BVUSDE APP

Android Based Application

A project

Submitted in partial fulfillment of the requirements for the

Award of degree of Bachelor of Computer Applications

2017-20

Submitted by: **Guided by:**

Rahul Baswal **Mr. Ajay Kumar**

BHARATI VIDYAPEETH DEEMED UNIVERSITY

Academic Study Center – BVUSDE, New Delhi

An ISO 9001:2008 Certified Institute

"A" Grade Accreditation by NAAC

STUDENT UNDERTAKING

I, Rahul Baswal have completed the Project titled “BVUSDE App” under the guidance of Mr. Ajay Kumar in the partial fulfillment of the requirement for the award of Degree of Bachelor of Computer Applications of Bharati Vidyapeeth University. This is an original piece of work and I have neither copied and nor submitted elsewhere.

RAHUL BASWAL

(Student’s Signature)

ACKNOWLEDGEMENT

It is a matter of great pleasure for me to undertake training from the online courses available to learn this skill. I take this opportunity with much pleasure to thank all the people who have helped me through the course and producing this report. I sincerely thank my teacher, Mr. Ajay Kumar, for his guidance, help and motivation. Apart from the subject of my course, I learnt a lot from him, which I am sure, will be useful in different stages of my life.

I would also like to express my gratitude to the all other members of my report advisory committee.

CONTENTS

Chapter-1

INTRODUCTION

1.1 PROJECT INTRODUCTION

The BVUSDE App is excellent app in the college management field. It manages section of the college like admission, message, courses, examination, about us, contact us etc.
The "BVUSDE App'' is a computerized college system. This app has been developed to form whole college system including admission, message, courses, examination etc. The proposed system will keep a record of counselling, examination, courses, admission and generation of report regarding the present exam marks. This project has pop-up-based dialog that will help in getting student information for the counselling process and admission and retrieving the information through various user-friendly navigational menu-driven events.

The purpose of the project entitled as "BVUSDE App" is to computerize the Front Office Management of college to develop app which is user friendly, simple, fast, and cost – effective. It deals with the collection of college information details, etc. Traditionally, it was done manually.
The main function of the app is to register and retrieve details as and when required, and also to access these details meaningfully System input contains courses, name, mobile no., email id; while system output is to get these counselling details.
Along with the app we also propose this project for the college, which will enable the organizations to project themselves on large scale and keep up to date with today's latest technology. Also, it provides a good efficiency and effectiveness to the organization.

1.2 NEED OF COMPUTERIZATION OF SYSTEM

Why is such an app required?
In BVUSDE App developer develop the app in that way the app may less noise full it is easy for students or other faculty. Student can easily get information and counselling at BVDU-SDE
Where all the courses are approved by UCG. Pop is on the main screen for admission counselling. This app provide ease for get the information on a click of navigation bar,

which take to page providing the information. This app decreases the gap between student queries and college counselling.

1.3 Proposed Software

Reduction in the time spent by the student to gather the information & to queries it in an organized manner. Also, reduction in the time for gathering reports related to various activity like examination, contact information, admission procedure. student can view their result on app with reduce of time management.

College are having thousands of students under various courses for student. It is difficult to take queries of all students. It will take lot of time to provide these types of information and required separate task. This is a project dealing with the computerization of services of College. This project deals with the providing information to the students, providing counselling to students.

1.4 Importance of the work

1.4.1 Advantages

The main advantage of the app is that it helps to improve the efficiency and quality of work. so that it can be enhance the fulfilment.

- Since the app is available on smart phone, hence it helps in minimizing the amount of paperwork done as compared to the earlier time. so that it may be led to money feature.
- As all the information is being in app, hence the time taken to provide information has been drastically reduced leading to productivity.
- It provides college information, contact information, admission procedure when needed.so in this case money, time save can be done.

1.4.2 Intangible Benefits

Intangible benefit of the app:

- It helps in getting better the work environment of a college.
- It helps in increasing the transparency in a college, where the report of student is clearly visible to him as well as to his faculty.
- It helps to make the student satisfied, which helps in getting the best from his Course details.
- It helps in getting accurate and timely result.
- Save time, effort & cost.

1.4.3 Enhanced Functionalities

- The system would be a totally generalized on the basis of all the functionalities could be module according to the needs and wants of the college.
- The system would be a secure one so as to make it more reliable and robustness against various risks on the login, because the authentication is done from the google firebase database.
- The student has the facility to check their exam status, which help them for easy working in a better manner.
- The student has a facility to view the report based on the PRN which being stored in the college database. This would help in providing the student record and its information.

Chapter-2

System Analysis

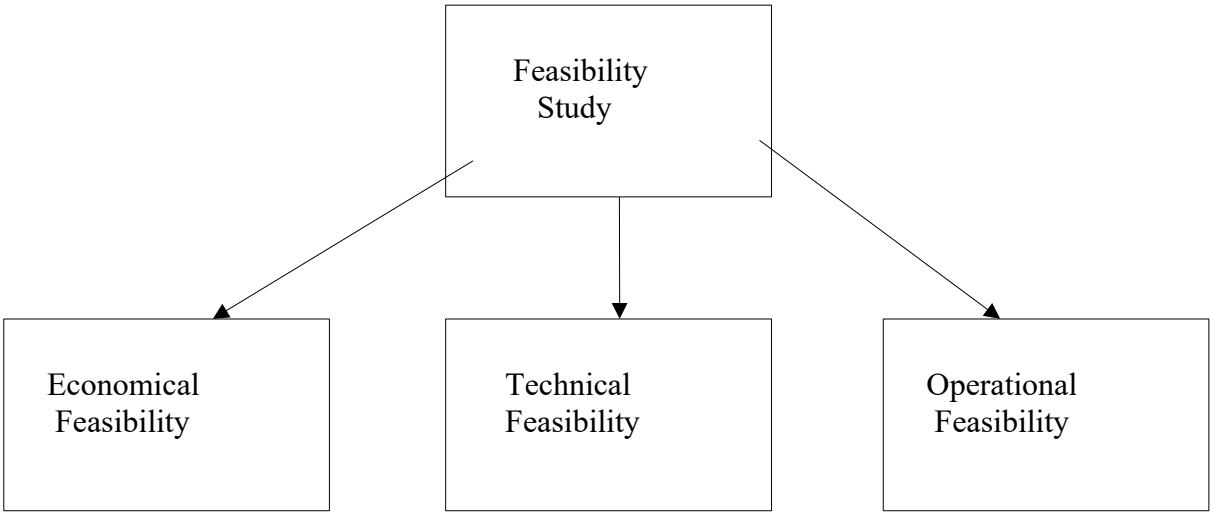
2.1 FEASIBILITY STUDY

A feasibility study determines whether the proposed solution is feasible based on the priorities of the requirements of the organization. A feasibility study culminates in a feasibility report that recommends a solution. It helps you to evaluate the cost-effectiveness of a proposed system.

The feasibility study is carried out to test if the proposed system is worth being implemented. Given unlimited resources and infinite time, all projects are feasible. After performing a Preliminary Investigation, gathering and interpreting data and details concerning the project, a Feasibility Check is done which involves a series of steps to check the Technical, Financial and Operational feasibilities.

During this phase, various solutions to the existing problems were examined. For each of these solutions the Cost and Benefits were the major criteria to be examined before deciding on any of the proposed systems. These Solutions would provide coverage of the following:

a) Specification of information to be made available by the system.
b) A clear-cut description of what tasks will be done manually and what needs to be handled by the automated system.
c) Specifications of new computing equipment needed.



A system that passes the feasibility tests is considered a feasible system. Let us see some feasible tests in my project.

2.1.1 TECHNICAL FEASIBILITY:

- It is related to the software and equipment specified in the design for implementing a new system. Technical feasibility is a study of function, performance and constraints that may affect the ability to achieve an acceptable system. During technical analysis, the analyst evaluates the technical merits of the system, at the same time collecting additional information about performance, reliability, maintainability and productivity. Technical feasibility is frequently the most difficult areas to assess.

- The main technical issue raised during feasibility is the existence of necessary technology and whether the proposed equipment has the capacity to hold required data. The technical guarantee of accuracy, reliability, ease and data were also investigated.

- Assessing System Performance: It involves ensuring that the system responds to user queries and is efficient, reliable, accurate and easy to use. Since we have the excellent network setup which is supported and excellent configuration of servers with 80 GB hard disk and 512 MB RAM, it satisfies the performance requirement.

- After the conducting the technical analysis we found that our project fulfills all the technical pre-requisites, the network environments if necessary are also adaptable according to the project.

2.1.2 ECONOMIC FEASIBILITY

- This feasibility has great importance as it can outweigh other feasibilities because costs affect organization decisions. The concept of Economic Feasibility deals with the fact that a system that can be developed and will be used on installation must be profitable for the Organization. The cost to conduct a full system investigation, the cost of hardware and

software, the benefits in the form of reduced expenditure are all discussed during the economic feasibility.

➢ Return on Investment:

- There will be revenue in terms of more Customer Subscriptions.
- There will be cost reduction in terms of maintaining huge amounts of paper records, stationary, humans.
- There will be tracking of the Subscribers from a centralized database.
- There will be awareness among not only the Subscribers, but general public regarding the good points of the issue.
- Subscriber satisfaction will lead to more upgrades and reduce the downgrades.

➢ Cost of No Change

- The cost will be in terms of utilization of resources leading to the cost to the company. Since our cost of project is our efforts, which is obviously less than the long-term gain for the company, the project should be made.

