**CAMPUS NOTIFICATION APPLICATION**

For Android Phones

REPORT

SUBMITTED FOR THE PARTIAL FULFILLMENT OF THE DEGREE

at

Government College for girls ,Gurgaon

SUBMITTED BY

MANISHA ( 4612817)

RASHI BHAKHRI (4612831)



COMPUTER SCIENCE DEPARTMENT

MAHARSHI DAYANAND UNIVERSITY

ROHTAK

**Certificate from the Guide**

This is to certify that the project report entitled “CAMPUS BASED NOTIFICATION” Submitted in partial fulfillment of the degree of **MASTERS OF COMPUTER APPLICATION (MCA)**  to Maharshi Dayanand University , Rohtak is carried out by Ms. Manisha (roll no. 4612817) and Ms. Rashi Bhakhri (roll no. 4612831) semester IV under my guidance.

**Head Of Department**

Name: Mr. SandeepMaan

Date:

**ACKNOWLEDGEMENT**

Project work in an assignment whereby the candidates coming out of the academic field to get exposure to the persons who are on the job. Therefore it affords an opportunity of learning from others experience and exposure.

We sincerely express our deep sense of gratitude to our project supervisor Mr. Sandeep mann sir who has been a great help to us, supporting & guiding us throughout the project work. We would once again like to express our heartiest gratitude to all who helped us directly or indirectly throughout the project work.

**TABLE OF CONTENTS**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | | **ABSTRACT** | **5** | | 1. **INTRODUCTION**     1. OBJECTIVE    2. SCOPE OF THE PROJECT    3. IDENTIFICATION OF NEEDS    4. SDLC MODEL | **6**  **7**  **8**  **9** | | 1. **SYSTEM ANALYSIS**   2.1 EXISTING SYSTEM & DRAWBACK  2.2 PROPOSED SYSTEM  2.3 SOFTWARE REQUIREMENT SPECIFICATION  2.3.1 DATA REQUIREMENTS  2.3.2 FUNCTIONAL REQUIREMENTS  2.3.3 SYSTEM DEPENDABILITY  2.3.4 MAINTAINABILITY REQUIREMENTS  2.4 VALIDATION | **12**  **14**  **16**  **18** | | **3. SYSTEM DESIGN**  3.1 DETAILED DESIGN  3.2 USER INTERFACE DESIGN  3.3 DATABASE DESIGN | **19** | | **4. SYSTEM IMPLEMENTATION**  4.1 MODULE DESCRIPTION  4.1.1 CODING:USER INTERFACE  4.1.2 CODING:FIREBASE  4.1.3 CODING:APP SERVER | **30** | | **5. SYSTEM TESTING**  5.1 TEST PLAN  5.2 TEST ACTIVITY | **45** | | **6. CONCLUSION & FUTURE SCOPE** | **48** | |

**ABSTRACT**

Campus Notification System is a project which aims in developing an android application to update the students and staff of daily campus activities . This project aims in digitization of notice boards .

To physically visit notice board can be tedious. This may result in delays and misinformation . To overcome this we have come up with an idea that will bring notices to you anytime , anywhere.

The purpose behind selecting android is its flexibility and simplicity. It is an end to end software platform that can be adapted to work on any configuration. By taking an advantage of this application students can get notifications on their cell phone.

An authorized person (the one who post the Notice) will not only send the notice but also attach any document. The application is designed in such a way that only specific branch of students will get notice when an authorized person of particular department will send it.

**INTRODUCTION**

Notice Application it is fluke for students and staff all can access this application through mobile phones ,there is no need to go and visit notice board.

This application will help students and staff to get all notifications regarding studies, institute and also they will get all updates about college department and Corporate .

It is helpful for all updates and notification, it is simply way to convey message for students and staff. It is best facility for everyone because at one click students and staff can get all notices , updates and Information. This system applied in educational institutes as an online path of notice board and sharing system that can access and use anytime ,anywhere that’s means all in you are hand.

It is about virtual system that everyone can access and fill any application form regarding college and studies. In today’s world everything is digitalized and paper is being used less every days and our application is also useful and helpful for reducing paper works in school, colleges, private and government organizations. This Application one step towards

**“Go Green , Save Trees...!”.**

These are the some features of Notice Application should have:

1) Notice can be posted with response obtained instantly as pop-up notification like other applications.

2) Notice administrator can push all important notices in to selected staffs email.

3) Notice administrator can create any notice category.

4 )Advanced dashboard for updates all information out of staff and students.

**1.1 OBJECTIVE**

The proposed system’s objectives are to overcome all the limitations and drawbacks of the existing system.

The online Notice application is user-friendly android application. The main objective of the application is its simplicity of design and ease of implementation that shows and helps to collect most of the information about events going on in college premises .The interface will be very user-friendly.

The main objectives of the proposed system can be enumerated as follows:

* Faster dissemination of notices regarding education, technical events, cultural events. Any lost/found going out in college.
* Easy way to broadcast your message.
* Helps you to be updated with what’s going on in College.
* Good way to advertise about Tuitions/ Coaching and Courses.
* User can also follow a group notice board.

**1.2 SCOPE OF THE PROJECT**

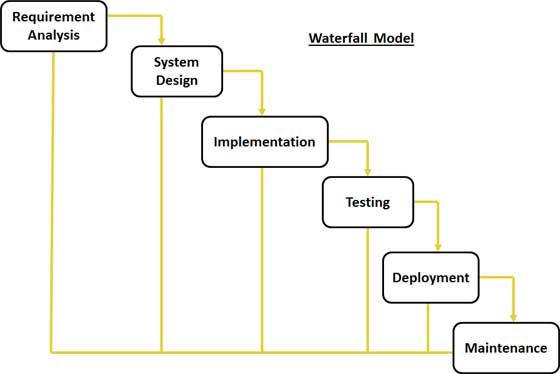
* This system is useful for accessing notice on android device.
* It informs students about the activities and various events related to their college.
* Student does not meet to visit the notice board every time.
* This system is also used in hospital and banking sector.

**1.3 IDENTIFICATION OF NEED**

* As we discussed earlier that manual maintenance of a notices is a tedious job. So to enhance the ease of working, we go for this package.
* Giving the facility to convey messages to all students anytime and anywhere.
* Making students updated about all the events and activities going on in the college.
* The student will not require to stand in the crowd to see the notice. There wil be no issue of fighting in order to see the notice first. Everyone is first to see that notice inside their own mobile phone anywhere and anytime.
* The least but most important it saves time.
* Utilizing less man power. As there are many persons involved in circulating the message. With this application, only one person is required to post the notice. Rest of the man power is saved in the entire process.

