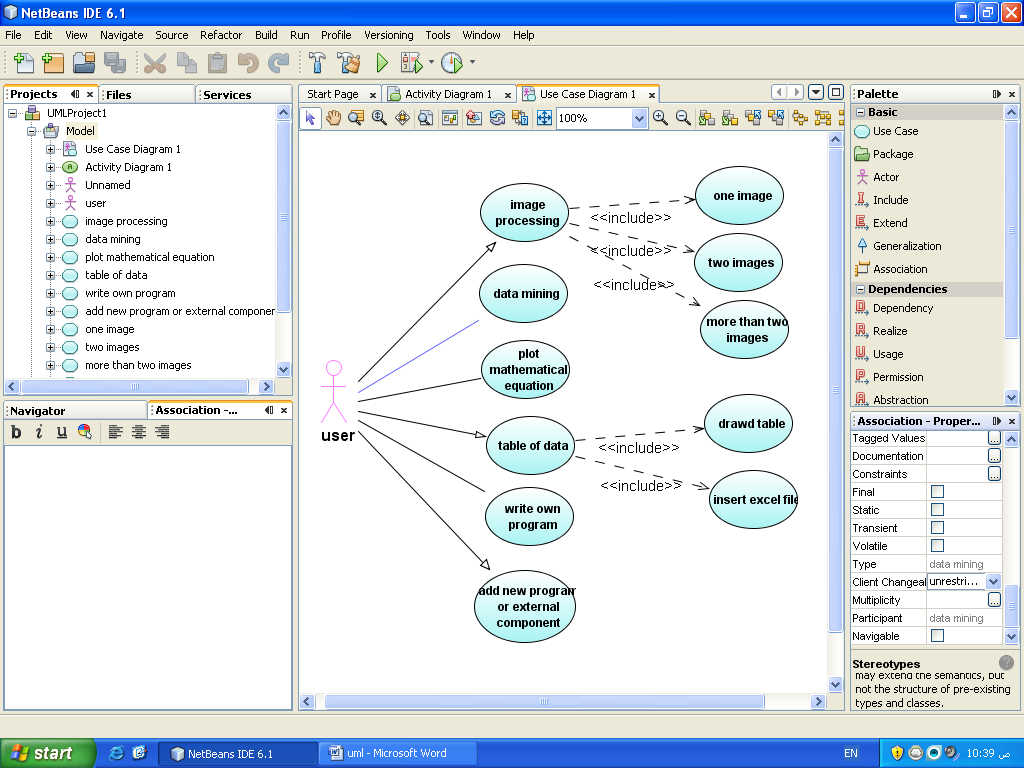
#### 

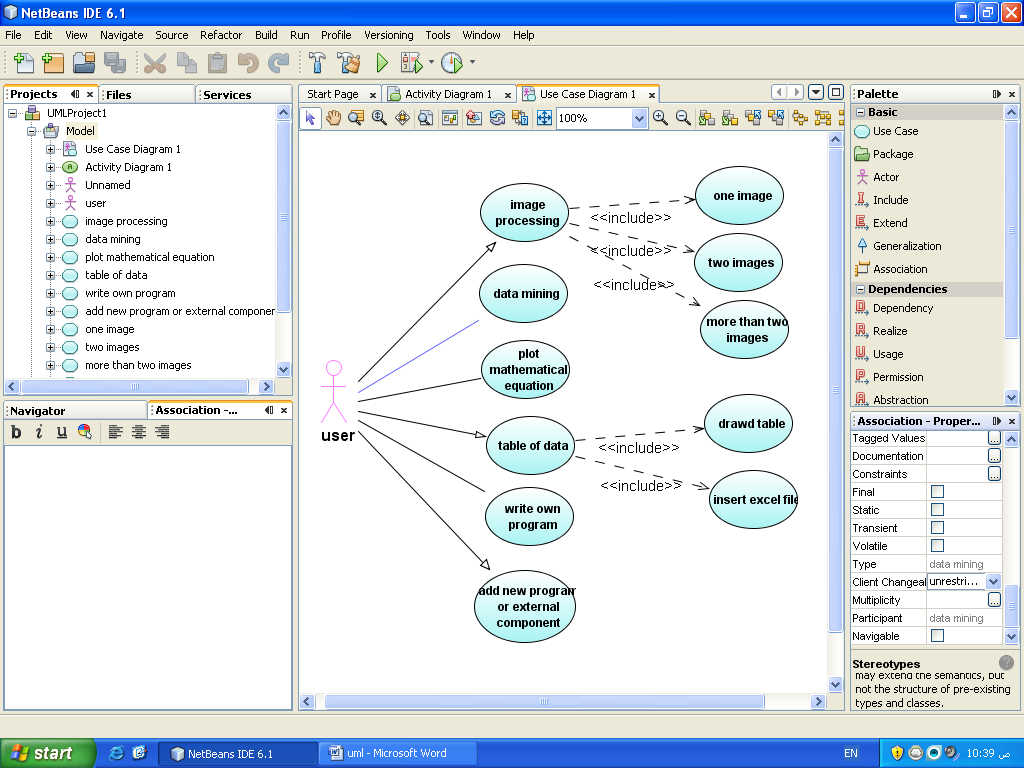
### 3.4.1.1) use case diagram:

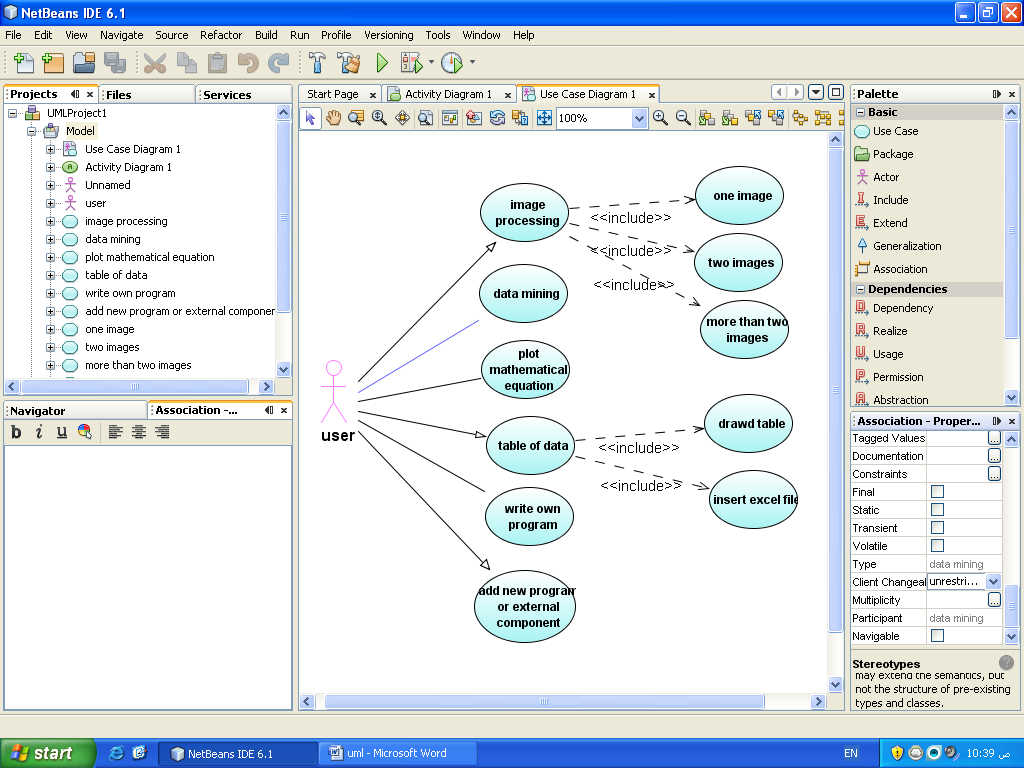
Administrator

Faculty

Student







## 3.5) System Architecture:

# 

Web Browser

Login

Role checking

Form & Menu Manager

Data Validation

Security Manager

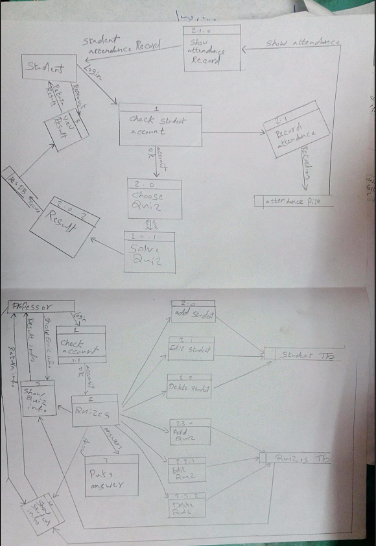
ES Appointment Manager

Data Import & Export

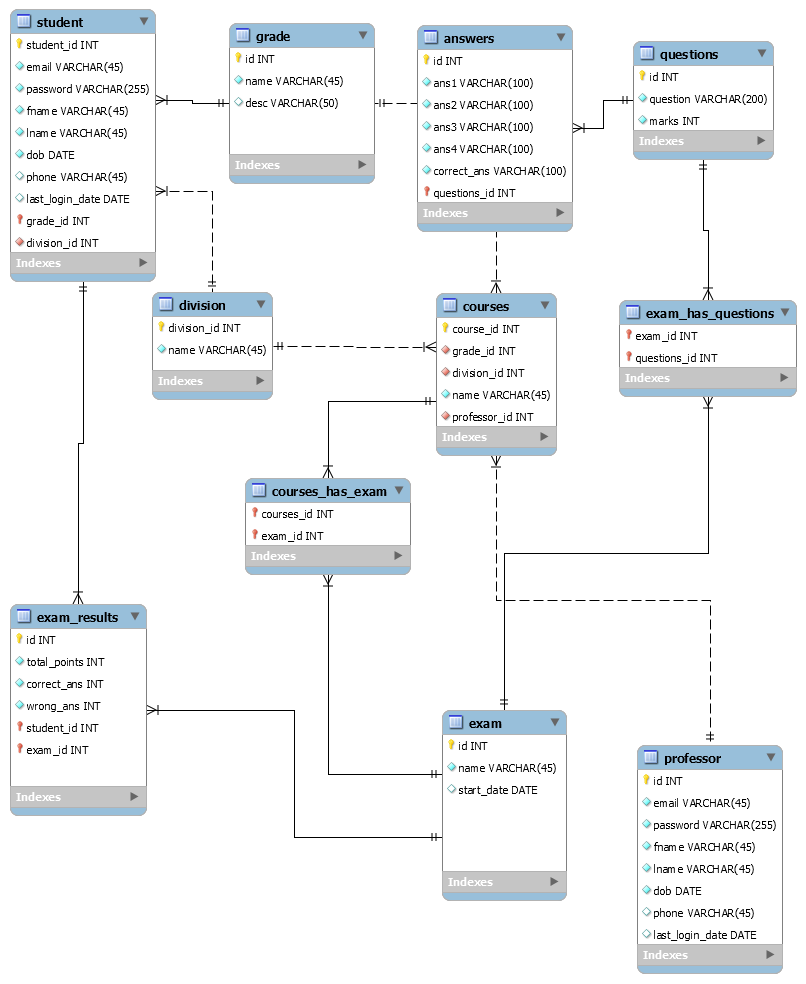
Report Generation

Transaction Management for ES Database

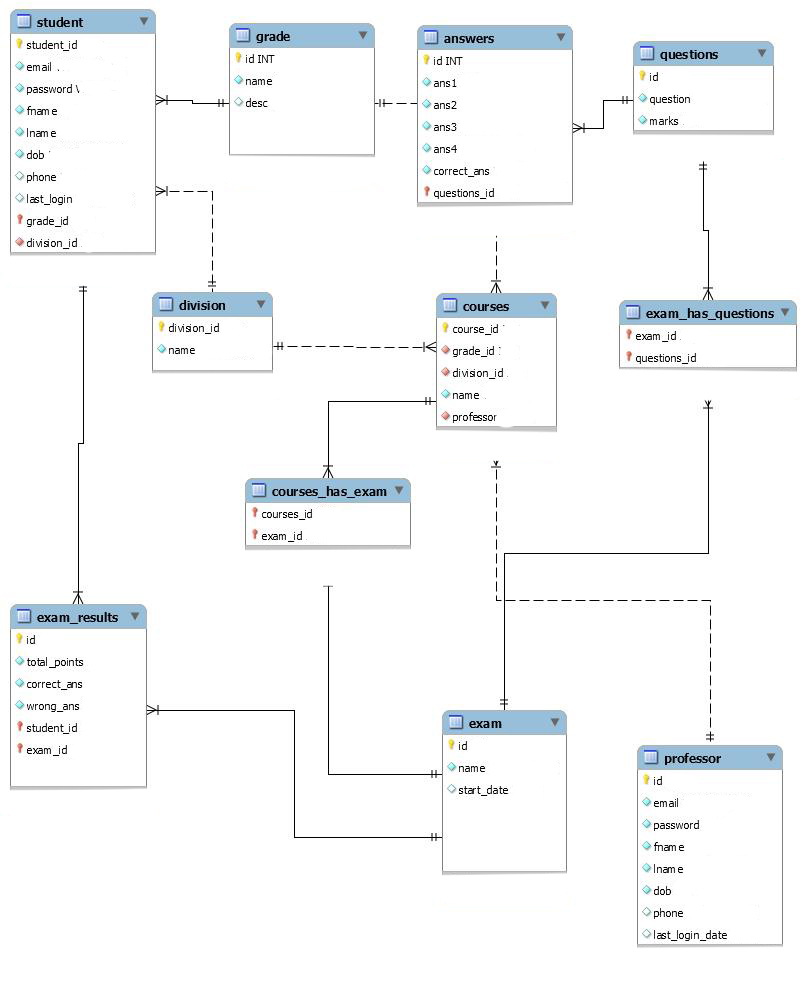
## 3.6)Data Flow Diagram(DFD):



## 3.7)DataBase Diagram:



## 3.8) ER Diagram (ERD):



# Chapter (4)

# Software Implementation

### Technology used:-

**Technology** :-

* **PHP (language)**
* **Mysql**
* **Android**

**Front end technology**:-

1. **HTML**
2. **CSS**
3. **JavaScribt(Jquery Lib)**

**4.Bootstrab**

**Hardware requirement :-** Basic computer system and mobile.

# Back end technology:-

1. **PHP**
2. **Mysql**

**Software Requirement:- (Any one(**

* WAMP Server
* XAMP Server
* Apache Server
* PhpmyAdmin

### Home Page for student:-

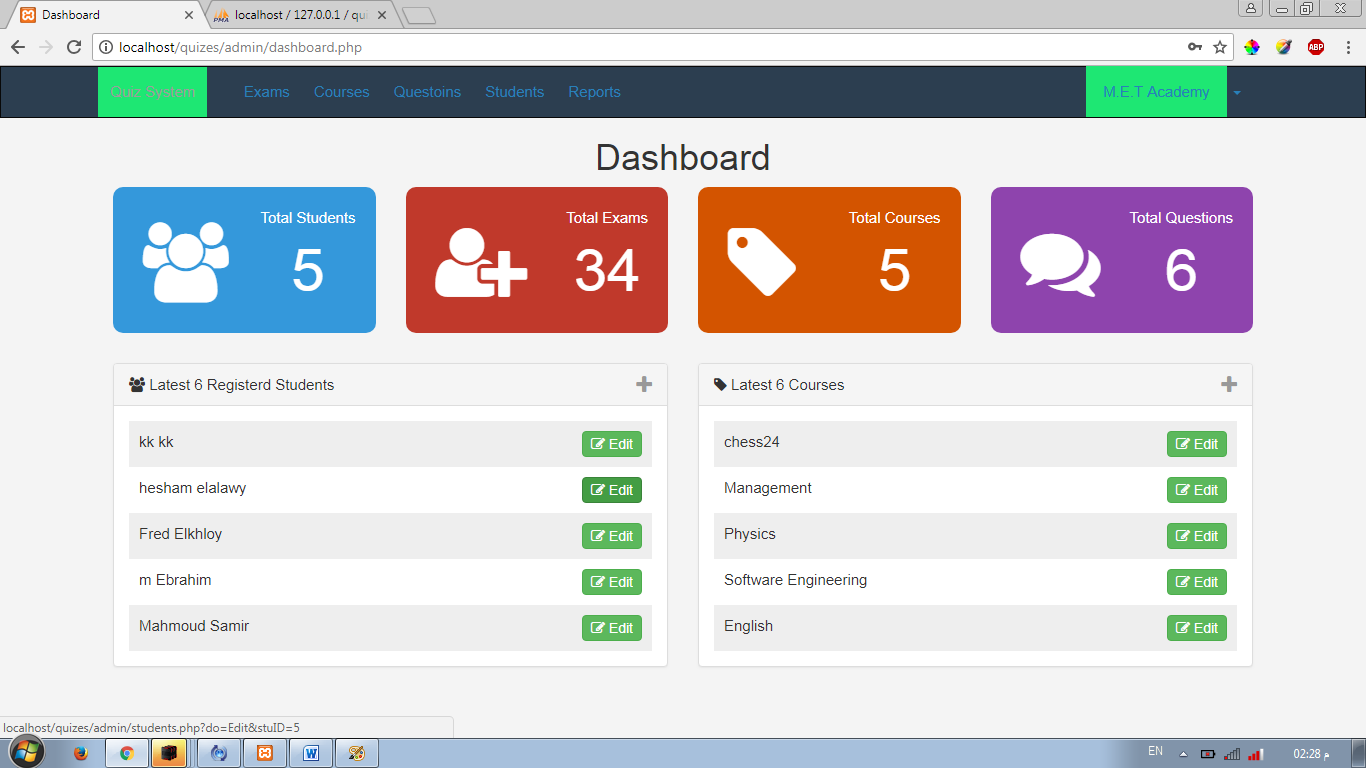
home option to go to questions , settings or previous results .



### Home Page for Admin:-

In home page available login section option in

Drop Down menu for log to admin



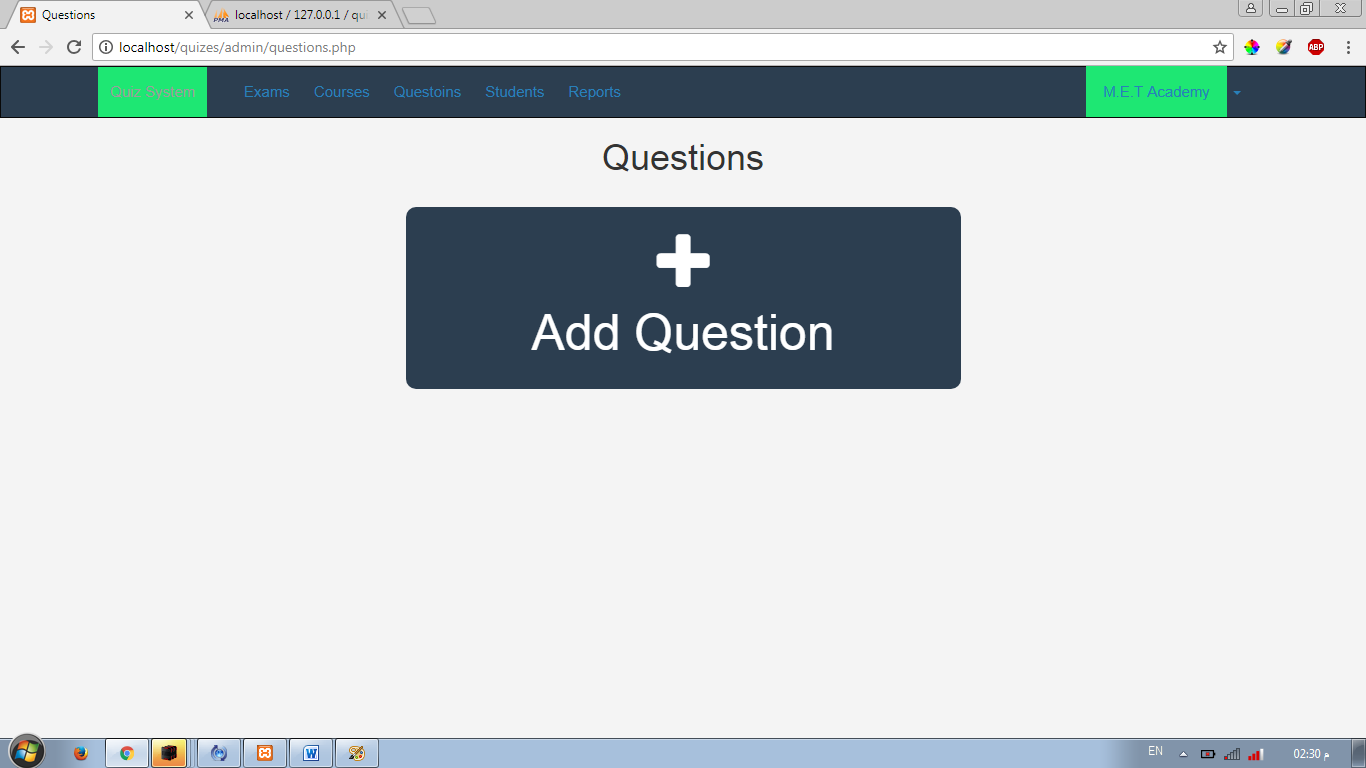
### Admin login:-

This is admin login page . If admin will enter right Card ID and password then here will open a new window (DashBoard)



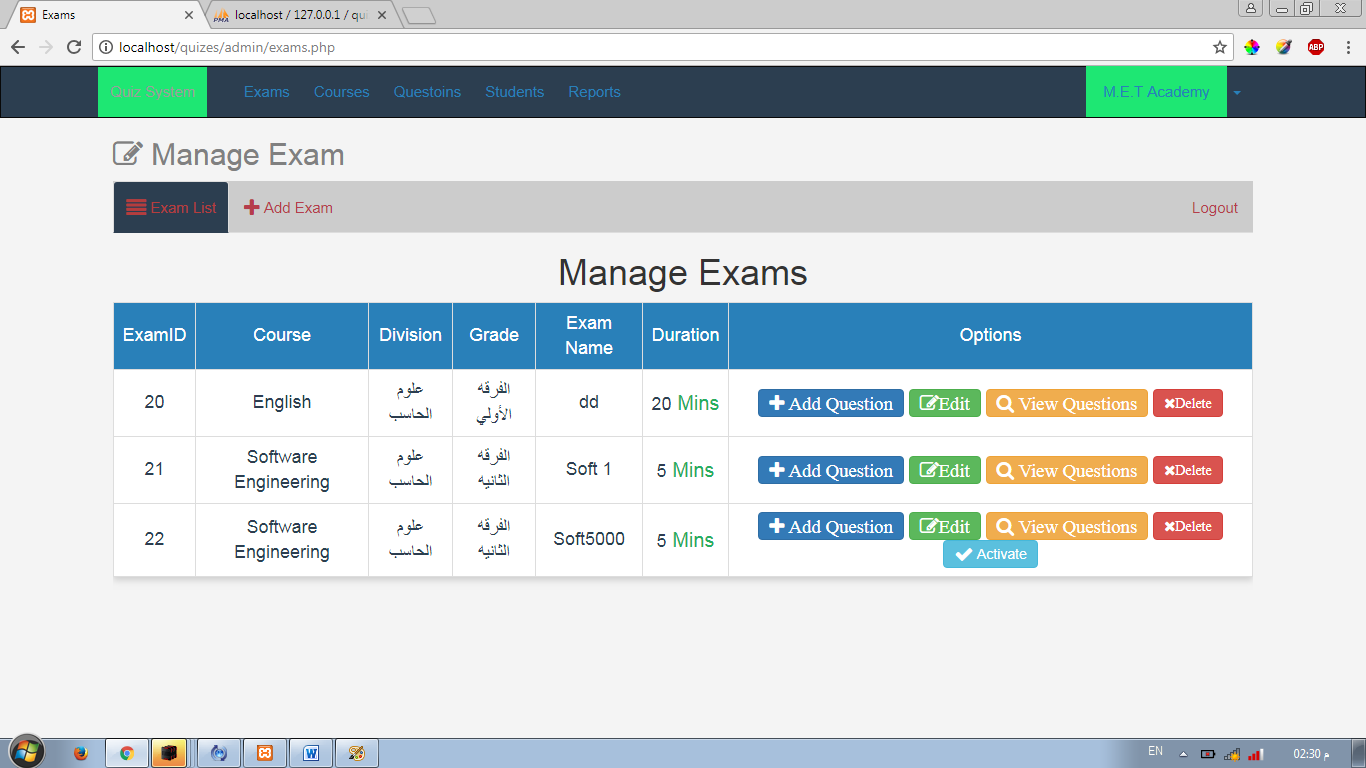
### Admin task:-

This is a admin task window here given all task of admin as, Add Question, Edit Question.