➢ Cost- benefit analysis:

- A cost-benefit analysis is necessary to determine economic feasibility. The primary objective of the cost benefit analysis is to find out whether it is economically worthwhile to invest in the project. If the returns on the investment are good, then the project is considered economically worthwhile.
- Cost benefit analysis is performed by first listing all the costs associated with the project cost which consists of both direct costs and indirect costs.
- Direct costs are those incurred by buying software, hiring people, cost of consumable items, rent for accommodation etc.
- Indirect costs include those involving time spent by user in discussing problems with system analysts, gathering data about problem etc.

2.1.3 OPERATIONAL FEASIBILITY

- Operation feasibility is a measure of how people feel about the system. Operational Feasibility criteria measure the urgency of the problem or the acceptability of a solution. Operational Feasibility is dependent upon determining human resources for the project. It refers to projecting whether the system will operate and be used once it is installed.

- If the ultimate users are comfortable with the present system and they see no problem with its continuance, then resistance to its operation will be zero.

- Behaviorally also the proposed system is feasible. A particular application may be technically and but may fail to produce the forecasted benefits, because the company is not able to get it to work. For the system, it is not necessary that the user must be a computer expert, but any computer operator given a little bit of knowledge and training can easily operate.

- Our Project is operationally feasible since there is no need for special training of staff member and whatever little instructing on this system is required can be done so quite easily and quickly as it is essentially.

- This project is being developed keeping in mind the general people who one has very little knowledge of computer operation, but can easily access their required database and other related information. The redundancies can be decreased to a large extent as the system will be fully auto.

2.2 CHOICE OF PLATFORMS

Operating System:	Windows 10
Language:	Java, Android Studio (IDE)
Database:	Google Firebase

CHAPTER 3

SYSTEM DESIGN

3.1 DESIGN METHODOLOGY

Every software development methodology approach acts as a basis for applying specific frameworks to develop and maintain software. Several software developments approaches have been used since the origin of information technology.

Broadly these are:

- Software development life cycle methodology
- Agile methodology

There are many models under these methodologies:

➢ **Software development life cycle:**

- Waterfall: a linear framework
- Spiral: a combined linear-iterative framework
- Incremental: a combined linear-iterative framework or V Model
- Prototyping: an iterative framework
- Rapid application development (RAD): an iterative framework

➢ **Agile methodology:**

- Scrum
- Extreme programming
- Adaptive software development.
- Dynamic system development method (DSDM)
- Waterfall development

The waterfall model is a sequential development approach, in which development is seen as flowing steadily downwards (like a waterfall) through the phases of requirements analysis, design, implementation, testing (validation), integration, and maintenance. The first formal description of the method is often cited as an article published by Winston W. Royce [3] in 1970 although Royce did not use the term "waterfall" in this article.

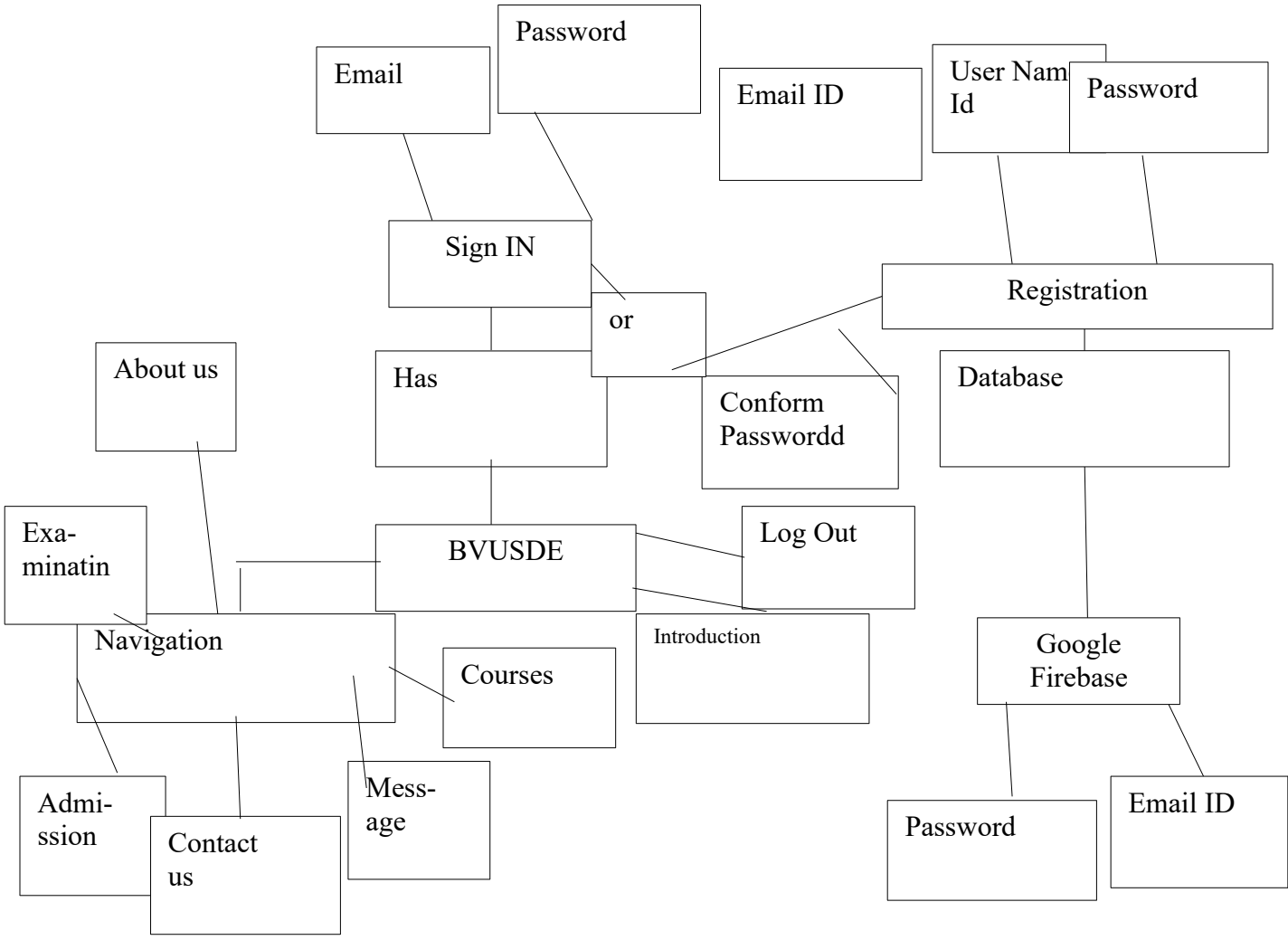
The basic principles are:

- Project is divided into sequential phases, with some overlap and splash back acceptable between phases.
- Emphasis is on planning, time schedules, target dates, budgets and implementation of an entire system at one time.
- Tight control is maintained over the life of the project via extensive written documentation, formal reviews, and approval/signoff by the user and information technology management occurring at the end of most phases before beginning the next phase.

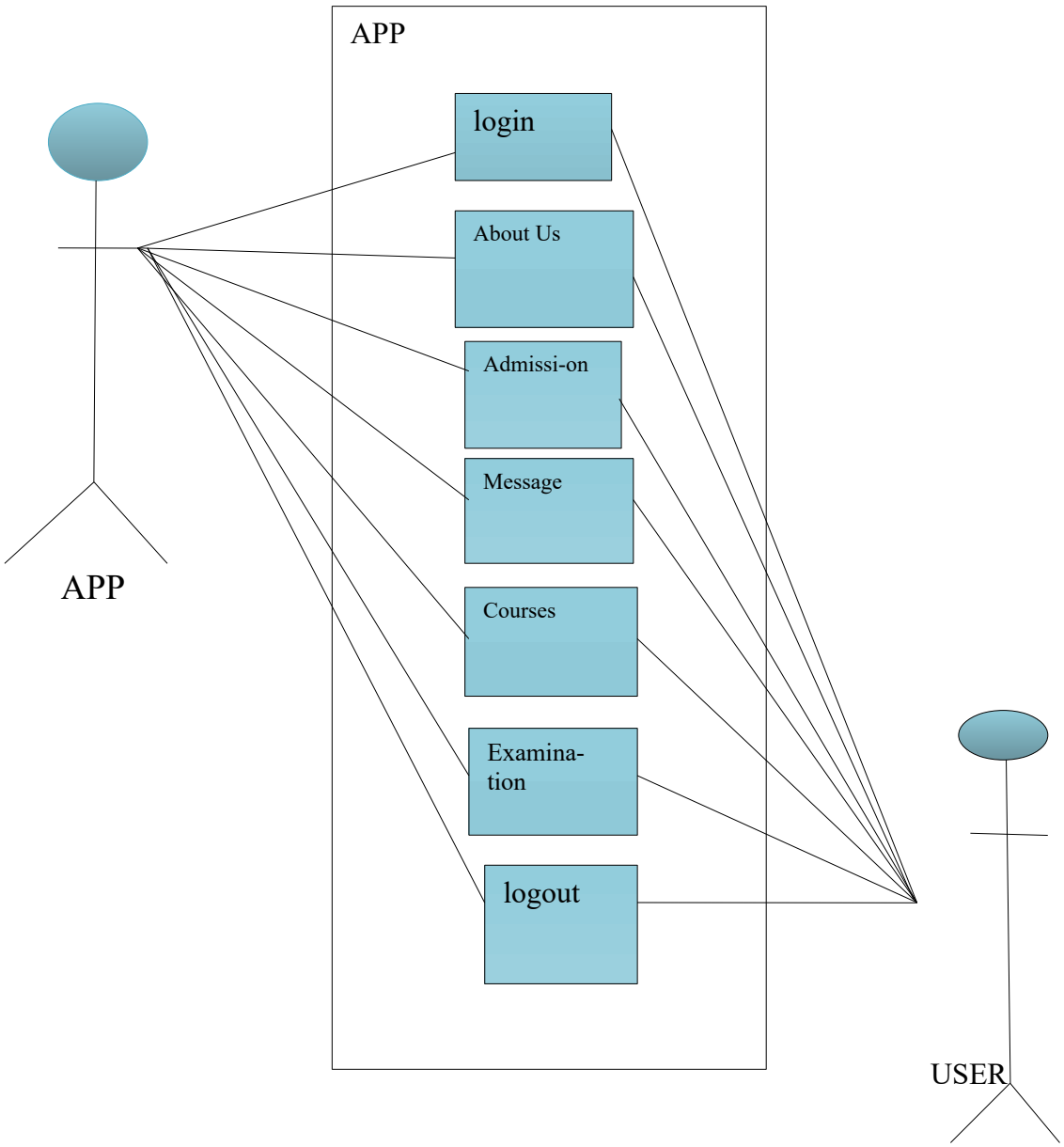
The Waterfall model is a traditional engineering approach applied to software engineering. It has been widely blamed for several large-scale government projects running over budget, over time and sometimes failing to deliver on requirements due to the Big Design Up Front approach. Except when contractually required, the Waterfall model has been largely superseded by more flexible and versatile methodologies developed specifically for software development.

