CHAPTER 1

INTRODUCTION

Academic success is important because it is strongly linked to the positive outcomes we value. Adults who are academically successful and with high levels of education are more likely to be employed, have stable employment, have more employment opportunities than those with less education and earn higher salaries, are more likely to have health insurance, are less dependent on social assistance, are less likely to engage in criminal activity, are more active as citizens and charitable volunteers and are healthier and happier. Academic success is important because working people will need higher levels of education to tackle the technologically demanding occupations of the future.

As Academic success plays important role, most students and also parents are much concerned about the academic exam results and will be willingly waiting to see the results. When we look into the existing system results are announced in the college's notice boards which may gather more crowds for the fewer days of result announcement.

To eliminate these problems and to provide efficiency it is necessary to computerize the result announcement process. FastResults is a website which can make the result announcement much easier and provides the students an opportunity to see their result from their location with the high privacy.

CHAPTER 2

LITERATURE SURVEY

2.1 EXISTING SYSTEM

We can find many online result websites which are particular to their domain and college. This project is very similar to other existing systems but with lot of improvements and additional features. Existing system lacks of security, privacy, user experience and automatic result delivery features. In proposed system there lacks are eliminated.

2.2 TECHNOLOGY SURVEY

2.2.1 ANDROID

Android is an open source and Linux-based Operating System for mobile devices such as smart phones and tablet computers. Android was developed by the Open Handset Alliance, led by Google, and other companies.

Android offers a unified approach to application development for mobile devices which means developers need only develop for Android, and their applications should be able to run on different devices powered by Android.

The first beta version of the Android Software Development Kit (SDK) was released by Google in 2007 where as the first commercial version, Android 1.0, was released in September 2008.

2.2.2 MY SQL

MySQL is a database system used on the web. Basically, a MySQL database allows us to create a relational database structure on a web-server in order to store data or automate procedures. PHP acts as your queries, and forms are basically web pages with fields in them.

Usage of MySQL

- ✓ Scalability and Flexibility: Can handle embedded applications and massive data warehouses.
- ✓ High Performance: Unique storage engine architecture.

- ✓ High Availability: High speed master/slave replication configurations.
- ✓ ACID (atomic, consistent, isolated, durable) transaction support.
- ✓ Distributed transaction capability.
- ✓ Multi-version transaction support.

2.2.3 Bootstrap

Bootstrap is a sleek, intuitive, and powerful, mobile first front-end framework for faster and easier web development. It uses HTML, CSS and Javascript.

Advantages

- ✓ Bootstrap 3, framework consists of Mobile first styles throughout the entire library instead them of in separate files.
- ✓ It is supported by all popular browsers.
- ✓ With just the knowledge of HTML and CSS anyone can get started with Bootstrap. Also the Bootstrap official site has a good documentation.
- ✓ Bootstrap's responsive CSS adjusts to Desktops, Tablets and Mobiles. More about the responsive design is in the chapter Bootstrap Responsive Design.
- ✓ Provides a clean and uniform solution for building an interface for developers.
- ✓ It contains beautiful and functional built-in components which are easy to customize.
- ✓ It also provides web based customization.
- ✓ Best of all it is an open source.

2.3 Node.js

Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.

Node.js is an open source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript, and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.

Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.

2.3.1 Features of Node.js

- ✓ All APIs of Node.js library are asynchronous, that is, non-blocking. It essentially means a Node.js based server never waits for an API to return data. The server moves to the next API after calling it and a notification mechanism of Events of Node.js helps the server to get a response from the previous API call.
- ✓ Being built on Google Chrome's V8 JavaScript Engine, Node.js library is very fast in code execution.
- ✓ Node.js uses a single threaded model with event looping. Event mechanism helps the server to respond in a non-blocking way and makes the server highly scalable as opposed to traditional servers which create limited threads to handle requests. Node.js uses a single threaded program and the same program can provide service to a much larger number of requests than traditional servers like Apache HTTP Server.
- ✓ Node.js applications never buffer any data. These applications simply output the data in chunks.
- ✓ Node.js is released under the MIT license.

CHAPTER 3

SYSTEM REQUIREMENT SPECIFICATION

3.1 NON-FUNCTIONAL REQUIREMENT

3.1.1 Services provided to Users

- ➢ Users can register to FastResults system.
- ➢ Users can get automatic result to their mobile systems.
- ➢ Data privacy.
- ➢ Android application.
- ➢ One click to website.

3.1.2 Services provided to Admin

- ➢ Friendly UI.
- ➢ Support for Excel format result sheet.

3.2 FUNCTIONAL REQUIREMENT SPECIFICATION

3.2.1 Services provided for Users

Registration

Registration process should take the following details

- o Name
- o USN
- o Mobile number
- o Email
- o Password

Login

Users should able to login via phone number or email id with his password.

Result view

Users should be able to see only his result.

Automatic result announcement

Users should register to our service and download android application to get result announcement to mobile.

One click to website

Android application provides a link which will take the users directly to our website.

3.2.2 Services provided for Admin.

Friendly UI

Bootstrap is a well known framework for web development which has a special feature, auto scaling to the screen resolution. Bootstrap framework will make the UI very friendly and keeps it understandable.

Support for Excel format result sheet

Excel result sheet can be given as input to FastResults. FastResults will take care of converting to convenient format making it easy to use.

3.3 APPENDICES

Hardware Interfaces for Server

- RAM: 512MB+
- Processor: Pentium 4+
- Processor Speed: 1ghz+
- Hard Disk: 4GB+

Software Interface for Server

• Operating System	:	Linux Debian
• Language	:	Node.js
• Database	:	MySQL

Requirements for Android Client

• Operating System	:	Android API 17+
• RAM	:	512MB

CHAPTER 4

SYSTEM ANALYSIS

4.1 EXISTING SYSTEM

As we have discussed the existing system in the Literature Survey which has dis-advantages. Proposed system is designed to eliminate the flaws or disadvantages in the existing system.

4.2 LIMITATIONS OF THE EXISTING SYSTEM

- **Privacy:** In most of the similar existing system any user can see anyone's result if he has USN.
- **Unfriendly UI:** UI and not easy understandable and good looking.
- **Registration:** Existing system doesn't have registration option. Result is available to everyone which in turn privacy problem.
- **No automated notifications:** Each one of them should visit the website to see the result.
- **No one click website:** Users should remember and enter the website URL each time. There is no button which takes them directly to website.