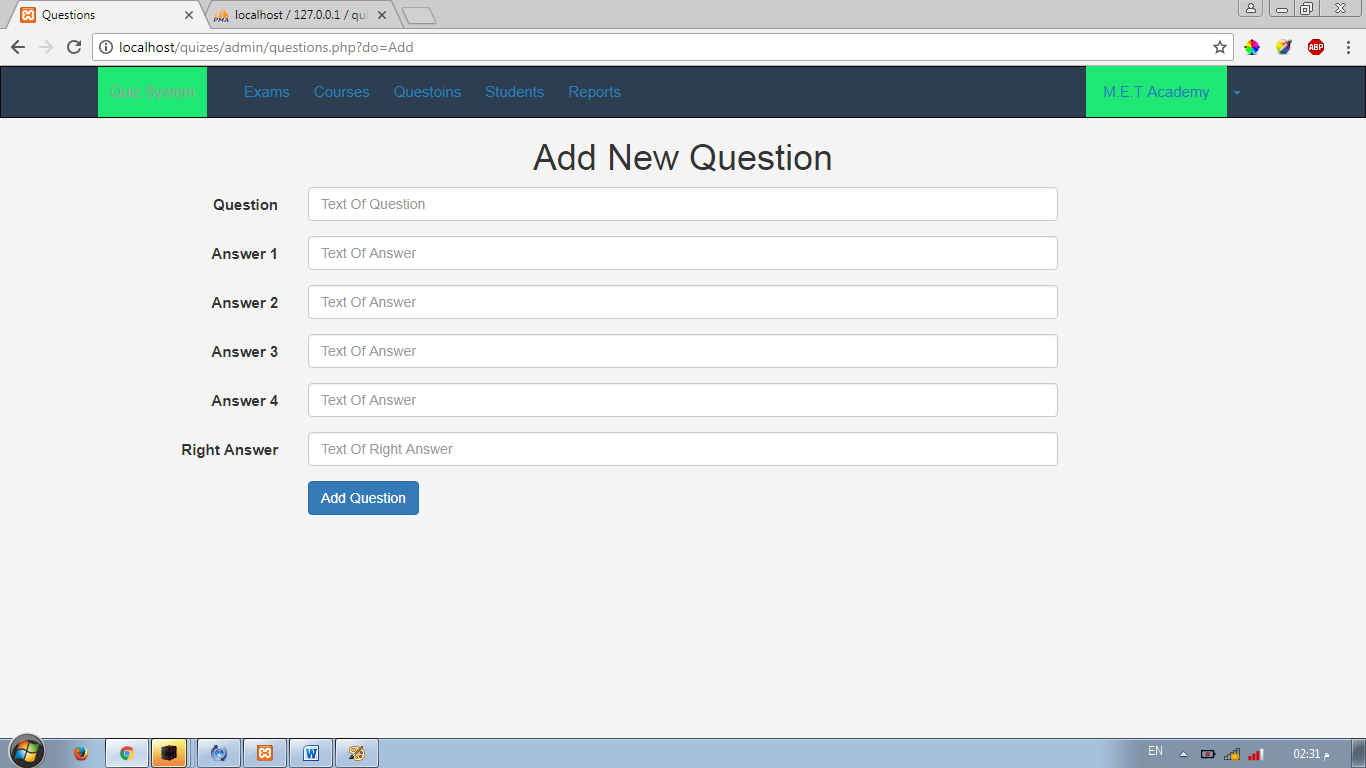
### Manage Exam:-

This is a manage exam window here given all task of admin as exam id, course , Division, Grade, Exam name, Options.



### Question Paper-:

This option uses for upload the question paper on webpage.



### Manage Courses-:

Manage course do edit and delete of courses

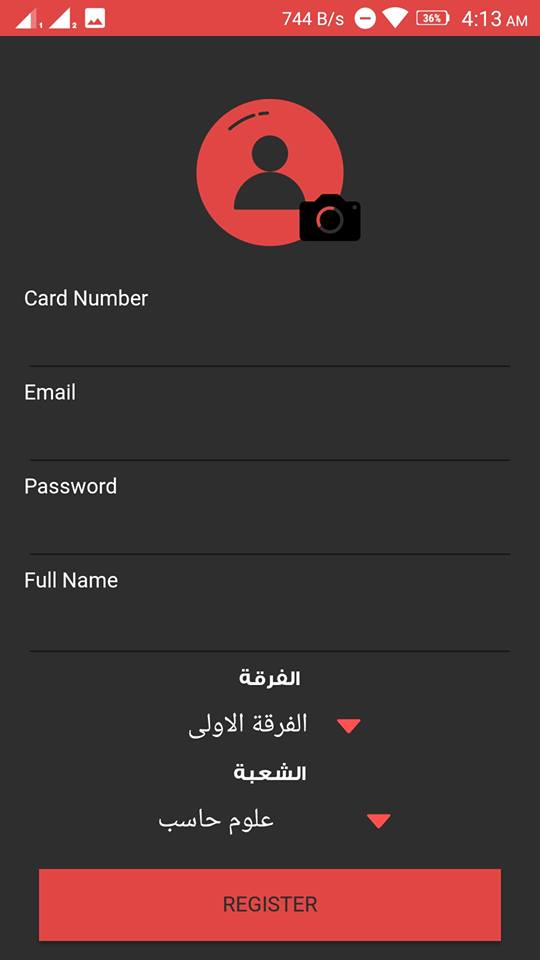


### Manage Student-:

### 

### Student Registration:-

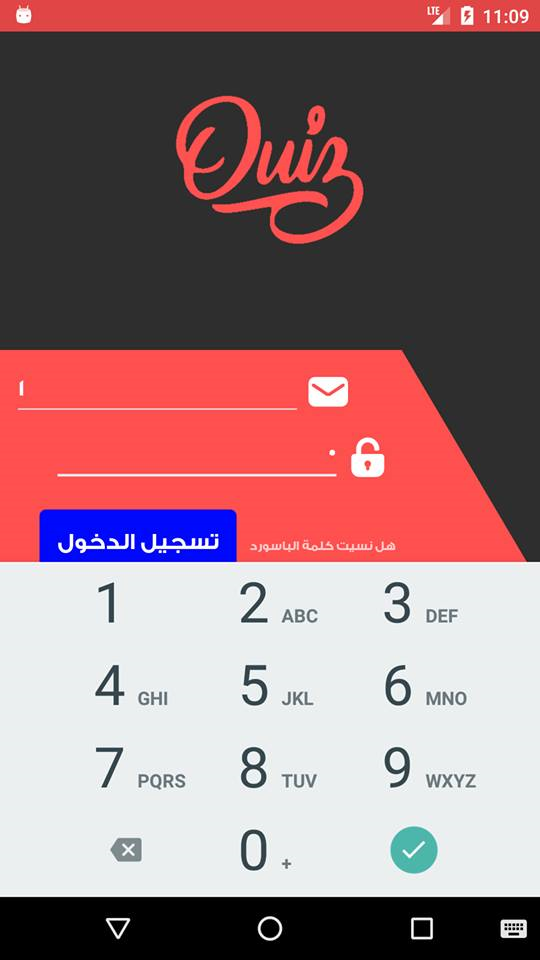
This registration form is available for all students for registering themselves via fill all requirement.



### Student Login:-

This is student login page

. If user enters right Card ID and password then here will open a new window for test.



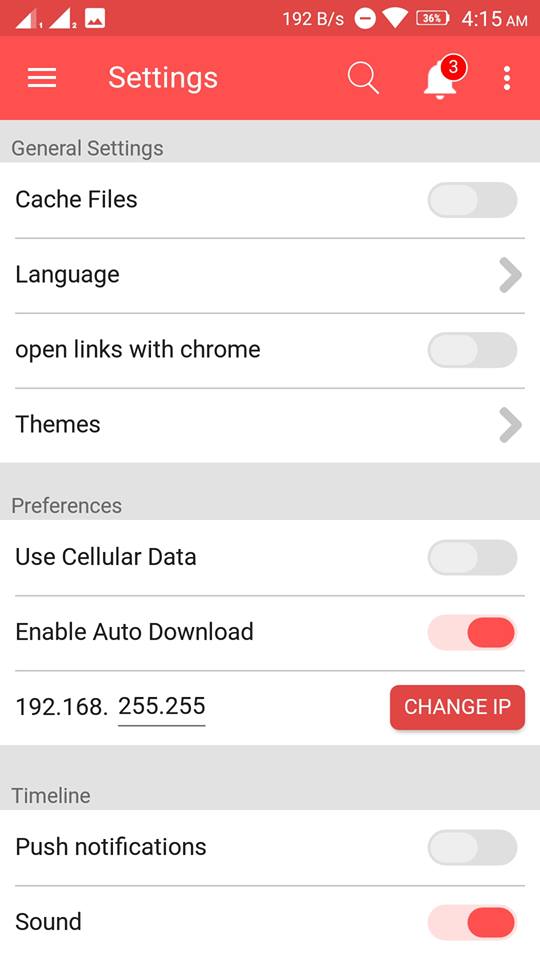
### Exam:-

* In this Activity page given question and choices which will be submitted by student and after test
* show the result.



### Setting:-

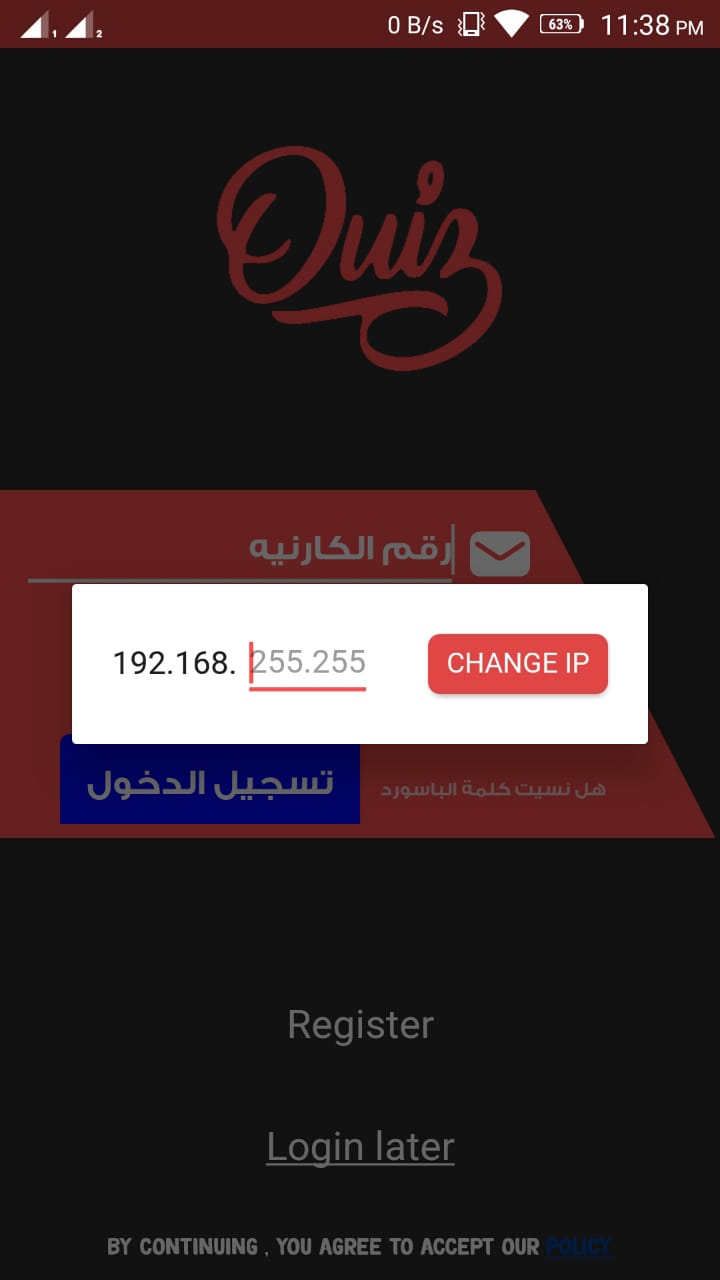
This is to control the application and change of the setting.



# Chapter (5)

**system test**

### Home Page for student:-

****

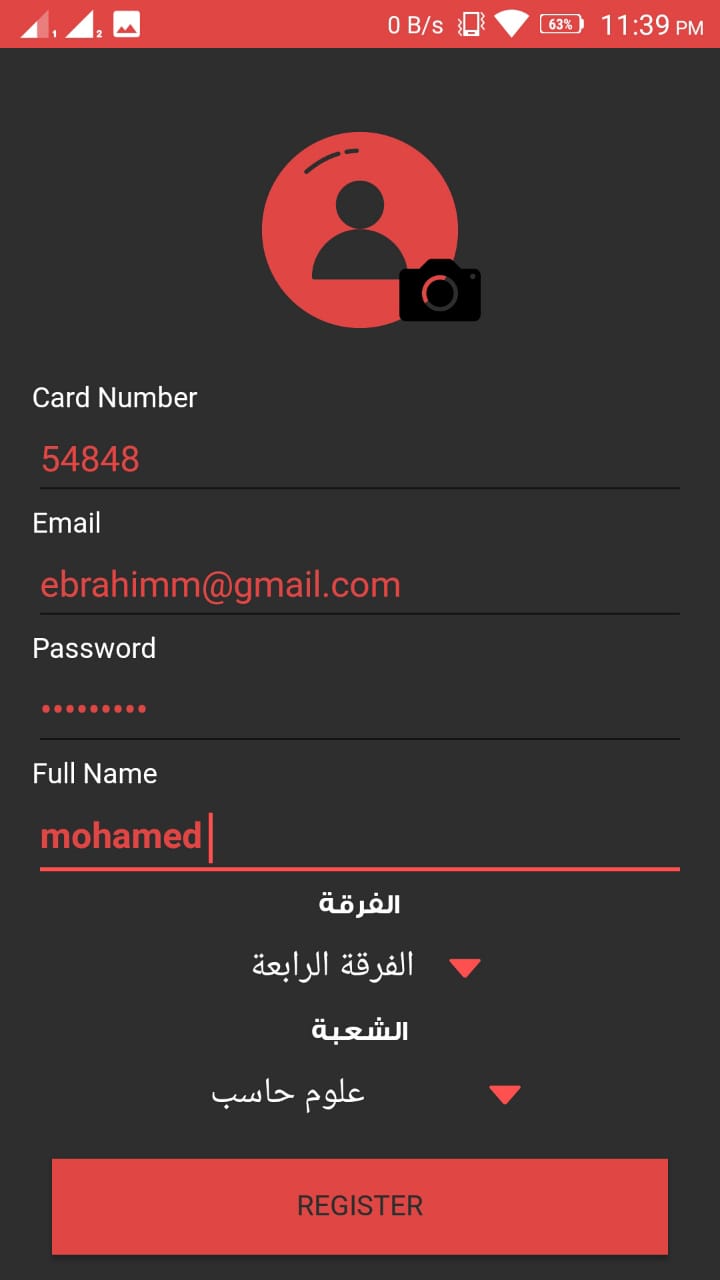
1. **In the beginning, you must enter the IP that is taken from the doctor.**
2. **Then press CHANGE IP.**

### Login interface:-

****

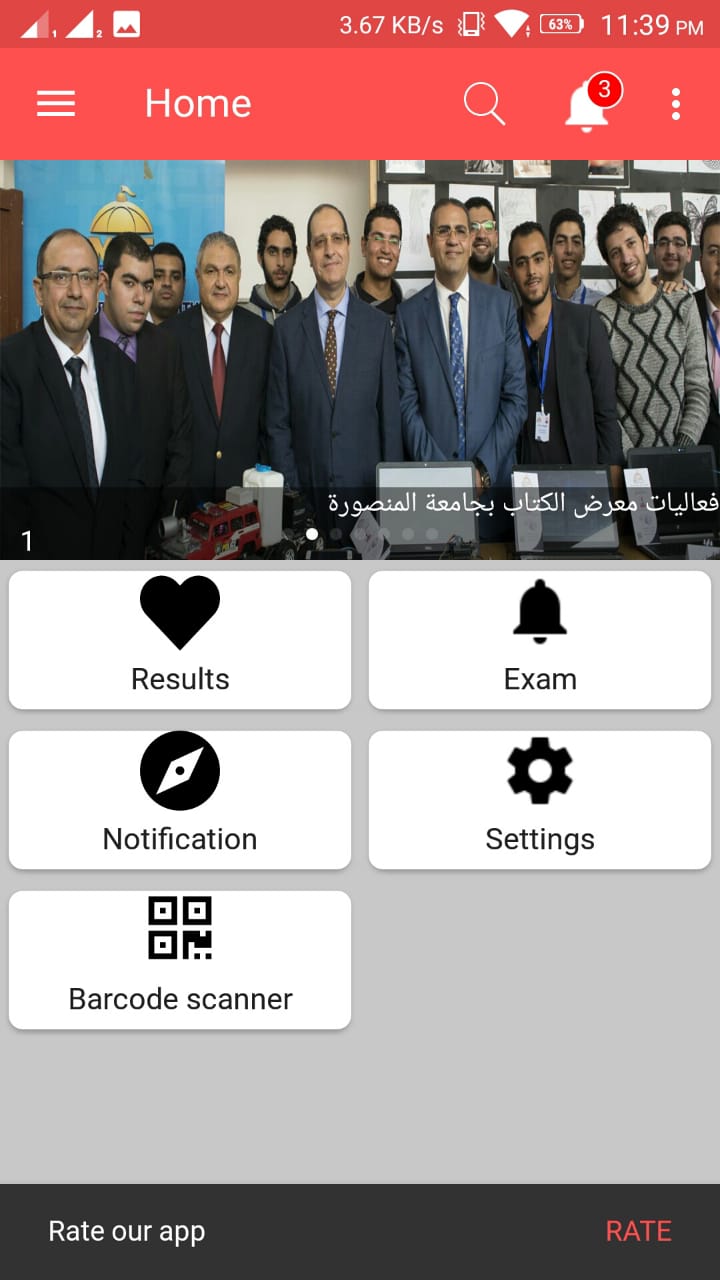
1. **Enter the Card Number.**
2. **Then enter your password.**
3. **Then press تسجيل الدخول**.

### New Student Login Interface:-



1. **Enter the card number.**
2. **Then enter the email.**
3. **Then enter the password.**
4. **Then skip the full name.**
5. **You specify the الفرقه. .**
6. **Also define the الشعبه .**
7. **Click on REGISTER.**

### home page for student:-



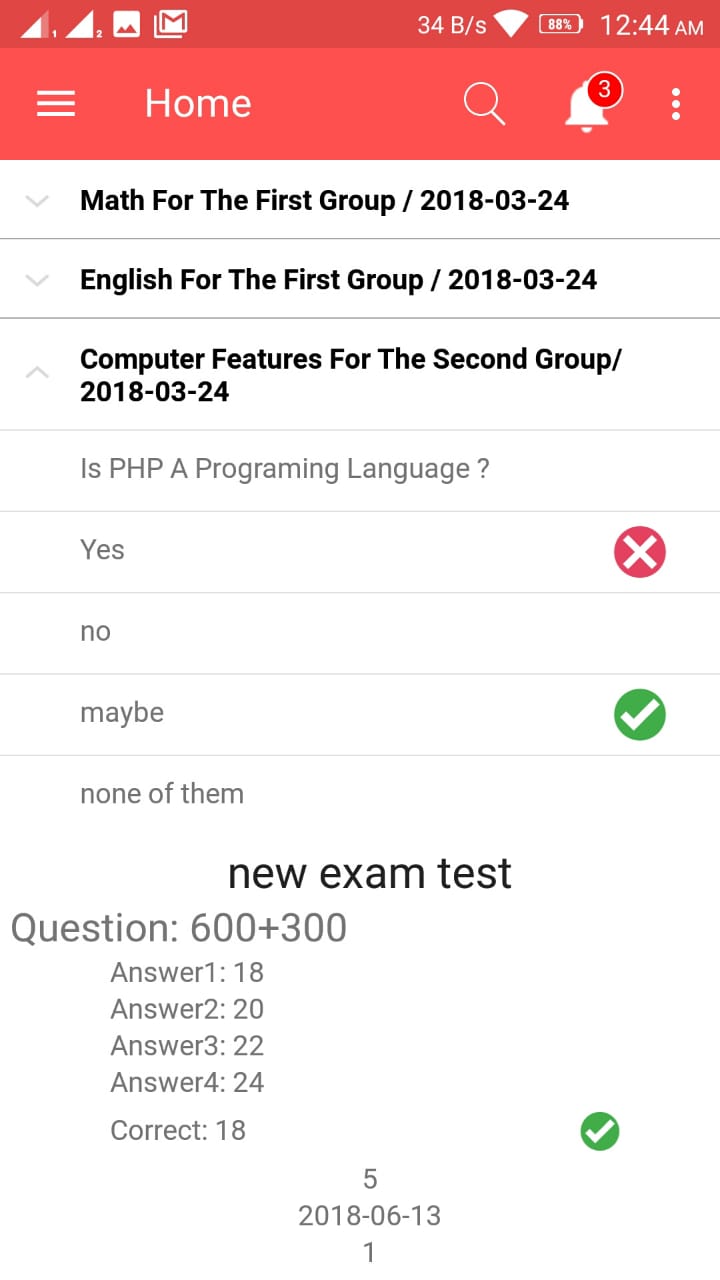
1. **You can see the results by clicking on RESULTES.**
2. **You can access questions by clicking on QUESTIONS.**
3. **You can learn the latest alerts and notifications by clicking on NOTIFICATION.**
4. **You can access the settings by clicking on SETTINGS.**

### Exam:-



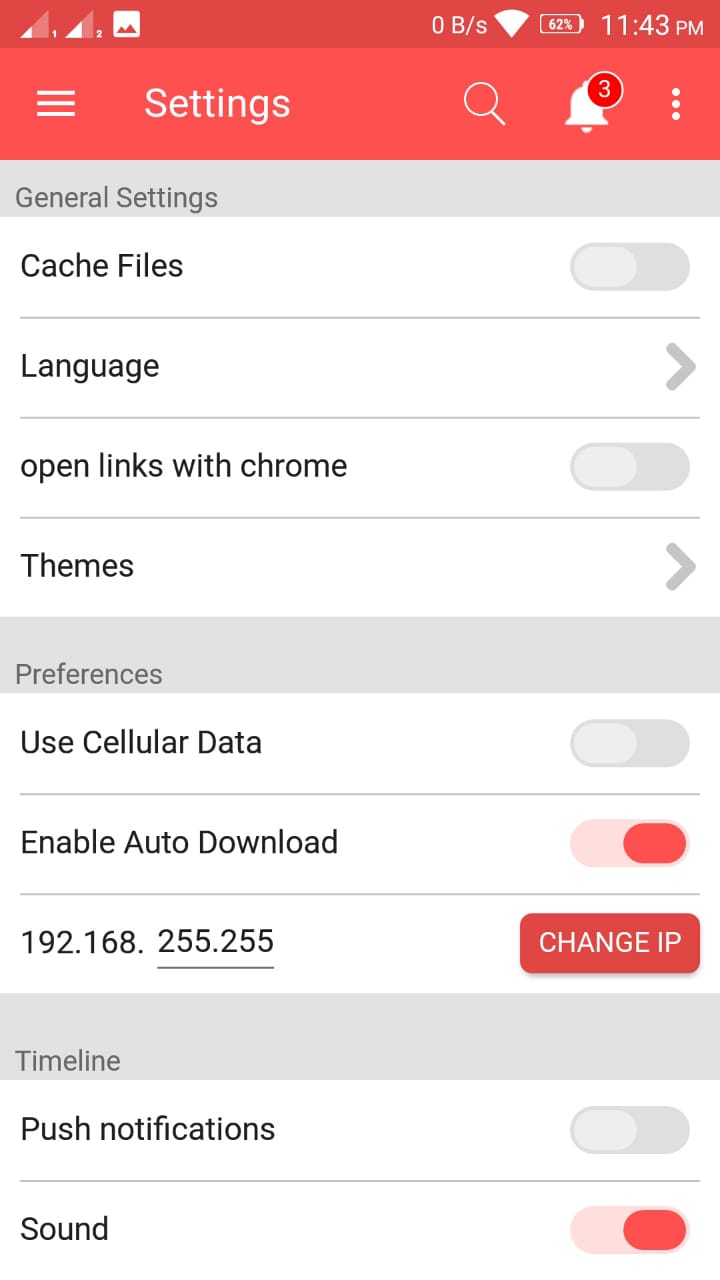
1. **The screen in the top of the name of the exam, which is determined by Dr.**
2. **Number of questions available and required.**
3. **Time to finish the exam.**
4. **The question and the answers to the question that is chosen between them.**