3.2 DATABASE DESIGN

3.2.1 ER DIAGRAMS

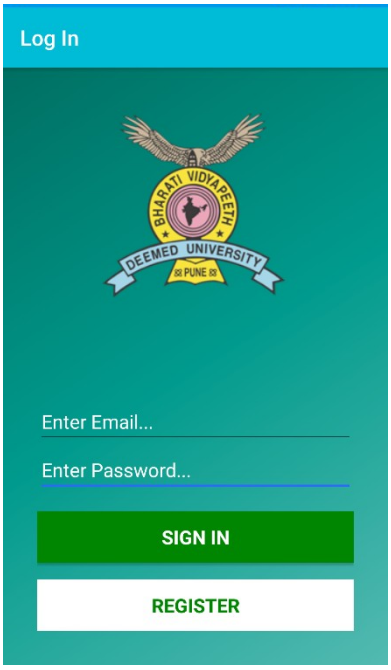


3.2.2 High Level Use Case Diagram

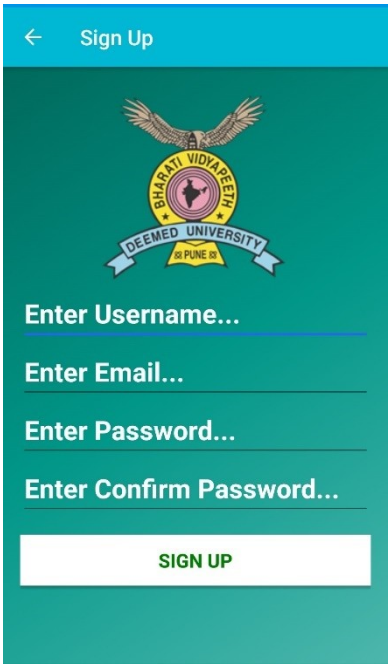


3.3 SCREEN DESIGN

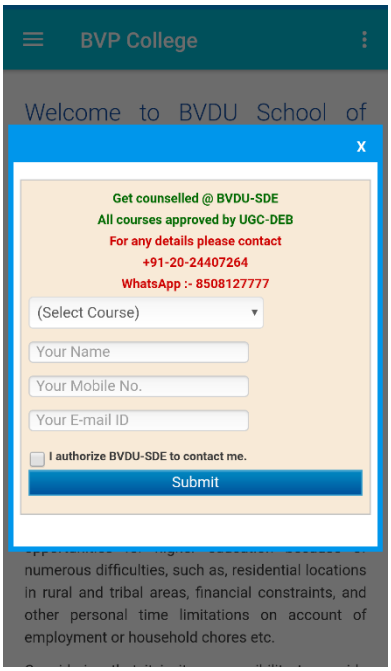
3.3.1 LOGIN FORM



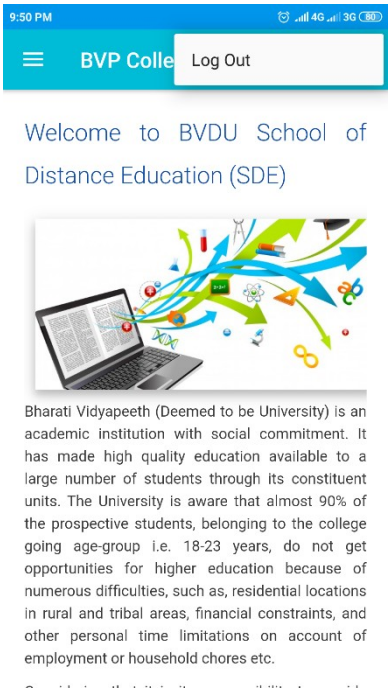
3.3.2 Registration Form



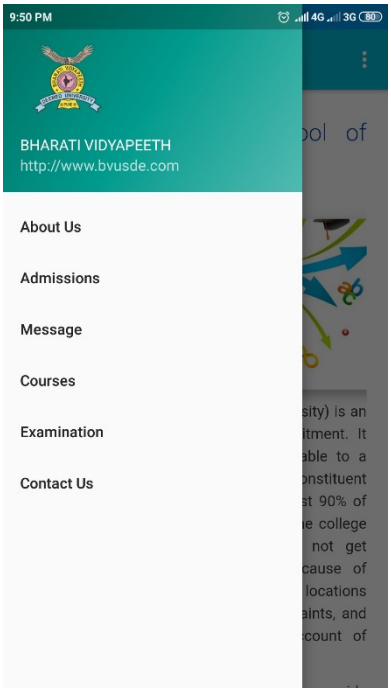
3.3.3 POP-UP



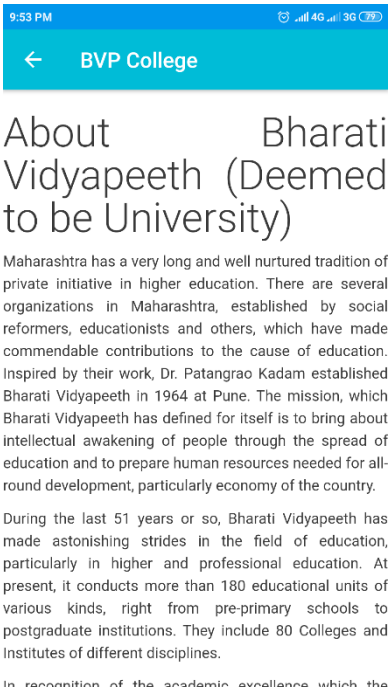
3.3.4 Main Screen



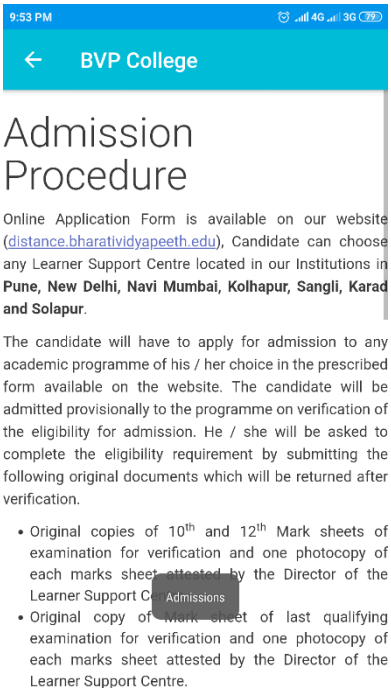
3.3.5 Navigational Panel



3.3.6 About US

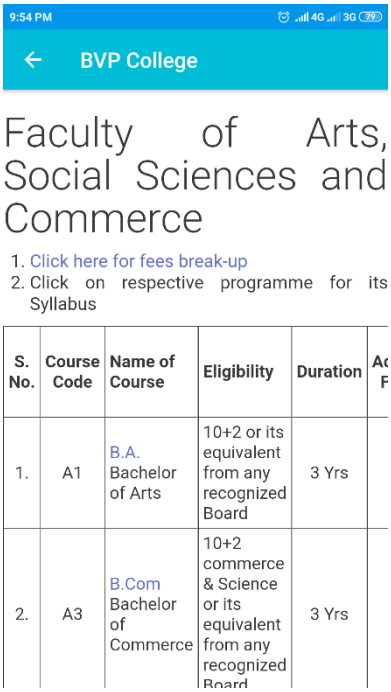


3.3.7 Admission

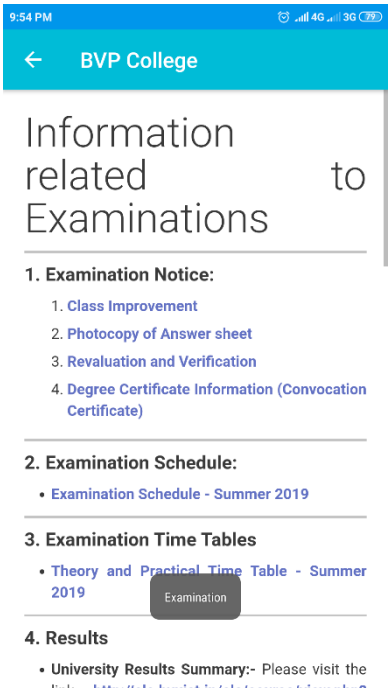


3.3.8 Message

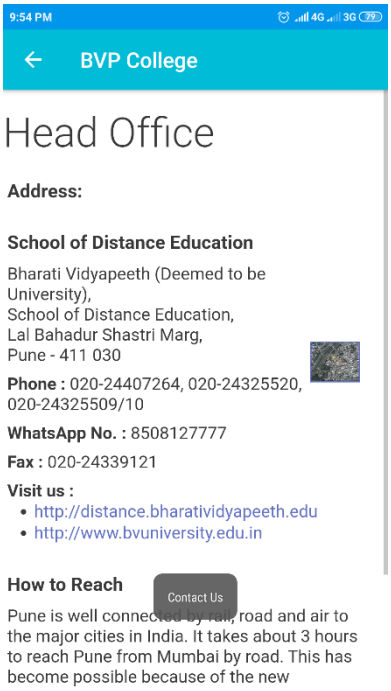
3.3.9 Courses



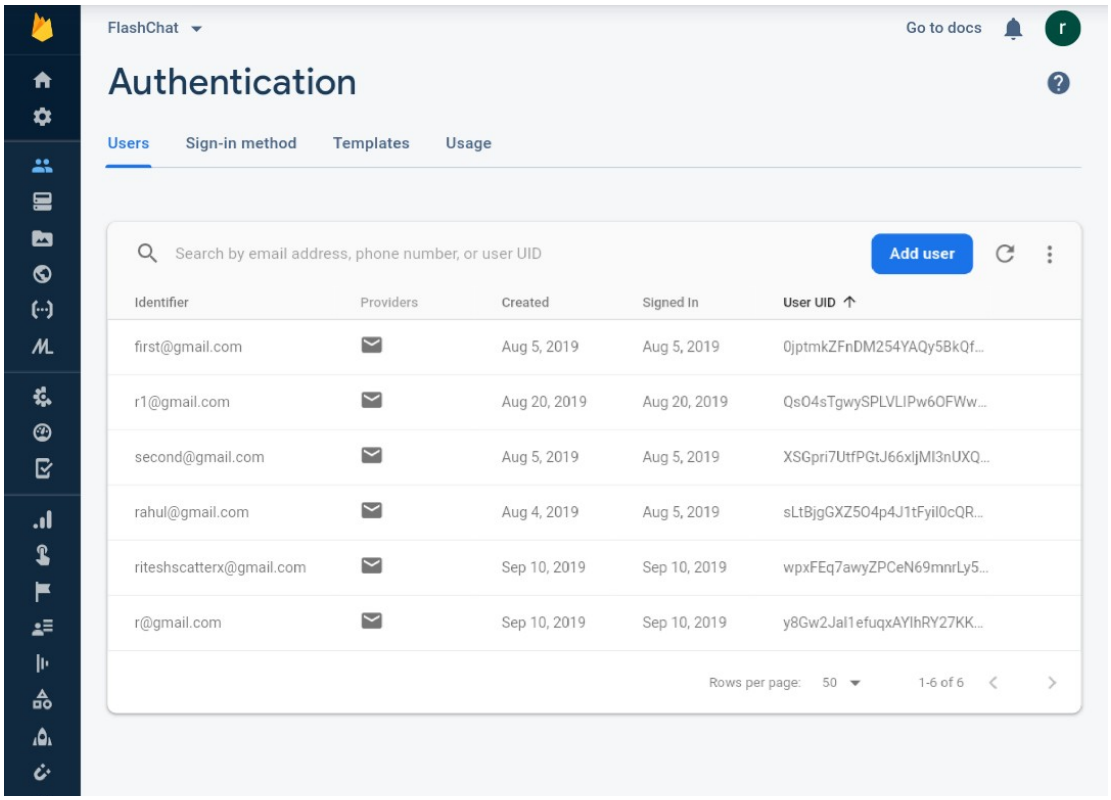
3.3.10 Examination



3.3.11 Contact US

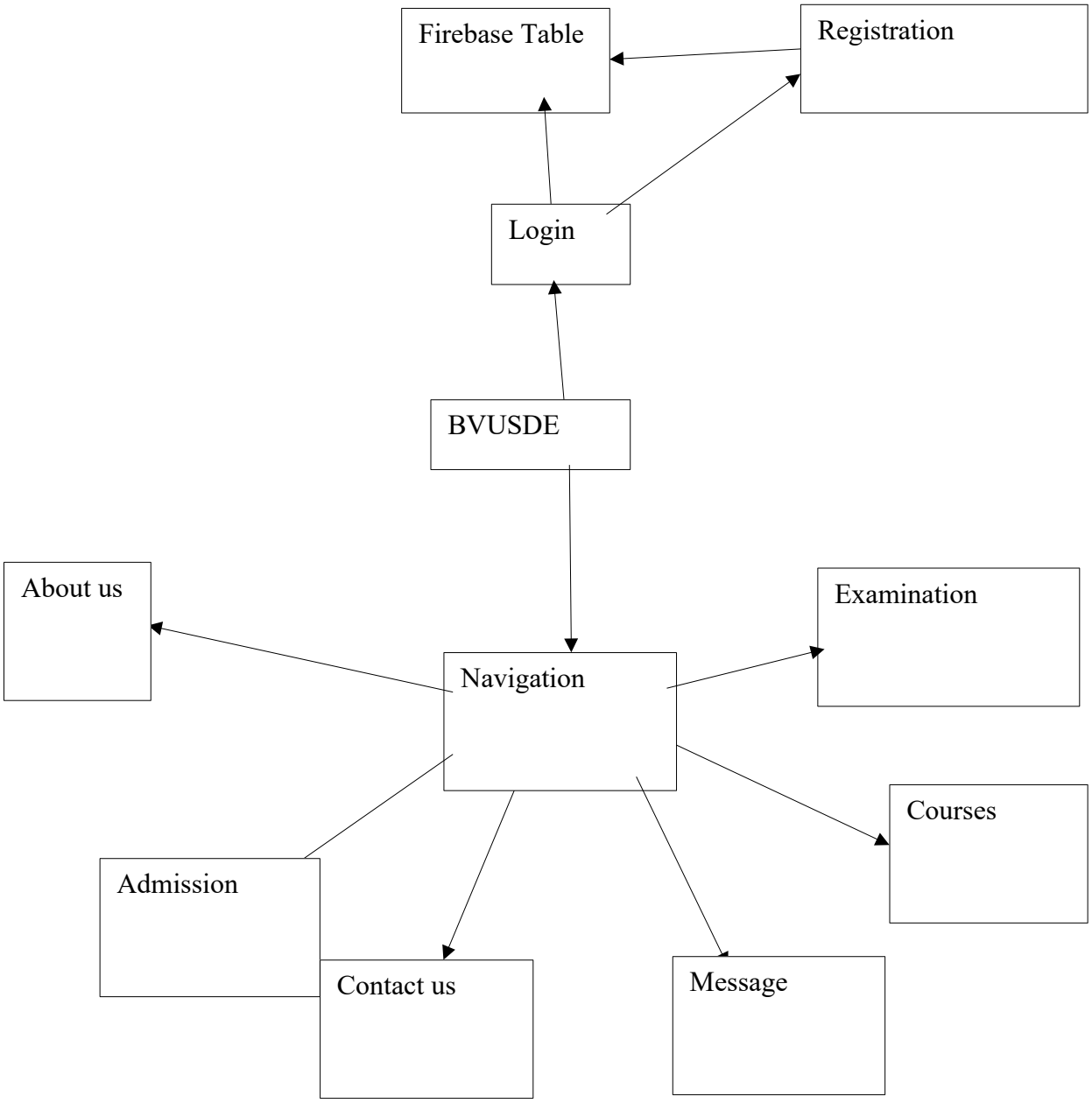

9:54 PM
BVP College
Head Office
Address:
School of Distance Education
Bharati Vidyapeeth (Deemed to be University),
School of Distance Education,
Lal Bahadur Shastri Marg,
Pune - 411 030
Phone : 020-24407264, 020-24325520, 020-24325509/10
WhatsApp No. : 8508127777
Fax : 020-24339121
Visit us :
http://distance.bharatividyapeeth.edu
http://www.bvuniversity.edu.in
How to Reach
Contact Us
Pune is well connected by rail, road and air to the major cities in India. It takes about 3 hours to reach Pune from Mumbai by road. This has become possible because of the new

3.4 TABLE


FlashChat
Go to docs
r
Authentication
Users
Sign-in method
Templates
Usage
Search by email address, phone number, or user UID
