CLOUD-BASED, SMART NOTICE DELIVERY SYSTEM USING OCR

Project report submitted in partial fulfillment of the requirement for the degree of Bachelor of Technology in
Computer Science and Engineering

By

Pulkit Aggarwal (141330)
Abhishek Giri (141332)

Under the supervision of
Mr. Amol Vasudeva

to

Department of Computer Science & Engineering
Jaypee University of Information Technology Waknaghat,
Solan-173234, Himachal Pradesh

CANDIDATE'S DECLARATION

We hereby declare that the work presented in this report entitled **"Cloud Based, Smart Notice Delivery System Using OCR"** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** submitted in the Department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat, is an authentic record of our own work carried out over a period from August 2017 to May 2018 under the supervision of **Mr. Amol Vasudeva** (Assistant Professor, Department of Computer Science & Engineering).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Pulkit Aggarwal, 141330

Abhishek Giri, 141332

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Amol Vasudeva
Assistant Professor,
Department of Computer Science & Engineering
Dated:

ACKNOGLEGEMENT

The completion of any project depends upon cooperation, co-ordination and combined efforts of several sources of knowledge. We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organisations. we would like to extend my sincere thanks to all of them.

We highly indebted to our project guide Mr Amol Vasudeva for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

We are also thankful to and fortunate enough to get constant encouragement, support and guidance from all Teaching staffs of Department of Computer Science which helped us in successfully completing our project work. Also, we would like to extend our sincere regards to all the non-teaching staff of Department of Computer Science for their timely support.

Pulkit Aggarwal
Abhishek Giri

TABLE OF CONTENTS:

LIST OF ABREVATIONS

- **OCR**: Optical Character Recognition
- **FCM:** Firebase Cloud Messaging
- **KDC**: Key Distribution Centre
- **OAuth**: Open Authentication
- **REST**: Representational State Transfer
- **MIT**: Massachusetts Institute of Technology
- **AS**: Authorization Server
- **RS:** Resource Serve
- **RP:** Resource Person
- **API**: Application Program Interface

LIST OF FIGURES

LIST OF GRAPHS

LIST OF TABLES

ABSTRACT

The project is based on to digitalize the notice delivery mechanism, in a smart and efficient way, by using various cloud services provided by firebase. The basic idea of this work was to get the daily updated of the university on time, and we need to keep track of all the notice boards, almost daily. Moreover, the life of a printed notice was hardly two three days, then it may get discarded, and even sometimes a very critical information goes un-noticed from our end. Also, sometimes due to Notice Boards located in different places there have been issues in past when one notice was displayed on the boards where such notices were not displayed usually and it creates problems for students. Moreover, in past we have seen that exam date-sheet are sometimes updated nearby the examination and sometimes at late evening which creates chaos.

1. INTRODUCTION

Our work in the final year project was to build up a Hybrid application which enables the students and faculties of our university to peruse effortlessly the Notices and Circulars in a more friendly and swift way at time and place of their choosing than Traditional notice board mechanism along with keeping a check on authenticity of sender and receiver. We intend that our work satisfy the needs of all Stakeholders involved.

1.1 Project Overview

The basic idea of this work was started from the incident, when we were unable to get the daily updated of the university on time, and we need to keep track of all the notice boards, almost daily. Moreover, the life of a printed notice was hardly two three days, then it may get discarded, and even sometimes a very critical information goes un-noticed from our end. Also, sometimes due to Notice Boards located in different places there have been issues in past when one notice was displayed on the boards where such notices were not displayed usually and it creates problems for students. Moreover, in past we have seen that exam date-sheet are sometimes updated nearby the examination and sometimes at late evening which creates chaos.

To work-around a new mechanism for students and faculties of Jaypee University of information technology to access notice and important information in a very efficient way instead of just scrutinizing the notice board periodically. We plan to propagate and broadcast such notices over a common platform, basically a mobile application, that will deliver and organize public notices on the go. A notice will be uploaded on the application portal only by an authorized person and each registered user will instantly receive the notification on their mobile devices.