4.3 PROPOSED SYSTEM

In the proposed system a student can register with our system and can see his academic result after the announcement. An Android application feature is also provided which brings the result related notifications.

4.4 ADVANTAGES OF THE PROPOSED SYSTEM

- Website adjustment to screen resolution.
- High speed.
- Privacy.
- Notification system.
- One click website facility.

4.5 FEASIBILITY STUDY:

All projects are feasible given resources and infinite time. It is both necessary and prudent to evaluate the feasibility of a project at the earliest possible time. Unfortunately the development of a computer based system it is more likely to be plagued by the scarcity of resources and difficult date.

The feasibility includes,

- Operational Feasibility
- Technical Feasibility
- Economic Feasibility

4.5.1 Operational Feasibility

Now a day almost 80% people are familiar with Android smart phones and web browsers. Since the Bootstrap and Android is providing an attractive user interface by using Java Script, XML to the operator or users, the users feel very easy to work onto it. The UI components includes Buttons, Text, Text input Box and many more. There UI components makes the user understand the operation of respective UI components very easily.

4.5.2 Technical Feasibility

At the backend NodeJS and MySQL Server are the most recent technologies to develop Web based systems and to design databases. The system offers greater levels of user friendliness combined with greater processing speed. So the whole system is very responsive and lesser chances to failure.

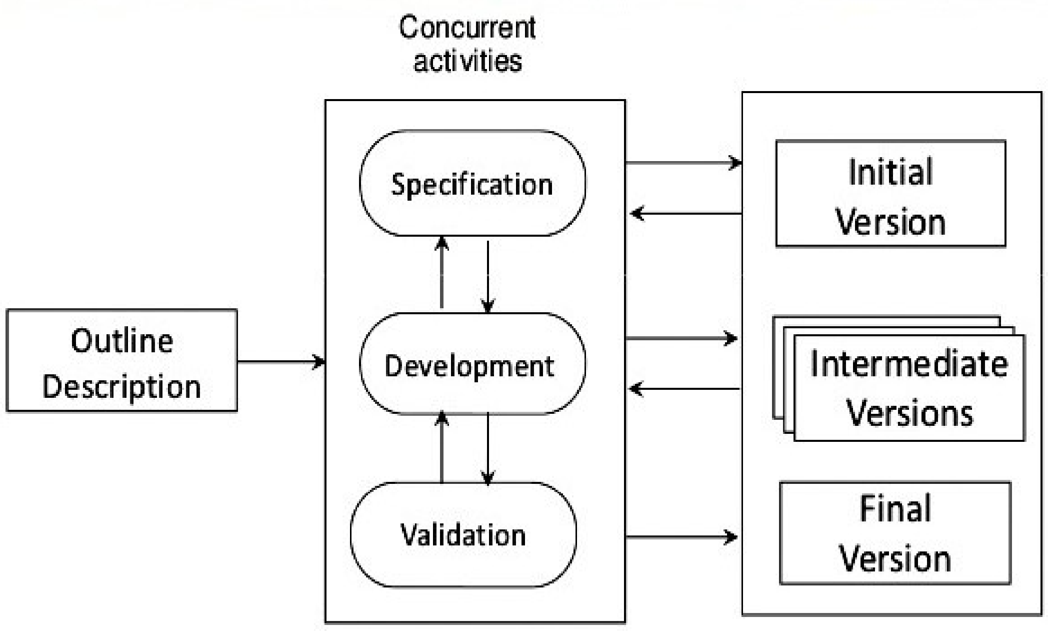
4.5.3 Economic Feasibility

Now a day's Internet costs are very low and almost every people will have it. As most people are using Android phone itself no extra resource is required by the end users as Android mobile will also have browsers. So it's economically very feasible.

4.6 APPROACH ADAPTED

Evolutionary development

Evolutionary development is based on the idea of developing an initial implementation, exposing this to user comment and refining it through many versions until an adequate system has been developed.



There are two fundamental types of evolutionary development:

1. Exploratory development where the objective of the process is to work with the customer to explore their requirements and deliver a final system. The development starts with the parts of the system that are understood. The system evolves by adding new features proposed by the customer.

2. Throwaway prototyping where the objective of the evolutionary development process is to understand the customer's requirements and hence develop a better requirements definition for the system. The prototype concentrates on experimenting with the customer requirements that are poorly understood.

Advantages

1. The advantage of a software process that is based on an evolutionary approach is that the specification can be developed incrementally.
2. In evolutionary approach users develop a better understanding of their problem, this can be reflected in the software system.
3. Suitable For small and medium-sized systems.

Disadvantages

1. The process is not visible Managers need regular deliverables to measure progress. If systems are developed quickly, it is not cost-effective to produce documents that reflect every version of the system.
2. Systems are often poorly structured Continual change tends to corrupt the software structure. Incorporating software changes becomes increasingly difficult and costly.
3. The problems of evolutionary development become particularly acute for large, complex, long-lifetime systems, where different teams develop different parts of the system. It is difficult to establish a stable system architecture using this approach, which makes it hard to integrate contributions from the teams.

CHAPTER 5

SYSTEM DESIGN

System design is the process of defining the elements of a system such as the architecture, modules and components, the different interfaces of those components and the data that goes through that system. We followed three tire architecture as it differs the data storage and business logic which makes it easy to use and maintain.

The design activity often results in three separate outputs –

- Architecture design.
- High level design.
- Detailed design.

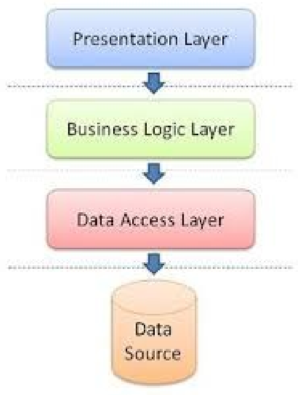
5.1 ARCHITECTURAL DESIGN

Architectural design is a set of principles- a coarse grained pattern that provides an abstract framework for a family of systems. An architectural style improves partitioning and promotes design reuse by providing solutions to frequently recurring problems.

5.1.1 THREE-TIER ARCHITECTURE

Three-tier architecture is a client-server architecture in which the functional process logic, data access, computer data storage and user interface are developed and maintained as independent modules on separate platforms.

Three-tier architecture:



The three tiers in three-tier architecture are:

- **Presentation Tier**:

We used Android as the presentation layer which is uses XML to compose UI components. The UI contains pictures, buttons and text.