Add user
Identifier
Providers
Created
Signed In
User UID
first@gmail.com
Aug 5, 2019
Aug 5, 2019
0jptmkZFnDM254YAQy5BkQf...
r1@gmail.com
Aug 20, 2019
Aug 20, 2019
QsO4sTgwySPLVLIPw6OFWw...
second@gmail.com
Aug 5, 2019
Aug 5, 2019
XSGpri7UtfPGtJ66xljMI3nUXQ...
rahul@gmail.com
Aug 4, 2019
Aug 5, 2019
sLtBjgGXZ5O4p4J1tFyil0cQR...
riteshscatterx@gmail.com
Sep 10, 2019
Sep 10, 2019
wpxFEq7awyZPCeN69mnrLy5...
r@gmail.com
Sep 10, 2019
Sep 10, 2019
y8Gw2Jal1efuqxAYlhRY27KK...
Rows per page: 50
1-6 of 6

3.5 REPORT DESIGN

3.5.1 DATA FLOW DIAGRAM



CHAPTER 4:

SYSTEM TESTING & IMPLEMENTATION

4.1 TESTING

- All software intended for public consumption should receive some level of testing. Without testing, you have no assurance that software will behave as expected. The results in public environment can be truly embarrassing.
- Testing is a critical element of software quality assurance and represents the ultimate review of specification, designing, and coding. Testing is done throughout the system development at various stages. If this is not done, then the poorly tested system can fail after installation. Testing is a very important part of SDLC and takes approximately 50%of the time.
- The first step in testing is developing a test plan based on the product requirements. The test plan is usually a formal document that ensures that the product meets the following standards:

➢ Is thoroughly Tested- Untested code adds an unknown element to the product and increases the risk of product failure.