### Results Exam Questions:-



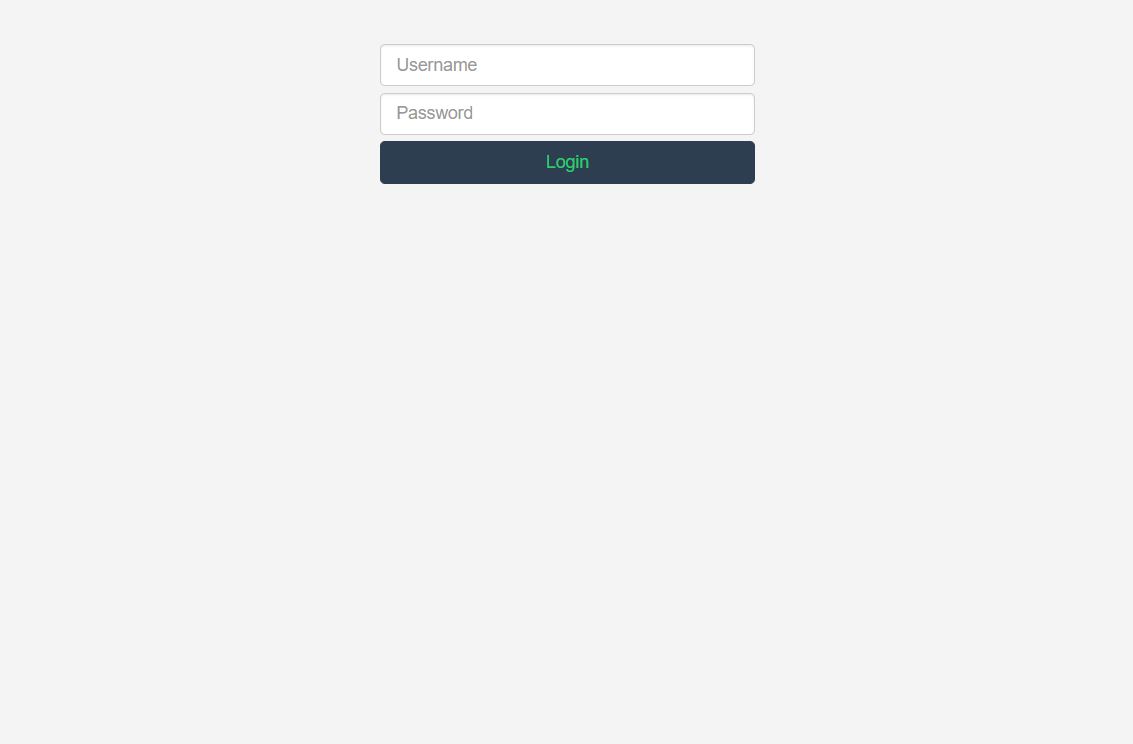
* **Exam Questions When the question is asked, the question is answered and the answer is determined correctly and the answer is wrong**

### Settings:-

****

1. **You will be able to modify your IP**
2. **To enter the exam and then click on CHANGE IP.**
3. **The language is changed by pressing LANGUAGE.**
4. **Shut down and turn on sound by pressing on SOUND.**

### Admin login:-

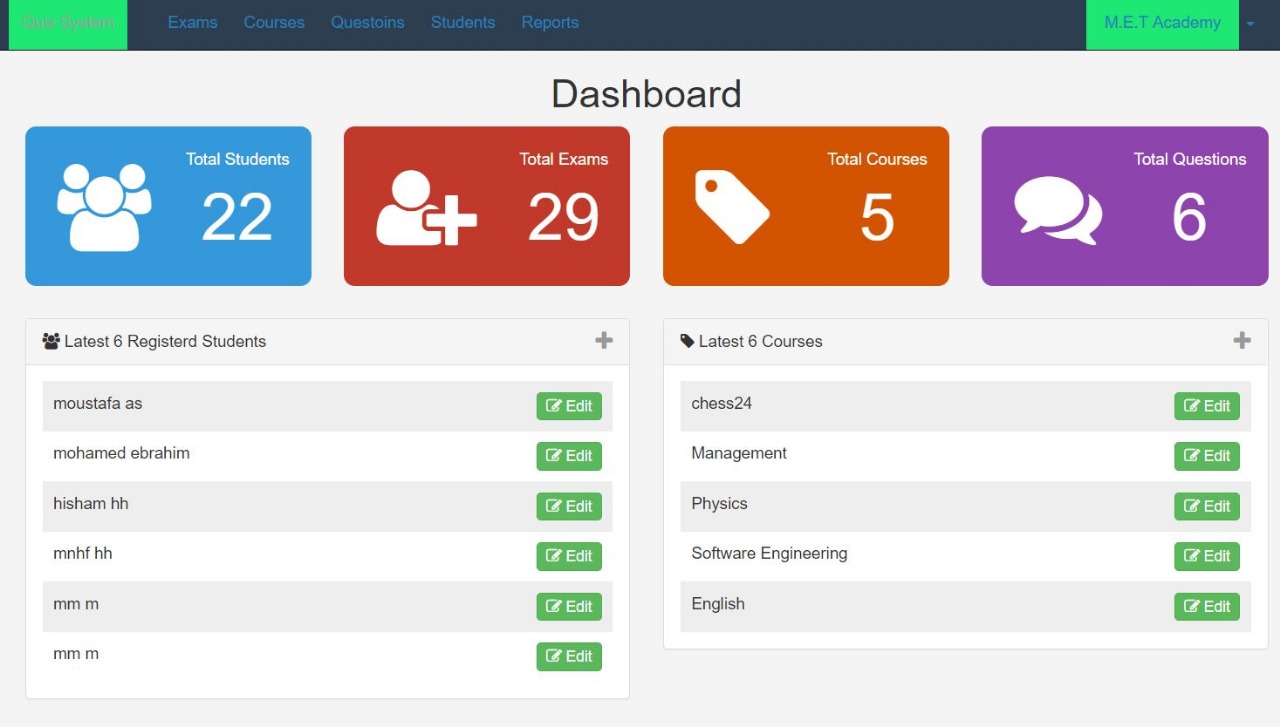


1. **The user name is entered**
2. **Then enter the password**
3. **Then press on login.**

### Home Page for Admin:-

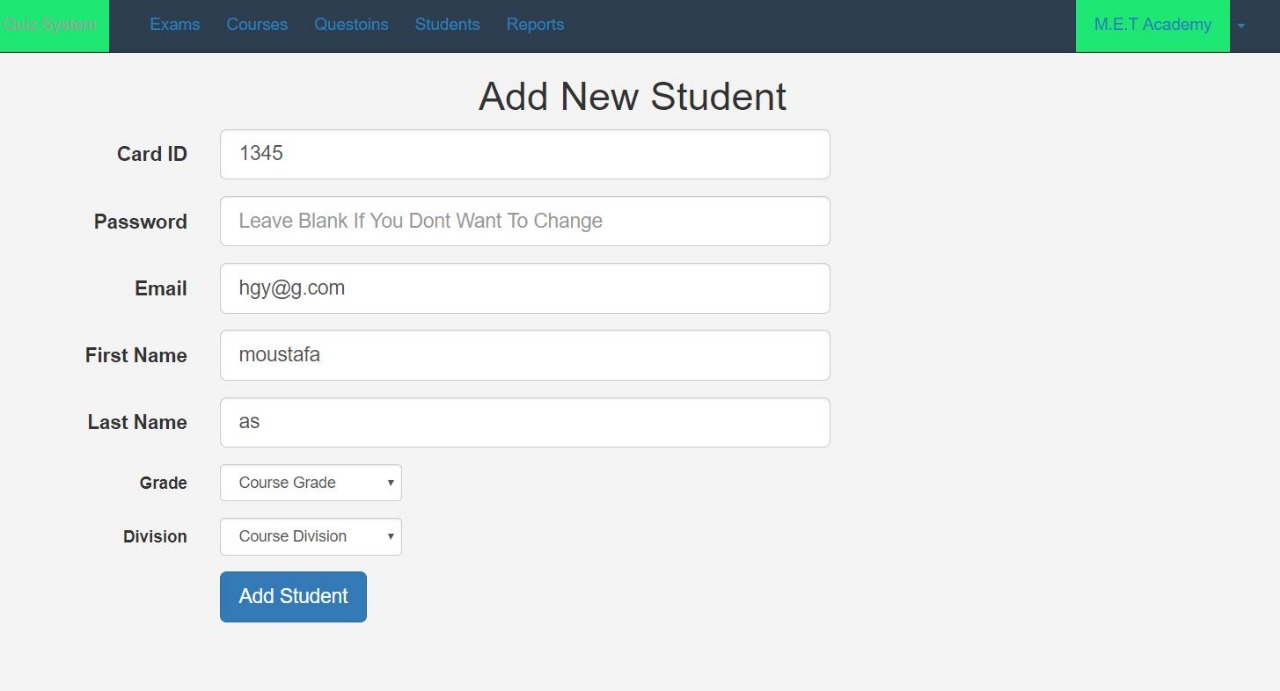
**It contains a task bar that contains a set of icons**

* **EXAMS.**
* **COURSES.**
* **QUESTIONS.**
* **STUDENTS.**
* **REPORTS.**

****

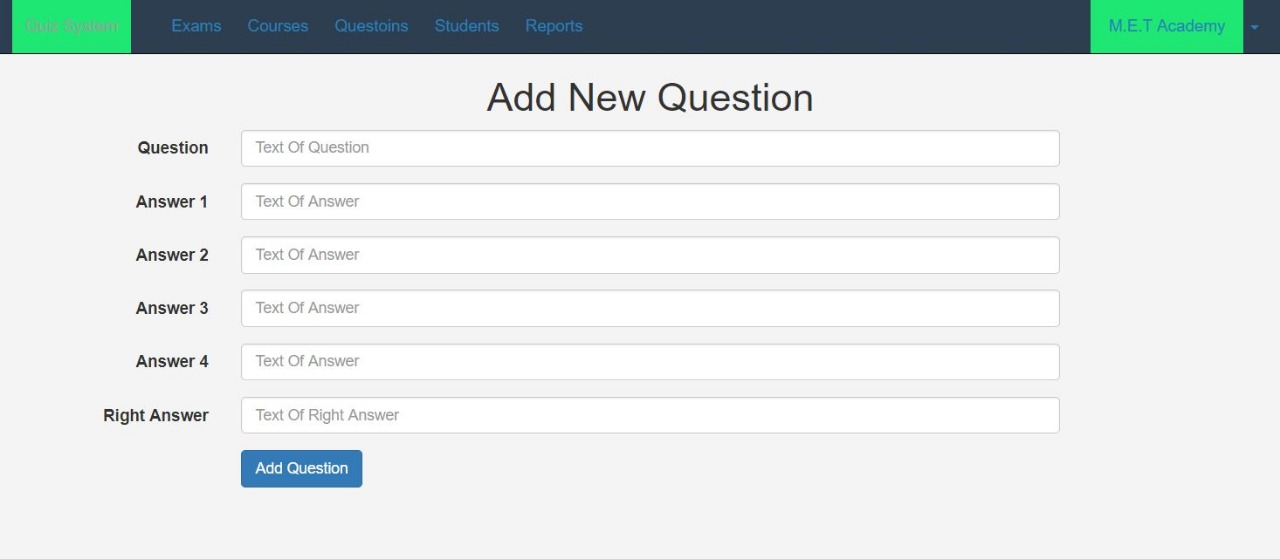
1. **You can find out how many students are enrolled by clicking on TOTAL STUDENTS.**
2. **Exams can be added by pressing TOTAL EXAMS.**
3. **Available courses can be found by clicking on TOTAL COURSES.**
4. **To view all stored questions click on TOTAL QUESTIONS.**
5. **To exit the page click on LOGIN the icon next to M.E.T ACDEEMY.**

### Add New Student:-

****

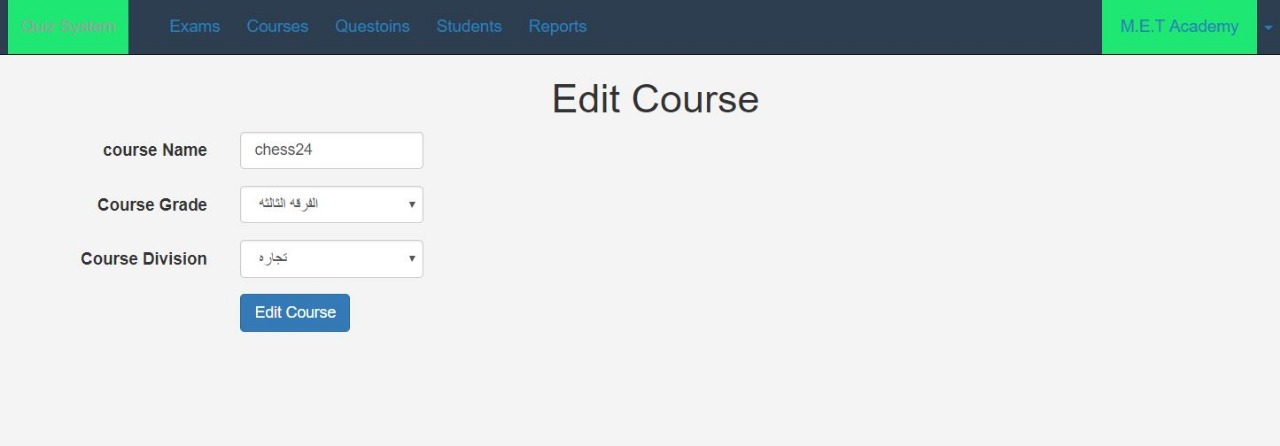
1. **Enter Card ID.**
2. **Then enter the Password.**
3. **Then enter the Email.**
4. **Then enter First Name.**
5. **Then enter Last Name.**
6. **Then select Division.**
7. **Click on Add Student.**

### Question Paper:-

****

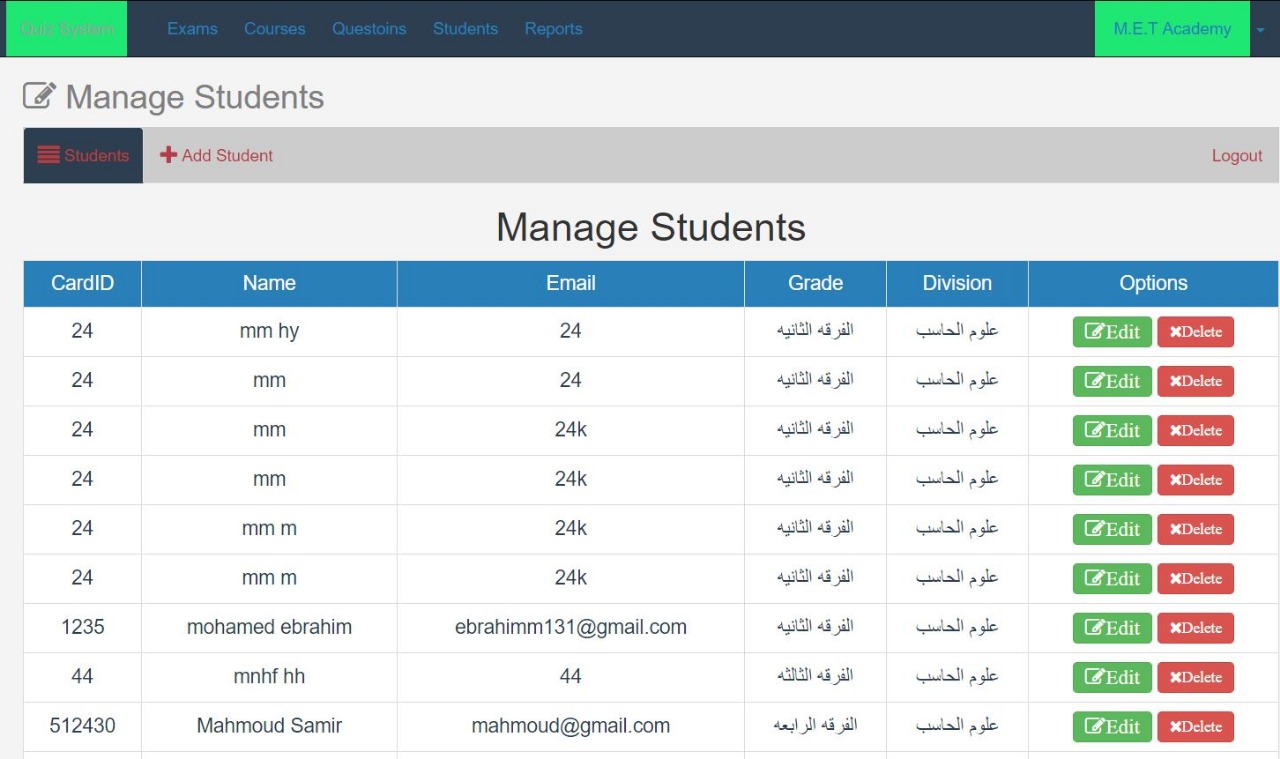
1. **Write the Question.**
2. **Then enter the answer 2.**
3. **Then enter the answer 3.**
4. **Then enter the answer 4.**
5. **Then enter the Right Answer.**
6. **Click on Add Question.**

### Edit Course:-



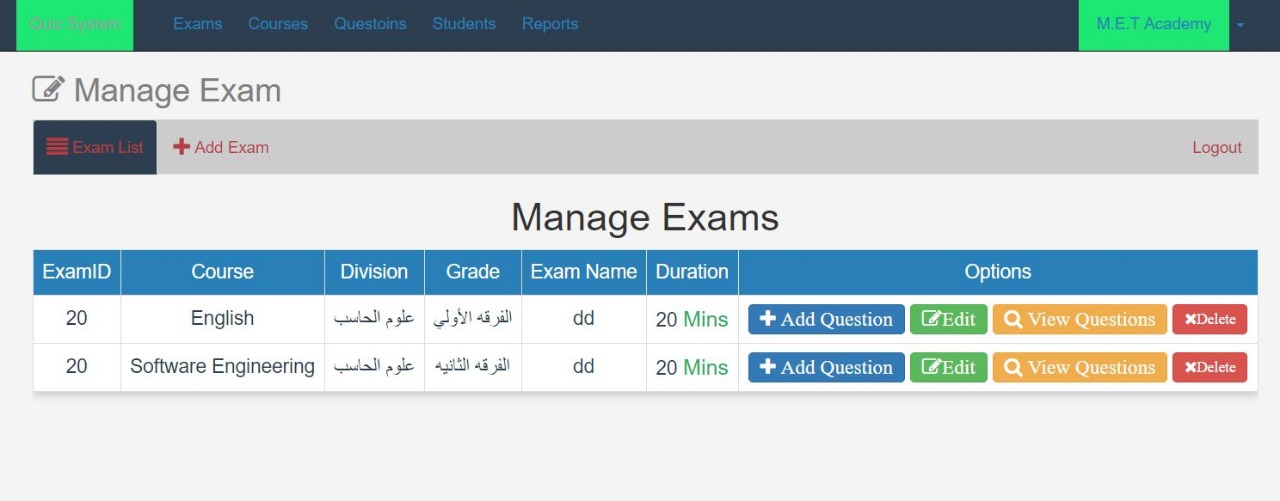
1. **Then enter Course Name.**
2. **Then select Course Grade.**
3. **Then select Course Division.**
4. **Click on Edit Course.**

### Manage Student:-

****

1. **Students are added by clicking on +Add Student.**
2. **You can modify the student data by clicking on Edit.**
3. **Student data can be removed by clicking on Delete.**

### Manage Exam:-

****

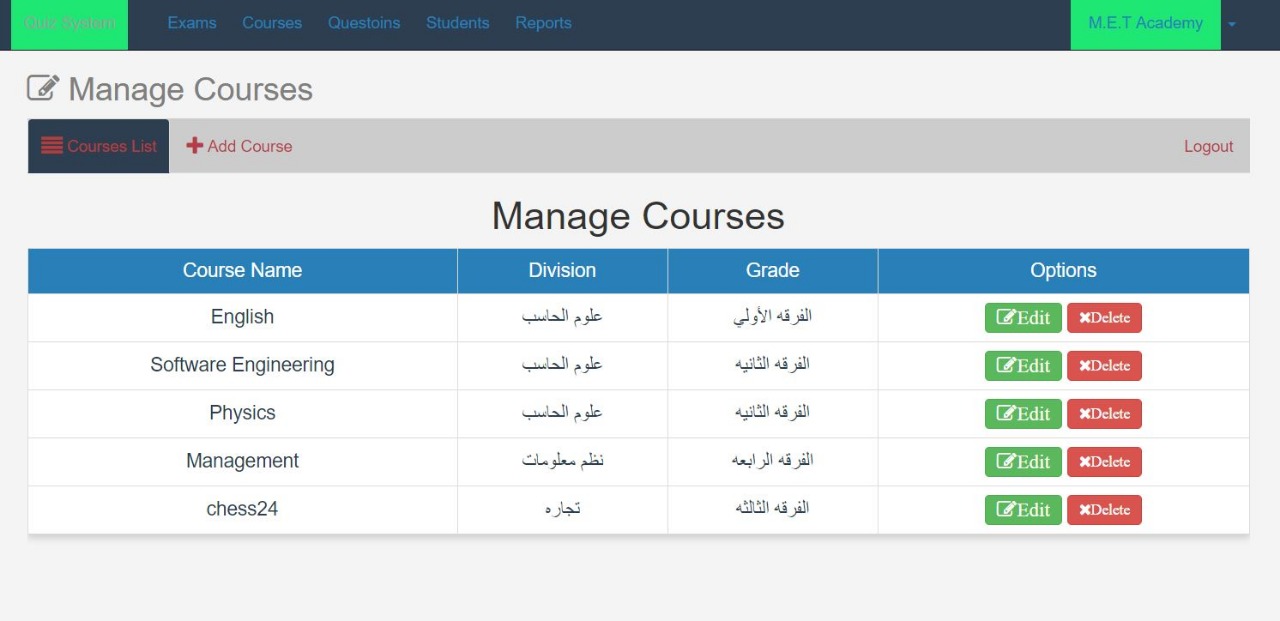
**You can add new exams by clicking on +Add Exam.**

**This screen shows the contents of the stored exams Exam ID,**

**COURSE, Division, Grade, Exam Name, Duration.**

1. **Can add question to exam store by clicking on +Add Question.**
2. **You can modify the exam by clicking on Edit.**
3. **The content of the exam can be checked by clicking on the View Questions.**
4. **Any exam can be deleted by clicking on Delete.**

### Manage Courses:-

****

1. **This screen shows the contents of the stored Courses Course Name, Division, Grade, Options.**
2. **You can add new Courses by clicking on +Add Courses.**
3. **You can modify the Courses by clicking on Edit.**
4. **Any Courses can be deleted by clicking on Delete.**

### The code of examination system for Web:-

Home page:-

<?php

session\_start();