Figure 1.1: A Printed notice

The project will not only reduce the paper wastage while also provide a quick and genuine source for the notices or information. Moreover, a person need not to be physically present to go through a new notice. The solution of all the problems is notice digitalization.so that notice can circulate across the globe.

1.1 Motivation

The whole idea this work has started from the incident, when we were unable to get the daily updated of the university on time, and we need to keep track of all the notice boards, almost daily. Moreover, the life of a printed notice was hardly two three days, then it may got discarded, and even sometimes a very critical information goes un-noticed from our end.

Then we think of to work-around a new mechanism for students and faculties of Jaypee University of information technology to access notice and important information in a very efficient way instead of just scrutinizing the notice board periodically .We all have seen large queues in front of notice board when schedule or other important instructions for students are to be passed on to them and moreover we have to periodically see the notice boards to check whether they are having some important Notices and when someone has gone out of university then he has to rely on his friends for providing him the information. Also, sometimes due to Notice Boards located in different places there have been issues in past when one notice was displayed on the boards where such notices were not displayed usually and it creates problems for students.

Figure 1.2: A regular notice board, full of notices

1.2 Problem Statement

In almost every college or organization, A single notice has to be pinned on all the available noticeboards (in print format) so that information can propagate to its max.
This results in frequent wastage of bunch of papers along with improper transmission of the important information.
Even sometimes, an important notice gets completely skipped for some specific reasons. Majorly due to delay in transmission and physical limitations of the print form of a notice.

1.3 Objectives

To reduce the paper wastage while also provide a quick and genuine source for the notices or information. Moreover, a person need not to be physically present to go through a new notice and to notify them for a notice published.
The notices of all kinds will be available on all the user's phones through a notification and will be visible in a sorted fashion according to the published time.

Figure 1.3: Identification symbol for the Notice Delivery Application

1.4 Target Audience

This project is a prototype for the notice delivery system and it is mainly targeted for the college students and faculties. To deliver public notices, on the go, directly to the user's device, is the main purpose to deal with. This project is useful for the college administration and as well as to the students.

The purpose of the smart notice delivery system is to efficiently deliver notices and to build a convenient & easy-to-use mobile application for students, usually navigate notice boards for daily updates. The whole mechanism is based on a NoSQL database, mainly FireStore database, with its notices storage and notification delivery functions. Moreover, we target to provide an impressive user experience along with the enhanced device capabilities.

1.5 Methodology

The process initializes when an admin logs into his account in admin portal using credentials that are his email id and password.

Then he clicks on the button "Draft A Notice" and the notice is uploaded on the server and the size of the uploaded file is checked if it is more than 2MB then the admin is prompted with an alert message or if the size is acceptable then the uploaded file moves to backend in a temporary location on Google cloud storage for further work.

Now it is checked whether the uploaded type is of image or not. If the condition does not match that of image then it is converted to JPG and we move onto image compression phase otherwise directly to image compression phase. Also, before moving to next phase the URL of image is sent to the admin portal where it is stored for further needs.

In backend the OCR i.e., optical content recognition is performed and plain text is collected and from this we scrutinize the important information which feeds into the admin portal's AutoFill detail part and after this some manual corrections if required are done the notice is finally published. In Backend a corresponding entry is made into database and a notification is triggered in the user's device.