**SDLC Model Used**

This section describes the project as per the various stages of the Software Development life cycle. The model of software development life cycle used in this project is the waterfall method. The Waterfall Method is comprised of a series of very definite phases, each one run intended to be started sequentially only after the last has been completed, with one or more tangible deliverables produced at the end of each phase of the waterfall method of SDLC. Essentially, it starts with a heavy, documented, requirements-planning phase that outlines all the requirements for the



project, followed by sequential phases of design, coding, test-casing, optional documentation, verification (alpha-testing), validation (beta-testing), and finally deployment/release.

**1. Requirement Analysis:**

Existing system is time consuming and it makes difficult to convey huge amount of users about any event, class or seminar almost instantly. Also there is always a big crowd in front of notice board. So it was hectic to read any useful instruction and information. Thus all the problems of the

existing system are summarized and proposing a new system that works as an online application. It is a value added solution to the problem. It resolves all the problems stated above. It will provide simple interface to the user to operate on and convey the intended users about events almost instantly, anytime and anywhere.

1. **Design:**

It includes translation of the requirements specified in the SRS into a logical structure that can be implemented in a programming language. The output of the design phase is a design document that acts as an input for all the subsequent SDLC phases. The design of this app is simple and user friendly containing six main activities, namely:

* Register
* Dashboard
* Details of Notices

**3.Coding / Implementation:**

It includes translation of the requirements specified in the SRS into a logical structure that can be implemented in a programming language. The output of the design phase is a design document that acts as an input for all the subsequent SDLC phases. The project is implemented using the Android virtual devise (AVD). This emulator helped to implement the project in a real-like environment and sketch out the details of how it will work on a real hardware. Each activity is linked with another and interconnectivity is transparent and smooth.

**4. Testing:**

It includes detection of errors in the application. The testing process starts with a test plan that recognizes test-related activities, such as test case generation, testing criteria, and resource alloca-tion for testing. The code is tested and mapped against the design document created in the design phase. The output of the testing phase is a test report containing errors that occurred while testing the application. Testing of the project has not been done on real hardware and also on the emulator or software environment. Testing has been done for each of the individual activities of the project.

**5.Maintenance:**

It includes implementation of changes that software might undergo over a period of time, or implementation of new requirements after the software is deployed at the customer location. The maintenance phase also includes handling the residual errors that may exist in the software even after the testing phase. The project maintenance is low cost and efficient as user will get this application at free of cost and also this application is shared over network, therefore maintenance is little bit difficult.

**SYSTEM ANALYSIS**

System analysis is conducted for the purpose of studying a system or its parts in order to identify its objectives.

It is a problem solving technique that improves the system and ensures that all the components of the system work efficiently to accomplish their purpose.

Analysis specifies what the system should do.

**2.1 EXISTING SYSTEM & DRAWBACKS**

Currently our college has manual system of putting notices on notice board.



* **Order of Data:** Notice can get out of order in traditional notice board system. If someone accidentally puts some data in the wrong place. it can lead to lost data. Automated notice management systems allow users to quickly check whether information already exists somewhere in the system, which helps avoid problems like redundant data.
* **Complexity:** Automated system is less complex than manual system of handling notices, which can make it easier for untrained people to access and manipulate data. Anyone having the basic knowledge of mobiles can work on the automated system.
* **Inconsistency of data:** There will be an unavailability for future use, since notice might get misplaced during manual notices management. So notice won’t be preserved properly for future use.
* **Damage:** Manual notices stack are vulnerable to damage, destruction and theft in ways that digital databases are not.. A manual database, however, may only exist in one place without any copies.

While a digital database will typically allow users to search the entire database for specific information in seconds, someone looking for information in a manual system may have to spend hours searching for a particular piece of data.

* **Editing and Communication:** Manual notices do not allow users to easily edit data or information. Manual notices often cannot be edited directly, forcing users to make new copies. To circulate notice on paper, users must require peons and other staff. Notice app allow users to edit information fields directly, and because data is stored digitally, it is already in a form that can be easily transmitted.

**2.2 PROPOSED SYSTEM**

Proposed System will be able to do the following:

1. **To eliminate wastage of time and energy:**

Notice app will be able to save lot of paper and time. It directs both teacher and pupils energy and attention to one thing at a time by placing proper persons at their proper places at the proper time. Everything will be instantaneous.

1. **To avoid duplication and overlapping:**

This application will help to remove duplicate notices. Only admin can post the notice. No one else would be able to do so. So, student and staff will be given correct information all the time.

1. **To ensure due attention of student to each and every notice:**

Notice App ensures that everyone has kind attention to every notice and updates going on in college. There will be a buzz at each and every notice to drive the attention of student to check it once. In this way, students will be well informed about their college activities.

1. **To bring system into college life:**

It would be dire need of all colleges as it’s easy and shortcut method to inform all the students. In the absence of proper notification system will make it very difficult to inform students at right time.

1. **Searching a particular Notice:**

This application allows you to search the notice very easily through title of notice. If anyone forgets about the notice details, he can search it out very easily.

1. **Free Service:**

It gives free service to notify all the students. There will be no cost of sending notification to all. Just have the good system implemented in college and that too free of cost.

1. **Prevent Crowd in College:**

As you can see, there is always a crowd at notice board. As notice board is one, and people to see notice are more. Everyone will be well informed even at their homes. So they are free to do there other work.

1. **Anytime Anywhere Service:**

With this application, notices will be delivered anytime and at any place. There is no restriction of time to send a notice.

1. **Keeping Notices at one place:**

This application allows you to have notices in one place only. If there is an attachment with that, all will be placed in a separate folder dedicated to that application. So there will be no here and there of notices.

**2.3 SOFTWARE REQUIREMENT SPECIFICATION**

**2.3.1 Data Requirements**

Data requirement is meant to be the data that will be used in our application. Data required in this project is all notices, that need to be conveyed to the user. This application also require the username and passwords of persons in order to register them and sending notification about updates. So two main requirements are:

* Notice Details
* User Details

**2.3.2 Functional Requirements**

In order to make this application functional, we require the following:

* Download mobile application:

A user should be able to download the mobile an application through either an application store or similar service on the mobile phone. The application should be free to download.