- **Application Tier/Business tier**:

Also called the middle tier, logic tier, business logic or logic tier, this tier is pulled from the presentation tier. In our project the middle layer is PHP which mainly has the business logic and manages the communication between Android application and Database

- **Data Tier**:

We used MySQL as database to store the large complex data.

Advantages:

- Easy to modify without affecting other modules
- Fast communication
- Performance will be good in three tier architecture.

5.2 HIGH LEVEL DESIGN

In high level design identifies the modules that should be built for developing the system and the specifications of these modules. At the end of system design all major data structures, file format, output formats, etc., are also fixed. The focus is on identifying the modules.

5.2.1 DATA FLOW DIAGRAM

A Data Flow Diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. As our project contains complex data flow we are using DFD to explain and to keep track of Data flow.

5.2.2 Context Data Flow Diagram

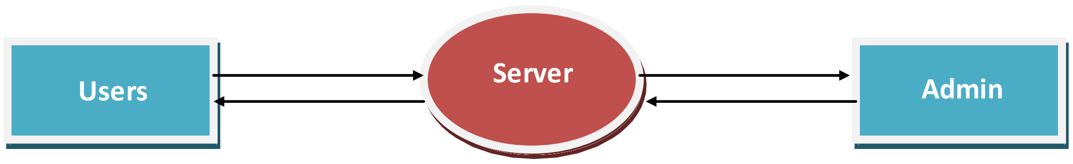


Fig: Context data flow diagram

.

5.2.3.1 Data Flow Diagram:

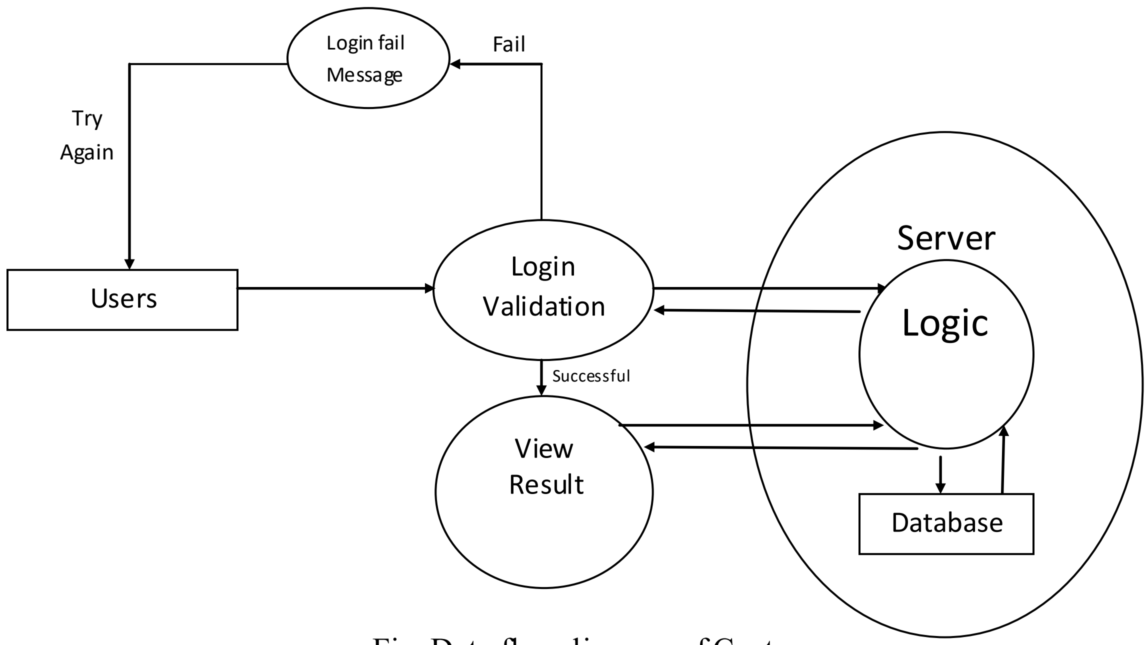


Fig: Data flow diagram of Customer

5.3 DETAILED DESIGN

5.3.1 Use case diagrams

Use case diagrams model the functionality of a system using actors and use cases. Use cases are services or functions provided by the system to its users.

Use cases:

In FastResults use cases includes registration, login, view result, register android application and many more.

Actors:

Actors in our project are users.

Associations:

Associations between actors and use cases are indicated in use case diagrams by solid lines. An association exists whenever an actor is involved with an interaction described by a use case.

5.3.1.1 Use case Diagram:

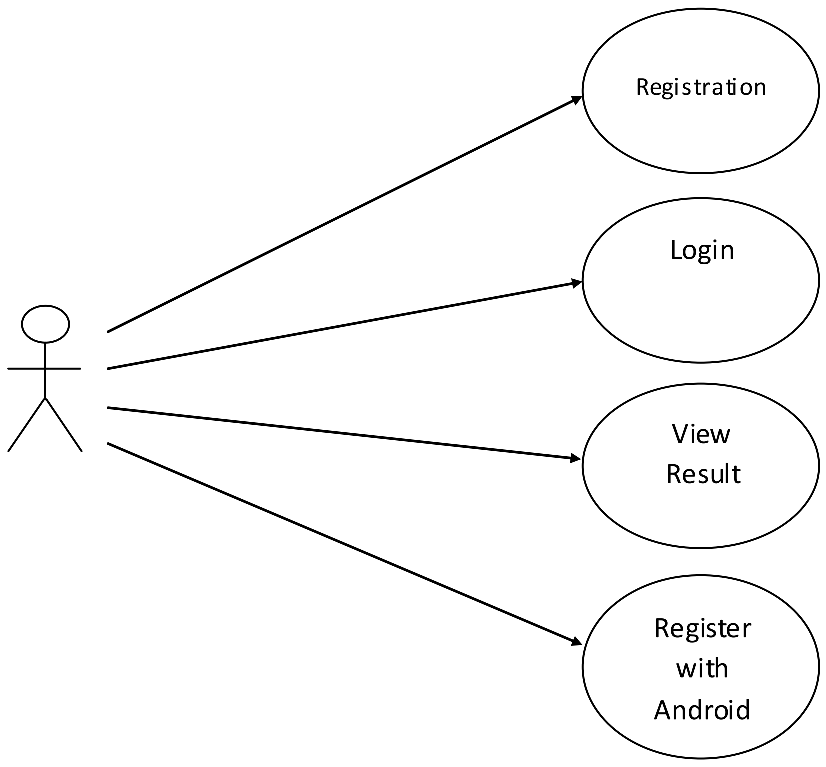


Fig: Use case diagram of Customer

- **Registration:** User is given with the option to register to FastResults system.
- **Login:** Users must login by giving credentials to view result.
- **View result:** User can view only his/her academic result.
- **Register with android:** User should login through Android application to get notification result feature.