$nonavbar = '' ;

$pageTitle = 'Home' ;

include 'init.php' ;

?>

<div class="bg"></div>

<div class="container">

<h4 class="title">Welcome To Our Exam System</h4>

</div>

<?php

$name = $\_SESSION['Username'] ;

// Getting User Profile

$stmt = $conn->prepare("SELECT \* FROM signup WHERE name = ?") ;

$stmt->execute(array($name)) ;

$row = $stmt->fetchAll() ;

// Getting Subjects

$stmt1 = $conn->prepare("SELECT \* FROM subject") ;

$stmt1->execute() ;

$subjects = $stmt1->fetchAll();

// Footer

include $tpl.'footer.php' ;

?>

<div class="container">

<h2></h2>

<ul class="nav nav-tabs">

<li class="active"><a data-toggle="tab" href="#home">Home</a></li>

<li><a data-toggle="tab" href="#menu1">Profile</a></li>

<li><a data-toggle="tab" href="#menu2">Menu 2</a></li>

<li style="float:right"><a href="logout.php">Logout</a></li>

</ul>

<div class="tab-content">

<div id="home" class="tab-pane fade in active">

<center><button type="button" class="starting btn btn-primary" data-toggle="tab" href="#select">Start Exam</button></center>

<div class="col-sm-4"></div>

<div class="col-sm-4">

<div id="select" class="tab-pane fade">

<form action="show\_ques.php" method="POST">

<select class="form-control sel-home" id="#select" name="subject">

<option>Select Subject ..</option>

<?php

if($stmt1->rowCount() > 0) {

foreach($subjects as $subject) { ?>

<option value="<?php echo $subject['id'] ; ?>" ><?php echo $subject['subj\_name'] ; ?></option>

<?php }

}

?>

</select>

<center><input type="submit" class="starting btn btn-primary"

value="Let's Go" id="go-btn"/></center>

</form>

</div>

</div>

</div>

<div id="menu1" class="tab-pane fade">

<h3>My Profile</h3>

<table class="table">

<thead>

<tr>

<th>ID</th>

<th>Name</th>

<th>Img</th>

</tr>

</thead>

<tbody>

<?php

foreach($row as $data) { ?>

<tr>

<td><?php echo $data['id'] ; ?></td>

<td><?php echo $data['name'] ; ?></td>

<td><img src="img/<?php echo $data['img'] ; ?>" width="35px" height="30px" alt="My Pic" /></td>

</tr>

<?php }

?>

</tbody>

</table>

</div>

<div id="menu2" class="tab-pane fade">

<h3>Menu 2</h3>

<p>Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam.</p>

</div>

<div id="menu3" class="tab-pane fade">

<h3>Menu 3</h3>

<p>Eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo.</p>

</div>

</div>

</div>

Index:-

<?php

session\_start();

$nonavbar = '' ;

$pageTitle = 'Login' ;

if(isset($\_SESSION['Username'])){

header('Location: home.php'); // Redirect to dashboard Page

exit();

}

include 'init.php';

// Loign Page

?>

<div class="bg"></div>

<div class="container">

<h4 class="title">Welcome To Our Exam System</h4>

<div class="row">

<div class="col-md-6 col-sm-12">

<div class="login-panel-box">

<div class="panel panel-danger">

<div class="panel-heading">Login</div>

<div class="panel-body">

<?php

if(isset($\_GET['run']) && $\_GET['run'] == 'faild') {

$faildMsg = "Username Or Password Is Wrong ." ;

echo msg("faild" , $faildMsg) ;

}

?>

<form role="form" class="login" action="signin\_sub.php" method="POST">

<input type="text" class="form-control" placeholder="Username" name="user" autocomplete="off" required="required"/>

<input type="password" class="form-control" placeholder="Password" name="pass" autocomplete="new-password" required="required" />

<input class="btn btn-primary btn-block" type="submit" value="login" />

</form>

</div>

</div>

</div>

</div>

<div class="col-md-6 col-sm-12">

<div class="login-panel-box">

<div class="panel panel-danger">

<div class="panel-heading">Register</div>

<div class="panel-body">

<?php

if(isset($\_GET['run'] )&& $\_GET['run'] == 'success'){

$msgSuccess = "SuccessFully Registered .. Will Activated Soon ." ;

echo msg("success" , $msgSuccess) ; }

?>

<form role="form" class="login" action="signup\_sub.php" method="POST" enctype="multipart/form-data">

<input type="text" class="form-control" placeholder="Your Username" name="name" autocomplete="off" required="required"/>

<input type="password" class="form-control" placeholder="Password" name="pass" autocomplete="new-password"required="required"/>

<input type="password" class="form-control" placeholder="Confirm Password" name="confirm" autocomplete="new-password" required="required"/>

<div class="form-group">

<label>Upload Your Image</label>

<input type="file" class="form-control" name="img" required="required" />

</div>

<input class="btn btn-primary btn-block" type="submit" value="SignUp" />

</form>

</div>

</div>

</div>

</div>

</div>

</div>

<?php

include $tpl.'footer.php';

?>

Connect:-

<?php

session\_start();

$nonavbar = '' ;

$pageTitle = 'Login' ;

if(isset($\_SESSION['Username'])){

header('Location: home.php'); // Redirect to dashboard Page

exit();

}

include 'init.php';

// Loign Page

?>

<div class="bg"></div>

<div class="container">

<h4 class="title">Welcome To Our Exam System</h4>

<div class="row">

<div class="col-md-6 col-sm-12">

<div class="login-panel-box">

<div class="panel panel-danger">

<div class="panel-heading">Login</div>

<div class="panel-body">

<?php

if(isset($\_GET['run']) && $\_GET['run'] == 'faild') {

$faildMsg = "Username Or Password Is Wrong ." ;

echo msg("faild" , $faildMsg) ;

}

?>

<form role="form" class="login" action="signin\_sub.php" method="POST">

<input type="text" class="form-control" placeholder="Username" name="user" autocomplete="off" required="required"/>

<input type="password" class="form-control" placeholder="Password" name="pass" autocomplete="new-password" required="required" />

<input class="btn btn-primary btn-block" type="submit" value="login" />

</form>

</div>

</div>

</div>

</div>

<div class="col-md-6 col-sm-12">

<div class="login-panel-box">

<div class="panel panel-danger">

<div class="panel-heading">Register</div>

<div class="panel-body">

<?php

if(isset($\_GET['run'] )&& $\_GET['run'] == 'success'){

$msgSuccess = "SuccessFully Registered .. Will Activated Soon ." ;

echo msg("success" , $msgSuccess) ; }

?>

<form role="form" class="login" action="signup\_sub.php" method="POST" enctype="multipart/form-data">

<input type="text" class="form-control" placeholder="Your Username" name="name" autocomplete="off" required="required"/>

<input type="password" class="form-control" placeholder="Password" name="pass" autocomplete="new-password"required="required"/>

<input type="password" class="form-control" placeholder="Confirm Password" name="confirm" autocomplete="new-password" required="required"/>

<div class="form-group">

<label>Upload Your Image</label>

<input type="file" class="form-control" name="img" required="required" />

</div>

<input class="btn btn-primary btn-block" type="submit" value="SignUp" />

</form>

</div>

</div>

</div>

</div>

</div>

</div>

<?php

include $tpl.'footer.php';

?>

Answer:-

<?php

session\_start() ;

include 'init.php' ;

$answers = $\_POST ;

echo "<pre>" ;

print\_r($answers) ;

echo "</pre>" ;

// Getting Answers

// $stmt = $conn->prepare("SELECT \* FROM questions WHERE subj\_id = ?");

// $stmt->execute(array($subject)) ;

// $questions = $stmt->fetchAll() ;

?>

<?php

**Init:-**

include 'connect.php';

//Routes

$tpl = 'includes/templates/'; // Template Directory

$lang = 'includes/lang/'; // Language Directory

$func = 'includes/func/'; //Functions Directory

$css = 'layout/css/'; // Css Directory

$js = 'layout/js/'; // JS Directory

//Include Important Things

include $func. 'functions.php';

include $lang. 'en.php';

include $tpl. 'header.php';

//Include Navbar In All Pages Except The One With noNavBar Variable

if(!isset($nonavbar)){

include $tpl.'navbar.php';

}

**Logout:-**

<?php

// Start The Session

session\_start();

// UnSet The Session

session\_unset();

// Destroy The Session

session\_destroy();

header('Location: index.php');

exit();

Show\_quiz:-

<?php

session\_start() ;

$nonavbar = '' ;

include 'init.php' ;

$subject = $\_POST['subject'] ;

// Getting The Question According To Subject ID

$stmt = $conn->prepare("SELECT \* FROM questions WHERE subj\_id = ?");

$stmt->execute(array($subject)) ;

$questions = $stmt->fetchAll() ;

// echo "<pre>" ;

// print\_r($questions) ;

// echo "</pre>" ;

?>

<div class="bg"></div>

<div class="container">

<h4 class="title">Welcome To Our Exam System</h4>

</div>

<div class="container">

<div class="col-sm-2"></div>

<div class="col-sm-8">

</br>

<form action="answers.php" method="POST">

<?php

$counter = 1 ;

foreach($questions as $question) { ?>

<table class="table table-bordered">

<thead>

<tr class="ques">

<th><?php echo $counter . ' - ' . $question['question'] ; ?></th>

</tr>

</thead>

<tbody>

<?php if(isset($question['ans1'])) { ?>

<tr class="ques-ans">

<td>&nbsp;<span class="qno">1</span>&emsp;<input type="radio"

value="0" name="<?php echo $question['id'] ;?>" />

&nbsp;<?php echo $question['ans1'] ; ?>

</td>

</tr>

<?php } ?>

<?php if(isset($question['ans2'])) { ?>

<tr class="ques-ans">

<td>&nbsp;<span class="qno">2</span>&emsp;<input type="radio"

value="1" name="<?php echo $question['id'] ;?>" />

&nbsp;<?php echo $question['ans2'] ; ?>

</td>

</tr>

<?php } ?>

<?php if(isset($question['ans3'])) { ?>

<tr class="ques-ans">

<td>&nbsp;<span class="qno">3</span>&emsp;<input type="radio"

value="2" name="<?php echo $question['id'] ;?>" />

&nbsp;<?php echo $question['ans3'] ; ?>

</td>

</tr>

<?php } ?>

<?php if(isset($question['ans4'])) { ?>

<tr class="ques-ans">

<td>&nbsp;<span class="qno">4</span>&emsp;<input type="radio"

value="3" name="<?php echo $question['id'] ;?>" />

&nbsp;<?php echo $question['ans4'] ; ?>

</td>

</tr>

<?php } ?>

<tr class="ques-ans">

<td><input type="radio" checked="checked" style="display:none"

value="no\_attempts" name="<?php echo $question['id'] ;?>" />

</td>

</tr>

</tbody>

</table>

<?php $counter = $counter + 1 ; }

?>

<center><input type="submit" value="End Exam" class="starting btn btn-primary"/></center>

</form>

</div>

</div>

?>

Signin\_sub:-

<?php

session\_start();

$pageTitle = 'SignUp' ;

$nonavbar = '' ;

include 'init.php' ;

if($\_SERVER['REQUEST\_METHOD'] == 'POST') {

extract($\_POST) ;

$formErrors = array() ;

if(empty($user)) {

$formErrors[] = 'User Name Can\'t Be <strong>Empty</Strong>' ;

}

if(strlen($user) < 4) {

$formErrors[] = 'User Name Can\'t Be Less Than <strong>4 Chars</Strong>' ;

}

if(empty($pass)) {

$formErrors[] = 'Empty <strong>Pass</strong>' ;

}

if(strlen($pass) < 6) {

$formErrors[] = 'Pass Less Than <strong>6</strong> Digits' ;

}

if(!empty($formErrors)) { ?>

<div class="container">

<div class="row">

<div class="login-panel-box center">

<div class="panel panel-danger">

<div class="panel-heading">Register Form Errors</div>

<div class="panel-body">

<?php

foreach($formErrors as $error) {

echo '<div class="alert alert-danger">'.$error.'</div>';

}

redirect("Redirecting ..");

?>

</div>

</div>

</div>

</div>

</div>

<?php }

if(empty($formErrors)) {

// Checking The User

$stmt = $conn->prepare("SELECT

id , name , pass

FROM

signup

WHERE

name= ?

AND

pass= ?

LIMIT 1") ;

$hashedPass = sha1($pass) ;

$stmt->execute(array($user , $hashedPass)) ;

$row = $stmt->fetch();

$count = $stmt->rowCount();

// if User Exists

if($count > 0) {

$\_SESSION['Username'] = $user; // Register Session

$\_SESSION['ID'] = $row['UserID'] ;

header('Location: home.php'); // Redirect to dashboard Page

exit();

}else {

header('Location: index.php?run=faild') ;

}

}

}

?>

Signup\_sub:-

<?php

$pageTitle = 'SignUp' ;

$noNavBar = '' ;

include 'init.php' ;

if($\_SERVER['REQUEST\_METHOD'] == 'POST') {

extract($\_POST) ;

$formErrors = array() ;

if(empty($name)) {

$formErrors[] = 'User Name Can\'t Be <strong>Empty</Strong>' ;

}

if(strlen($name) < 4) {

$formErrors[] = 'User Name Can\'t Be Less Than <strong>4 Chars</Strong>' ;

}

if(empty($pass)) {

$formErrors[] = 'Empty <strong>Pass</strong>' ;

}

if(strlen($pass) < 6) {

$formErrors[] = 'Pass Less Than <strong>6</strong> Digits' ;

}

if(empty($confirm)) {

$formErrors[] = 'Empty <strong>Pass</strong>' ;

}

if(strlen($confirm) < 6) {

$formErrors[] = 'Pass Less Than <strong>6</strong> Digits' ;

}

if(!empty($formErrors)) { ?>

<div class="container">

<div class="row">

<div class="login-panel-box center">

<div class="panel panel-danger">

<div class="panel-heading">Register Form Errors</div>

<div class="panel-body">

<?php

Foreach ($formErrors as $error) {

echo '<div class="alert alert-danger">'.$error.'</div>';

}

redirect("Redirecting ..");

?>

</div>

</div>

</div>

</div>

</div>

<?php }

if(empty($formErrors)) {

$img\_name = $\_FILES['img']['name'] ;

$tmp\_name = $\_FILES['img']['tmp\_name'];

move\_uploaded\_file($tmp\_name,'img/'.$img\_name) ;

$hashedPass = sha1($pass) ;

$query = "INSERT INTO signup VALUES ('','$name','$hashedPass','$img\_name')" ;

$stmt = $conn->prepare($query) ;

$stmt->execute();

if($stmt->rowCount() > 0) {

header('Location: index.php?run=success'); // Redirect to dashboard Page

exit();

}

}

}

?>

examination system for Android™



**#Welcome**

Welcome to examination's open source Android app! Come on in, take your shoes

off, stay a while explore how examination's 's native squad has built and

continues to build the app, \_discover\_ our implementation of [RxJava](https://github.com/ReactiveX/RxJava) in logic-

filled [view models](https://github.com/mohamedebrahim96/MET-Quiz),

and maybe even create an issue or two.

**#Purpose of the Project**

- Attempting all the quizzes will be easy for everyone.

- Access to the quiz for the revision even after the 3 months challenge.

- Easy to manage quizzes and keeping the track of all attempted quizzes.

- Android Resources: Useful resources related to Android based on a search keyword

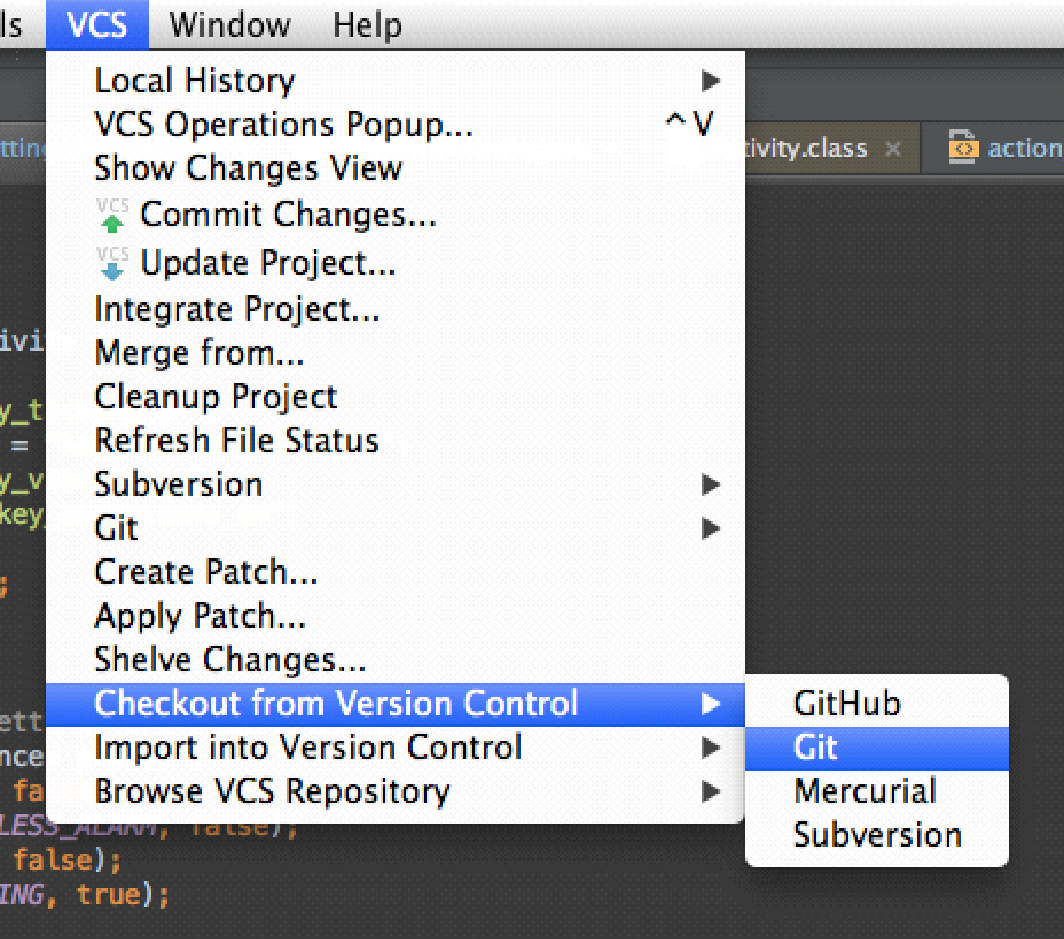
- replace using paper.

**#Getting Started**

Follow these instructions to build and run the project without data; note that

the app will be blank.

1. Clone this repository from git hub link ([**https://github.com/mohamedebrahim96/MET-Quiz**](https://github.com/mohamedebrahim96/MET-Quiz))



2. Download the appropriate [JDK](**http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html**)

for your system. We are currently on JDK 8.

3.Install Android Studio (<https://developer.android.com/sdk/index.html>).

4. `cd` into the project repo and run `Quize` to your Android

  development environment. Keep an eye on the output to see if   any manual steps

  are required.

5. Import the project. Open Android Studio, click `Open an existing Android

  Studio project` and select the project. Gradle will build the project.

6. Run the app. Click `Run > Run 'app'`. After the project builds you'll be

  prompted to build or launch an emulator.

**#Features**

- Educational (based on Exam).

- Added new Exams and question Done by the Professor

- It has database to store questions , Exams and previous results

- Every exam have it's own duration (manage by Professor)

- Timmer of 500 mill sec for each question (Timer at top center)

- Random questions in exam (It will peek random ques and will show it to the user)

- Good and Extensible UI.

- It's offline exam besed on local server

- Compatibility with Android API-14 and above

- minSdkVersion 14

- targetSdkVersion 25 (android Nougat)

#**Application Requirements**

Basic authentication & Details -

1) Splash Screen

2) Login Screen

We intend to keep only Google login for authenticating users as of now.

=====================================================

3) Sign up

One time screen after the user authenticates. This screen will take user details. The fields will include:

card number

email

password

first Name

last Name

Photo/Select Avatar

=====================================================

4) Home Screen

There will be a list of all the quizzes and each quiz item will contain the following information:

Screen title

Student's Name

Exam  
settings  
results

barcode Fragment

=====================================================

5)Results

Overall Score.

Number of quizzes attempted.

Attempted Quizzes history & score.

Course Notes / External resources.

**#Contributing**

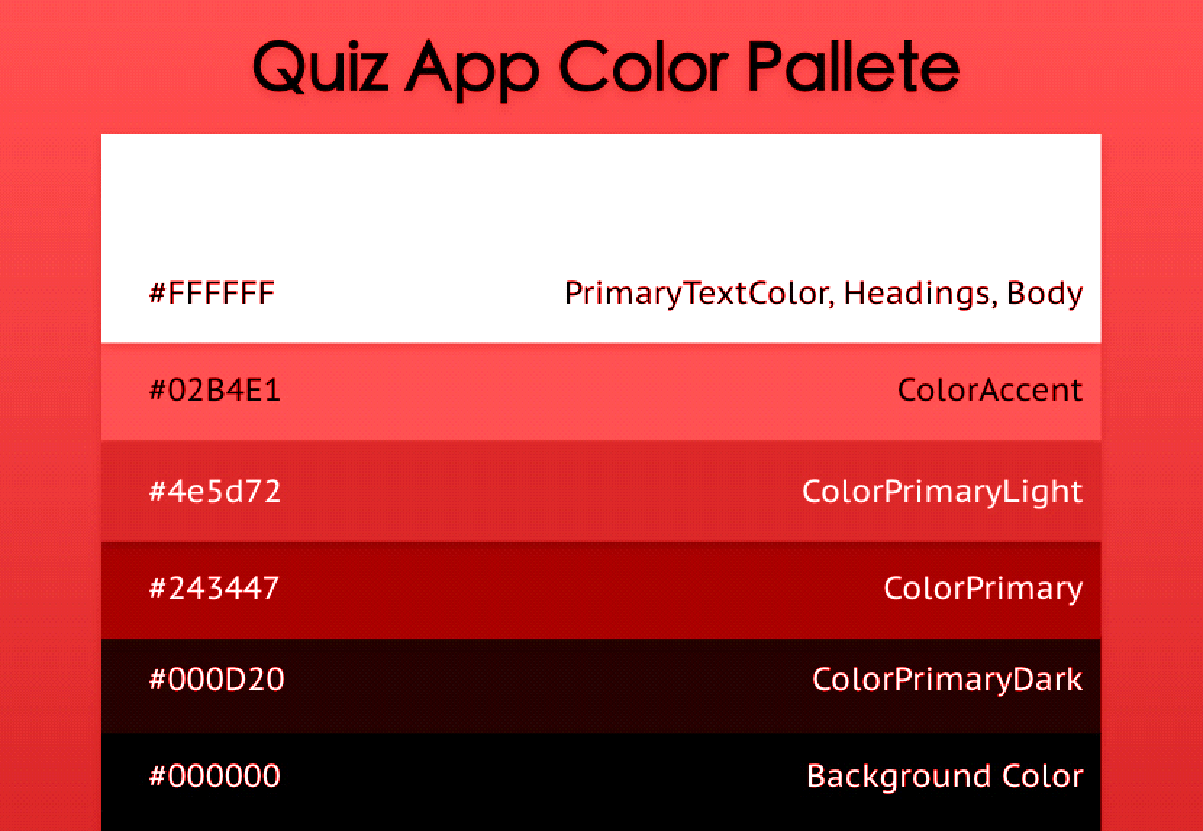
We intend for this project to be an educational resource: we are excited to share our wins, mistakes, and methodology of Android development as we work in the open. Our primary focus is to continue improving the app for our users in line with our roadmap.

The best way to submit feedback and report bugs is to open a Github issue. Please be sure to include your operating system, device, version number, and steps to reproduce reported bugs. Keep in mind that all participants will be expected to follow our code of conduct.

**#Code of Conduct**

We aim to share our knowledge and findings to the professor as we work daily to improve our product, for Examnition community, in a safe and open space. We work as we live, as kind and considerate students who learn and grow from giving and receiving positive, constructive feedback. professor reserve the right to delete or edit any of his students.