* User registration:

Given that a user has downloaded the mobile application, then the user should be able to register through the mobile application. The user must provide user-name, password and e-mail address. The user can choose to provide a regularly used phone number.

* Dashboard:

Given that a user is logged in to the mobile application, then the first page that is shown should be the dashboard page. The user should be able to see all the college notices.

* View Notice:

The user should be able to view for a notice by its title.

* Posting Notices:

The registered person of this application should be able to post the notices.

* **Notification Alert:**

All the registered users should be able to have a ping or notification on their mobile phone whenever a new notice is posted.

**2.3.4 System Dependability**

Following are the requirements that an application require from the device/mobile on which it is installed.

* Internet Permission:

Application developed, require full internet permissions of mobile so that it can fetch notices from the server. At the same time, it should be able to receive buzz or notification tone whenever new notice is posted by admin.

* External SD Card Writable Permissions:

This application would be requiring read write access to SD card. It is required in order to download the notices attachment and save in SD card of mobile phone.

* Hardware Control:

It uses vibrator of mobile phone whenever any notification arrives.

* Account Info:

It also fetches your goggle account information in order to get the user registered with Firebase Cloud Messaging.

**2.3.5 Maintainability Requirements**

Following are the maintainability requirement of Notice mobile application

* Application extendibility:

The application should be easy to extend. The code should be written in a way that it favors implementation of new functions. It is requires in order for future functions to be implemented easily to the application.

* Application testability:

Test environments should be built for the application to allow testing of the applications different functions

.

**2.4 Validation**

Any application is useless without validation. There should be a way to validate the user input first before sending the user request to the server. Following are the validations implemented in proposed system:

* Validations During Registration:

There are a lot of validations that needs to be implemented in the application. They are as follow:

* + User type:

User type is an option to select the type user who has to register for the application

Ex. HOD, student, staff.

* + Username:

The username can contain only alphabets, digits, underscore and hyphen.

* Validating During Posting Notices:

The application should validate the notice posting fields before posting any notice. It should check whether title and description fields are filled or not. if not, it should tell the user to fill up the required fields while posting the noti

**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. It is meant to satisfy specific needs and requirements of a business or organization through the engineering of a coherent and well-running system.

**USE CASE DIAGRAM**

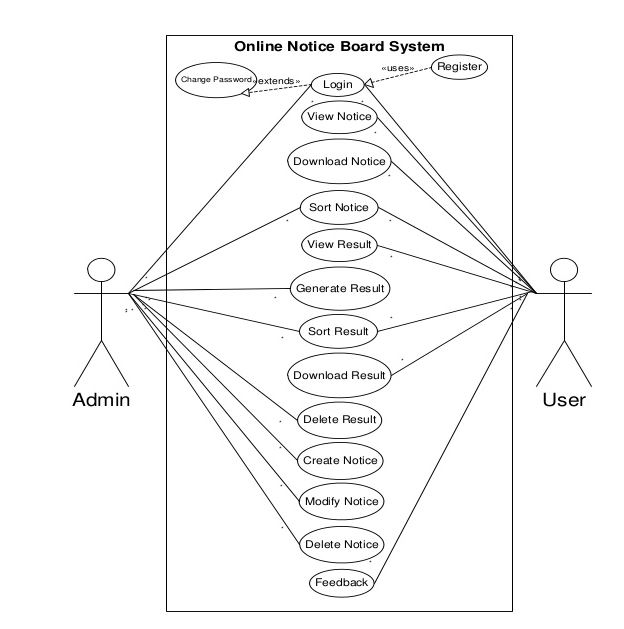
Actor

Actor

**USE CASE :**

A use case diagram is a representation of a user’s interaction with the system that shows the relationship between the actors and the different use cases in which user is involved.

**SEQUENCE DIAGRAM**

****

Interface

App server

Fcm

Send notification

Users ..n

username register device

store token

device registered

send (title,notice)

Add recepients

Receive notification

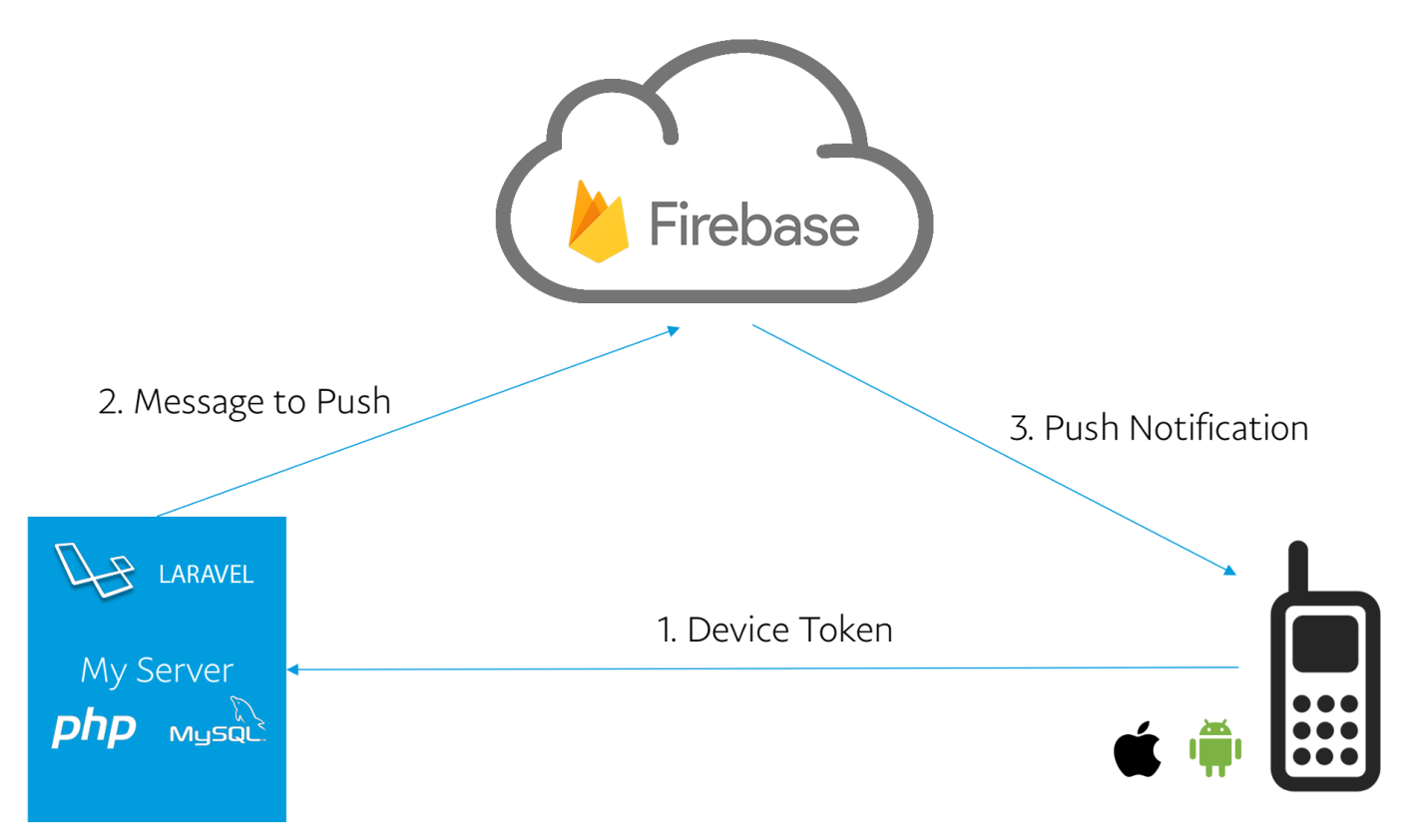
View (notification)