5.3.2 Sequence Diagram:

A Sequence diagram is an interaction diagram that shows how processes operate with one another and in what order. It describes interactions among classes in terms of an exchange of messages over time. Sequence diagrams are used to show how objects interact in a given situation. An important characteristic of a sequence diagram is that time passes from top to bottom: the interaction starts near the top of the diagram and ends at the bottom.

Targets/Class roles/State:

Objects as well as classes can be targets on a sequence diagram, which means that messages can be sent to them. A target is displayed as a rectangle with some text in it. Below the target, its lifeline extends for as long as the target exists. Targets can be actor, boundary, control, entity and database.

Messages:

Messages are arrows that represent communication between objects.

Lifelines:

Lifelines are vertical dashed lines that indicate the object's presence over time.

5.3.2 .1 Sequence Diagram - User:

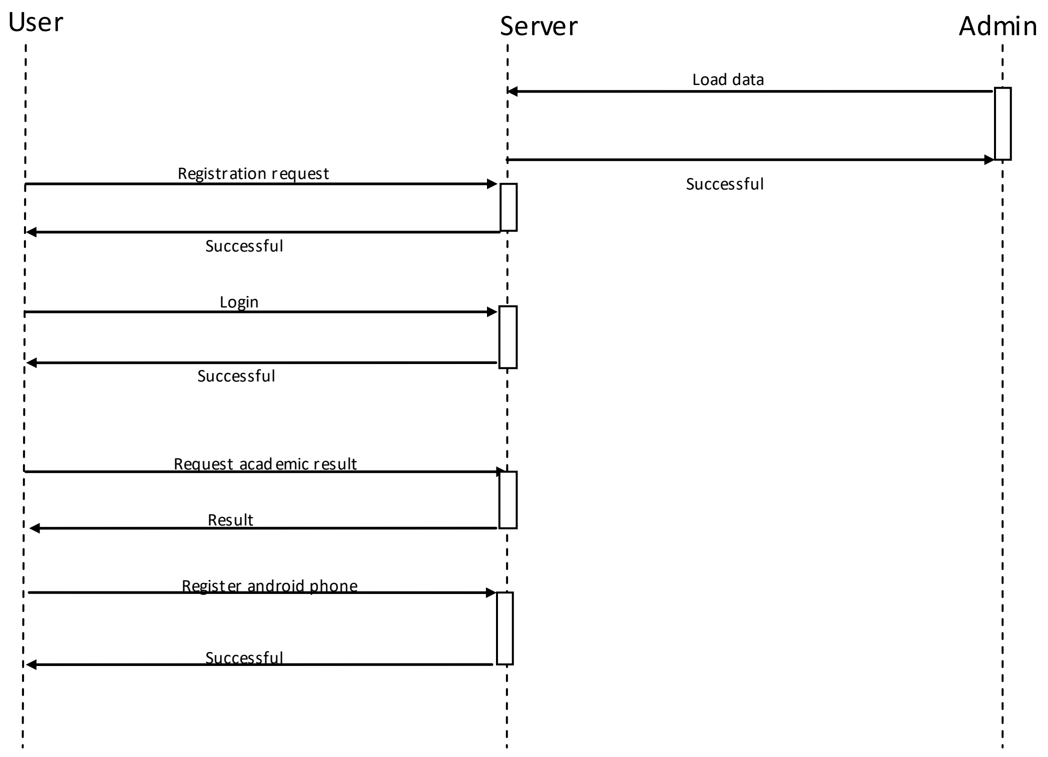


Fig: Sequence Diagram of the users.

5.4 ENTITY RELATIONSHIP DIAGRAM

In software engineering, an entity-relationship model (ERM) is an abstract and conceptual representation of data. Entity-relationship modelling is a database modelling method, used to produce a type of conceptual schema or semantic data model of a system, often a relational database, and its requirements in a top-down fashion. Diagrams created by this process are called entity-relationship diagrams, ER diagrams, or ERDs. A relationship is how the data is shared between entities. There are three types of relationships between entities

1. One-to-one

One instance of an entity (A) is associated with one other instance of another entity (B). For example, in a database of employees, each employee name (A) is associated with only one social security number (B).