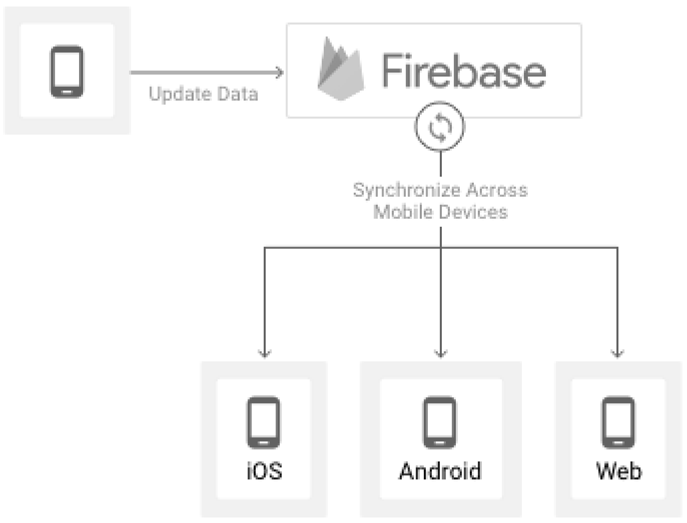


Figure 1.4: Cross Platform Accessibility of a Notice

When the user wishes to use our application or is prompted by a notification from our side, we first authenticate the user by his credentials and a device token is generated which is sent to our backend where it is stored as it is needed for sending notifications and the client is granted to access notice. Whenever a new entry is added into database the notice is real time synchronised from database to client and a notification pops out on the client side.

1.6 MODULES ORGANIZATION

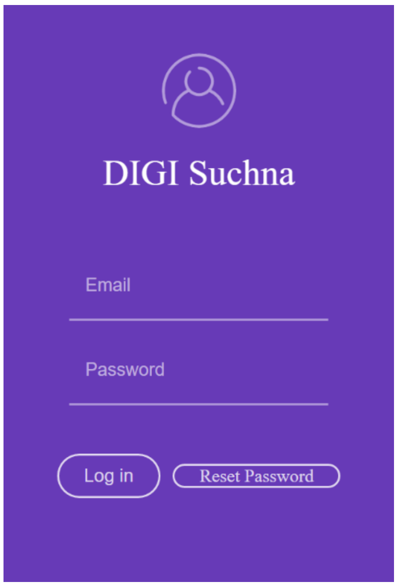
The project has been broadly organized in three phases of development, followed by a deep study on the image compression and optical character recognition to extract important information from the notices. The major phases of our project development are as follows:

1.6.1 The Client Application (DIGI Suchna Mobile App)

A Cross platform mobile application to digitalise the frequently used content and to outreach the information with all the students. We come up with an optimized solution, basically an android app, that delivers the public notices digitally and also notify end users as soon as a notice being published. digitalize the public notices of the university. This app will provide and target all the public notices, highlights their important facts and also provides important links at the same place.

1.6.2 The Admin Portal

A separate web portal connected with the firebase cloud services to provide privileges to the administrator. It grants authorized person to draft a new notice that is to be broadcast via the android application.



3.4.3 Cloud Backend (Firebase Functions)

It is the backbone of the whole system and plays a central role to connect the mobile application, database and other third-party services.

The entire notice delivery system is build up with eight major modules, which empowers the application to be a smart solution. These modules are as follows:

- ✓ Authentication
- ✓ Realtime Synchronization
- ✓ Quick Search
- ✓ Time Since (Notice Published)
- ✓ Drop Zone to upload file to the cloud storage
- ✓ Image Compression
- ✓ OCR to extract useful information
- ✓ Automated notifications
- ✓ Offline persistence and cache
- ✓ User Friendly Interface

1.7 Key Challenges

During the development of the whole project, there were lots of challenges, we faced and observed, a variety of challenges, and tried to solve out most of them. Some of the key challenges we had were:

- ❖ No idea of Android Development
- ❖ Choice of cloud service provider
- ❖ Choice of technology stack
- ❖ Database modelling
- ❖ Database efficiency
- ❖ Product name and branding
- ❖ Events tracking and analytics
- ❖ Deliver build to the targeted users
- ❖ Deliver updates after production used

We worked upon all these challenges. Some have been already resolved, while few are still a big challenge to be solved out.

1.8 Future Prospects

The ‘DigiSuchna’ app has a lot of potential and scope in the present scenario where almost every institution relies majorly on print media to circulate official notices. In the first step, we plan to take our services to other JUIT sister institutes in Noida, Guna, Anoopshehar and nearby universities in Himachal Pradesh. Here we will have our experience in running the service in JUIT as an added benefit. Further we also plan to take our service to various government organizations.