**SEQUENCE DIAGRAM**

The sequence diagram is a good diagram  to use to document a system's requirements and to flush out a system's design. It is so useful is because it shows the interaction logic between the objects in the system in the time order that the interactions take place.

3.2 **Detail Design**

The detailed design of this application is as follow:



\

**Firebase Cloud Messaging (FCM)** is a google cloud messaging service it is used for sending notification and messages in cross-platform for android, web application and ios application, which is available at no cost.

**Firebase** is a mobile development platform built by the Google to help the developers quickly **develop**high-quality apps,**grow** the user base, and**earn** extra cash by displaying ads.

Firebase comes with**everything that mobile developers need**to build, maintain and scale a mobile app on all platforms including ios: from Storage, Hosting and Databases to innovative tools like Remote Configuration and Test Lab.

**Firebase Cloud Messaging (FCM)** platform is another important service based on the precedent**Google Cloud Messaging (GCM)** service. FCM **inherits core infrastructure of GCM**but simplifies the client development. FCM integration can be completed with just lines of code instead of writing own registration or subscription retry complex logic as you did for GCM.

In this blog, we’re going to see server side integration of Firebase Cloud Messaging (FCM) in android app using php . If you want to use this service in your application please just follow below simple steps and then you can send notification using google FCM (Firebase Cloud Messaging) service.

**ABOUT FCM SERVER**

The server side of FCM consists of two main components:

* FCM connection servers provided by Google. These servers take messages from an app server and send them to a client app running on a device. Google provides connection servers for HTTP and XMPP.
* An app server that you must implement in your environment. This app server sends data to a client app via the chosen FCM connection server, using the appropriate XMPP or HTTP protocol.

A full FCM implementation requires both a client implementation and a server implementation to work perfectly.

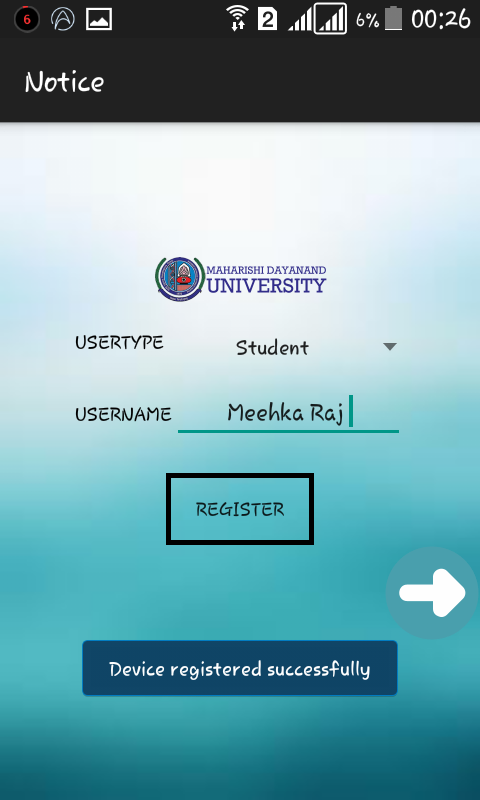
**3.3 USER INTERFACE DESIGN**

User Interface Design means the design of application with which the user interacts. So it should be kept in mind that UI should be very simple and easy to use. It should be simple enough in look and feel also.

1. **Registering a User:**

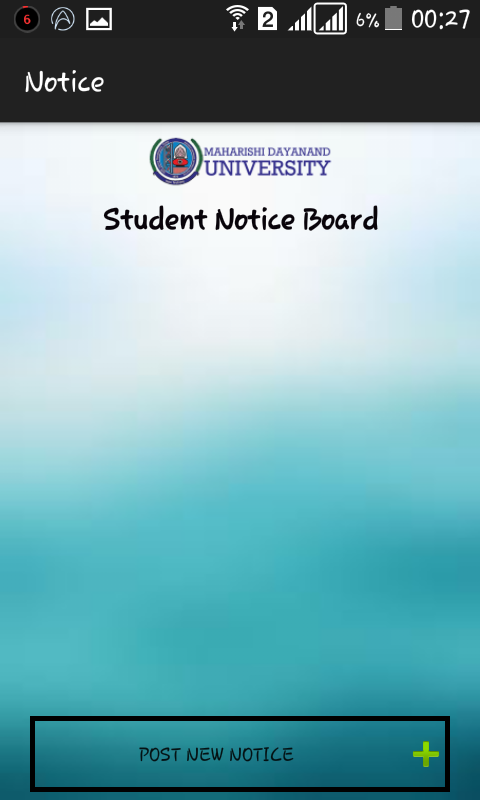
The first step in this application is to get the users registered to both FCM Server and to Remote Web Server. For this, user will provide all the necessary details and press the register button. The request will first go to Firebase Cloud Messaging Server. FCM Server will provide the registration id for that device. After that, all the information along with registration id is stored on Web Server and the user gets registered

.