**#color palette**



<color name="colorPrimary">#ff5050</color>  
<color name="colorPrimaryDark">#e14646</color>  
<color name="*coloractionbar*">#ff5050 </color>  
<color name="colorAccent">#ff5050</color>  
<color name="year">#999999</color>  
<color name="title">#222222</color>  
<color name="black">#000</color>  
  
<color name="material\_light\_primary\_text">#DE000000</color>  
<color name="material\_light\_hint\_text">#61000000</color>  
<color name="material\_light\_active\_icon">#8A000000</color>  
<color name="material\_ripple\_light">#1F000000</color>  
<color name="divider\_color">#1F000000</color>  
<color name="active\_icon\_color">#000000</color>  
<color name="cards\_and\_dialogs\_color">@android:color/white</color>  
<color name="quantum\_grey\_600">#757575</color>  
<color name="premium">#ffb90b</color>  
<color name="gray\_quiz">#2E2E2E</color>  
<color name="white">#fff</color>  
<color name="dark\_white">#c8c8c8</color>

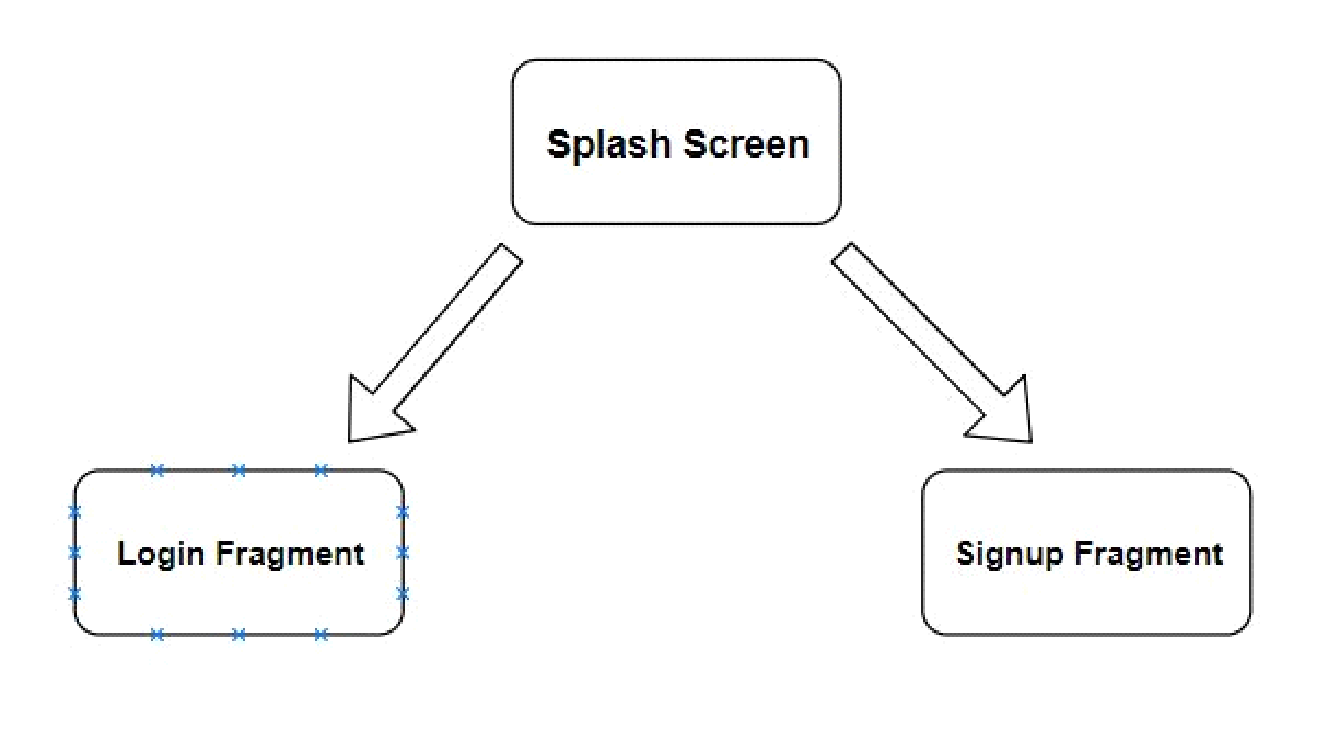
//=================================================================

//=================================================================

//=================================================================

**app activities and fragments**

**Splash screen**



public class SplashScreen extends AppCompatActivity {  
  
    public static final String *MY\_PREFS\_NAME* = "MyPrefsFile";  
    Context *mContext*;  
    public  SharedPreferences *prefs*;  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.*activity\_splash*);  
  
        *mContext* = this.getApplicationContext();  
        CalligraphyConfig.initDefault(new CalligraphyConfig.Builder()  
                .setDefaultFontPath("fonts/*airbnb*.ttf")  
                .setFontAttrId(R.attr.*fontPath*)  
                .build());  
        remmber\_me();  
    }  
  
    private void *remmber*\_me() {  
        *prefs* = *mContext*.getSharedPreferences(*MY\_PREFS\_NAME*, *MODE\_PRIVATE*);  
        String *remmber* = *prefs*.getString("card\_id",null);  
        if (remmber != null){  
            Intent i = new Intent(*mContext*, MainActivity.class);  
            startActivity(i);  
            finish();  
        }else {  
            login\_fragment();  
            CustomDialog();  
        }  
  
    }  
  
    private void CustomDialog() {  
        CustomDialogClass cdd=new CustomDialogClass(SplashScreen.this);  
        cdd.setCancelable(false);  
        cdd.show();  
    }  
  
    private void login\_fragment() {  
        Fragment fragment = new LoginFragment();  
        FragmentTransaction fragmentTransaction = getSupportFragmentManager().beginTransaction();  
        fragmentTransaction.setCustomAnimations(android.R.anim.*fade\_in*,  
                android.R.anim.*fade\_out*);  
        fragmentTransaction.replace(R.id.*frame*, fragment, *LOGIN\_FRAGMENT\_TAG*);  
        fragmentTransaction.commitAllowingStateLoss();  
    }  
  
    @Override  
    protected void attachBaseContext(Context newBase) {  
        super.attachBaseContext(CalligraphyContextWrapper.wrap(newBase));  
    }  
  
  
}

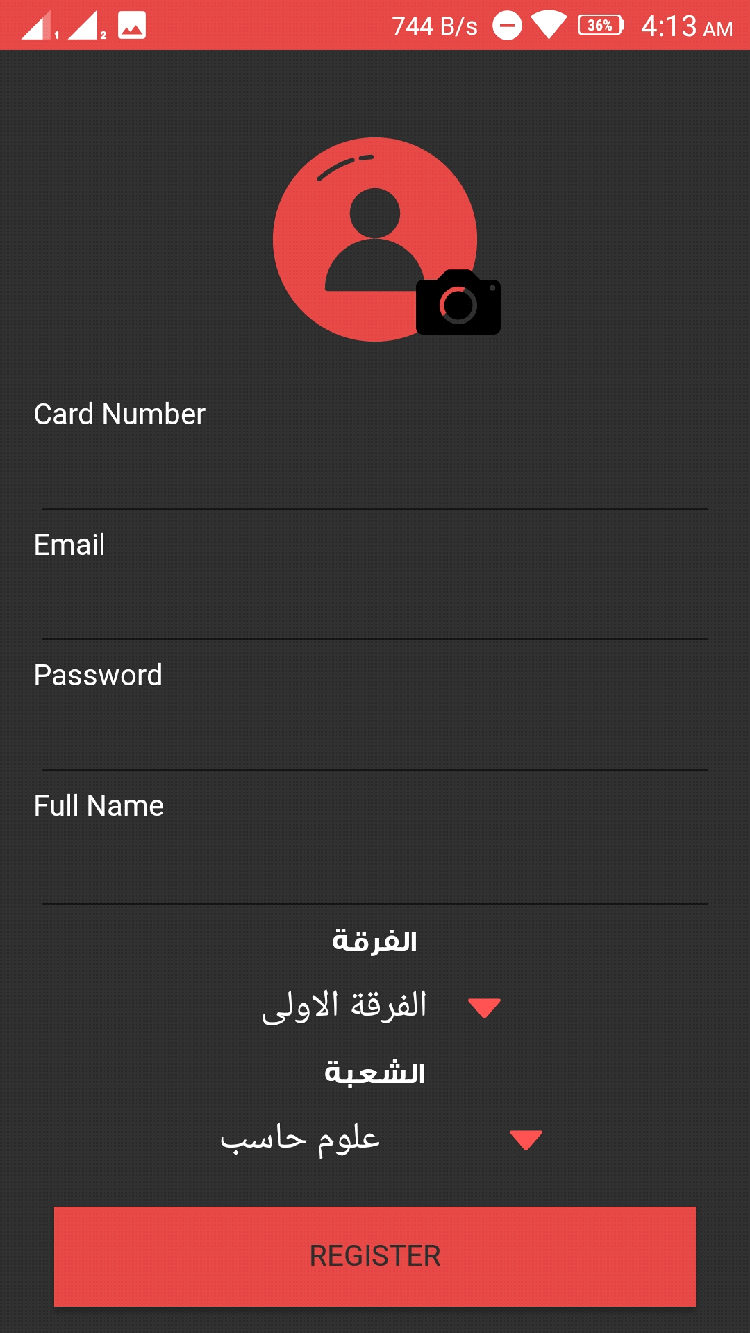
**==============================**

**Login Fragment**

public class LoginFragment extends Fragment implements View.OnClickListener{  
  
    final static String *LOGIN\_FRAGMENT\_TAG* ="LOGIN\_FRAGMENT\_TAG";  
  
    String *ROOT\_URL* ;  
    private EditText *login\_cardnumber*,*login\_password*;  
    TextView *later*,*terms*,*terms2*,*register*;  
    Button *login\_btn*;  
    Context *mContext*;  
    @Override  
    public View onCreateView(LayoutInflater inflater, ViewGroup container,  
                             Bundle savedInstanceState) {  
        View view = inflater.inflate(R.layout.*login\_fragment*, container, false);  
        *terms* =  view.findViewById(R.id.*terms*);  
        *terms2* =  view.findViewById(R.id.*terms2*);  
        *login\_btn* =  view.findViewById(R.id.*login\_btn*);  
        *register* =  view.findViewById(R.id.*register*);  
  
        *login\_cardnumber* =  view.findViewById(R.id.*login\_cardnumber*);  
        *login\_password* =  view.findViewById(R.id.*login\_password*);  
  
  
        *mContext* = this.getActivity();  
  
        *login\_password*.setOnEditorActionListener(new EditText.OnEditorActionListener() {  
            @Override  
            public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {  
                if (actionId == EditorInfo.*IME\_ACTION\_DONE*) {  
                   login();  
                    return true;  
                }  
                return false;  
            }  
        });  
  
  
        *later* =  view.findViewById(R.id.*later*);  
  
  
        *register*.setOnClickListener(this);  
        *login\_btn*.setOnClickListener(this);  
        *terms2*.setOnClickListener(this);  
        *later*.setOnClickListener(this);  
  
        return view;  
    }  
  
    @Override  
    public void onClick(View view) {  
  
        switch (view.getId())  
        {  
            case R.id.*login\_btn*:  
                login();  
                break;  
  
            case R.id.*terms2*:  
                startActivity(new Intent(getActivity(), PrivacyPolicyActivity.class));  
                getActivity().overridePendingTransition(R.anim.*fade\_in*, R.anim.*fade\_out*);  
                break;  
            case R.id.*register*:  
                Fragment fragment = new SignupFragment();  
                FragmentTransaction fragmentTransaction = getActivity().getSupportFragmentManager().beginTransaction();  
                fragmentTransaction.setCustomAnimations(android.R.anim.*fade\_in*,  
                        android.R.anim.*fade\_out*);  
                fragmentTransaction.replace(R.id.*frame*, fragment, *SIGNUP\_FRAGMENT\_TAG*);  
                fragmentTransaction.addToBackStack(null);  
                fragmentTransaction.commitAllowingStateLoss();  
                break;  
            case R.id.*later*:  
                skipSplash();  
                break;  
        }  
    }  
  
    private void login() {  
  
        SharedPreferences prefs = *mContext*.getSharedPreferences(*MY\_PREFS\_NAME*, *MODE\_PRIVATE*);  
        *ROOT\_URL* = prefs.getString("ip", "<http://192.168.1.5/>");  
  
        Retrofit retrofit = new Retrofit.Builder()  
                .baseUrl(*ROOT\_URL*)  
                .addConverterFactory(GsonConverterFactory.create())  
                .build();  
  
        RegisterAPI api = retrofit.create(RegisterAPI.class);  
        api.loging\_user(  
                *login\_cardnumber*.getText().toString(),  
                *login\_password*.getText().toString()  
        ).enqueue(new Callback<User>() {  
            @Override  
            public void onResponse(Call<User> call, Response<User> response) {  
  
                if(response.isSuccessful()) {  
                    User u = response.body();  
  
                    //System.out.println("====================================================");  
                    //System.out.println(*responsse*);  
                    Toast.makeText(*mContext*,"Hello, "+u.getFname(), Toast.*LENGTH\_SHORT*).show();  
                    //Log.e("TAG", *responsse*.toString());  
                    SharedPreferences.Editor editor = *mContext*.getSharedPreferences(*MY\_PREFS\_NAME*, *MODE\_PRIVATE*).edit();  
                    editor.putString("student\_id",u.getStudentId());  
                    editor.putString("card\_id",u.getCardId());  
                    editor.putString("email",u.getEmail());  
                    editor.putString("password",u.getPassword());  
                    editor.putString("*fname*",u.getFname());  
                    editor.putString("*lname*",u.getLname());  
                    editor.putInt("grade\_id",Integer.parseInt(u.getGradeId()));  
                    editor.putInt("division\_id",Integer.parseInt(u.getDivisionId()));  
                    editor.apply();  
                    skipSplash();  
                }  
            }  
  
            @Override  
            public void onFailure(Call<User> call, Throwable t) {  
                Log.e("TAG", "Unable to submit post to API.");  
            }  
        });  
    }  
    private void skipSplash()  
    {  
        Intent i = new Intent(getActivity(), MainActivity.class);  
        startActivity(i);  
        getActivity().finish();  
    }

**==============================**

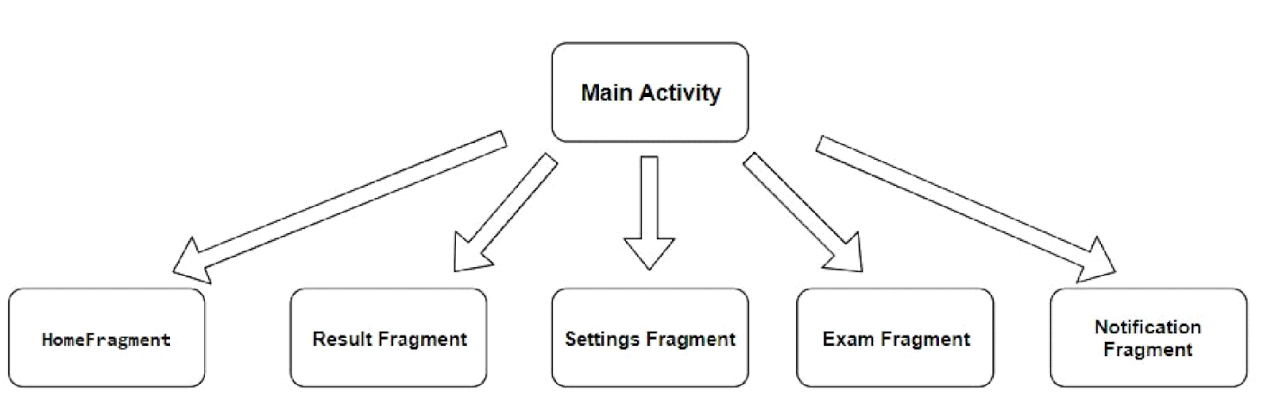
**Signup Fragment**



public class *Signup*Fragment extends Fragment implements View.OnClickListener{  
  
    final static String *SIGNUP*\_FRAGMENT\_TAG = "*SIGNUP*\_FRAGMENT\_TAG";  
    private EditText *card\_id*,*email*,*password*,*fname*,*lname*;  
    Button *buttonRegister*;  
    String *ROOT\_URL*;  
    Context *mContext*;  
    Spinner *spinner\_grad*,*spinner\_division*;  
    static int *grade\_id*,*division\_id*;  
  
    @Override  
    public View onCreateView(LayoutInflater inflater, ViewGroup container,  
                             Bundle savedInstanceState) {  
        View view = inflater.inflate(R.layout.*signup\_layout*, container, false);  
        *mContext* = this.getActivity();  
        *card\_id* =  view.findViewById(R.id.*card\_id*);  
        *email* =  view.findViewById(R.id.*email*);  
        *password* =  view.findViewById(R.id.*password*);  
        *fname* =  view.findViewById(R.id.*fname*);  
        //*lname* =  view.findViewById(R.id.*lname*);  
        *buttonRegister* =  view.findViewById(R.id.*buttonRegister*);  
        *spinner\_grad*= view.findViewById(R.id.*spinner\_grad*);  
        *spinner\_division*= view.findViewById(R.id.*spinner\_division*);  
  
  
  
        spinner\_setup();  
  
        *buttonRegister*.setOnClickListener(this);  
        return view;  
    }  
  
    private void spinner\_setup() {  
        List<String> grades = new ArrayList<>();  
        grades.add("الفرقة الاولى");  
        grades.add("الفرقة الثانية");  
        grades.add("الفرقة الثالثه");  
        grades.add("الفرقة الرابعة");  
  
  
        ArrayAdapter<String> adapter = new ArrayAdapter<>(*mContext*, R.layout.*spinner\_item\_layout*, grades);  
        *spinner\_grad*.setAdapter(adapter);  
        *spinner\_grad*.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {  
            @Override  
            public void onItemSelected(AdapterView<?> parentView, View selectedItemView, int position, long id) {  
                *grade\_id* = position+1;  
            }  
            @Override  
            public void onNothingSelected(AdapterView<?> parentView) {  
            }  
        });  
        //=======================================================  
        //*Devision*final List<String> division\_list = new ArrayList<>();  
        division\_list.add("علوم حاسب");  
        division\_list.add("شعبة نظم المعلومات الإدارية");  
        division\_list.add("محاسبة");  
        division\_list.add("إدارة الأعمال");  
  
        ArrayAdapter<String> adapter\_division = new ArrayAdapter<String>(*mContext*, R.layout.*spinner\_item\_layout*, division\_list);  
        *spinner\_division*.setAdapter(adapter\_division);  
        *spinner\_division*.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {  
            @Override  
            public void onItemSelected(AdapterView<?> parentView, View selectedItemView, int position, long id) {  
                *division\_id* = position+1;  
            }  
            @Override  
            public void onNothingSelected(AdapterView<?> parentView) {  
            }  
        });  
  
  
    }  
  
  
    private void insertUser() {  
        SharedPreferences prefs = *mContext*.getSharedPreferences(*MY\_PREFS\_NAME*, *MODE\_PRIVATE*);  
        *ROOT\_URL* = prefs.getString("ip", "<http://192.168.1.5/>");  
  
        Retrofit retrofit = new Retrofit.Builder()  
                .baseUrl(*ROOT\_URL*)  
                .addConverterFactory(GsonConverterFactory.create())  
                .build();  
  
        RegisterAPI api = retrofit.create(RegisterAPI.class);  
        api.insertUser(  
                *card\_id*.getText().toString(),  
                *email*.getText().toString(),  
                *password*.getText().toString(),  
                *fname*.getText().toString(),  
                "hh",  
                *grade\_id*,  
                *division\_id*).enqueue(new Callback<User>() {  
            @Override  
            public void onResponse(Call<User> call, Response<User> response) {  
  
                if(response.isSuccessful()) {  
                    User u = response.body();  
                        Toast.makeText(*mContext*,"Registered Successfully", Toast.*LENGTH\_SHORT*).show();  
  
                        SharedPreferences.Editor editor = *mContext*.getSharedPreferences(*MY\_PREFS\_NAME*, *MODE\_PRIVATE*).edit();  
                        editor.putString("student\_id",u.getCardId());  
                        editor.putString("card\_id",u.getCardId());  
                        editor.putString("email",u.getEmail());  
                        editor.putString("password",u.getPassword());  
                        editor.putString("*fname*",u.getFname());  
                        editor.putString("*lname*",u.getLname());  
                        editor.putInt("grade\_id",Integer.parseInt(u.getGradeId()));  
                        editor.putInt("division\_id",Integer.parseInt(u.getDivisionId()));  
                        editor.apply();  
                        skipSplash();  
  
                }  
            }  
  
            @Override  
            public void onFailure(Call<User> call, Throwable t) {  
                Log.e("TAG", "insertUser():: Unable to submit post to API.");  
            }  
        });  
    }  
  
    private void validateFields() {  
        if (*card\_id*.getText().length() == 0) {  
            *card\_id*.setError("Empty Field");  
        }else if (*email*.getText().length() == 0){  
            *email*.setError("Empty Field");  
  
        }else {  
            insertUser();  
        }  
    }  
  
    @Override  
    public void onClick(View view) {  
        switch (view.getId()) {  
            case R.id.*buttonRegister*:  
                validateFields();  
  
                break;  
        }  
    }  
  
    private void skipSplash()  
    {  
        Intent i = new Intent(getActivity(), MainActivity.class);  
        startActivity(i);  
        getActivity().finish();  
    }  
  
}

**==============================**

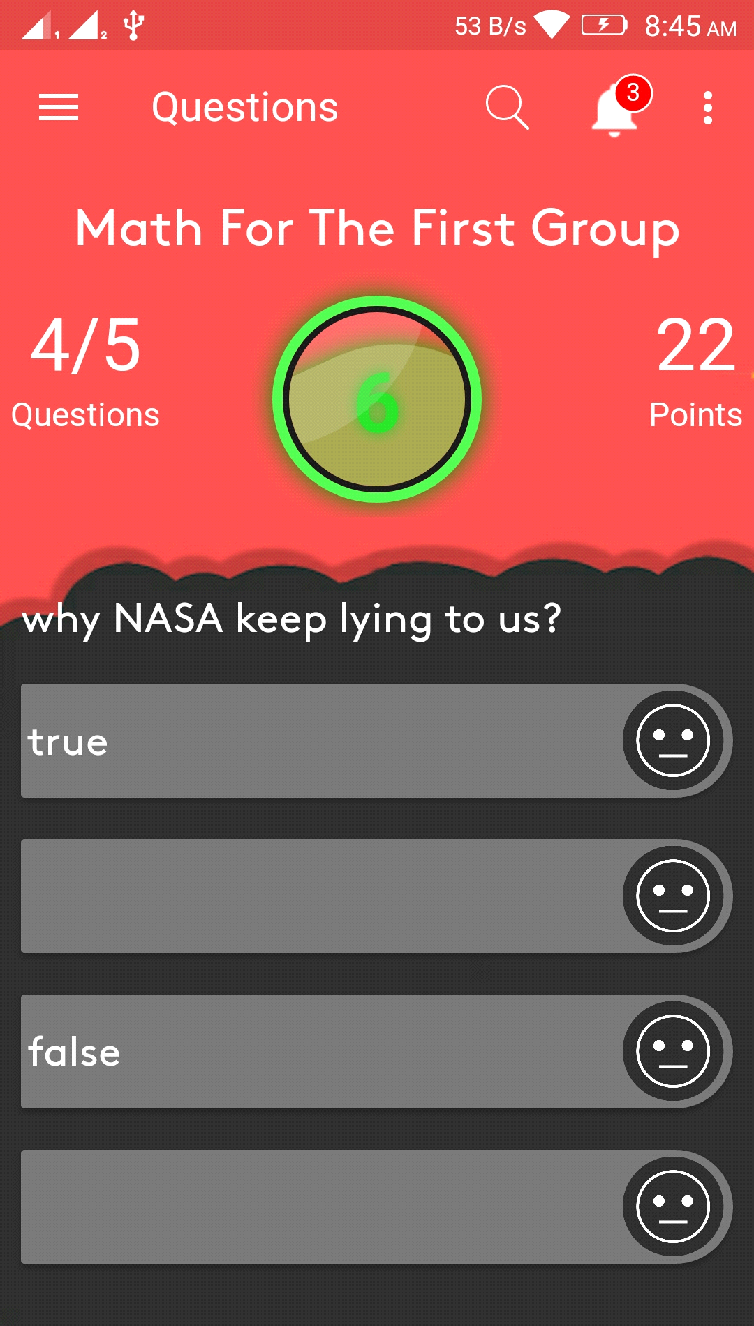
**Main Activity**



public class MainActivity extends AppCompatActivity {  
  
    Context *mContext*;  
    public Toolbar *mToolbar*;  
    public ImageView *notify\_layout*;  
    public SearchView *editsearch*;  
    public RelativeLayout *red\_badge*;  
    //=======================================================  
    public static NavigationView *navigationView*;  
    private DrawerLayout *drawer*;  
    private View *navHeader*;  
    private ImageView *imgNavHeaderBg*, *imgProfile*;  
    private TextView *txtName*, *txtWebsite*;  
    ActionBarDrawerToggle *actionBarDrawerToggle*;  
    View *parentLayout*;  
    // index to identify current nav menu item  
    public static int *navItemIndex* = 0;  
  