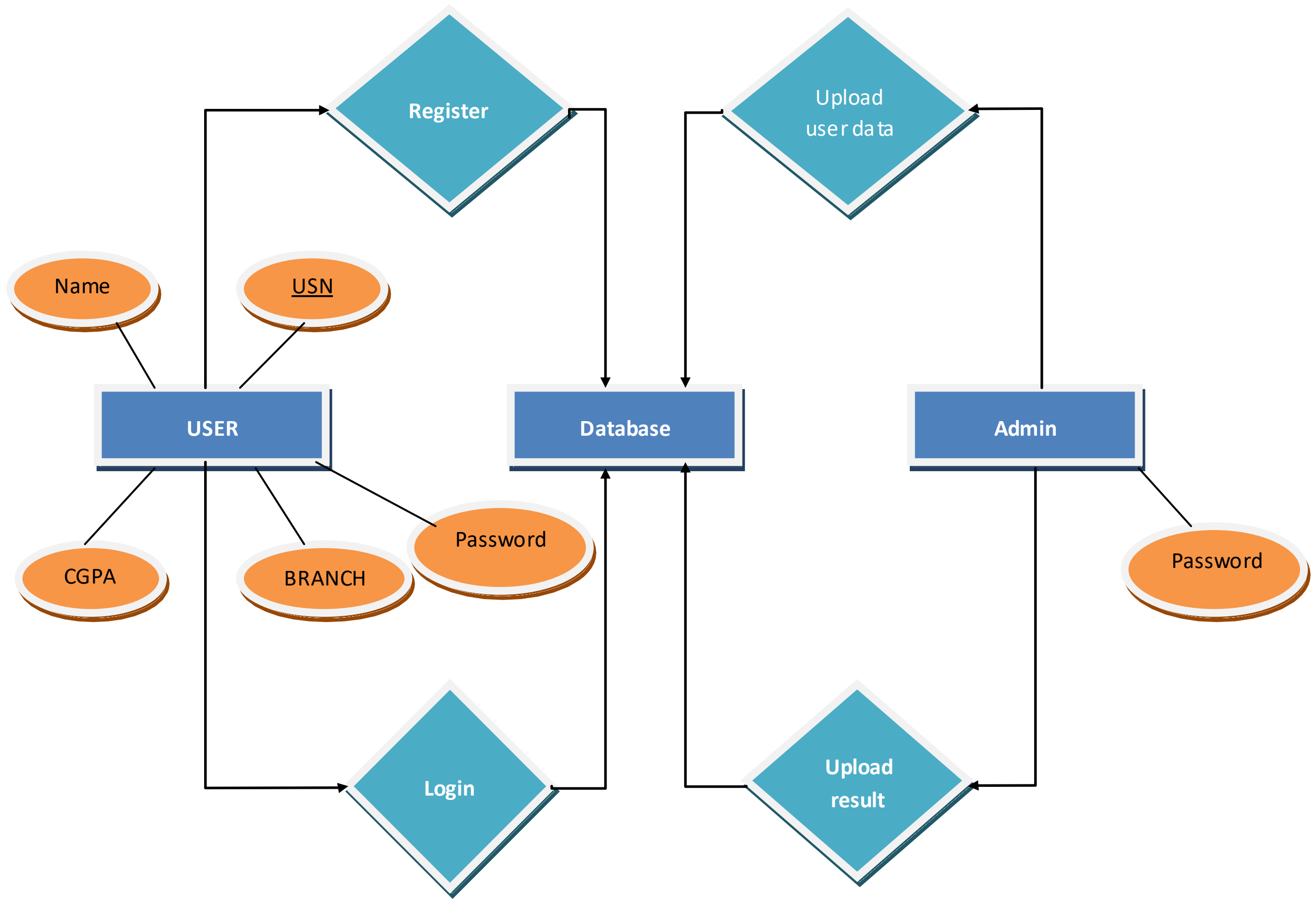
2. One-to-Many

One instance of an entity (A) is associated with zero, one or many instances of another entity (B), but for one instance of entity B there is only one instance of entity A. For example, for a company with all employees working in one building, the building name (A) is associated with many different employees (B), but those employees all share the same singular association with entity A.

3. Many-to-Many

One instance of an entity (A) is associated with one, zero or many instances of another entity (B), and one instance of entity B is associated with one, zero or many instances of entity A. For example, for a company in which all of its employees work on multiple projects, each instance of an employee (A) is associated with many instances of a project (B), and at the same time, each instance of a project (B) has multiple employees (A) associated with it.

5.4.1 ER Diagrams



CHAPTER 6

IMPLEMENTATION

6.1 FRAMEWORKS USED:

Apache HttpCore 4.4.6:

HttpCore is a set of low level HTTP transport components that can be used to build custom client and server side HTTP services.

PEAR Mail Package:

PEAR stands for PHP Extension and Application Repository PEAR's Mail package defines an interface for implementing mailers under the PEAR hierarchy. It also provides supporting functions useful to multiple mailer backend.

6.2 CLIENT SIDE ANDROID APIs IMPLEMENTATION

6.2.1 Intent

Intent is an abstract description of an operation to be performed. Intent provides a facility for performing late runtime binding between the code in different applications. Its most significant use is in the launching of activities, where it can be thought of as the glue between activities. It is basically a passive data structure holding an abstract description of an action to be performed.

Intent Structure

The primary pieces of information in an intent are:

- Action: The general action to be performed, such as ACTION_VIEW, ACTION_EDIT, ACTION_MAIN, etc.
- Data: The data to operate on, such as a person record in the contacts database, expressed as an Uri.

6.2.2 Activity

An activity is a single, focused thing that the user can do. Almost all activities interact with the user, so the Activity class takes care of creating a window for you in which you can place your UI with setContentView(View). While activities are often presented to the user as full-screen windows.

Methods used

- **void onCreate(Bundle savedInstanceState):** Used to initialize activity. Most importantly, here we usually sets content of Activity.
- **void onResume():** Called when the activity will start interacting with the user.
- **void onPause():** Called when the system is about to start resuming a previous activity.
- **Intent getIntent():** Return the intent that started this activity.
- **void startActivity(Activity a):** Used to start another activity.
- **void onBackPressed():** Called when the activity has detected the user's press of the back key.
- **void startActivityForResult(Activity a):** Used to start another activity which returns the result.
- **void onActivityResult(int requestCode, int resultCode, Intent data):** A callback method which is automatically called when an Activity returns result which is called using startActivityForResult().
- **void setContentView(R.layout.activity_main):** Used to set the UI contents of Activity.
- **void finish():** End the activity.
- **int findViewById(R.id.toolbar):** Finds a view that was identified by the id attribute from the XML that was processed in onCreate(Bundle).

6.2.5 Firebase Cloud Messaging

Firebase Cloud Messaging (FCM) is a cross-platform messaging solution that lets you reliably deliver messages at no cost. Using FCM, you can notify a client app that new email or other data is available to sync. You can send notification messages to drive user reengagement and retention. For use cases such as instant messaging, a message can transfer a payload of up to 4KB to a client app.

Services used for Google Cloud Messaging

- **MyFirebaseMessagingService**: It extends FirebaseMessagingService. This is required if you want to do any message handling beyond receiving notifications on apps in the background. To receive notifications in foregrounded apps, to receive data payload, to send upstream messages, and so on, you must extend this service.

- **MyFirebaseInstanceIdService**: It extends FirebaseInstanceIdService to handle the creation, rotation, and updating of registration tokens. This is required for sending to specific devices or for creating device groups.

Methods used for Google Cloud Messaging

- **void onTokenRefresh():** The onTokenRefreshcallback fires whenever a new token is generated, so calling getToken in its context ensures that you are accessing a current, available registration token.
- **Token getToken():** Returns a token that authorizes an Entity to perform an action on behalf of the application identified by Instance ID.
- **void deleteInstanceId():** Resets Instance ID and revokes all tokens.

6.2.6 AsyncTask

AsyncTask enables proper and easy use of the UI thread. This class allows you to perform background operations and publish results on the UI thread without having to manipulate threads and/or handlers.