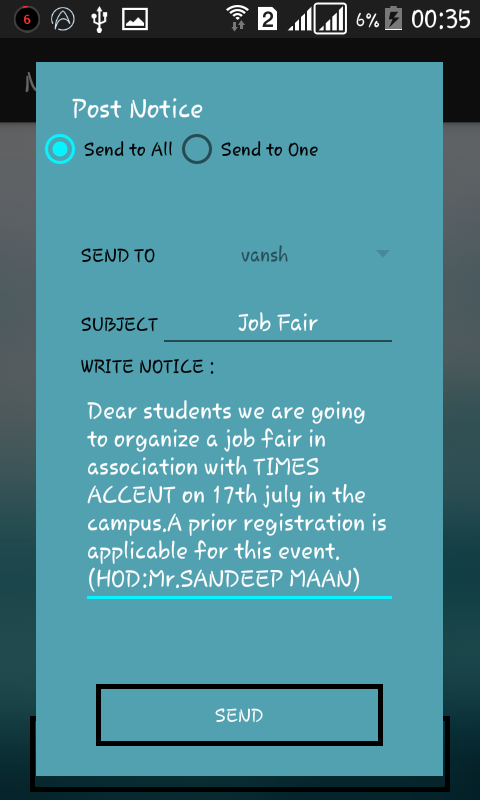
1. **Dashboard**

Given that a user is logged in to the mobile application, then the first page that is shown should be the dashboard page. The user should be able to see all the college notices.. it provides you interface for posting notices .



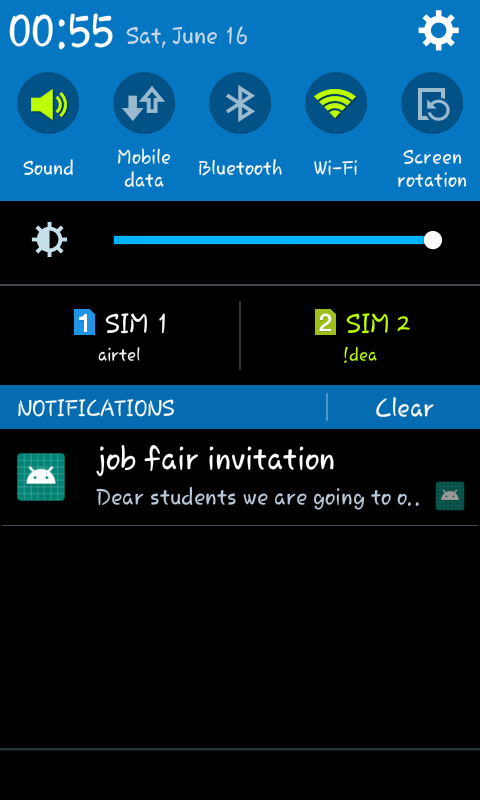
1. **Posting a Notice:**

If a user is an admin, he is able to post the notice. In order to post the notices, he has three option. One option is that, he can post a simple text notice. Another option allows him to send some attachment image with the notice. In this, he has two options. Either he can pick the image from the gallery or he can click a picture on the spot by using camera. After that, press the post button to post the notice.



1. **Notification Buzz:**

As soon as the admin post a notice, the script will run with which request is made by FCM Server to WebServer for all the registered IDs. After getting all the registered IDs, notification is sent to all the users registered with this application. Notification has a tune and vibration that runs whenever there is a notification received by the user from FCM Server.



4. **Viewing the Notices:**

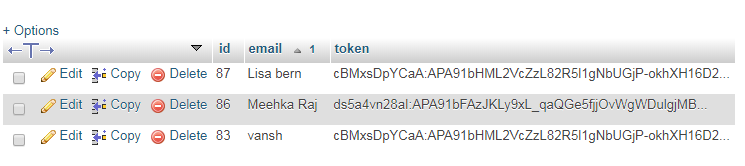
when you are using this application for the first time ,it will fetch all the notices from server. In all the other case, all previous notices are fetched from application’s own database stored inside client’s mobile. it then checks for new notices from the server. If there are new notices on the server, it will fetch all those notices.



Database Design

Database design is the process of producing a detailed data model of a database. This logical data model contains all the needed logical and physical design choices and physical storage parameters needed to generate a design in a data definition language, which can then be used to create a database. A fully attributed data model contains detailed attributes for each entity.

The term database design can be used to describe many different parts of the design of an overall database system. Principally, and most correctly, it can be thought of as the logical design of the base data structures used to store the data. In the relational model these are the tables and views. In an object database the entities and relationships map directly to object classes and named relationships. However, the term database design could also be used to apply to the overall process of designing, not just the base data structures, but also the forms and queries used as part of the overall database application within the database management system (DBMS).



**Implementation, Testing and Maintenance**

**4.1 Java**

Java is a very popular programming language developed by Sun Microsystems (now owned by Oracle). Developed long after C and C++, Java incorporates many of the powerful features of those powerful languages while addressing some of their drawbacks. Still, programming languages are only as powerful as their libraries. These libraries exist to help developers build applications. Some of the Java’s important core features are:

* Its easy to learn and understand.
* Its designed to be platform-independent and secure, using virtual machines. Its object-oriented.

Android relies heavily on these Java fundamentals. The Android SDK includes many standard Java libraries (data structure libraries, math libraries, graphics libraries, networking libraries and everything else you could want) as well as special Android libraries that will help you develop awesome Android applications.

Platform Independence Importance:

With many programming languages, you need to use a compiler to reduce your code down into machine language that the device can understand. While this is well and good, different devices use different ma-chine languages. This means that you might need to compile your applications for each different device or machine language in other words, your code isnt very portable. This is not the case with Java. The Java compilers convert your code from human readable Java source files to something called bytecode in the Java world. These are interpreted by a Java Virtual Machine, which operates much like a physical CPU might operate on machine code, to actually execute the compiled code. Although it might seem like this is inefficient, much effort has been put into making this process very fast and efficient. These efforts have paid off in that Java performance in generally second only to C/C++ in common language performance comparisons.

Android applications run in a special virtual machine called the Dalvik VM. While the details of this VM are unimportant to the average developer, it can be helpful to think of the Dalvik VM as a bubble in which your Android application runs, allowing you to not have to worry about whether the device is a Motorola Droid, an HTC Evo, or the latest toaster running Android. You dont care so long as the device is Dalvik VM friendly and that’s the device manufacturers job to implement, not yours.

Why is Java Secure?