1.9 Project Outreach

Our project is based on providing a timely and efficient way to students of our university through which they can view notices at time and place of their choice. But the basic idea of our project has been to transmit information to a desired set of people by an organization so that the essence of the facts is not lost and misinformation could be avoided along with keeping a check on the authenticity of sender and receiver. So, our project can be acclimatized to the needs of various other organization and government initiatives which can be seen as our project was recognized by various organizations and government departments, including the **Department of Industries** Himanchal Pradesh, **IT department of Rajasthan** Government and the **Microsoft’s** world-wide championship, the **Imagine cup**.

2. LITERATURE SURVEY

2.1 Optical Character Recognition (OCR) System

From this research paper we learnt that OCR is of 2 types online and offline. In offline the text is generally caught by a scanner and which is then made accessible as a picture but in online mode the 2D coordinates of consecutive points are constituted as a function of time. The on-line methods appear better than offline counterparts in recognizing handwritten characters due to the temporal information available with the former.

To perform the character recognition, our application has to go through three important steps:

1-Segmentation: Given input image, identify individual glyphs (basic units representing one or more characters, usually contiguous).

2-Feature Extraction: From each glyph image, extract features to be used as input of ANN. This is the most critical part of this approach.

3-Classification: Train ANN using training sample. Then given new glyph, classify it. The second step is the most difficult in the sense that there is no obvious way to obtain these features.

The basic steps involved in Optical Character Recognition are: -

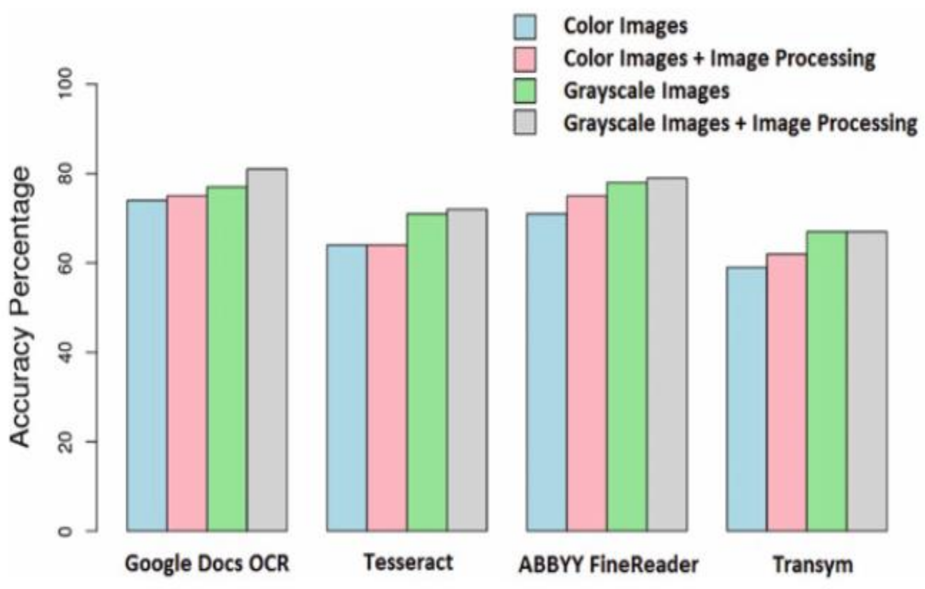
1. Image Acquisition
2. Preprocessing
3. Document Page Analysis
4. Feature Extraction
5. Training and Recognition
6. Post Processing

Optical Character Recognition (OCR) System
Najib Ali Mohamed Isheawy And Habibul Hasan
IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p-ISSN: 2278-8727,

2.2 Detailed Analysis of OCR Technology

From this research paper we mainly learnt about the challenges in OCR technology. For better quality and high exactness character acknowledgment, OCR strategies expect high calibre or high-determination pictures with some fundamental auxiliary properties, for example, high separating content and foundation. The way pictures are created is a vital and deciding variable in the exactness and accomplishment of OCR, since this frequently influences the nature of pictures significantly. Normally OCR with pictures delivered by scanners gives high exactness and great execution. Conversely, pictures delivered by cameras more often than not will be not as great of a contribution as filtered pictures to be utilized for OCR due to the natural or camera related components. Many faults may arise some are listed below:

1. Scene Complexity
2. Conditions of Uneven Lighting
3.Skewness (Rotation)
4.Warping
5. Multilingual Environments
6.Fonts



Graph 2.1: Comparison of various OCR Techniques.