Methods

- **void onPreExecute():** Called on the UI thread before the thread starts running. This method is usually used to setup the task, for example by displaying a progress bar.
- **String doInBackground(strs[]…):** This is the method that runs on the background thread. In this method implemented operation code to perform in background.

- **void onPostExecute(String s)**: Called on the UI thread after the background thread finishes. It takes as parameter the result received from doInBackground().

6.2.7 Http Post

- **InputStream:** This abstract class is the superclass of all classes representing an input stream of bytes.
- **List<>:** The List interface extends Collection and declares the behavior of a collection that stores a sequence of elements.
- **ArrayList<>():** Java ArrayList class uses a dynamic array for storing the elements. It inherits AbstractList class and implements List interface.
- **List.add(new BasicNameValuePair("id", id.trim())):** Inserts object into the invoking list at the index passed in the index. Any pre-existing elements at or beyond the point of insertion are shifted up. Thus, no elements are overwritten.
- **BasicNameValuePair:** A name / value pair parameter used as an element of HTTP messages.
- **HttpClient:** This interface represents only the most basic contract for HTTP request execution. It imposes no restrictions or particular details on the request execution process and leaves the specifics of state management, authentication and redirect handling up to individual implementations.
- **DefaultHttpClient:** Default implementation of HttpClient pre-configured for most common use scenarios.
- **HttpPost:** The POST method is used to request that the origin server accept the entity enclosed in the request as a new subordinate of the resource identified by the Request-URI in the Request-Line.
- **HttpPost.setEntity(new UrlEncodedFormEntity(nameValuePairs)):** Used to set Entity to HttpPost request.
- **UrlEncodedFormEntity(nameValuePairs):** An entity composed of a list of url-encoded pairs. This is typically useful while sending an HTTP POST request.
- **HttpResponse:** After receiving and interpreting a request message, a server responds with an HTTP response message.

- **HttpResponse HttpClient.execute(httpPost):** Executes HTTP request.
- **HttpEntity:** An entity that can be sent or received with an HTTP message. Entities can be found in some requests and in responses, where they are optional.
- **HttpEntity HttpResponse.getEntity():** Obtains the message entity of this response, if any.
- **InputStream HttpEntity.getContent():** Returns a content stream of the entity.
- **BufferedReader:** This is used to read the text from a character-based input stream
- **InputStreamReader:** This is a bridge from byte streams to character streams. It reads bytes and decodes them into characters using a specified charset.
- **StringBuilder:** This class is mutable sequence of characters. This provides an API compatible with StringBuffer, but with no guarantee of synchronization.
- **String BufferedReader.readLine():** It is used for reading a line of text.
- **StringBuilder.append(line + "\n"):** This method appends the string representation of the boolean argument to the sequence.

6.2.8 Other Methods and Classes

Methods

- **String String.trim():** The java string trim() method eliminates leading and trailing spaces
- **boolean String.isEmpty():** This method checks if this string is empty.
- **Toast Toast.makeText():** Make a standard toast that just contains a text view with the text from a resource.
- **void Toast.show(context,"Register your Service",Toast. LENGTH_ LONG)**: Show the view for the specified duration.
- **Snackbar SnackBar.make(view, "Register your Service", SnackBar. LENGTH _LONG):** Make a Snackbar to display a message.
- **void SnackBar.show():** Shows Snackbar.
- **void SharedPreferences.Editor.putString("id",id):** Set a String value in the preferences editor.

Classes

- **Toast:** A toast provides simple feedback about an operation in a small popup. It only fills the amount of space required for the message.
- **SnackBar:** Snackbars provide lightweight feedback about an operation. They show a brief message at the bottom of the screen on mobile and lower left on larger devices.
- **AlertDialog:** A class that contains properties and methods related to AlertDialog.
- **NetworkInfo:** Describes the status of a network interface.
- **NotificationCompat.Builder:** Builder class for NotificationCompat objects. Allows easier control over all the flags, as well as help constructing the typical notification layouts.
- **PendingIntent:** A description of an Intent and target action to perform with it.
- **NotificationManager:** Class to notify the user of events that happen. This is how you tell the user that something has happened in the background.
- **SharedPreferences.Editor:** Interface used for modifying values in a SharedPreferences object. All changes you make in an editor are batched, and not copied back to the original SharedPreferences until you call commit() or apply().
- **NavigationView:** Represents a standard navigation menu for application. The menu contents can be populated by a menu resource file.
- **RecyclerView:** A flexible view for providing a limited window into a large data set.
- **ProgressDialog:** Use a progress indicator such as ProgressBar inline inside of an activity rather than using this modal dialog.
- **Bundle:** A mapping from String keys to various parcelable values.
- **MenuInflater:** This class is used to instantiate menu XML files into Menu objects.
- **AppCompatActivity:** Base class for activities that use the support library action bar features. You can add an ActionBar to your activity when running on API level 7 or higher by extending this class for your activity and setting the activity theme to Theme.AppCompat or a similar theme.

6.3 SERVER SIDE NODEJS IMPLEMENTATION

➢ **express**

Express is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications. It facilitates the rapid development of Node based Web applications.

- **app.use([path,] callback [, callback...])**

Mounts the specified middleware function or functions at the specified path: the middleware function is executed when the base of the requested path matches path.

- **app.get(path, callback [, callback ...])**

Routes HTTP GET requests to the specified path with the specified callback functions.

- **app.listen(port, [hostname], [backlog], [callback])**

Binds and listens for connections on the specified host and port. This method is identical to Node's http.Server.listen().

- **app.post(path, callback [, callback ...])**

Routes HTTP POST requests to the specified path with the specified callback functions

- **express.static(root, [options])**

This is a built-in middleware function in Express. It serves static files and is based on serve-static.

➢ **request**

- **req.body**

Contains key-value pairs of data submitted in the request body. By default, it is undefined, and is populated when you use body-parsing middleware such as body-parser and multer.

- **req.cookies**

When using cookie-parser middleware, this property is an object that contains cookies sent by the request. If the request contains no cookies, it defaults to {}.

- **req.params**

This property is an object containing properties mapped to the named route "parameters".

➢ **response**

The res object represents the HTTP response that an Express app sends when it gets an HTTP request.

- **res.sendFile(path [, options] [, fn])**

Transfers the file at the given path. Sets the Content-Type response HTTP header field based on the filename’s extension. Unless the rootoption is set in the options object, path must be an absolute path to the file.