Lets take this bubble idea a bit further. Because Java applications run within the bubble that is a virtual machine, they are isolated from the underlying device hardware. Therefore, a virtual machine can encapsulate, contain, and manage code execution in a safe manner compared to languages that operate in machine code directly. The Android platform takes things a step further. Each Android application runs on the (Linux- based) operating system using a different user account and in its own instance of the Dalvik VM. Android applications are closely monitored by the operating system and shut down if they don’t play nice (e.g. use too much processing power, become unresponsive, waste resources, etc.).

Therefore, its important to develop applications that are stable and responsive. Applications can commu-nicate with one another using well- defined protocol.

Android Development Tools

* Android SDK:

The Android Software Development Kit (Android SDK) contains the necessary tools to create, compile and package Android applications. Most of these tools are command line based. The primary way to develop Android applications is based on the Java programming language.

* Android debug bridge (adb):

The Android SDK contains the Android debug bridge (adb), which is a tool that allows you to connect to a virtual or real Android device, for the purpose of managing the device or debugging your application.

* Android Developer Tools and Android Studio:

Google provides two integrated development environments (IDEs) to develop new applications .The Android Developer Tools (ADT) are based on the Eclipse IDE. ADT is a set of components (plug-ins), which extend the Eclipse IDE with Android development capabilities. Google also supports an IDE called Android Studio for creating Android applications. This IDE is based on the Intelligent J IDE.

Both IDEs contain all required functionality to create, compile, debug and deploy Android applications. They also allow the developer to create and start virtual Android devices for testing. Both tools provide specialized editors for Android specific files. Most of Android’s configuration files are based on XML. In this case these editors allow you to switch between the XML representation of the file and a structured user interface for entering the data .Dalvik Virtual Machine The Android system uses a special virtual machine, i.e., the Dalvik Virtual Machine (Dalvik) to run Java based applications. Dalvik uses a custom bytecode format which is different from Java bytecode.

Therefore you cannot run Java class files on Android directly; they need to be converted into the Dalvik bytecode format.

Android RunTime (ART):

With Android 4.4, Google introduced the Android RunTime (ART) as optional runtime for Android 4.4. It is expected that versions after 4.4 will use ART as default runtime. ART uses Ahead Of Time compilation. During the deployment process of an application on an Android device, the application code is translated into machine code. This results in approx. 30% larger compile code, but allows faster execution from the beginning of the application.

**4.1.1 Coding : user interface**

**Registration,xml**

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:background="@drawable/download1"

android:orientation="vertical" >

<ImageView

android:id="@+id/logo"

android:layout\_width="match\_parent"

android:layout\_height="58dp"

android:layout\_marginTop="70dp"

app:srcCompat="@mipmap/mdulogo" />

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal"

android:layout\_marginLeft="50dp"

android:layout\_marginRight="50dp"

android:layout\_marginTop="10dp">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="USERTYPE"

android:textAlignment="center"

android:textColor="#000000"

/>

<Spinner

android:id="@+id/type"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="left"

android:textAlignment="center"

android:layout\_weight="1"/>

</LinearLayout>

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal"

android:layout\_marginTop="10dp"

android:layout\_marginLeft="50dp"

android:layout\_marginRight="50dp >

<TextView

android:id="@+id/email\_id"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:textAlignment="center"

android:text="@string/username"

android:textColor="#000000"

/>

<EditText

android:id="@+id/edit\_email"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:textAlignment="center"

/>

</LinearLayout>

<Button

android:id="@+id/register"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="center\_horizontal"

android:text="@string/register"

android:layout\_marginTop="20dp"

android:paddingLeft="20dp"

android:paddingRight="20dp"

android:background="@layout/button\_border\_settings"/>

<ImageView

android:id="@+id/next"

android:layout\_width="53dp"

android:layout\_height="wrap\_content"

android:layout\_gravity="end"

android:clickable="true"

android:onClick="onClick"

app:srcCompat="@mipmap/arrow" />

</LinearLayout>

**Push notication.xml**

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical"

>

android:layout\_width="match\_parent"

android:layout\_height="match\_parent">

<RadioGroup

android:id="@+id/radioGroup"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal">

<RadioButton

android:id="@+id/radioButtonSendAll"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Send to All" />

<RadioButton

android:id="@+id/radioButtonSendOne"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Send to One" />

</RadioGroup>

<LinearLayout

android:layout\_marginTop="30dp"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:orientation="horizontal"

android:paddingLeft="30dp"

android:paddingRight="30dp"

>

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="SEND TO"

android:textAlignment="center"

android:textColor="#000000"/>

<Spinner

android:id="@+id/spinnerDevices"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:textAlignment="center"

></Spinner>

</LinearLayout>

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingLeft="30dp"

android:paddingRight="30dp">

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="SUBJECT"

android:textAlignment="center"

android:textColor="#000000"/>

<EditText

android:id="@+id/editTextTitle"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:textAlignment="center" />

</LinearLayout>

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:paddingRight="30dp"

android:paddingLeft="30dp"

android:orientation="vertical" >

<TextView

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="WRITE NOTICE :"

android:textColor="#000000"

android:textAlignment="center"

/>

<EditText

android:id="@+id/editTextMessage"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:scrollbars="vertical"

/>

</LinearLayout>

<Button

android:id="@+id/buttonSendPush"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="Send"

android:layout\_marginLeft="40dp"

android:layout\_marginRight="40dp"

android:layout\_marginTop="50dp"

android:background="@layout/button\_border\_settings"

android:layout\_marginBottom="20dp"/>

</LinearLayout>

**4.1.2. Coding: firebase**

**Firebasetoken.java**

package com.example.acer.mynoticeboardapp;

import android.content.Intent;

import android.util.Log;

import com.google.firebase.iid.FirebaseInstanceId;

import com.google.firebase.iid.FirebaseInstanceIdService;

public class FirebaseToken extends FirebaseInstanceIdService

{

public static final String TOKEN\_BROADCAST ="mytoken";

@Override

public void onTokenRefresh() {

// Get updated InstanceID token.