    // tags used to attach the fragments  
    public static final String *TAG\_HOME* = "home";  
    public static final String *TAG\_browse* = "browse";  
    public static final String *TAG\_Barcodescanner* = "*barcodescanner*";  
    public static final String *TAG\_NOTIFICATIONS* = "notifications";  
    public static final String *TAG\_SETTINGS* = "settings";  
    public static final String *TAG\_QUESTIONS* = "QUESTIONS";  
  
    public static String *CURRENT\_TAG* = *TAG\_HOME*;  
    // toolbar titles respected to selected nav menu item  
    public static String[] *activityTitles*;  
  
    // flag to load home fragment when user presses back key  
    private boolean *shouldLoadHomeFragOnBackPress* = true;  
  
  
    public static MainActivity *INSTANCE*;  
    private static final String *DATABASE\_NAME* = "MyDatabase";  
    private static final String *PREFERENCES* = "RoomDemo.preferences";  
    private static final String *KEY\_FORCE\_UPDATE* = "force\_update";  
  
    private MyDatabase *database*;  
  
    public static MainActivity get() {  
        return *INSTANCE*;  
    }  
  
  
    public MyDatabase getDB() {  
        return *database*;  
    }  
  
    public boolean isForceUpdate() {  
        return getSP().getBoolean(*KEY\_FORCE\_UPDATE*, true);  
    }  
  
    public void setForceUpdate(boolean force) {  
        SharedPreferences.Editor edit = getSP().edit();  
        edit.putBoolean(*KEY\_FORCE\_UPDATE*, force);  
        edit.apply();  
    }  
  
    private SharedPreferences getSP() {  
        return getSharedPreferences(*PREFERENCES*, *MODE\_PRIVATE*);  
    }  
  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.*activity\_main*);  
        *mContext* = this.getApplicationContext();  
        *parentLayout* = findViewById(android.R.id.*content*);  
        *mToolbar* =  findViewById(R.id.*toolbar*);  
        setSupportActionBar(*mToolbar*);  
        *editsearch* =  findViewById(R.id.*search\_view*);  
        *notify\_layout* =  findViewById(R.id.*image*);  
        *red\_badge* =  findViewById(R.id.*red\_badge*);  
        *drawer* =  findViewById(R.id.*drawer\_layout*);  
        *navigationView* =  findViewById(R.id.*nav\_view*);  
        // Navigation view header  
        *navHeader* = *navigationView*.getHeaderView(0);  
        *txtName* =  *navHeader*.findViewById(R.id.*name*);  
        *txtWebsite* =  *navHeader*.findViewById(R.id.*website*);  
        *imgNavHeaderBg* =  *navHeader*.findViewById(R.id.*img\_header\_bg*);  
        *imgProfile* =  *navHeader*.findViewById(R.id.*img\_profile*);  
  
  
  
        // create database  
        *database* = Room.databaseBuilder(getApplicationContext(), MyDatabase.class, *DATABASE\_NAME*)  
                .addMigrations(MyDatabase.*MIGRATION\_1\_2*)  
                .build();  
  
        *INSTANCE* = this;  
  
        CalligraphyConfig.initDefault(new CalligraphyConfig.Builder()  
                .setDefaultFontPath("fonts/*airbnb*.ttf")  
                .setFontAttrId(R.attr.*fontPath*)  
                .build()  
        );  
  
        *notify\_layout*.setOnClickListener(new View.OnClickListener() {  
            @Override  
            public void onClick(View view) {  
                //Toast.makeText(MainActivity.this, "Notify", Toast.LENGTH\_LONG).show();  
                //red\_badge.setVisibility(View.GONE);  
                NotifyFragment notifyFragment = new NotifyFragment();  
                getSupportFragmentManager().beginTransaction()  
                        .replace(R.id.*container*, notifyFragment,*TAG\_NOTIFICATIONS*)  
                        .commitAllowingStateLoss();  
                *navItemIndex* = 3;  
                *CURRENT\_TAG* = *TAG\_NOTIFICATIONS*;  
                selectNavMenu();  
                setToolbarTitle();  
            }  
        });  
        /\**editsearch*.setOnSearchClickListener(new View.OnClickListener() {  
            @Override  
            public void onClick(View view) {  
                searchFragment = new SearchFragment();  
                getSupportFragmentManager().beginTransaction()  
                        .replace(R.id.container, searchFragment,TAG\_NOTIFICATIONS)  
                        .commitAllowingStateLoss();  
                navItemIndex = 3;  
                CURRENT\_TAG = TAG\_NOTIFICATIONS;  
            }  
        });\*/  
        //================================================  
  
        // load toolbar titles from string resources  
        *activityTitles* = getResources().getStringArray(R.array.*nav\_item\_activity\_titles*);  
  
  
        Snackbar.make(*parentLayout*, "Rate our app", Snackbar.*LENGTH\_LONG*)  
                        .setAction("Rate", new View.OnClickListener() {  
                            @Override  
                            public void onClick(View view) {  
                                Rate\_us();  
                            }  
                        }).show();  
  
        loadNavHeader();  
        setUpNavigationView();  
        if (savedInstanceState == null) {  
            *navItemIndex* = 0;  
            *CURRENT\_TAG* = *TAG\_HOME*;  
            loadHomeFragment();  
            *actionBarDrawerToggle*.syncState();  
        }  
  
        //==================================================  
        Menu m = *navigationView*.getMenu();  
        for (int i=0;i<m.size();i++) {  
            MenuItem mi = m.getItem(i);  
  
            //for *aapplying* a font to subMenu ...  
            SubMenu subMenu = mi.getSubMenu();  
            if (subMenu!=null && subMenu.size() >0 ) {  
                for (int j=0; j <subMenu.size();j++) {  
                    MenuItem subMenuItem = subMenu.getItem(j);  
                    applyFontToMenuItem(subMenuItem);  
                }  
            }  
            applyFontToMenuItem(mi);  
        }  
    }  
  
  
  
    @Override  
    public boolean onCreateOptionsMenu(Menu menu) {  
        if (*navItemIndex* == 2) {  
            getMenuInflater().inflate(R.menu.*favourite*, menu);  
        }else if (*navItemIndex* == 3) {  
            getMenuInflater().inflate(R.menu.*notifications*, menu);  
        }else {  
            getMenuInflater().inflate(R.menu.*menu\_menu*, menu);  
        }  
  
        //============================================================  
        for(int i=0;i<menu.size();i++)  
        {  
            MenuItem mi = menu.getItem(i);  
            applyFontToMenuItem(mi);  
        }  
  
        /\*Menu menu1 = navigationView.getMenu();  
        Drawable *yourdrawable* = menu1.getItem(4).getIcon(); // change 0 with 1,2 ...  
        *yourdrawable*.mutate();  
        *yourdrawable*.setColorFilter(getResources().getColor(R.color.premium), PorterDuff.Mode.SRC\_ATOP);\*/  
        return true;  
    }  
  
    @Override  
    public boolean onOptionsItemSelected(MenuItem item) {  
        // Handle item selection  
        int id = item.getItemId();  
  
        if (id == R.id.*log\_out*) {  
            log\_out();  
        }  
        if (id == R.id.*action\_mark\_all\_read*) {  
            Toast.makeText(getApplicationContext(), "All notifications marked as read!", Toast.*LENGTH\_LONG*).show();  
        }  
        if (id == R.id.*action\_clear\_notifications*) {  
            Toast.makeText(getApplicationContext(), "Clear all notifications!", Toast.*LENGTH\_LONG*).show();  
        }  
  
        if (id == R.id.*about*) {  
            startActivity(new Intent(MainActivity.this, AboutUsActivity.class));  
            *drawer*.closeDrawers();  
        }  
        if (id == R.id.*settings*) {  
            *navItemIndex* = 4;  
            *CURRENT\_TAG* = *TAG\_SETTINGS*;  
            loadHomeFragment();  
        }  
        if (id == R.id.*Notification*) {  
            *navItemIndex* = 3;  
            *CURRENT\_TAG* = *TAG\_NOTIFICATIONS*;  
            loadHomeFragment();  
        }  
        return super.onOptionsItemSelected(item);  
    }  
  
    @Override  
    public void onBackPressed() {  
        if (!*editsearch*.isIconified()) {  
            *editsearch*.onActionViewCollapsed();  
            loadHomeFragment();  
        }  
  
        if (*drawer*.isDrawerOpen(GravityCompat.*START*)) {  
            *drawer*.closeDrawers();  
            return;  
        }  
  
  
        if (*shouldLoadHomeFragOnBackPress*) {  
            // checking if user is on other navigation menu  
            // rather than home  
            if (*navItemIndex* != 0) {  
                *navItemIndex* = 0;  
                *CURRENT\_TAG* = *TAG\_HOME*;  
                loadHomeFragment();  
                return ;  
            }  
        }  
        if (*CURRENT\_TAG* == *TAG\_HOME*) {  
            super.onBackPressed();  
        }  
  
    }  
  
  
  
    //====================================================================================  
    //====================================================================================  
    //====================================================================================  
  
    private void loadNavHeader() {  
        // name, website  
  
        SharedPreferences prefs = getSharedPreferences(*MY\_PREFS\_NAME*, *MODE\_PRIVATE*);  
        String name = prefs.getString("*fname*", "@*mohamedebrahim*96");//"No name defined" is the default value.  
        String email = prefs.getString("email", "ebrahimm132@gmail.com");  
        String *imageurl* = prefs.getString("*imageurl*", "<https://avatars2.githubusercontent.com/u/16405013?s=460&v=4>");  
        String cover = prefs.getString("cover", "defaultStringIfNothingFound");  
  
  
        *txtName*.setText(name);  
        *txtWebsite*.setText(email);  
  
        RequestOptions options = RequestOptions  
                .circleCropTransform()  
                .placeholder(R.drawable.*thin128*);  
        RequestOptions options2 = new RequestOptions()  
                .placeholder(R.drawable.*nav*);  
        Glide.with(this).load(imageurl)  
                    .thumbnail(0.5f)  
                    .apply(options)  
                    .into(*imgProfile*);  
        Glide.with(this)  
                //.load(urlNavHeaderBg)  
                .load(cover)  
                .apply(options2)  
                .into(*imgNavHeaderBg*);  
        *navigationView*.getMenu().getItem(3).setActionView(R.layout.*menu\_dot*);  
        *navigationView*.setItemIconTintList(null);  
  
    }  
  
    /\*\*\*  
     \* Returns respected fragment that user  
     \* selected from navigation menu  
     \*/  
    private void loadHomeFragment() {  
        // selecting appropriate nav menu item  
        selectNavMenu();  
  
        // set toolbar title  
        setToolbarTitle();  
  
        // if user select the current navigation menu again, don't do anything  
        // just close the navigation drawer  
        if (getSupportFragmentManager().findFragmentByTag(*CURRENT\_TAG*) != null) {  
            *drawer*.closeDrawers();  
  
            // show or hide the fab button  
            return;  
        }  
  
        // Sometimes, when fragment has huge data, screen seems hanging  
        // when switching between navigation menus  
        // So using runnable, the fragment is loaded with cross fade effect  
        // This effect can be seen in GMail app  
  
                // update the favourite content by replacing fragments  
                Fragment fragment = getHomeFragment();  
                FragmentTransaction fragmentTransaction = getSupportFragmentManager().beginTransaction();  
                fragmentTransaction.setCustomAnimations(android.R.anim.*fade\_in*,  
                        android.R.anim.*fade\_out*);  
                fragmentTransaction.replace(R.id.*container*, fragment, *CURRENT\_TAG*);  
                fragmentTransaction.commitAllowingStateLoss();  
  
  
        //Closing drawer on item click  
        *drawer*.closeDrawers();  
  
        // refresh toolbar menu  
        invalidateOptionsMenu();  
    }  
  
    private Fragment getHomeFragment() {  
        switch (*navItemIndex*) {  
            case 0:  
                // home  
                HomeFragment homeFragment = new HomeFragment();  
                return homeFragment;  
            case 1:  
                // photos  
                HomeFragment homeFragment2 = new HomeFragment();  
                return homeFragment2;  
            case 2:  
                // movies fragment  
                FavoriteFragment favoriteFragment = new FavoriteFragment();  
                return favoriteFragment;  
            case 3:  
                // notifications fragment  
                NotifyFragment notificationsFragment = new NotifyFragment();  
                return notificationsFragment;  
  
            case 4:  
                // settings fragment  
                SettingsFragment settingsFragment = new SettingsFragment();  
                return settingsFragment;  
            default:  
                return new HomeFragment();  
        }  
    }  
  
    public  void setToolbarTitle() {  
        getSupportActionBar().setTitle(*activityTitles*[*navItemIndex*]);  
    }  
  
    public  void selectNavMenu() {  
        *navigationView*.getMenu().getItem(*navItemIndex*).setChecked(true);  
    }  
  
    public void setUpNavigationView() {  
        //Setting Navigation View Item Selected Listener to handle the item click of the navigation menu  
        *navigationView*.setNavigationItemSelectedListener(new NavigationView.OnNavigationItemSelectedListener() {  
  
            // This method will trigger on item Click of navigation menu  
            @Override  
            public boolean onNavigationItemSelected(MenuItem menuItem) {  
  
                //Check to see which item was being clicked and perform appropriate action  
                switch (menuItem.getItemId()) {  
                    //Replacing the favourite content with ContentFragment Which is our Inbox View;  
                    case R.id.*nav\_home*:  
                        *navItemIndex* = 0;  
                        *CURRENT\_TAG* = *TAG\_HOME*;  
                        break;  
                    case R.id.*nav\_browse*:  
                        *navItemIndex* = 1;  
                        *CURRENT\_TAG* = *TAG\_browse*;  
                        break;  
                    case R.id.*nav\_favourite*:  
                        *navItemIndex* = 2;  
                        *CURRENT\_TAG* = *TAG\_Barcodescanner*;  
                        break;  
                    case R.id.*nav\_notifications*:  
                        *navItemIndex* = 3;  
                        *CURRENT\_TAG* = *TAG\_NOTIFICATIONS*;  
                        break;  
                    case R.id.*nav\_settings*:  
                        *navItemIndex* = 4;  
                        *CURRENT\_TAG* = *TAG\_SETTINGS*;  
                        break;  
                    case R.id.*nav\_about\_us*:  
                        // launch new intent instead of loading fragment  
                        startActivity(new Intent(MainActivity.this, AboutUsActivity.class));  
                        *drawer*.closeDrawers();  
                        return true;  
                    case R.id.*Terms\_Conditions*:  
                        // launch new intent instead of loading fragment  
                        startActivity(new Intent(MainActivity.this, TermsConditions.class));  
                        overridePendingTransition(android.R.anim.*fade\_in*, android.R.anim.*fade\_out*);  
                        *drawer*.closeDrawers();  
                        return true;  
                    case R.id.*nav\_privacy\_policy*:  
                        // launch new intent instead of loading fragment  
                        startActivity(new Intent(MainActivity.this, PrivacyPolicyActivity.class));  
                        overridePendingTransition(R.anim.*fade\_in*, R.anim.*fade\_out*);  
                        *drawer*.closeDrawers();  
                        return true;  
                    case R.id.*log\_out*:  
                        // launch new intent instead of loading fragment  
                        log\_out();  
                        *drawer*.closeDrawers();  
                        return true;  
                    case R.id.*gopremium*:  
                        Toast.makeText(*mContext*, "Go Premium", Toast.*LENGTH\_SHORT*).show();  
                        *drawer*.closeDrawers();  
                        return true;  
                    case R.id.*rateus*:  
                        Rate\_us();  
                        *drawer*.closeDrawers();  
                        return true;  
                    default:  
                        *navItemIndex* = 0;  
                }  
  
                //Checking if the item is in checked state or not, if not make it in checked state  
                if (menuItem.isChecked()) {  
                    menuItem.setChecked(false);  
                } else {  
                    menuItem.setChecked(true);  
                }  
                menuItem.setChecked(true);  
  
                loadHomeFragment();  
  
                return true;  
            }  
        });  
  
        *actionBarDrawerToggle* = new ActionBarDrawerToggle(this, *drawer*, *mToolbar*, R.string.*openDrawer*, R.string.*closeDrawer*) {  
  
            @Override  
            public void onDrawerClosed(View drawerView) {  
                // Code here will be triggered once the drawer closes as we *dont* want anything to happen so we leave this blank  
                super.onDrawerClosed(drawerView);  
            }  
  
            @Override  
            public void onDrawerOpened(View drawerView) {  
                // Code here will be triggered once the drawer open as we *dont* want anything to happen so we leave this blank  
                super.onDrawerOpened(drawerView);  
            }  
        };  
  
        //Setting the actionbarToggle to drawer layout  
        *drawer*.setDrawerListener(*actionBarDrawerToggle*);  
        *actionBarDrawerToggle*.syncState();  
  
        //calling sync state is necessary or else your hamburger icon wont show up  
    }  
  
    private void log\_out() {  
        SharedPreferences preferences = getSharedPreferences(*MY\_PREFS\_NAME*, *MODE\_PRIVATE*);  
        preferences.edit().remove("card\_id").commit();  
        startActivity(new Intent(MainActivity.this, SplashScreen.class));  
    }  
  
  
  
    private void applyFontToMenuItem(MenuItem mi) {  
  
        if(mi.getItemId() != R.id.*gopremium*){  
            Typeface font = Typeface.createFromAsset(getAssets(), "fonts/*airbnb*.ttf");  
            SpannableString mNewTitle = new SpannableString(mi.getTitle());  
            mNewTitle.setSpan(new CustomTypefaceSpan("" , font), 0 , mNewTitle.length(),  Spannable.*SPAN\_INCLUSIVE\_INCLUSIVE*);  
            mi.setTitle(mNewTitle);  
        }else{  
            SpannableString s = new SpannableString(mi.getTitle());  
            s.setSpan(new ForegroundColorSpan(getResources().getColor(R.color.*premium*)), 0, s.length(), 0);  
            mi.setTitle(s);  
        }  
  
    }  
  
  
    private void Rate\_us() {  
        Uri uri = Uri.parse("market://details?id=" + *mContext*.getPackageName());  
        Intent goToMarket = new Intent(Intent.*ACTION\_VIEW*, uri);  
        // To count with Play market *backstack*, After pressing back button,  
        // to taken back to our application, we need to add following flags to intent.  
        goToMarket.addFlags(Intent.*FLAG\_ACTIVITY\_NO\_HISTORY* |  
                Intent.*FLAG\_ACTIVITY\_NEW\_DOCUMENT* |  
                Intent.*FLAG\_ACTIVITY\_MULTIPLE\_TASK*);  
        try {  
            startActivity(goToMarket);  
        } catch (ActivityNotFoundException e) {  
            startActivity(new Intent(Intent.*ACTION\_VIEW*,  
                    Uri.parse("http://play.google.com/store/apps/details?id=" + *mContext*.getPackageName())));  
        }  
    }  
    @Override  
    protected void attachBaseContext(Context newBase) {  
        super.attachBaseContext(CalligraphyContextWrapper.wrap(newBase));  
    }  
}

**==============================**

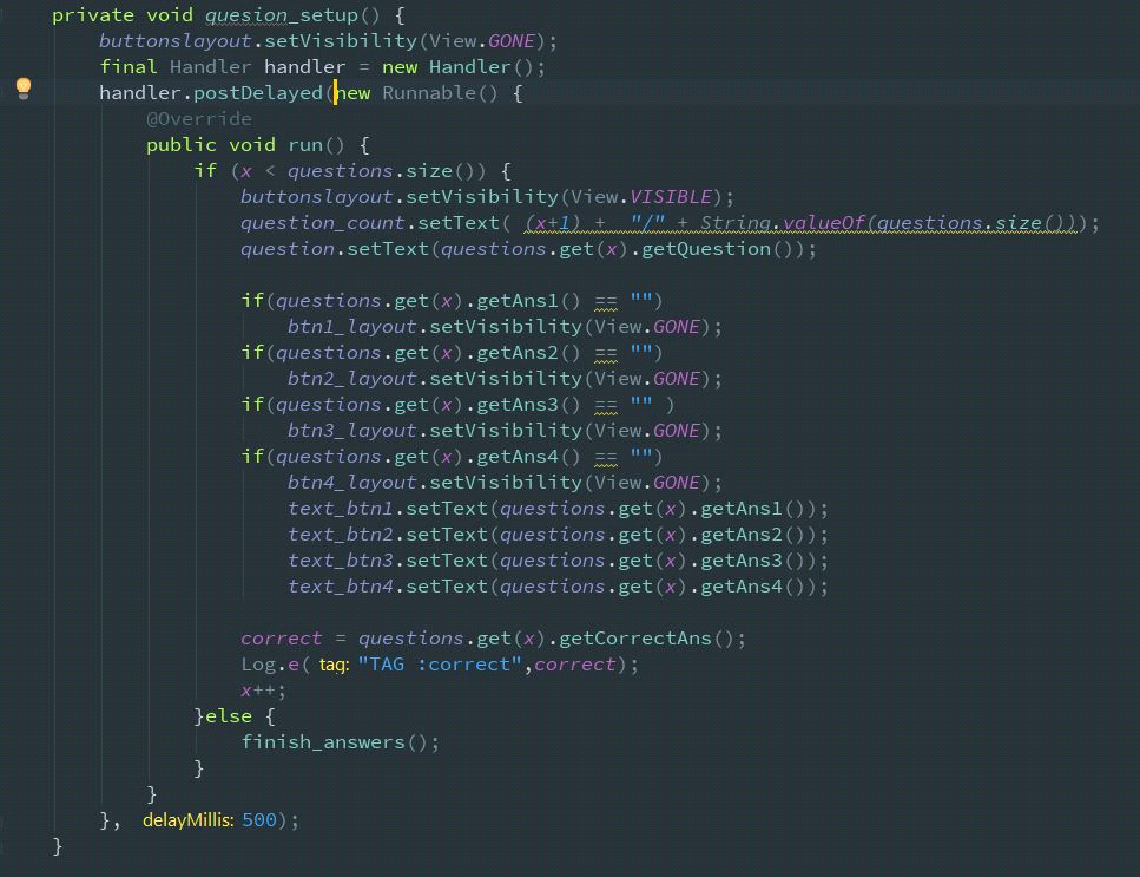
**Exam Fragment**



public class QuestionsFragment extends Fragment implements View.OnClickListener{  
  
    WebView *mWebView*;  
    List<QuestionModel> *questions* ;  
    TextView *question\_count*,*question*,*text\_btn1*,*text\_btn2*,*text\_btn3*,*text\_btn4*,*points*,*exam\_name*,*total\_score*;  
    LinearLayout *buttonslayout*;  
    RelativeLayout *result\_layout*,*btn1\_layout*,*btn2\_layout*,*btn3\_layout*,*btn4\_layout*;  
    Button *btn1*,*btn2*,*btn3*,*btn4*,*home*;  
    public static int *x* = 0;  
    int *exam\_duration*,*points\_number*,*total\_questions\_points*;  
    static String *correct*;  
    Context *mContext*;  
    int *number\_of\_questions*,*student\_id*;  
    static int *degree* = 0;  
    String *exam\_start\_date*,*exam\_name2*,*ROOT\_URL*;  
    CountDownTimer *countDownTimer*;  
    static int *correct\_ans*;  
  