- **res.send([body])**

Sends the HTTP response. This method performs many useful tasks for simple non-streaming responses: For example, it automatically assigns the Content-Length HTTP response header field (unless previously defined) and provides automatic HEAD and HTTP cache freshness support.

- **res.redirect([status,] path)**

Redirects to the URL derived from the specified path, with specified status, a positive integer that corresponds to an HTTP status code . If not specified, status defaults to “302 “Found”.

➢ **body-parser**

Node.js body parsing middleware. Parse incoming request bodies in a middleware before your handlers, available under the req.bodyproperty.

- **bodyParser.urlencoded([options])**

Returns middleware that only parses urlencoded bodies and only looks at requests where the Content-Type header matches the type option.

➢ **excel**

Native node.js Excel file parser. Only supports xlsx for now.

- **XLSX.readFile():** To readthe file.
- **workbook.Sheets():** To read a sheet of excel workbook.

➢ **fcm-node**

A Node.JS simple interface to Google's Firebase Cloud Messaging (FCM). Supports both android and iOS, including topic messages, and parallel calls.

Aditionally it also keeps the callback behavior for the new firebase messaging service.

- **fcm.send():** To send notification data to FCM server.

➢ **mysql**

This is a node.js driver for mysql. It is written in JavaScript, does not require compiling, and is 100% MIT licensed.

- **mysql.createConnection() :** To create database connection.
- **con.query() :** To execute SQL query.

➢ **nodemailer**

A framework which supports to send emails.

- **transporter.sendMail():** Sends email to recipient.

➢ **formidable**

A Node.js module for parsing form data, especially file uploads.

- **IncomingForm():** Creates a new incoming form.

➢ **events**

Many objects in a Node emit events, for example, a net.Server emits an event each time a peer connects to it, an fs.readStream emits an event when the file is opened. All objects which emit events are the instances of events.EventEmitter.

- **addListener(event, listener):** Adds a listener at the end of the listeners array for the specified event.
- **emit(event, [arg1], [arg2], [...]):** Execute each of the listeners in order with the supplied arguments. Returns true if the event had listeners, false otherwise.

➢ **Other methods**.

- **require():** This will include or import NodeJS frameworks.

CHAPTER 7

TESTING

Testing is a process of executing a program to ensure that defined input will produce actual results that agree with required outputs. In developing a software project, error can be initiated at any stage during the development. For each phase of the software development cycle there are different techniques for detecting and elimination errors that originate in that phase. However some errors will reflect in the code. Testing performs a very crucial role for quality assurance and for ensuring the reliabilities of the software. The quality of the system depends on its design, development, testing and implementation. Weaknesses in any of these areas will seriously affect the quality and therefore value of the system to its users. Once the code has been generated, testing of the modules begins implementation ends with formal tests.

7.1 BLACK BOX TESTING:

Internal system design is not considered in this type of testing. Tests are based on requirements and functionality. This method is named so because the software program, in the eyes of the tester, is like a black box; inside which one cannot see. Black box testing is a testing technique that ignores the internal mechanism of the system and focuses on the output generated against any input and execution of the system. It is also called functional testing.

7.2 SYSTEM TESTING

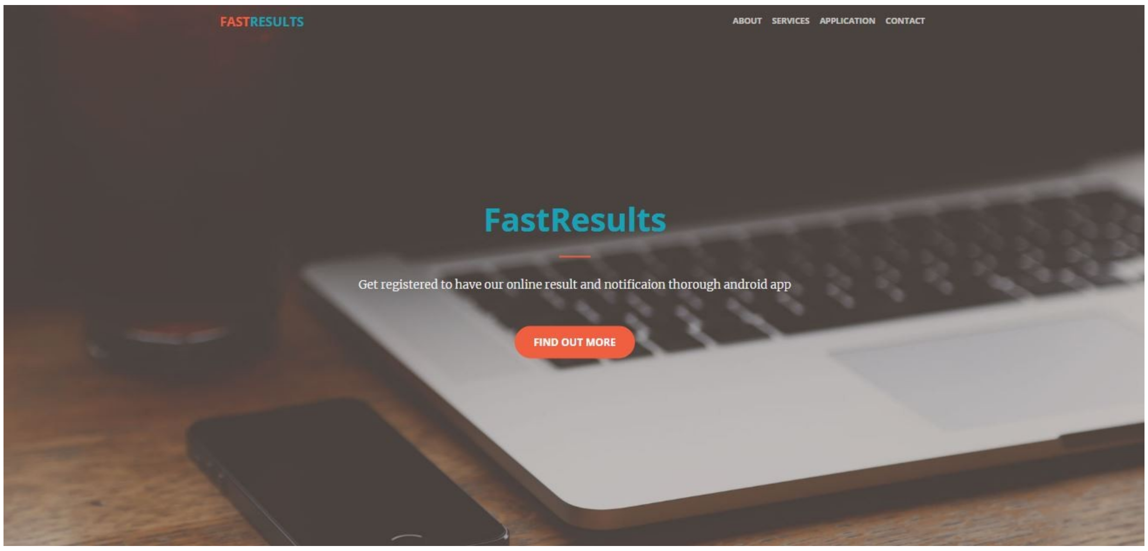
7.2.1 TEST CASES

ID	Test case name	Test case description	Test steps				Test status P/F
			Step	I/p given	Expected o/p	Actual o/p	
1	Registration	To check registration process	Select Register button from main	Required details for registration	Registration Successful	Registration Successful	Pass

			page				
2	Registration	To check that the user has entered all required details	Selects Register button from main page	Any required details for registration is missing	Please fill all details	Please fill all details	Pass
3	Login	To verify login process	Selects Login from register page	Username & Password	Login Successful	Login Successful	Pass
4	Login	Unregistered person login try	Selects Login from register page	Non valid Username & Password	User not registered	User not registered	Pass
5	Result view	Check reliability of result feature	After successful login	-	Academic result.	Result of logged in person.	Pass
6	Notification	Check notification process	Send notification button	-	Result Notification	Result Notification	Pass