String refreshedToken = FirebaseInstanceId.getInstance().getToken();

Log.d("mytoken", "Refreshed token: " + refreshedToken);

getApplicationContext().sendBroadcast(new Intent(TOKEN\_BROADCAST));

storeToken(refreshedToken);

}

private void storeToken(String token) {

//saving the token on shared preferences

SharedprefManager.getInstance(getApplicationContext()).saveDeviceToken(token);

}}}

**Pushnotification.java**

package com.example.acer.mynoticeboardapp;

import android.app.Activity;

import android.app.ProgressDialog;

import android.content.DialogInterface;

import android.os.Bundle;

import android.support.v7.app.AlertDialog;

import android.support.v7.app.AppCompatActivity;

import android.text.TextUtils;

import android.view.LayoutInflater;

import android.view.View;

import android.widget.ArrayAdapter;

import android.widget.Button;

import android.widget.EditText;

import android.widget.RadioGroup;

import android.widget.Spinner;

import android.widget.Toast;

import com.android.volley.AuthFailureError;

import com.android.volley.Request;

import com.android.volley.Response;

import com.android.volley.VolleyError;

import com.android.volley.toolbox.StringRequest;

import org.json.JSONArray;

import org.json.JSONException;

import org.json.JSONObject;

import java.util.ArrayList;

import java.util.HashMap;

public class ActivitySendPushNotification extends AppCompatActivity implements RadioGroup.OnCheckedChangeListener, View.OnClickListener {

private Button buttonSendPush;

private RadioGroup radioGroup;

private Spinner spinner;

private ProgressDialog progressDialog;

private EditText editTextTitle, editTextMessage, editTextImage;

private boolean isSendAllChecked;

private List<String> devices;

private void killActivity() {

finish();

}

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.send\_push\_notification);

this.setFinishOnTouchOutside(true);

radioGroup = (RadioGroup) findViewById(R.id.radioGroup);

spinner = (Spinner) findViewById(R.id.spinnerDevices);

buttonSendPush = (Button) findViewById(R.id.buttonSendPush);

editTextTitle = (EditText) findViewById(R.id.editTextTitle);

editTextMessage = (EditText) findViewById(R.id.editTextMessage);

// editTextImage = (EditText) findViewById(R.id.editTextImageUrl);

devices = new ArrayList<>();

radioGroup.setOnCheckedChangeListener(this);

buttonSendPush.setOnClickListener(this);

loadRegisteredDevices();

}

//method to load all the devices from database

private void loadRegisteredDevices() {

progressDialog = new ProgressDialog(this);

progressDialog.setMessage("Fetching Devices...");

progressDialog.show();

StringRequest stringRequest = new StringRequest(Request.Method.GET, EndPoints.URL\_FETCH\_DEVICES,

new Response.Listener<String>() {

@Override

public void onResponse(String response) {

progressDialog.dismiss();

JSONObject obj = null;

try {

obj = new JSONObject(response);

if (!obj.getBoolean("error")) {

JSONArray jsonDevices = obj.getJSONArray("devices");

for (int i = 0; i < jsonDevices.length(); i++) {

JSONObject d = jsonDevices.getJSONObject(i);

devices.add(d.getString("email"));

}

ArrayAdapter<String> arrayAdapter = new ArrayAdapter<String>(

ActivitySendPushNotification.this,

android.R.layout.simple\_spinner\_dropdown\_item,

devices);

spinner.setAdapter(arrayAdapter);

}

} catch (JSONException e) {

e.printStackTrace();

}

}

},

new Response.ErrorListener() {

@Override

public void onErrorResponse(VolleyError error) {

}

}) {

};

MyVolley.getInstance(this).addToRequestQueue(stringRequest);

}

//this method will send the push

//from here we will call sendMultiple() or sendSingle() push method

//depending on the selection

private void sendPush() {

if (isSendAllChecked) {

sendMultiplePush();

} else {

sendSinglePush();

}

}

private void sendMultiplePush() {

final String title = editTextTitle.getText().toString();

final String message = editTextMessage.getText().toString();

// final String image = editTextImage.getText().toString();

progressDialog.setMessage("Sending Push");

progressDialog.show();

StringRequest stringRequest = new StringRequest(Request.Method.POST, EndPoints.URL\_SEND\_MULTIPLE\_PUSH,

new Response.Listener<String>() {

@Override

public void onResponse(String response) {

progressDialog.dismiss();

Toast.makeText(ActivitySendPushNotification.this, response, Toast.LENGTH\_LONG).show();

}

},

new Response.ErrorListener() {

@Override

public void onErrorResponse(VolleyError error) {

}

}) {

@Override

protected Map<String, String> getParams() throws AuthFailureError {

Map<String, String> params = new HashMap<>();

params.put("title", title);

params.put("message", message);

/\* if (!TextUtils.isEmpty(image))

params.put("image", image);\*/

return params;

}

};

MyVolley.getInstance(this).addToRequestQueue(stringRequest);

}

private void sendSinglePush() {

final String title = editTextTitle.getText().toString();

final String message = editTextMessage.getText().toString();

// final String image = editTextImage.getText().toString();

final String email = spinner.getSelectedItem().toString();

progressDialog.setMessage("Sending Notice...");

progressDialog.show();

StringRequest stringRequest = new StringRequest(Request.Method.POST, EndPoints.URL\_SEND\_SINGLE\_PUSH,

new Response.Listener<String>() {

@Override

public void onResponse(String response) {

progressDialog.dismiss();

Toast.makeText(ActivitySendPushNotification.this, response, Toast.LENGTH\_LONG).show();

}

},

new Response.ErrorListener() {

@Override

public void onErrorResponse(VolleyError error) {

}

}) {

@Override

protected Map<String, String> getParams() throws AuthFailureError {

Map<String, String> params = new HashMap<>();

params.put("title", title);

params.put("message", message);

/\*if (!TextUtils.isEmpty(image))

params.put("image", image);\*/

params.put("email", email);

return params;

}

};

MyVolley.getInstance(this).addToRequestQueue(stringRequest);

}

@Override

public void onCheckedChanged(RadioGroup radioGroup, int i) {

switch (radioGroup.getCheckedRadioButtonId()) {

case R.id.radioButtonSendAll:

isSendAllChecked = true;

spinner.setEnabled(false);

break;

case R.id.radioButtonSendOne:

isSendAllChecked = false;

spinner.setEnabled(true);

break;

}

}

@Override

public void onClick(View view) {

//calling the method send push on button click

String title = editTextTitle.getText().toString();

String msg = editTextMessage.getText().toString();

if (TextUtils.isEmpty(title) || TextUtils.isEmpty(msg)) {

Toast.makeText(getApplicationContext(), "nothing to send", Toast.LENGTH\_LONG).show();