    @Override  
    public View onCreateView(LayoutInflater inflater, ViewGroup container,  
                             Bundle savedInstanceState) {  
        // Inflate the layout for this fragment  
        View view = inflater.inflate(R.layout.*questions\_fragment*, container, false);  
  
        *mContext* = this.getActivity();  
  
        *mWebView* =  view.findViewById(R.id.*webview*);  
        *buttonslayout* =  view.findViewById(R.id.*buttonslayout*);  
        *question\_count* =  view.findViewById(R.id.*question\_count*);  
        *result\_layout* =  view.findViewById(R.id.*result\_layout*);  
        *points* =  view.findViewById(R.id.*points*);  
        *exam\_name* =  view.findViewById(R.id.*exam\_name*);  
        *total\_score* =  view.findViewById(R.id.*total\_score*);  
        *home* =  view.findViewById(R.id.*home*);  
  
  
  
        *question* =  view.findViewById(R.id.*question*);  
        *btn1* =  view.findViewById(R.id.*btn1*);  
        *btn2* =  view.findViewById(R.id.*btn2*);  
        *btn3* =  view.findViewById(R.id.*btn3*);  
        *btn4* =  view.findViewById(R.id.*btn4*);  
        *text\_btn1* =  view.findViewById(R.id.*text\_btn1*);  
        *text\_btn2* =  view.findViewById(R.id.*text\_btn2*);  
        *text\_btn3* =  view.findViewById(R.id.*text\_btn3*);  
        *text\_btn4* =  view.findViewById(R.id.*text\_btn4*);  
  
        *btn1\_layout* =  view.findViewById(R.id.*btn1\_layout*);  
        *btn2\_layout* =  view.findViewById(R.id.*btn2\_layout*);  
        *btn3\_layout* =  view.findViewById(R.id.*btn3\_layout*);  
        *btn4\_layout* =  view.findViewById(R.id.*btn4\_layout*);  
  
  
  
        *btn1*.setOnClickListener(this);  
        *btn2*.setOnClickListener(this);  
        *btn3*.setOnClickListener(this);  
        *btn4*.setOnClickListener(this);  
  
  
  
  
        WebSettings webSettings = *mWebView*.getSettings();  
        webSettings.setJavaScriptEnabled(true);  
        webSettings.setJavaScriptCanOpenWindowsAutomatically(true);  
        //mWebView.clearCache(true);  
        //mWebView.clearHistory();  
        // mWebView.setWebChromeClient(new WebChromeClient());  
        // mWebView.setWebViewClient(new WebViewClient());  
        //mWebView.getSettings().setJavaScriptEnabled(true);  
        webSettings.setDomStorageEnabled(true);  
        //mWebView.getSettings().setJavaScriptEnabled(true);  
        //mWebView.getSettings().setDomStorageEnabled(true);  
        *mWebView*.setBackgroundColor(0);  
  
        *mWebView*.loadUrl("file:///android\_asset/index.html");  
  
  
        //====================================================================  
        MainActivity.*CURRENT\_TAG* =*TAG\_QUESTIONS*;  
        *navItemIndex* = 5;  
        ((AppCompatActivity)getActivity()).getSupportActionBar().setTitle(*activityTitles*[5]);  
//====================================================================  
  
        *home*.setOnClickListener(new View.OnClickListener() {  
            @Override  
            public void onClick(View v) {  
                getActivity().getFragmentManager().popBackStack();  
                popBackStack(getActivity().getSupportFragmentManager());  
                //getActivity().getSupportFragmentManager().beginTransaction().remove(QuestionsFragment.this).commit();  
            }  
        });  
  
  
        setup\_questions();  
        return view;  
    }  
    public static void popBackStack(FragmentManager manager){  
        FragmentManager.BackStackEntry first = manager.getBackStackEntryAt(0);  
        manager.popBackStack(first.getId(), FragmentManager.*POP\_BACK\_STACK\_INCLUSIVE*);  
    }

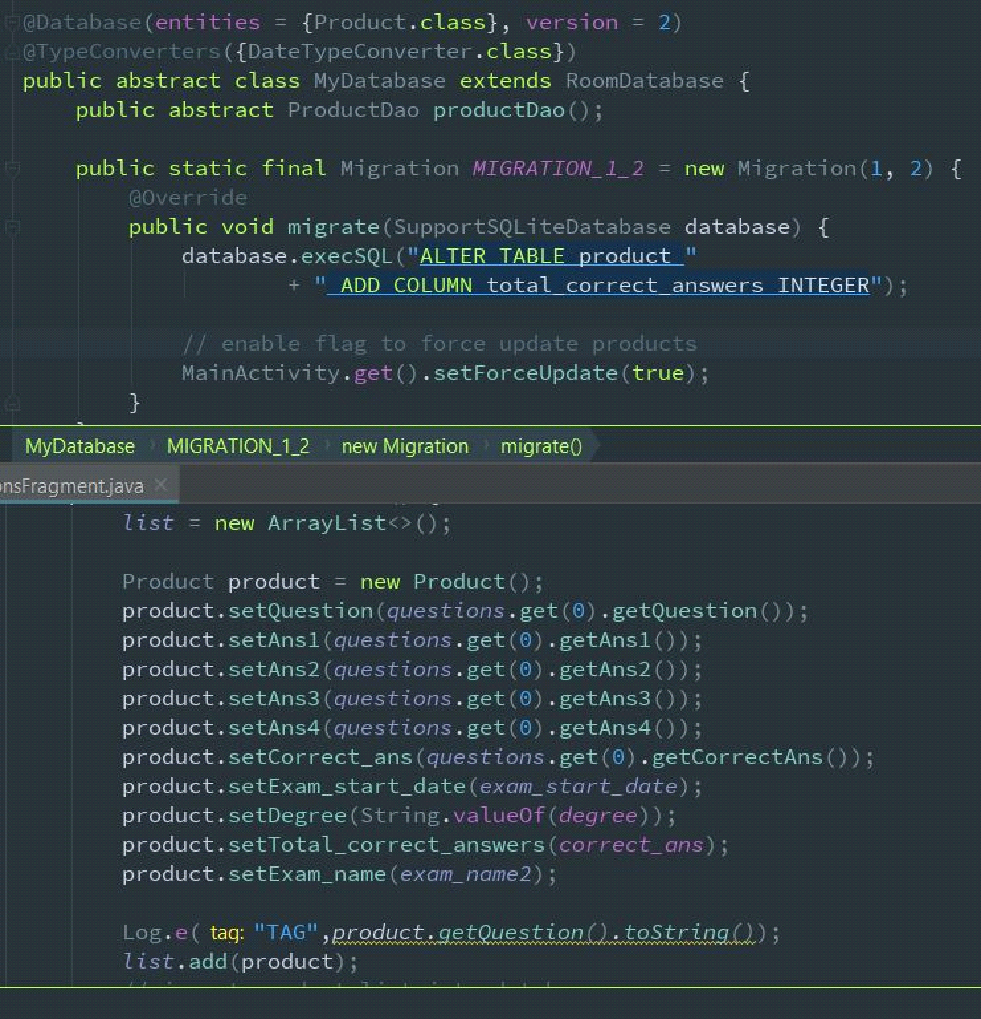
**algorithm changing the questions**

changing the questions every 500 mill second by loop and in a thread to avoid crash in the main thread.



private void setup\_questions() {  
  
        SharedPreferences prefs = *mContext*.getSharedPreferences(*MY\_PREFS\_NAME*, *MODE\_PRIVATE*);  
        *ROOT\_URL* = prefs.getString("ip", "<http://192.168.1.5/>");  
        *student\_id* = Integer.parseInt(prefs.getString("student\_id","911"));  
  
        Retrofit retrofit = new Retrofit.Builder()  
                .baseUrl(*ROOT\_URL*)  
                .addConverterFactory(GsonConverterFactory.create())  
                .build();  
  
        RegisterAPI api = retrofit.create(RegisterAPI.class);  
        api.getQuestions().enqueue(new Callback<Example>() {  
            @Override  
            public void onResponse(Call<Example> call, Response<Example> response) {  
  
                *questions* = new ArrayList<>();  
                for (QuestionModel fruit : response.body().getQuestionModel()) {  
                    *questions*.add(fruit);  
                }  
                *exam\_start\_date* = response.body().getStartDate();  
  
                *points\_number* = response.body().getPoints();  
                //points.setText(String.valueOf(response.body().getPoints()));  
                *exam\_duration* = Integer.parseInt(response.body().getExamDuration());  
                *exam\_name*.setText(response.body().getExamName());  
                *exam\_name2* = response.body().getExamName();  
                *total\_questions\_points* = *questions*.size() \* *points\_number*;  
                *number\_of\_questions* = *questions*.size();  
                quesion\_setup();  
                counter();  
            }  
            @Override  
            public void onFailure(Call<Example> call, Throwable t) {  
                Log.e("TAG",t.toString());  
            }  
        });  
    }  
    private void *quesion*\_setup() {  
        *buttonslayout*.setVisibility(View.*GONE*);  
        final Handler handler = new Handler();  
        handler.postDelayed(new Runnable() {  
            @Override  
            public void run() {  
                if (*x* < *questions*.size()) {  
                    *buttonslayout*.setVisibility(View.*VISIBLE*);  
                    *question\_count*.setText( (*x*+1) +  "/" + String.valueOf(*questions*.size()));  
                    *question*.setText(*questions*.get(*x*).getQuestion());  
  
                    if(*questions*.get(*x*).getAns1() == "")  
                        *btn1\_layout*.setVisibility(View.*GONE*);  
                    if(*questions*.get(*x*).getAns2() == "")  
                        *btn2\_layout*.setVisibility(View.*GONE*);  
                    if(*questions*.get(*x*).getAns3() == "" )  
                        *btn3\_layout*.setVisibility(View.*GONE*);  
                    if(*questions*.get(*x*).getAns4() == "")  
                        *btn4\_layout*.setVisibility(View.*GONE*);  
                        *text\_btn1*.setText(*questions*.get(*x*).getAns1());  
                        *text\_btn2*.setText(*questions*.get(*x*).getAns2());  
                        *text\_btn3*.setText(*questions*.get(*x*).getAns3());  
                        *text\_btn4*.setText(*questions*.get(*x*).getAns4());  
  
                    *correct* = *questions*.get(*x*).getCorrectAns();  
                    Log.e("TAG :correct",*correct*);  
                    *x*++;  
                }else {  
                    finish\_answers();  
                }  
            }  
        }, 500);  
    }  
  
    @Override  
    public void onClick(View v) {  
        switch (v.getId()){  
            case R.id.*btn1*:  
                if(*text\_btn1*.getText().toString().equals(*correct*)){  
                    *degree*= *degree* + *points\_number*;  
                    *correct\_ans*++;  
                }  
                quesion\_setup();  
                break;  
            case R.id.*btn2*:  
                if(*text\_btn2*.getText().toString().equals(*correct*)){  
                    *degree*= *degree* + *points\_number*;  
                    *correct\_ans*++;  
                }  
                quesion\_setup();  
                break;  
            case R.id.*btn3*:  
                if(*text\_btn3*.getText().toString().equals(*correct*)){  
                    *degree*= *degree* + *points\_number*;  
                    *correct\_ans*++;  
                }  
                quesion\_setup();  
                break;  
            case R.id.*btn4*:  
                if(*text\_btn4*.getText().toString().equals(*correct*)){  
                    *degree*= *degree* + *points\_number*;  
                    *correct\_ans*++;  
                }  
                quesion\_setup();  
              break;  
  
        }  
    }  
  
    private void counter() {  
        *countDownTimer* = new CountDownTimer(*exam\_duration*\*100000, 1000) {  
            public void onTick(long millisUntilFinished) {  
                *points*.setText(String.valueOf(millisUntilFinished / 1000));  
            }  
            public void onFinish() {  
                finish\_answers();  
            }  
        }.start();  
    }  
  
    private void finish\_answers() {  
        *total\_score*.setText(*total\_questions\_points* + "/"+*degree*);  
        *result\_layout*.setVisibility(View.*VISIBLE*);  
        *countDownTimer*.cancel();  
        publish\_answer();  
        new Thread(new Runnable() {  
            @Override  
            public void run() {  
                AddProducts();  
            }  
        }).start();  
    }  
  
    private void publish\_answer() {  
  
        Retrofit retrofit = new Retrofit.Builder()  
                .baseUrl(*ROOT\_URL*)  
                .addConverterFactory(GsonConverterFactory.create())  
                .build();  
        RegisterAPI api = retrofit.create(RegisterAPI.class);  
        api.publish(  
                *degree*,  
                *correct\_ans*,  
                (*number\_of\_questions*-*correct\_ans*),  
               *student\_id*,  
                5).enqueue(new Callback<ResponseBody>() {  
            @Override  
            public void onResponse(Call<ResponseBody> call, Response<ResponseBody> response) {  
                try {  
                    Toast.makeText(*mContext*, response.body().string(), Toast.*LENGTH\_SHORT*).show();  
                } catch (IOException e) {  
                    e.printStackTrace();  
                }  
            }  
            @Override  
            public void onFailure(Call<ResponseBody> call, Throwable t) {  
                Log.e("TAG", "insertUser():: Unable to submit post to API.");  
            }  
        });  
    }  
  
**Local Database**

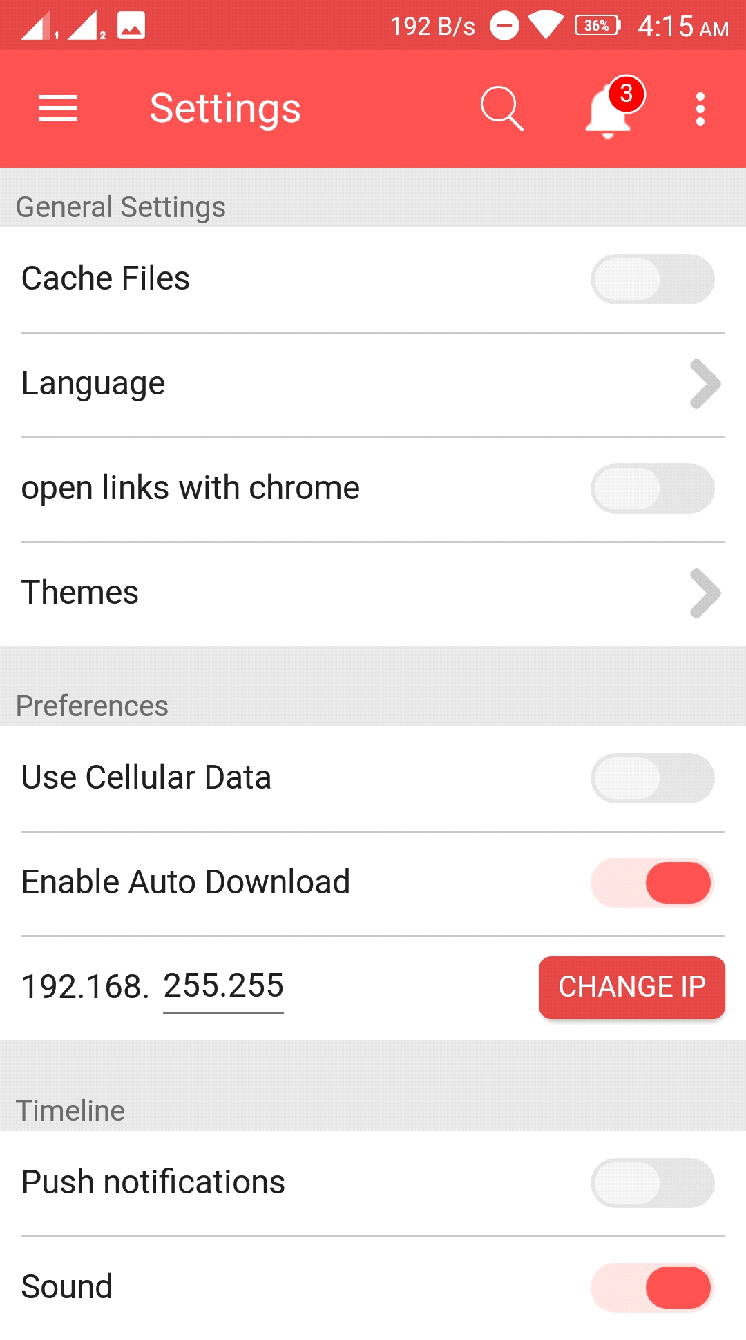
i have used a local database (**Room**) in android app to store the results locally on the devics.



   List<Product> *list* ;  
    private void AddProducts() {  
        *list* = new ArrayList<>();  
  
        Product product = new Product();  
        product.setQuestion(*questions*.get(0).getQuestion());  
        product.setAns1(*questions*.get(0).getAns1());  
        product.setAns2(*questions*.get(0).getAns2());  
        product.setAns3(*questions*.get(0).getAns3());  
        product.setAns4(*questions*.get(0).getAns4());  
        product.setCorrect\_ans(*questions*.get(0).getCorrectAns());  
        product.setExam\_start\_date(*exam\_start\_date*);  
        product.setDegree(String.valueOf(*degree*));  
        product.setTotal\_correct\_answers(*correct\_ans*);  
        product.setExam\_name(*exam\_name2*);  
  
        Log.e("TAG",product.getQuestion().toString());  
        *list*.add(product);  
        // insert product list into database  
        MainActivity.get().getDB().productDao().insertAll(*list*);  
        // disable flag for force update  
        MainActivity.get().setForceUpdate(false);  
    }  
  
}

**==============================**

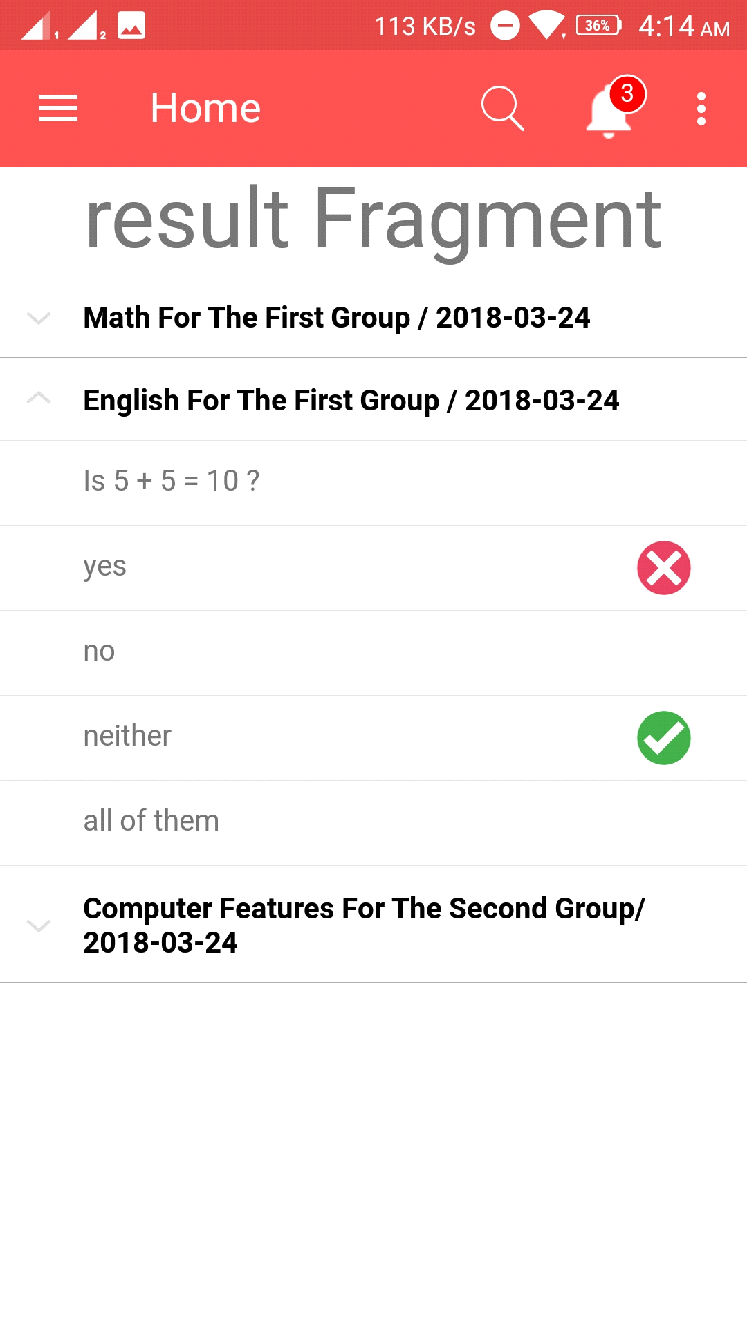
**Settings Fragment**



public class SettingsFragment extends Fragment implements CompoundButton.OnCheckedChangeListener,View.OnClickListener {  
  
    Button *clearcach*,*button\_ip*;  
    SharedPreferences.Editor *editor*;  
    Context *mContext*;  
    SwitchButton *sb1*,*sb3*,*sb4*,*sb5*,*sb6*,*sb7*,*sb8*;  
    Vibrator *vibe*;  
    RelativeLayout *terms*,*policy*,*Themes*,*Language*;  
    EditText *edittext*\_ip;  
  
    @Override  
    public View onCreateView(LayoutInflater inflater, ViewGroup container,  
                             Bundle savedInstanceState) {  
        *mContext* = this.getActivity();  
        final View view = inflater.inflate(R.layout.*settings\_fragment*, container, false);  
        *mContext* = this.getActivity();  
  
  
        *vibe* = (Vibrator)*mContext*.getSystemService(Context.*VIBRATOR\_SERVICE*);  
        *clearcach* =  view.findViewById(R.id.*clearcach*);  
        *terms* = view.findViewById(R.id.*terms*);  
        *policy* = view.findViewById(R.id.*policy*);  
        *Themes* = view.findViewById(R.id.*Themes*);  
        *Language* = view.findViewById(R.id.*Language*);  
        *edittext\_ip* = view.findViewById(R.id.*edittext\_ip*);  
        *button\_ip* = view.findViewById(R.id.*button\_ip*);  
  
  
//====================================================================  
        MainActivity.*CURRENT\_TAG* =*TAG\_SETTINGS*;  
        *navItemIndex* = 4;  
        MainActivity.*navigationView*.getMenu().getItem(4).setChecked(true);  
        ((AppCompatActivity)getActivity()).getSupportActionBar().setTitle(*activityTitles*[4]);  
//====================================================================  
  
  
        *sb1*=  view.findViewById(R.id.*sb\_default\_measure1*);  
        *sb3* =  view.findViewById(R.id.*sb\_default\_measure3*);  
        *sb4* =  view.findViewById(R.id.*sb\_default\_measure4*);  
        *sb5* =  view.findViewById(R.id.*sb\_default\_measure5*);  
        *sb6* =  view.findViewById(R.id.*sb\_default\_measure6*);  
        *sb7* =  view.findViewById(R.id.*sb\_default\_measure7*);  
        *sb8* =  view.findViewById(R.id.*sb\_default\_measure8*);  
  
  
  
  
        *editor* = *mContext*.getSharedPreferences(*MY\_PREFS\_NAME*, *MODE\_PRIVATE*).edit();  
        *clearcach*.setOnClickListener(new View.OnClickListener() {  
            @Override  
            public void onClick(View view) {  
                *editor*.clear().commit();  
                Toast.makeText(*mContext*,"Cache Cleared",Toast.*LENGTH\_SHORT*).show();  
  
            }  
        });  
  