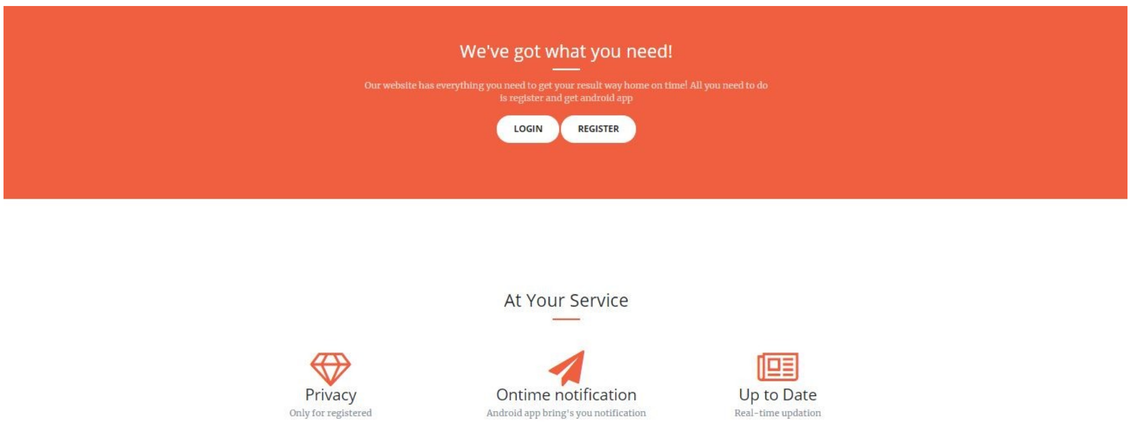
7.3 MANUAL TESTING

7.3.1 SCREENSHOTS



FastResults

Get registered to have our online result and notificaion thorough android app



We've got what you need!

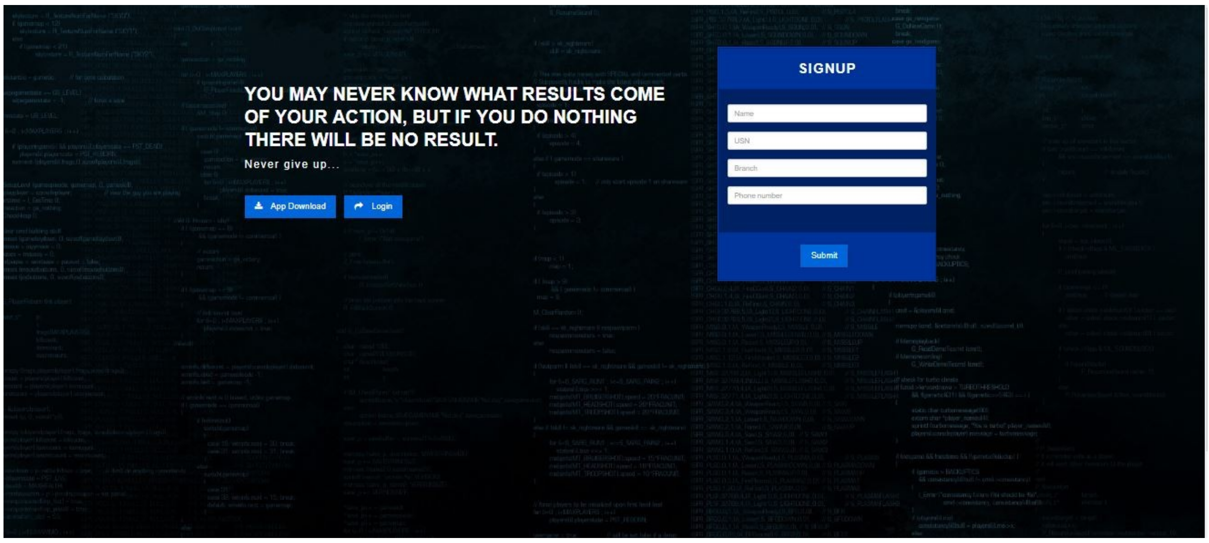
Our website has everything you need to get your result way home on time! All you need to do is register and get android app

At Your Service

Privacy
Only for registered

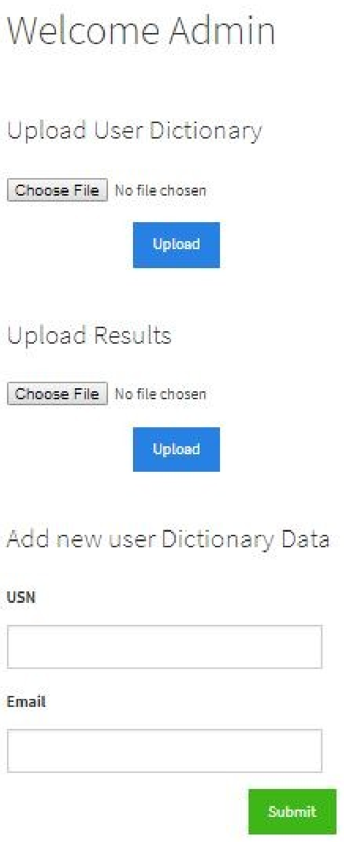
Ontime notification
Android app bring's you notification

Up to Date
Real-time updation



YOU MAY NEVER KNOW WHAT RESULTS COME OF YOUR ACTION, BUT IF YOU DO NOTHING THERE WILL BE NO RESULT.

Never give up...



Welcome Admin

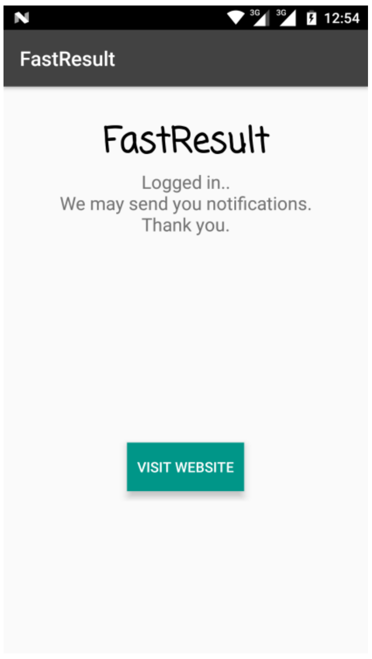
Upload User Dictionary

Upload Results

Add new user Dictionary Data

USN

Email



FastResult

Logged in..
We may send you notifications.
Thank you.

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENT

8.1 Conclusion

FastResults is helpful for the colleges to announce academic results and also to the students to view their respective academic scores. It also helps to bridge the gap between Examination department and the students while exchanging information and notifications.

8.2 Future Enhancements

FastResults can be enhanced as a connecting portal for alumni students and students who are currently perceiving in the college. It can also be developed as a resource centre for academic resources.