➢ Meets product requirements- To meet customer needs, the product must provide the features and behavior described in the product specification.

➢ Does not contain defects- Features must work within established quality standards and those standards should be clearly stated within the test plan.

Testing Techniques

4.1.1 Black box Testing

It aims to test a given program's behavior against its specification or component without making any reference to the internal structures of the program or the algorithms used. Therefore, the source code is not needed, and so even purchased modules can be tested. We study the system by examining its inputs and related outputs. The key is to devise inputs that have a higher likelihood of causing outputs that reveal the presence of defects. We use experience and knowledge of the domain to identify such test cases. Failing this a systematic approach may be necessary. Equivalence partitioning is where the input to a program falls into a number of classes. e.g. positive numbers vs. negative numbers. Programs normally behave the same way for each member of a class. Partitions exist for both input and output. Partitions may be discrete or overlap. Invalid data (i.e. outside the

normal partitions) is one for which partitions should be tested. Test cases are chosen to exercise each portion. Also test boundary cases (atypical, extreme, zero) should be considered since these frequently show up defects. For completeness, test all combinations of partitions. Black box testing is rarely exhaustive (because one doesn't test every value in an equivalence partition) and sometimes fails to reveal corruption defects caused by weird combination of inputs. Black box testing should not be used to try and reveal corruption defects caused, Example, by assigning a pointer to point to an object of the wrong type. Static inspection (or using a better programming language) is preferred.

4.1.2 White box Testing

It was used as an important primary testing approach. Code is tested using code scripts, drivers, stubs, etc. which are employed to directly interface with it and drive the code. The tester can analyze the code and use the knowledge about the structure of a component to derive test data. This testing is based on the knowledge of structure of component (e.g. by looking at source code). The advantage is that structure of code can be used to find out how many test cases needed to be performed. Knowledge of the algorithm (examination of the code) can be used to identify the equivalence partitions. Path testing is where the tester aims to exercise every independent execution path through the component. All conditional statements tested for both true and false cases. If a unit has no control statements, there will be up to 2n possible paths through it. This demonstrates that it is much easier to test small program units than large ones. Flow graphs are a pictorial representation of the paths of control through a program (ignoring assignments, procedure calls and I/O statements). We use a flow graph to design test cases that execute each path. Static tools may be used to make this easier in programs that have a complex branching structure. Dynamic program analyzers instrument a program with additional code. Typically, this will count how many times each statement is executed. At end, print out report showing which statements have and have not been executed.

Possible methods:

- Usual method is to ensure that every line of code is executed at least once.
- Test capabilities rather than components (e.g. concentrate on tests for data loss over ones for screen layout).
- Test old in preference to new (users less affected by failure of new capabilities).

- Test typical cases rather than boundary ones (ensure normal operation works properly).

Debugging: Debugging is a cycle of detection, location, repair and test. Debugging is a hypothesis testing process. When a bug is detected, the tester must form a hypothesis about the cause and location of the bug. Further examination of the execution of the program (possible including many returns of it) will usually take place to confirm the hypothesis. If the hypothesis is demonstrated to be incorrect, a new hypothesis must be formed. Debugging tools that show the state of the program are useful for this, but inserting print statements is often the only approach. Experienced debuggers use their knowledge of common and/or obscure bugs to facilitate the hypothesis testing process. After fixing a bug, the system must be reset to ensure that the fix has worked and that no other bugs have been introduced. In principle, all tests should be performed again but this is often too expensive to do.

4.2 TEST PLANNING

Testing needs to be planned to be cost and time effective. Planning is setting out standards for tests. Test plans set the context in which individual engineers can place their own work. Typical test plan contains:

- Overview of Testing Process.
- Recording procedures so that tests can be audited.
- Hardware and Software Requirements.
- Constraints.

4.3 OVERVIEW OF TESTING STRATEGIES

A strategy for software testing integrates test case design methods into a well-planned series of steps that result in the successful construction of software. It provides a road map for the software developer, the quality assurance organization and the customer- a road map that describes the steps to be conducted as part of testing, when these steps are planned and then undertaken, and how much effort, time and resources will be required. Therefore,

any testing strategy must incorporate test planning, test case design, test execution, and resultant data collection and evaluation.

Large system is usually tested using a mixture of strategies. Different strategies may be needed for different parts of the system or at a stage of the process.

4.3.1 Testing Strategies

Test Type	Description
Unit Test	Each independent piece of code works correctly.
Integration Test	All units work together without errors.
Interface Test	Usually done at integration stage when modules or sub-systems are combined. Objective is to detect errors or invalid assumptions about interfaces between modules. Reason these are not shown up in unit testing is that test case may perpetuate same incorrect assumption made by module designer. Particularly important when OO development has been used. Global variable) One places data there and the other retrieves it. Architectures.

Regression Test	Newly Added features do not introduce errors to other features that are already working.
Load Test (also called Stress Test)	The product continues to work under extreme usage. Test system's ability to cope with a specified load (e.g. transactions per second). Plan tests to increase load incrementally. Go beyond design limit until system fails (this test particularly important for distributed systems).
Platform Test	The product works on all the target hardware and software platforms.
Top Down Test	This approach tests high levels of system before detailed components. This is appropriate when developing the system top-down as it is likely to show up structural design errors early. Validation (as distinct from verification) can begin early. Its disadvantage is that stubs needs to be generated (extra effort) and might be impractical if component is complex (e.g. converting an array into a linked list; unrealistic to generate random list;

	therefore, end up implementing unit anyway). Test output may be difficult to observe (needs creation of artificial environment). This is not appropriate for OO systems (except within a class).
Bottom Up Test	This is opposite of top-down testing. This testing test low-level unit then works up hierarchy. Its advantages and disadvantages of bottom-up mirror those of top-down. In this testing there is need to write test drivers for each unit. These are as reusable as the unit itself. Combining top-down development with bottom-up testing means that all parts of system must be implemented before testing can begin, therefore does not accord with incremental approach discussed above.
Back to Back Test	Comparison of test results from different versions of the system (e.g. comparing the prototype with previous version or different configuration). The process

	involves running the first system, saving test case results. Then running the second system, also saving its results. Finally comparing the results files. It is important to note that no difference doesn't imply no bugs. Both systems may have made the same mistake.

Testing Done in our System

The best testing is to test each subsystem separately as we have done in our project. It is best to test a system during the implementation stage in form of small sub steps rather than large chunks. We have tested each module separately i.e. have completed unit testing first and system testing was done after combining /linking all different Modules with different menus and thorough testing was done. Once each lowest level unit has been tested, units are combined with related units and retested in combination. This proceeds hierarchically bottom-up until the entire system is tested as a whole. Hence, we have used the **Top Up** approach for testing our system.

Typical levels of testing in our system:

- Unit -procedure, function, method
- Module -package, abstract data type
- Sub-system - collection of related modules, method-message paths
- Acceptance Testing - whole system with real data (involve customer, user, etc.)

4.3.2 Beta Testing

It is acceptance testing with a single client. It is conducted at the developer's site by a customer. The software is used in a natural setting with the developer "looking over the shoulder" of the user and recording errors and usage problems. Conducted in a controlled environment. usually comes in after the completion of basic design of the program. The

project guide who looks over the program or other knowledgeable officials may make suggestions and give ideas to the designer for further improvement. They also report any minor or major problems and help in locating them and may further suggest ideas to get rid of them. Naturally a number of bugs are expected after the completion of a program and are most likely to be known to the developers only after the alpha testing.

involves distributing the system to potential customers to use and provide feedback. It is conducted at one or more customer sites by the end-user of the software. Unlike alpha testing, the developer is generally not present. Therefore, the beta test is a "live" application of the software in an environment that cannot be controlled by the developer. The customer records all problems (real or imagined) that are encountered during beta testing and reports these to the developer at regular intervals. As a result of problems reported during beta test, software engineers make modifications and then prepare for release of the software product to the entire customer base.

In, this project, this exposes system to situations and errors that might not be anticipated by us.

4.3.3 IMPLEMENTATION

In this project, we implemented a base model for Automated app using adaptive method. We emphasize that it is a base model and we can still do a lot of work to add more functions in order to improve its user friendliness. If this project can be integrated with the firebase database system, it will truly serve as an automated tool for college process.