Phases in OCR: There are six major phases in OCR recognition:

1) Pre-processing Phase
2) Segmentation Phase
3) Normalization Phase
4) Feature Extraction Phase
5) Classification Phase
6) Postprocessing Phase

.

A Detailed Analysis of OCR Technology

By: Hamad, Karez & Kaya, Mehmet

2.3 A Comprehensive Review of Image Compressions Techniques

Here we learnt how to compress an image, based on various image compression techniques. The basic process is as follows:

1. Specifying the Rate (bits available) and Distortion (tolerable error) parameters for the target image.
2. Dividing the image data into various classes, based on their importance.
3. Dividing the available bit budget among these classes, such that the distortion is a minimum.
4. Quantize each class separately using the bit allocation information derived in step3.
5. Encode each class separately using an entropy coder and write to the file.

Reconstructing the image from the compressed data is usually a faster process than compression. The steps involved are:

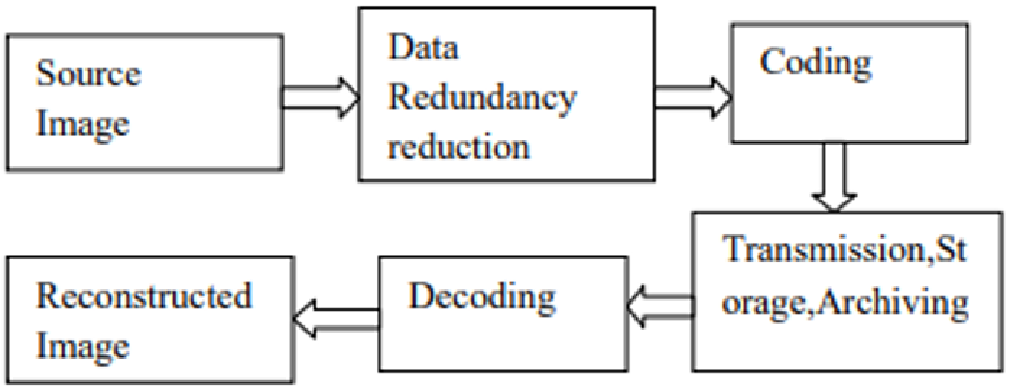


Figure 2.1: Basic steps of Image Compression Techniques

6. Read in the quantized data from file, using an entropy decoder. (Reverse of step 5).
7. Dequantize the data. (Reverse of step 4).
8. Rebuild the image. (Reverse of step 2)

Need of Compression:

- Sufficient amount of storage space increased.
- To reduce the time taken for transmission of an image to be sent over the internet or download from the web pages.
- Image Archiving: Satellite Data
- Image Transmission: Web Data

When research into image compression began in the late 1970s, most compression concentrated on using conventional lossless techniques. However, such types of compression, which included statistical and dictionary methods of compression, did not tend to perform well on photographic, or continuous tone images. The primary problem with statistical techniques stemmed from the fact that pixels in photographic images tend to be well spread out over their entire range.