}

Else

{

sendPush();

progressDialog.dismiss();

killActivity();

}

}}

**4.1.3 Coding : app server**

**Dbconnect.php**

<?php

//Class DbConnect

class DbConnect

{

//Variable to store database link

private $con;

//Class constructor

function \_\_construct()

{

}

//This method will connect to the database

function connect()

{

//Including the config.php file to get the database constants

include\_once dirname(\_\_FILE\_\_) . '/Config.php';

//connecting to mysql database

$this->con = new mysqli(DB\_HOST, DB\_USERNAME, DB\_PASSWORD, DB\_NAME);

//Checking if any error occured while connecting

if (mysqli\_connect\_errno()) {

echo "Failed to connect to MySQL: " . mysqli\_connect\_error();

}

//finally returning the connection link

return $this->con;

}

}

**Sendmultipledevices.php**

<?php

//importing required files

require\_once 'DbOperation.php';

require\_once 'Firebase.php';

require\_once 'Push.php';

$db = new DbOperation();

$response = array();

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

//hecking the required params

if(isset($\_POST['title']) and isset($\_POST['message'])) {

//creating a new push

$push = null;

//first check if the push has an image with it

if(isset($\_POST['image'])){

$push = new Push(

$\_POST['title'],

$\_POST['message'],

$\_POST['image']

);

}else{

//if the push don't have an image give null in place of image

$push = new Push(

$\_POST['title'],

$\_POST['message'],

null

);

}

//getting the push from push object

$mPushNotification = $push->getPush();

$devicetoken = $db->getAllTokens();

$firebase = new Firebase();

echo $firebase->

send($devicetoken, $mPushNotification);

}else{

$response['error']=true;

$response['message']='Parameters missing';

}

}else{

$response['error']=true;

$response['message']='Invalid request';

}

echo json\_encode($response);

5 **System Testing**

**5.1 Test Plan**

A test plan can be defined as a document describing the scope, approach, resources, and schedule of intended testing activities. It identifies test items, the features to be tested, the testing tasks, who will do each task, and any risks requiring contingency planning. In software testing, a test plan gives detailed testing information regarding an upcoming testing effort

Scope of testing Schedule

Test Deliverables

Release Criteria

**Risks and Contingencies**

It is also be described as a detail of how the testing will proceed, who will do the testing, what will be tested, in how much time the test will take place, and to what quality level the test will be performed.

The process of defining a test project so that it can be properly measured and controlled. The test planning process generates a high level test plan document that identifies the software items to be tested, the degree of tester independence, the test environment, the test case design and test measurement tech-niques to be used, and the rationale for their choice.

A testing plan is a methodological and systematic approach to testing a system such as a machine or software. It can be effective in finding errors and flaws in a system. In order to find relevant results, the plan typically contains experiments with a range of operations and values, including an understanding of what the eventual workflow will be.Test plan is a document which includes, introduction, assumptions, list of test cases, list of features to be tested, approach, deliverables, resources, risks and scheduling. A test plan is a systematic approach

to testing a system such as a machine or software. The plan typically contains a detailed understanding of what the eventual workflow will be. A record of the test planning process detailing the degree of tester indedendence, the test environment, the test case design techniques and test measurement techniques to be used, and the rationale for their choice.

**5.2 Test Activities**

Various Testing Activities are as follow:

1. Black box testing Internal system design is not considered in this type of testing. Tests are based on requirements and functionality.
2. White box testing This testing is based on knowledge of the internal logic of an applications code. Also known as Glass box Testing. Internal software and code working should be known for this type of testing. Tests are based on coverage of code statements, branches, paths, conditions.
3. Unit testing Testing of individual software components or modules. Typically done by the pro-grammer and not by testers, as it requires detailed knowledge of the internal program design and code. may require developing test driver modules or test harnesses.
4. Incremental integration testing Bottom up approach for testing i.e continuous testing of an appli-cation as new functionality is added; Application functionality and modules should be independent enough to test separately. done by programmers or by testers.
5. Integration testing Testing of integrated modules to verify combined functionality after integra-tion. Modules are typically code modules, individual applications, client and server applications on a network, etc. This type of testing is especially relevant to client/server and distributed systems.
6. Functional testing This type of testing ignores the internal parts and focus on the output is as per requirement or not. Black-box type testing geared to functional requirements of an application.
7. System testing Entire system is tested as per the requirements. Black-box type testing that is based on overall requirements specifications, covers all combined parts of a system.
8. End-to-end testing Similar to system testing, involves testing of a complete application environ-ment in a situation that mimics real-world use, such as interacting with a database, using network communications, or interacting with other hardware, applications, or systems if appropriate.
9. Acceptance testing - Normally this type of testing is done to verify if system meets the customer specified requirements. User or customer do this testing to determine whether to accept application.
10. Usability testing User-friendliness check. Application flow is tested, Can new user understand the application easily, Proper help documented whenever user stuck at any point. Basically system navigation is checked in this testing.

Conclusion and Future Scope

6.1 Future Scope

The future scope of the project is that it can be used as any news giving application or it can be used

to advertise your products, telling the customers about new schemes and products coming to your shop.

This application of e-Notice can be further extended to include the following features:

**1. Categorization of Notice:**

Notices can be categorized in different categories, so that its possible for user to easily manage the

notices. Categorization can also be done by making groups. Defining the notice to be circulated in

a particular group can make it more secure.

**2. Documents and PDF files:**

The attachments can be further improved to include PDF files or Doc files. Then there will not be

much need to send images with the notices. A single file would serve all the purposes.

**3. Feedback:**

Feedback on the notices can also be taken. It can increase communication among connected members and any issue can be easily sorted out on the spot.

**6.2 Conclusion**

I learned a lot by doing this project.

\_ Operating system: Ubuntu

\_ Languages used: Java, Android UI, PHP for backend

\_ Servers Used: WAMP Server, FCM Server

\_ Database: MySQL, SQLite

So during this project I learned all the above things. Before this project, I had no idea about Java and Android for making application. Although I had little bit knowledge of Ubuntu before. But now I learned a lot about Ubuntu and got knowledge of using Android and Java for developing mobile application and PHP for server side scripting. Now I prefer to work on command line rather than graphically. I learned

**how to work on shell script.**

If I talk about the project, e-Notice Application has reduced lot of manual work. It has made notififying each and every user very easy and taht too with no time and place restrictions.