4.4 SOFTWARE/HARDWARE SPECIFICATION

Hardware Specification

- Android API 16 and Above
- 56 MB RAM or More
- 2 GB Disk Space

Software Specification

- Android Studio (IDE)
- Java

DATABASE

Introduction to

Firebase

Introduction

A database is a collection of information that's related. Access allows you to manage your information in one database file.

• Tables store your data in your database

• Queries ask questions about information stored in your tables

• Forms allow you to view data stored in your tables

• Reports allow you to print data based on queries/tables that you have created

CHAPTER 5:

CONCLUSION

5.1 CONCLUSIONS

This project has been a rewarding experience in more than one way. The entire project work has enlightened us in the following areas.

- We have gained an insight into the working of the COLLEGE. This represents a typical real world situation.

- Our understanding of database design has been strengthened this is because in order to authenticate the user to database designing has to be properly followed.

- Scheduling a project and adhering to that schedule creates a strong sense of time management.

- Sense of teamwork has developed and confidence of handling real life project has increased to a great extent.

- Initially, there were problem with the validation but with discussions, we were to implement validations.

5.2 LIMITATION OF SYSTEM

- Log out done on client end
- Certain Tabs can be searched
- Required good internet connection

5.3 FUTURE SCOPE

- After adding some more useful modules in the project it can be used for many dynamic queries in the app.

- This project can be developed for online transaction of fee, by which any student can see its payment status anywhere.

- This project can be developed with centralized database so that data storage and backup services will be easy.

5.4 BIBLIOGRAPHY

- Herbert Scheldt, Java Complete Reference, Fifth Edition, Tata McGraw Hill Edition.

- Phil Hanna, JSP 2.0: The Complete Reference, Tata McGraw Hill Edition, 2003.

- Elmarsi and Navathe, Fundamentals of Database System (Third Edition), Addison Wesley.

- Ian Somerville, Software Engineering, Third Edition, Pearson Education.

- Ali Bahrami, Object-Oriented System Development, Third Edition, Tata McGraw Hill Edition.

- Ivan Bayross, SQL, PL/SQL programming language of Oracle, Second Edition, BPB Publication.

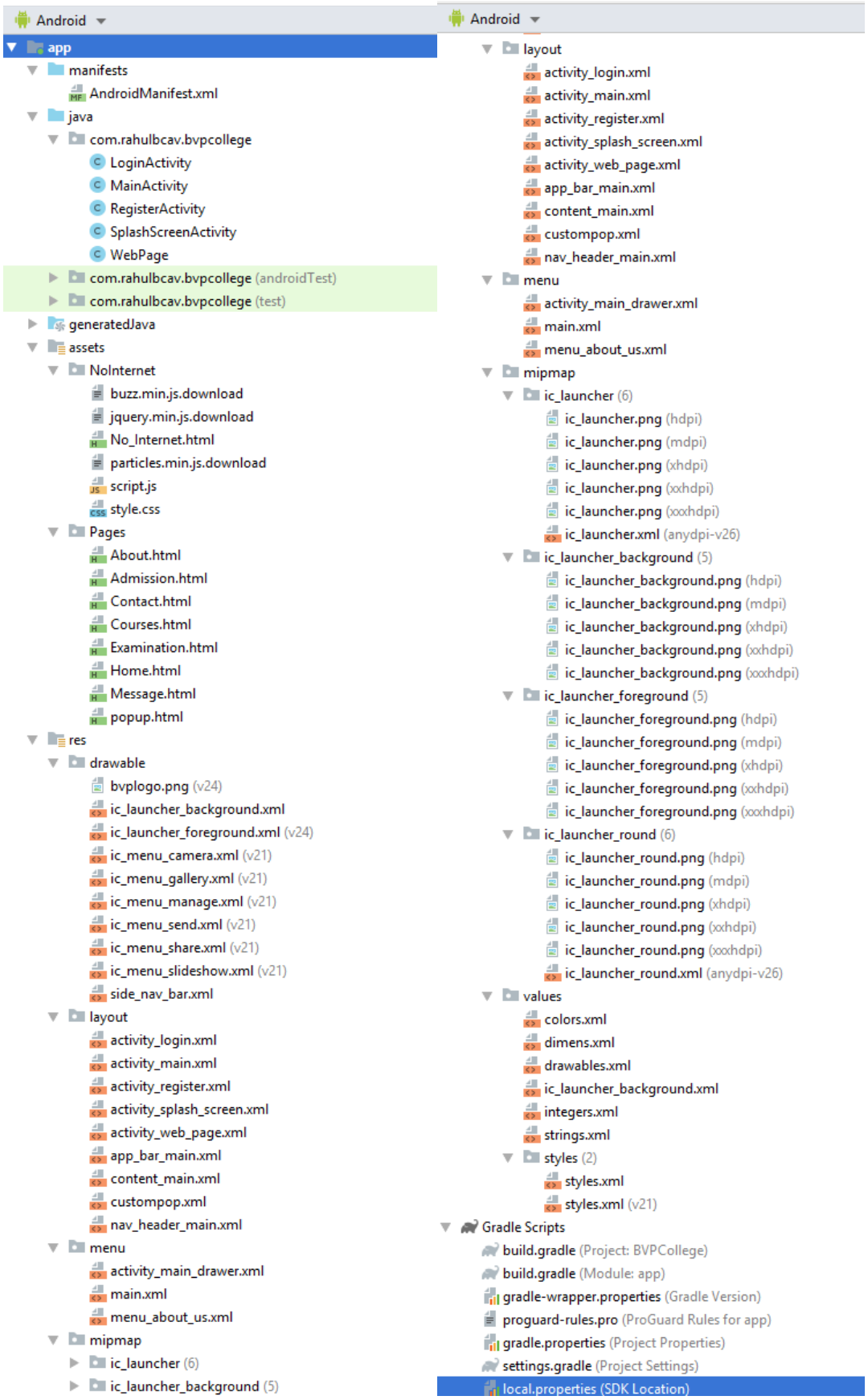
WEB REFERENCES

- www.google.com
- www.udemy.com

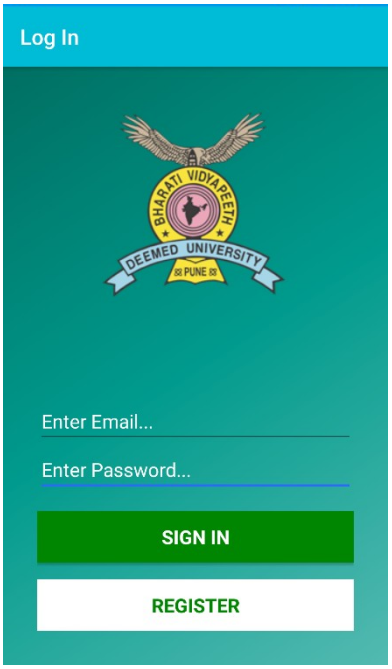
CHAPTER 6

ANNEXURES

6.1 SAMPLE INPUT & OUTPUT WITH CODE -->File Structure :



- Login Form



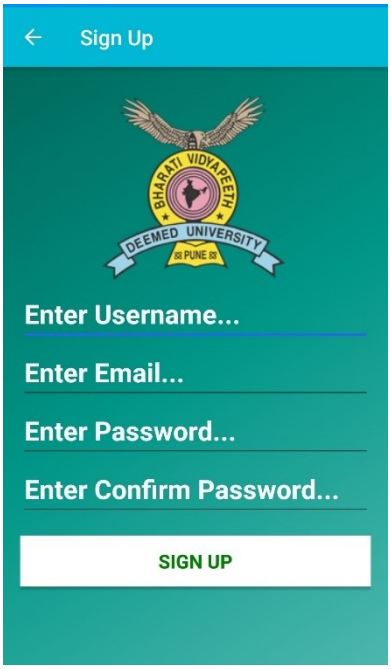
//Code

```
package com.rahulbcav.bvpcollege;

import android.content.Intent;
import android.content.SharedPreferences;
import android.os.Bundle;
import android.view.KeyEvent;
import android.view.View;
import android.view.inputmethod.EditorInfo;
import android.widget.AutoCompleteTextView;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;
import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;

public class LoginActivity extends AppCompatActivity {

    //Constant
    public static final String TAG = "session";
    // TODO: Add member variables here:
    private FirebaseAuth mAuth;
    // UI references.
    private AutoCompleteTextView mEmailView;
    private EditText mPasswordView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);

        SharedPreferences prefs = getSharedPreferences(TAG,0);

        mEmailView = (AutoCompleteTextView) findViewById(R.id.login_email);
        mPasswordView = (EditText) findViewById(R.id.login_password);

        mEmailView.setText(prefs.getString("displayEmail",""));
        mPasswordView.setText(prefs.getString("displayPassword",""));

        mPasswordView.setOnEditorActionListener(new TextView.OnEditorActionListener() {
            @Override
            public boolean onEditorAction(TextView textView, int id, KeyEvent keyEvent) {
                if (id == R.integer.login || id == EditorInfo.IME_NULL) {
                    attemptLogin();
                    return true;
                }
                return false;
            }
        });
        // TODO: Grab an instance of FirebaseAuth
        mAuth = FirebaseAuth.getInstance();
}
    // Executed when Sign in button pressed
    public void signInExistingUser(View v)   {
        // TODO: Call attemptLogin() here
        attemptLogin();
    }
// Executed when Register button pressed
    public void registerNewUser(View v) {
        Intent intent = new Intent(this, com.rahulbcav.bvpcollege.RegisterActivity.class);
        finish();
        startActivity(intent);
    }
// TODO: Complete the attemptLogin() method
    private void attemptLogin() {
        String email,password;
```

```
            email = mEmailView.getText().toString();
        password = mPasswordView.getText().toString();
            if(email.equals("") && password.equals("")) return;
                    Toast.makeText(this, "Login in Progress...", Toast.LENGTH_SHORT).show();
        String displayEmail = email;
        String displayPassword = password;
        SharedPreferences prefs = getSharedPreferences(TAG,0);
        prefs.edit().putString("displayEmail",displayEmail).apply();
        prefs.edit().putString("displayPassword",displayPassword).apply();

        // TODO: Use FirebaseAuth to sign in with email & password
    mAuth.signInWithEmailAndPassword(email,password).addOnCompleteListener(this, new
    OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {

              if(!task.isSuccessful()){
                showErrorDialog("There was a problem signing in");
              } else {
                Intent intent = new Intent(LoginActivity.this, MainActivity.class);
                finish();
                startActivity(intent);
              }
            }
        });
      }
    // TODO: Show error on screen with an alert dialog
      private void showErrorDialog(String msg){
        new AlertDialog.Builder(this)
            .setTitle("Oops!")
            .setMessage(msg)
            .setPositiveButton(android.R.string.ok,null)
            .setIcon(android.R.drawable.ic_dialog_alert)
            .show();
      }
    }
```