  
        *edittext\_ip*.setOnEditorActionListener(new EditText.OnEditorActionListener() {  
            @Override  
            public boolean onEditorAction(TextView v, int actionId, KeyEvent event) {  
                if (actionId == EditorInfo.*IME\_ACTION\_DONE*) {  
                    Done(view);  
                    return true;  
                }  
                return false;  
            }  
        });  
        *sb1*.setOnCheckedChangeListener(this);  
        *sb3*.setOnCheckedChangeListener(this);  
        *sb4*.setOnCheckedChangeListener(this);  
        *sb5*.setOnCheckedChangeListener(this);  
        *sb6*.setOnCheckedChangeListener(this);  
        *sb7*.setOnCheckedChangeListener(this);  
        *sb8*.setOnCheckedChangeListener(this);  
  
        *terms*.setOnClickListener(this);  
        *policy*.setOnClickListener(this);  
        *Themes*.setOnClickListener(this);  
        *Language*.setOnClickListener(this);  
        *button\_ip*.setOnClickListener(this);  
        return view;  
    }  
  
  
  
    @Override  
    public void onCheckedChanged(final CompoundButton compoundButton, boolean b) {  
        new Handler().postDelayed(new Runnable() {  
            @Override  
            public void run() {  
                if(compoundButton.isChecked()){  
                    compoundButton.setChecked(false);  
                    Toast.makeText(*mContext*, "Go *Preimum* first", Toast.*LENGTH\_SHORT*).show();  
                    *vibe*.vibrate(100);  
                }  
            }  
        }, 500);  
  
    }  
  
    @Override  
    public void onClick(View view) {  
        if(view == *policy*){  
            startActivity(new Intent(*mContext*, PrivacyPolicyActivity.class));  
            getActivity().overridePendingTransition(R.anim.*fade\_in*, R.anim.*fade\_out*);  
        }else if(view==*Themes*||view==*Language*){  
            Toast.makeText(*mContext*, "Go *Preimum* first", Toast.*LENGTH\_SHORT*).show();  
            *vibe*.vibrate(100);  
        }else if(view == *button\_ip*){  
            Done(view);  
        }   else {  
            startActivity(new Intent(*mContext*, TermsConditions.class));  
            getActivity().overridePendingTransition(R.anim.*fade\_in*, R.anim.*fade\_out*);  
        }  
    }  
  
    private void Done(View view) {  
        *editor*.putString("ip", "[http://192.168.](about:blank)"+ *edittext\_ip*.getText().toString()+ "/" );  
        *editor*.apply();  
        View view2 = getActivity().getCurrentFocus();  
        if (view2 != null) {  
            InputMethodManager imm = (InputMethodManager)*mContext*.getSystemService(Context.*INPUT\_METHOD\_SERVICE*);  
            imm.hideSoftInputFromWindow(view.getWindowToken(), 0);  
        }  
        Toast.makeText(*mContext*, "Successfully Changed", Toast.*LENGTH\_SHORT*).show();  
    }  
}

**==============================**

**Results Fragment**



public class ResultFragment extends Fragment {  
  
  
    Context *mContext*;  
    RecyclerView *recyclerView*;  
    ExpandableListView *expandableListView*;  
    ExpandableListAdapter *expandableListAdapter*;  
    List<String> *expandableListTitle*;  
    HashMap<String, List<String>> *expandableListDetail*;  
  
  
  
  
  
    @Override  
    public View onCreateView(LayoutInflater inflater, ViewGroup container,  
                             Bundle savedInstanceState) {  
        // Inflate the layout for this fragment  
        View view = inflater.inflate(R.layout.*result\_fragment*, container, false);  
  
        *mContext* = this.getActivity();  
        *recyclerView* = (RecyclerView) view.findViewById(R.id.*recycler\_view*);  
  
  
  
        *expandableListView* = (ExpandableListView) view.findViewById(R.id.*expandableListView*);  
        *expandableListDetail* = ExpandableListDataPump.getData();  
        *expandableListTitle* = new ArrayList<String>(*expandableListDetail*.keySet());  
        *expandableListAdapter* = new CustomExpandableListAdapter(*mContext*, *expandableListTitle*, *expandableListDetail*);  
        *expandableListView*.setAdapter(*expandableListAdapter*);  
        *expandableListView*.setOnGroupExpandListener(new ExpandableListView.OnGroupExpandListener() {  
  
            @Override  
            public void onGroupExpand(int groupPosition) {  
                Toast.makeText(*mContext*,  
                        *expandableListTitle*.get(groupPosition) + " List Expanded.",  
                        Toast.*LENGTH\_SHORT*).show();  
            }  
        });  
  
        *expandableListView*.setOnGroupCollapseListener(new ExpandableListView.OnGroupCollapseListener() {  
  
            @Override  
            public void onGroupCollapse(int groupPosition) {  
                Toast.makeText(*mContext*,  
                        *expandableListTitle*.get(groupPosition) + " List Collapsed.",  
                        Toast.*LENGTH\_SHORT*).show();  
  
            }  
        });  
  
        *expandableListView*.setOnChildClickListener(new ExpandableListView.OnChildClickListener() {  
            @Override  
            public boolean onChildClick(ExpandableListView parent, View v,  
                                        int groupPosition, int childPosition, long id) {  
                Toast.makeText(  
                        *mContext*,  
                        *expandableListTitle*.get(groupPosition)  
                                + " -> "  
                                + *expandableListDetail*.get(  
                                *expandableListTitle*.get(groupPosition)).get(  
                                childPosition), Toast.*LENGTH\_SHORT*).show();  
                return false;  
            }  
        });  
  
  
        // run the sentence in a new thread  
        new Thread(new Runnable() {  
            @Override  
            public void run() {  
                List<Product> products = MainActivity.get().getDB().productDao().getAll();  
                boolean force = MainActivity.get().isForceUpdate();  
                retrieveProducts();  
  
                /\*if (force || products.isEmpty()) {  
                    retrieveProducts();  
                } else {  
                    populateProducts(products);  
                }\*/  
  
            }  
        }).start();  
        return view;  
    }  
  
    List<Product> *list* ;  
  
  
    private void retrieveProducts() {  
        /\*list = new ArrayList<>();  
  
        for (int i = 0; i < 10; i++) {  
            Product product = new Product();  
            product.setName(getString(R.string.name\_format, String.valueOf(i)));  
            product.setImageUrl("https://picsum.photos/500/500?image=" + i);  
            product.setPrice(i == 0 ? 50 : i \* 100);  
            Log.e("TAG",product.getImageUrl().toString());  
            list.add(product);  
        }  
  
        // insert product list into database  
        MainActivity.get().getDB().productDao().insertAll(list);\*/  
  
        // disable flag for force update  
        MainActivity.get().setForceUpdate(false);  
        List<Product> list2 = new ArrayList<>();  
        list2 = MainActivity.get().getDB().productDao().getAll();  
        populateProducts(list2);  
    }  
  
    private void populateProducts(final List<Product> products) {  
        getActivity().runOnUiThread(new Runnable() {  
            @Override  
            public void run() {  
                Log.e("TAG","Set *adabpter* ================================");  
                LinearLayoutManager llm = new LinearLayoutManager(*mContext*);  
                *recyclerView*.setLayoutManager(llm);  
                *recyclerView*.setAdapter(new ProductAdapter(products));  
            }  
        });  
    }  
  
}

**==============================**

**gradle dependencies**

android {  
    compileSdkVersion 27  
    defaultConfig {  
        applicationId "com.vacuum.app.metquiz"  
        minSdkVersion 19  
        targetSdkVersion 27  
        versionCode 1  
        versionName "1.0"  
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"  
    }  
    buildTypes {  
        release {  
            minifyEnabled false  
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'  
        }  
    }  
}  
  
dependencies {  
  
    def supportVersion = "27.0.2"  
    def roomVersion = "1.1.0"  
  
    implementation fileTree(include: ['\*.jar'], dir: 'libs')  
    implementation "com.android.support:appcompat-v7:$supportVersion"  
    implementation "com.android.support:design:$supportVersion"  
    implementation "com.android.support:recyclerview-v7:$supportVersion"  
    implementation "com.android.support:cardview-v7:$supportVersion"  
    implementation 'com.android.support.constraint:constraint-layout:1.0.2'  
    testImplementation 'junit:junit:4.12'  
    androidTestImplementation 'com.android.support.test:runner:1.0.1'  
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.1'  
    implementation 'org.jsoup:jsoup:1.10.3'  
    implementation 'com.github.bumptech.glide:glide:4.2.0'  
    annotationProcessor 'com.github.bumptech.glide:compiler:4.2.0'  
    implementation 'com.kyleduo.switchbutton:library:2.0.0'  
    implementation 'uk.co.chrisjenx:calligraphy:2.3.0'  
  
    implementation 'com.squareup.picasso:picasso:2.3.2'  
    implementation 'com.nineoldandroids:library:2.4.0'  
    implementation 'com.daimajia.slider:library:1.1.5@aar'  
    implementation 'org.jsoup:jsoup:1.11.2'  
    implementation 'com.squareup.retrofit2:retrofit:2.4.0'  
  
    implementation 'com.squareup.retrofit2:converter-gson:2.4.0'  
    implementation 'me.dm7.barcodescanner:zbar:1.9.8'  
  
  
  
    // Room  
    implementation "android.arch.persistence.room:runtime:$roomVersion"  
    annotationProcessor "android.arch.persistence.room:compiler:$roomVersion"  
    implementation 'com.squareup.picasso:picasso:2.5.2'  
  
    /\*implementation "android.arch.persistence.room:runtime:$rootProject.roomVersion"  
    annotationProcessor "android.arch.persistence.room:compiler:$rootProject.roomVersion"  
    androidTestImplementation "android.arch.persistence.room:testing:$rootProject.roomVersion"  
  
// Lifecycle components  
    implementation "android.arch.lifecycle:extensions:$rootProject.archLifecycleVersion"  
    annotationProcessor "android.arch.lifecycle:compiler:$rootProject.archLifecycleVersion"\*/  
}

**==============================**

**Question Model(POJO)**

public class QuestionModel {  
  
    @SerializedName("questions\_id")  
    @Expose  
    private String *questionsId*;  
    @SerializedName("question")  
    @Expose  
    private String *question*;  
  
    @SerializedName("ans1")  
    @Expose  
    private String *ans1*;  
    @SerializedName("ans2")  
    @Expose  
    private String *ans2*;  
    @SerializedName("ans3")  
    @Expose  
    private String *ans3*;  
    @SerializedName("ans4")  
    @Expose  
    private String *ans4*;  
  
    @SerializedName("correct\_ans")  
    @Expose  
    private String *correctAns*;  
      
    public String getQuestionsId() {  
        return *questionsId*;  
    }  
  
    public void setQuestionsId(String questionsId) {  
        this.*questionsId* = questionsId;  
    }  
  
    public String getQuestion() {  
        return *question*;  
    }  
  
    public void setQuestion(String question) {  
        this.*question* = question;  
    }  
  
    public String getAns1() {  
        return *ans1*;  
    }  
  
    public void setAns1(String ans1) {  
        this.*ans1* = ans1;  
    }  
  
    public String getAns2() {  
        return *ans2*;  
    }  
  
    public void setAns2(String ans2) {  
        this.*ans2* = ans2;  
    }  
  
    public String getAns3() {  
        return *ans3*;  
    }  
  
    public void setAns3(String ans3) {  
        this.*ans3* = ans3;  
    }  
  
    public String getAns4() {  
        return *ans4*;  
    }  
  
    public void setAns4(String ans4) {  
        this.*ans4* = ans4;  
    }  
  
    public String getCorrectAns() {  
        return *correctAns*;  
    }  
  
    public void setCorrectAns(String correctAns) {  
        this.*correctAns* = correctAns;  
    }

**==============================**

**User Model(POJO)**

public class User {  
  
    @SerializedName("student\_id")  
    @Expose  
    private String *studentId*;  
    @SerializedName("card\_id")  
    @Expose  
    private String *cardId*;  
    @SerializedName("email")  
    @Expose  
    private String *email*;  
    @SerializedName("password")  
    @Expose  
    private String *password*;  
    @SerializedName("*fname*")  
    @Expose  
    private String *fname*;  
    @SerializedName("*lname*")  
    @Expose  
    private String *lname*;  
    @SerializedName("grade\_id")  
    @Expose  
    private String *gradeId*;  
    @SerializedName("division\_id")  
    @Expose  
    private String *divisionId*;  
  
    public String getStudentId() {  
        return *studentId*;  
    }  
  
    public void setStudentId(String studentId) {  
        this.*studentId* = studentId;  
    }  
  
    public String getCardId() {  
        return *cardId*;  
    }  
  
    public void setCardId(String cardId) {  
        this.*cardId* = cardId;  
    }  
  
    public String getEmail() {  
        return *email*;  
    }  
  
    public void setEmail(String email) {  
        this.*email* = email;  
    }  
  
    public String getPassword() {  
        return *password*;  
    }  
  
    public void setPassword(String password) {  
        this.*password* = password;  
    }  
  
    public String getFname() {  
        return *fname*;  
    }  
  
    public void setFname(String *fname*) {  
        this.*fname* = fname;  
    }  
  
    public String getLname() {  
        return *lname*;  
    }  
  
    public void setLname(String *lname*) {  
        this.*lname* = lname;  
    }  
  
    public String getGradeId() {  
        return *gradeId*;  
    }  
  
    public void setGradeId(String gradeId) {  
        this.*gradeId* = gradeId;  
    }  
  
    public String getDivisionId() {  
        return *divisionId*;  
    }  
  
    public void setDivisionId(String divisionId) {  
        this.*divisionId* = divisionId;  
    }

**#App Architecture**

This app follows Model-View-Presenter (MVP) architecture pattern. MVP is one of the widely used architectural patterns for android apps and offers the following advantages:

Readable and maintainable code

Clear separation of concerns

Modular code which provides high degree of decoupling

More testable code

Code which is fun to work with

That is exactly what we want. An android app that has clean, maintainable, testable and robust code.

Model-View-Presenter basics

Model-View-Presenter (MVP) is a derivation of Model-View-Controller (MVC) architectural pattern and is mostly used for building user interfaces. It consists of 3 components

Model - Responsible for maintaining state of the data in the application. It is usually an interface that acts as abstraction

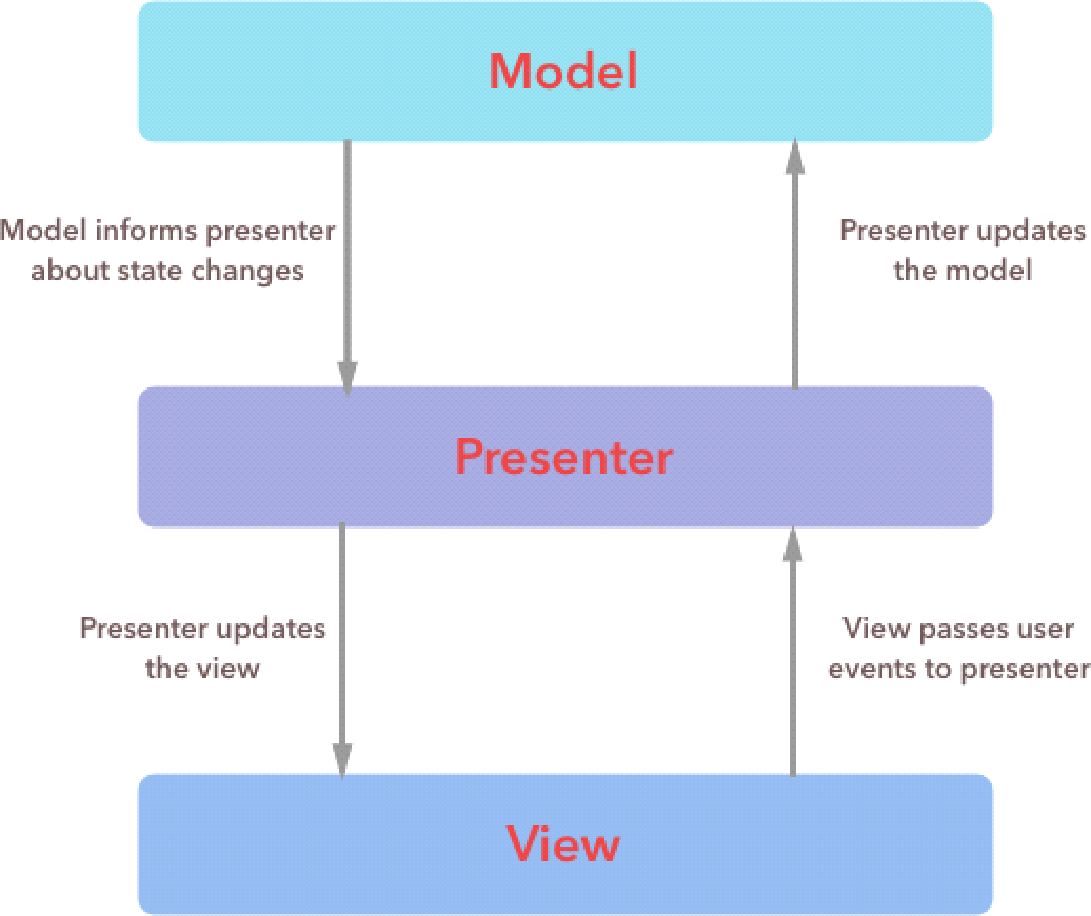
between actual data (from remote or local) and presenter. There is usually one model layer per android app.

Presenter - Acts as a bridge between Model and the View. Responsible for retrieving data from Model, formatting the data and passing it to the View. Ideally presenter should not contain any android dependencies to enable isolated unit testing. Presenter is also responsible for accepting actions from view and taking actions on model accordingly.

View - View is passive. View should be kept as dumb as possible. Meaning it should not contain any business logic and should act based on Presenter's instructions. Responsible for displaying data passed on by presenter and routing user actions to presenter.

View is kept independent of business logic. Basically view acts as an obedient slave that follows Presenter's instruction.

MVP architecture



Model, View and Presenter communicate with each other via interfaces to keep implementation logic separate. This makes the components loosely coupled and makes the code maintainable. Testing each separate layer becomes very easy with MVP, as the components talk to each other via interfaces. To test View, presenter can be mocked, to test View and Presenter, data or model layer can be mocked. Yes, it takes a little extra effort to implement this in apps, but the benefits we get surpasses the effort required by a huge margin.

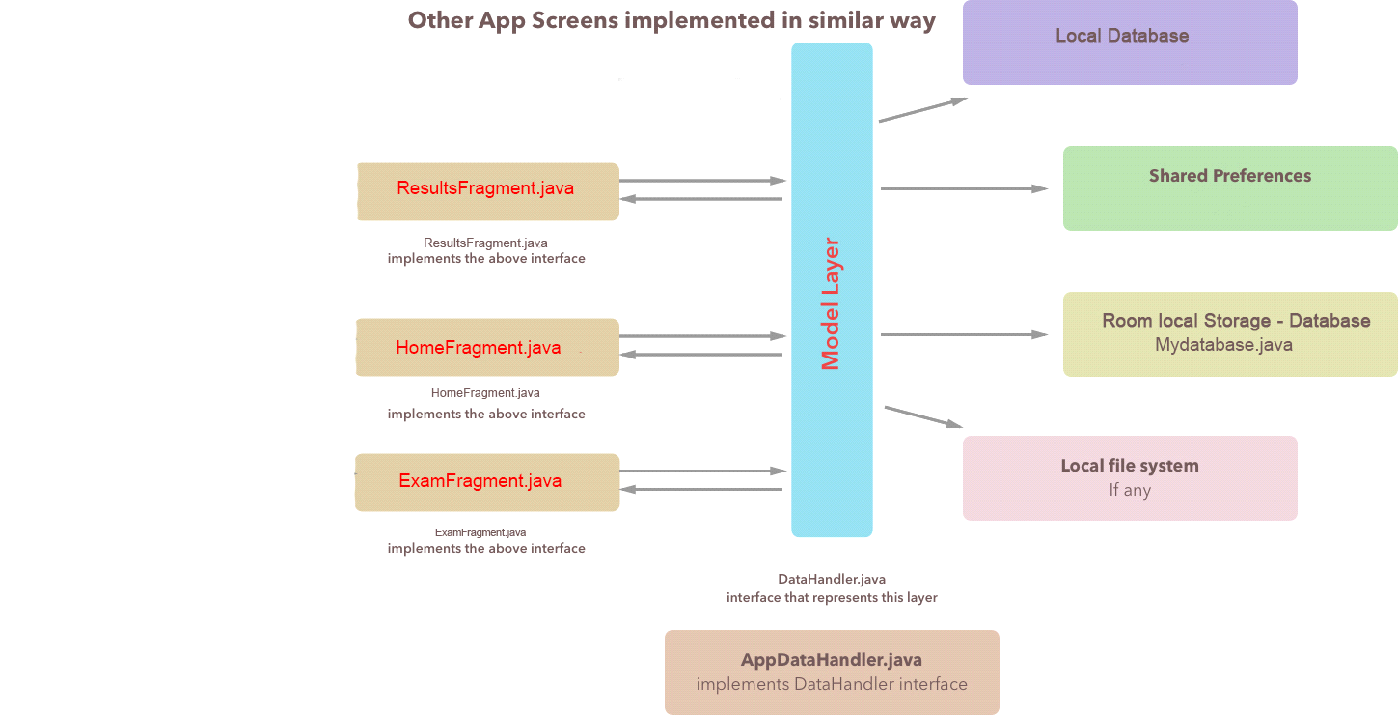
Implementation

In Android, activities and fragments are Views. Activity and fragment implement the View interface and contain a reference to Presenter. Presenter instructs the view what data to be displayed based on interaction with Model. Many a times Model is confused with Model class or POJOs, Model signifies model layer, whereas Model class and POJOs are simple classes that represent the structure of the data.

Model - In this app, we have a single interface that represents data or Model Layer (DataHandler.java). This is the single point of interaction for the layer. The implementation of this layer (AppDataHandler.java) is responsible for communicating with various data sources like remote database, local SQLite database, Shared Preferences etc. This encapsulates all the app data and state behind the model layer.

View & Presenter - There is one view and one presenter interface per screen, there is one contract that keeps related interfaces together. For example, for Sign In there is a contract called SignInContract.java that has 2 nested interface View and Presenter. SignInActivity.java, implements SignInContract.View interface while SignInPresenter.java implements the SignInContract.Presenter interface.

Here is the visual representation of Quiz App Architecture



Data flow

As clear from the above picture, the state is hidden inside the Model layer. Data transfers from Model layer to Presenter and eventually View. Presenter should not care whether the data is coming from internet or from local cache, neither should View. This business logic (Where data should come from?), is a part of Model layer.

Backend

For quiz app we are using Local server as backend. User information, quizzes, results etc. are all part of database. None of the database dependencies propagates beyond model layer. So the View and the Presenter need not know where the data is coming from.  
  
**#Supported Devices**

\* android Kitkat 4.4.2 (API-19)

\* android Lollipop 5.0/5.1(API-21)

\* android Marshmallow (API-23)

\* android nougat 7.0 (API-24)

\* android tablets

**#Third-party**

\* Room Persistence Library V1.1.0

\* espresso-core:3.0.1 (<https://github.com/chiuki/espresso-samples>)

\* picasso 2.3.2 (<https://github.com/square/picasso>)

\* Calligraphy 2.3.0 (<https://github.com/chrisjenx/Calligraphy>)

\* SwitchButton 2.0.0 (<https://github.com/kyleduo/SwitchButton>)

\* Slider 1.1.5 (<https://github.com/daimajia/AndroidImageSlider>)

\* Retrofit 2.4.0 (<https://github.com/square/retrofit>)

\* Glide 4.2.0 (<https://github.com/bumptech/glide>)

\* Jsoup 1.10.3 (<https://github.com/jhy/jsoup>)

**#Updates log**

april 29, 2018

-updating Exam layers and layouts.

-make some improvements and enhancement.  
-upload results on the server.  
-make the android app repo on Github.com

mar 9, 2018  
     -add a local Database (Room).  
     -store the results and view them

Sep 26, 2017  
     -Fix some bugs.  
     -new Activitys new layouts.

Aug 29, 2017  
     -add advanced settings of the user to add ip etc..

    -add login&signup layouts/fragmnts.

Apr 4, 2014  
     Add exam timers and points.

**#License**