- Registration Form



//Code

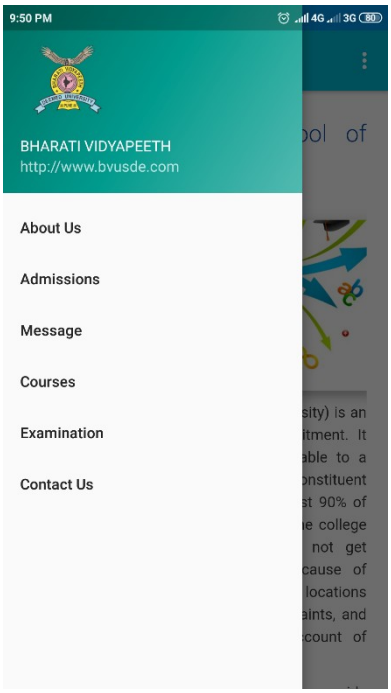
```
package com.rahulbcav.bvpcollege;

public class RegisterActivity extends AppCompatActivity {

    // Constants
    public static final String TAG = "session";
    public static final String DISPLAY_NAME_KEY = "username";
    // UI references.
    private AutoCompleteTextView mEmailView;
    private AutoCompleteTextView mUsernameView;
    private EditText mPasswordView;
    private EditText mConfirmPasswordView;
    // Firebase instance variables
    FirebaseAuth mAuth;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_register);

        getSupportActionBar().setDisplayShowHomeEnabled(true);

        mEmailView = (AutoCompleteTextView) findViewById(R.id.register_email);
        mPasswordView = (EditText) findViewById(R.id.register_password);
        mConfirmPasswordView = (EditText) findViewById(R.id.register_confirm_password);
        mUsernameView = (AutoCompleteTextView) findViewById(R.id.register_username);

        // Keyboard sign in action
        mConfirmPasswordView.setOnEditorActionListener(new TextView.OnEditorActionListener() {
            @Override
            public boolean onEditorAction(TextView textView, int id, KeyEvent keyEvent) {
```

```
            if (id == R.integer.register_form_finished || id == EditorInfo.IME_NULL) {
                attemptRegistration();
                return true;
            }
            return false;
        }
    });

    // TODO: Get hold of an instance of FirebaseAuth
    mAuth = FirebaseAuth.getInstance();
}
  // Executed when Sign Up button is pressed.
  public void signUp(View v) {
    attemptRegistration();
  }

  private void attemptRegistration() {
    // Reset errors displayed in the form.
    mEmailView.setError(null);
    mPasswordView.setError(null);

    // Store values at the time of the login attempt.
    String email = mEmailView.getText().toString();
    String password = mPasswordView.getText().toString();

    boolean cancel = false;
    View focusView = null;

    // Check for a valid password, if the user entered one.
    if (TextUtils.isEmpty(password) || !isPasswordValid(password)) {
      mPasswordView.setError(getString(R.string.error_invalid_password));
      focusView = mPasswordView;
      cancel = true;
    }

    // Check for a valid email address.
    if (TextUtils.isEmpty(email)) {
      mEmailView.setError(getString(R.string.error_field_required));
      focusView = mEmailView;
      cancel = true;
    } else if (!isEmailValid(email)) {
      mEmailView.setError(getString(R.string.error_field_required));
      focusView = mEmailView;
      cancel = true;
    }

    if (cancel) {
      // There was an error; don't attempt login and focus the first
      // form field with an error.
      focusView.requestFocus();
    } else {
      // TODO: Call create FirebaseUser() here
```

```
            createFirebaseUser();
        }
    }

    private boolean isEmailValid(String email) {
        // You can add more checking logic here.
        return email.contains("@");
    }

    private boolean isPasswordValid(String password) {
        //TODO: Add own logic to check for a valid password (minimum 6 characters)
        String confirmPassword = mConfirmPasswordView.getText().toString();
        return confirmPassword.equals(password) && password.length() > 6;
    }

    // TODO: Create a Firebase user
    private void createFirebaseUser(){
        String email = mEmailView.getText().toString();
        String password = mPasswordView.getText().toString();
            mAuth.createUserWithEmailAndPassword(email,password).addOnCompleteListener(this, new
            OnCompleteListener<AuthResult>() {
            @Override
            public void onComplete(@NonNull Task<AuthResult> task) {

                if(!task.isSuccessful()){
                    showErrorDialog("Registration attempt failed");
                } else {
                    saveDisplayName();
                    Intent intent = new Intent(RegisterActivity.this, LoginActivity.class);
                    finish();
                    startActivity(intent);
                }
            }
        });
    }

    // TODO: Save the display name to Shared Preferences
    private void saveDisplayName(){
        String displayName = mUsernameView.getText().toString();
        SharedPreferences prefs = getSharedPreferences(TAG,0);
        prefs.edit().putString(DISPLAY_NAME_KEY,displayName).apply();
    }

    // TODO: Create an alert dialog to show in case registration failed
    private void showErrorDialog(String msg){
        new AlertDialog.Builder(this)
                .setTitle("Oops!")
                .setMessage(msg)
                .setPositiveButton(android.R.string.ok,null)
                .setIcon(android.R.drawable.ic_dialog_alert)
                .show();
}
```

```
@Override
  public boolean onSupportNavigateUp() {
     onBackPressed();
     return true;
  }

  public void onBackPressed(){
     startActivity(new Intent(RegisterActivity.this, LoginActivity.class));
     RegisterActivity.this.finish();
  }

}
```

- Navigational Panel



//Code

```
package com.rahulbcav.bvpcollege;

import android.app.Dialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.content.SharedPreferences;
import android.graphics.Color;
import android.graphics.drawable.ColorDrawable;
import android.os.Bundle;
import androidx.appcompat.app.AlertDialog;
import androidx.core.view.GravityCompat;
import androidx.appcompat.app.ActionBarDrawerToggle;
import android.view.MenuItem;
import com.google.android.material.navigation.NavigationView;
import androidx.drawerlayout.widget.DrawerLayout;
import androidx.appcompat.app.AppCompatActivity;
import androidx.appcompat.widget.Toolbar;
import android.view.Menu;
import android.view.View;
import android.webkit.WebView;
import android.webkit.WebViewClient;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends AppCompatActivity
        implements NavigationView.OnNavigationItemSelectedListener {
            Dialog mDialog;

            WebView page;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        ShowPopup();

        Toolbar toolbar = findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        //start
        page = findViewById(R.id.page);
        page.getSettings().setJavaScriptEnabled(true);
        page.getSettings().setAppCacheEnabled(true);
        page.loadUrl("file:///android_asset/Pages/Home.html");

        page.setWebViewClient(new WebViewClient(){
            public void onReceivedError(WebView view, int errorCode, String description, String failingUrl){
                page.loadUrl("file:///android_asset/NoInternet/No_Internet.html");
            }
        });
```

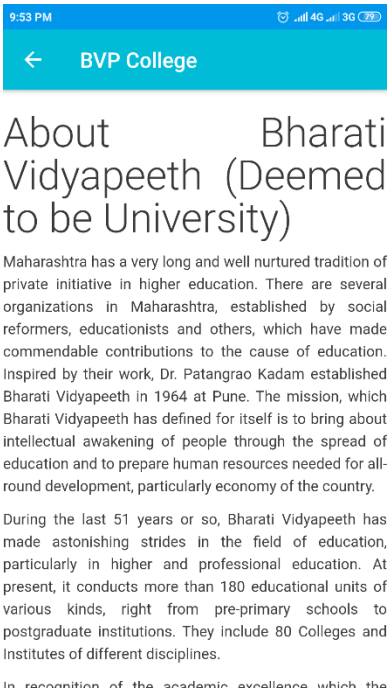
```
        //end

        DrawerLayout drawer = findViewById(R.id.drawer_layout);
        NavigationView navigationView = findViewById(R.id.nav_view);
        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
                this, drawer, toolbar, R.string.navigation_drawer_open, R.string.navigation_drawer_close);
        drawer.addDrawerListener(toggle);
        toggle.syncState();
        navigationView.setNavigationItemSelectedListener(this);
        }

       @Override
       public void onBackPressed() {
        DrawerLayout drawer = findViewById(R.id.drawer_layout);
        if (drawer.isDrawerOpen(GravityCompat.START)) {
            drawer.closeDrawer(GravityCompat.START);
        } else {
            super.onBackPressed();
        }
       }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is present.
        getMenuInflater().inflate(R.menu.main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {

            AlertDialog.Builder alertDialog = new AlertDialog.Builder(MainActivity.this);

            alertDialog.setTitle("Log Out Alert");
            alertDialog.setMessage("Do you want to Log Out")
                    .setCancelable(false)
                    .setPositiveButton("Yes", new DialogInterface.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialogInterface, int i) {

                            SharedPreferences pref = getSharedPreferences("session", 0 );
                            pref.edit().clear().commit();

                            Toast.makeText(MainActivity.this,"Log Out
Session!!",Toast.LENGTH_SHORT).show();
```

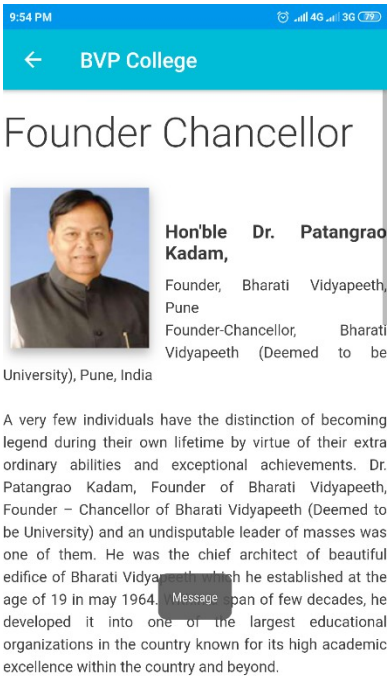
```
                finishAffinity();
                System.exit(0);
            }
        })
        .setNeutralButton("Cancel", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                dialogInterface.cancel();
            }
        })
        .setNegativeButton("No", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                dialogInterface.cancel();
            }
        });
    AlertDialog dialog = alertDialog.create();
    dialog.show();
    return true;
  }

  return super.onOptionsItemSelected(item);
}

@SuppressWarnings("StatementWithEmptyBody")
@Override
public boolean onNavigationItemSelected(MenuItem item) {
  // Handle navigation view item clicks here.
  int id = item.getItemId();

  if (id == R.id.nav_aboutus) {
    Toast.makeText(MainActivity.this,item.getTitle(),Toast.LENGTH_SHORT).show();
    Intent intent = new Intent(MainActivity.this, WebPage.class);
    intent.putExtra("name", "About");
    startActivity(intent);
  } else if (id == R.id.nav_admissions) {
    Toast.makeText(MainActivity.this,item.getTitle(),Toast.LENGTH_SHORT).show();
    Intent intent = new Intent(MainActivity.this, WebPage.class);
    intent.putExtra("name", "Admission");
    startActivity(intent);
  } else if (id == R.id.nav_courses) {
    Toast.makeText(MainActivity.this,item.getTitle(),Toast.LENGTH_SHORT).show();
    Intent intent = new Intent(MainActivity.this, WebPage.class);
    intent.putExtra("name", "Courses");
    startActivity(intent);
  } else if (id == R.id.nav_Message) {
    Toast.makeText(MainActivity.this,item.getTitle(),Toast.LENGTH_SHORT).show();
    Intent intent = new Intent(MainActivity.this, WebPage.class);
    intent.putExtra("name", "Message");
    startActivity(intent);
  } else if (id == R.id.nav_examination) {
    Toast.makeText(MainActivity.this,item.getTitle(),Toast.LENGTH_SHORT).show();
```

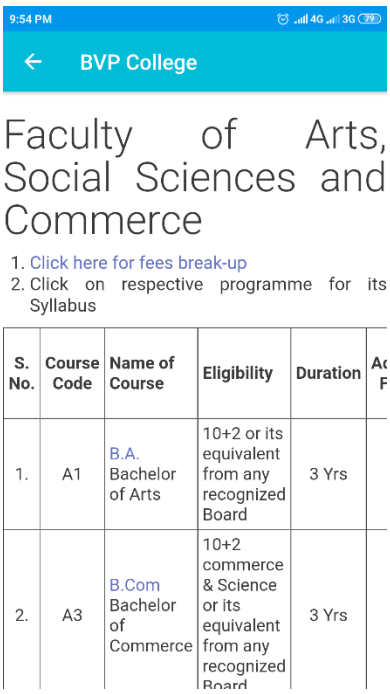
```
        Intent intent = new Intent(MainActivity.this, WebPage.class);
        intent.putExtra("name", "Examination");
        startActivity(intent);
    } else if (id == R.id.nav_contactus) {
        Toast.makeText(MainActivity.this,item.getTitle(),Toast.LENGTH_SHORT).show();
        Intent intent = new Intent(MainActivity.this, WebPage.class);
        intent.putExtra("name", "Contact");
        startActivity(intent);
    }

    DrawerLayout drawer = findViewById(R.id.drawer_layout);
    drawer.closeDrawer(GravityCompat.START);
    return true;
}

public void ShowPopup() {

    mDialog = new Dialog(MainActivity.this);
    TextView txtclose;
    final WebView webView;

    mDialog.setContentView(R.layout.custompop);
    mDialog.setTitle("Admission Open");

    txtclose =(TextView) mDialog.findViewById(R.id.txtclose);
    txtclose.setText("X");
    txtclose.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            mDialog.dismiss();
        }
    });

    //

    webView = mDialog.findViewById(R.id.webpage);
    webView.getSettings().setJavaScriptEnabled(true);

    webView.loadUrl("file:///android_asset/Pages/popup.html");

    webView.setWebViewClient(new WebViewClient(){

        public void onReceivedError(WebView view, int errorCode, String description, String failingUrl){
            webView.loadUrl("file:///android_asset/NoInternet/No_Internet.html");
        }

    });

   mDialog.getWindow().setBackgroundDrawable(new ColorDrawable(Color.TRANSPARENT));
    mDialog.show();
}
```

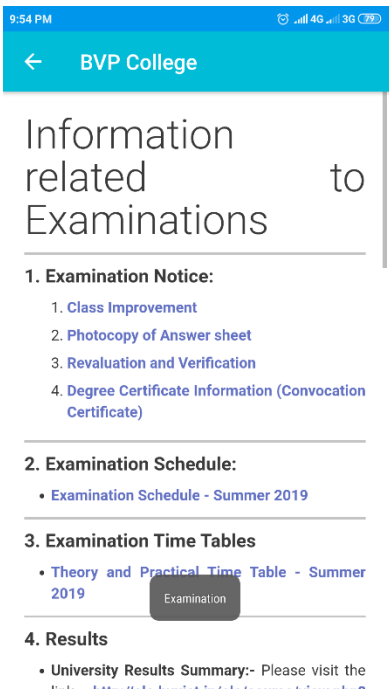
}

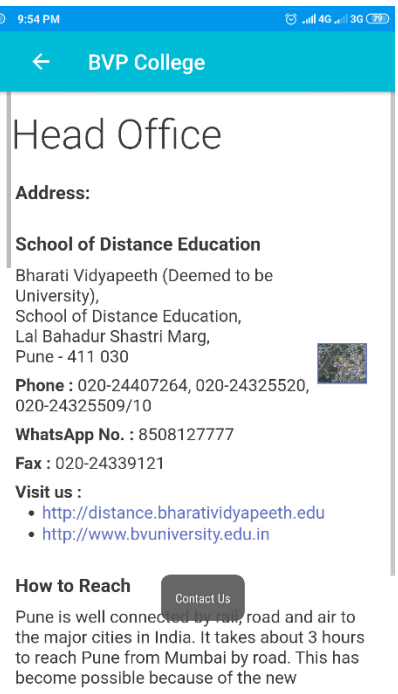
- Navigational Pages











//Code

```
package com.rahulbcav.bvpcollege;

import androidx.appcompat.app.ActionBar;
import androidx.appcompat.app.AppCompatActivity;
```

```
import android.content.Intent;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.webkit.WebView;
import android.webkit.WebViewClient;
import android.widget.Toolbar;
import java.io.IOException;
import java.net.URL;

public class WebPage extends AppCompatActivity {

    private final String googleDocs = "https://docs.google.com/viewer?url=";
    WebView webView;
    String name;

    @Override
    public boolean onSupportNavigateUp() {
        onBackPressed();
        return true;
    }

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_web_page);

        getSupportActionBar().setDisplayShowHomeEnabled(true);

        Intent intent = getIntent();
        name = intent.getStringExtra("name");

        webView = findViewById(R.id.webpage);
        webView.getSettings().setJavaScriptEnabled(true);
        webView.getSettings().setAppCacheEnabled(true);
        webView.getSettings().setBuiltInZoomControls(true);
        webView.loadUrl("file:///android_asset/Pages/" + name + ".html");

        webView.setWebViewClient(new WebViewClient(){

            public boolean shouldOverrideUrlLoading(WebView webView, String url){
                if(url.endsWith(".pdf")){
                    String pdfUrl = googleDocs + url;
                    webView.loadUrl(pdfUrl);
                } else {
                    webView.loadUrl(url);
                }
                return true;
            }

            public void onReceivedError(WebView view, int errorCode, String description, String failingUrl){
                webView.loadUrl("file:///android_asset/NoInternet/No_Internet.html");
            }

        });

    }
    public void onBackPressed(){
        finish();
```

```
    }
}
```