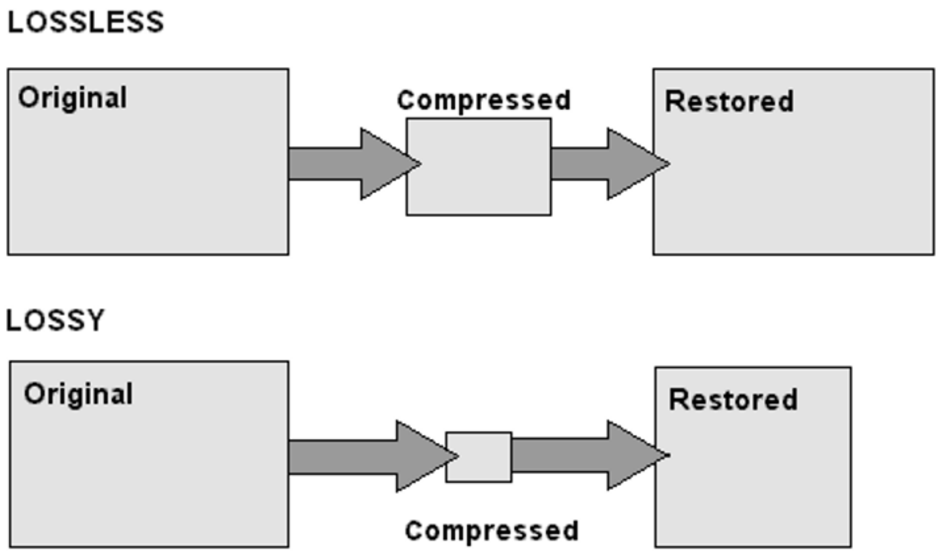


Figure 2.2: Lossy and Lossless image compression

If the colors in an image are plotted as a histogram based on frequency, the histogram is not as "spiky" as one would like for statistical compression to be effective. Each pixel code has approximately the same chance of appearing as any other, negating any opportunity for exploiting entropy differences

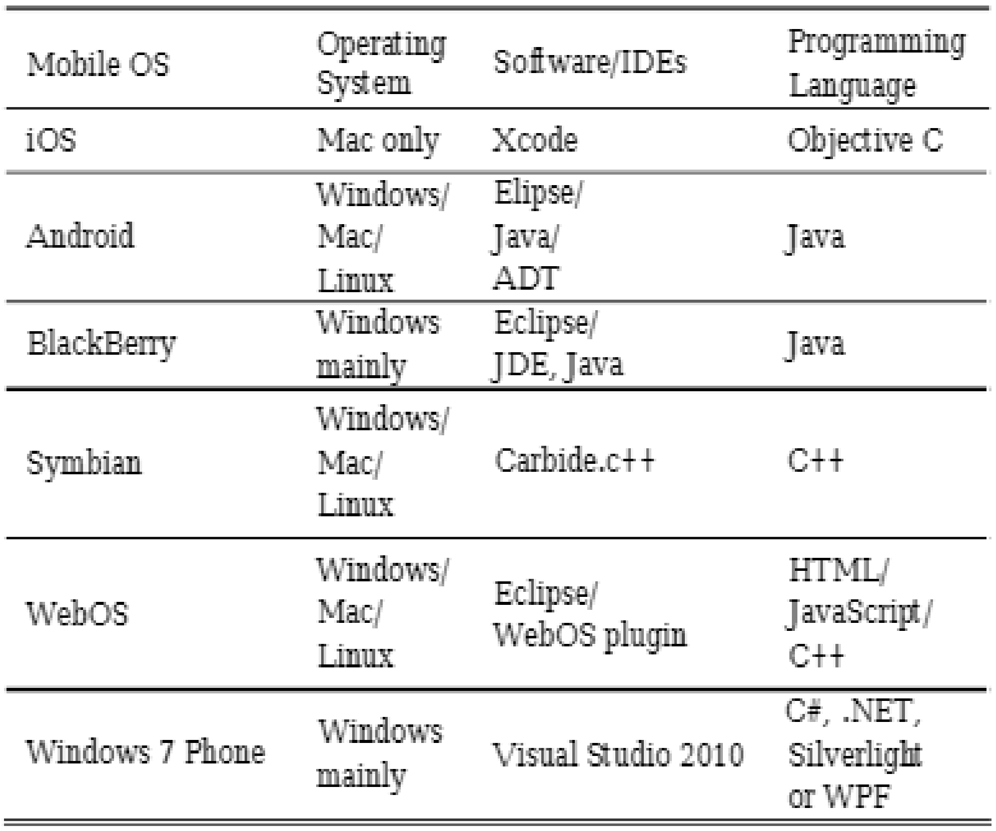
A Comprehensive Review of Image Compression Techniques
Kitty Arora, Manshi Shukla
(IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (2), 2014, 1169-1172

2.4 PhoneGap APIs Accessing the Native Mobile Platform APIs

From this research paper we learnt PhoneGap API's and how it works - The application's UI comprises of basically a distinct screen that contains a distinct web view that devours the majority of the accessible space on the gadget's screen.

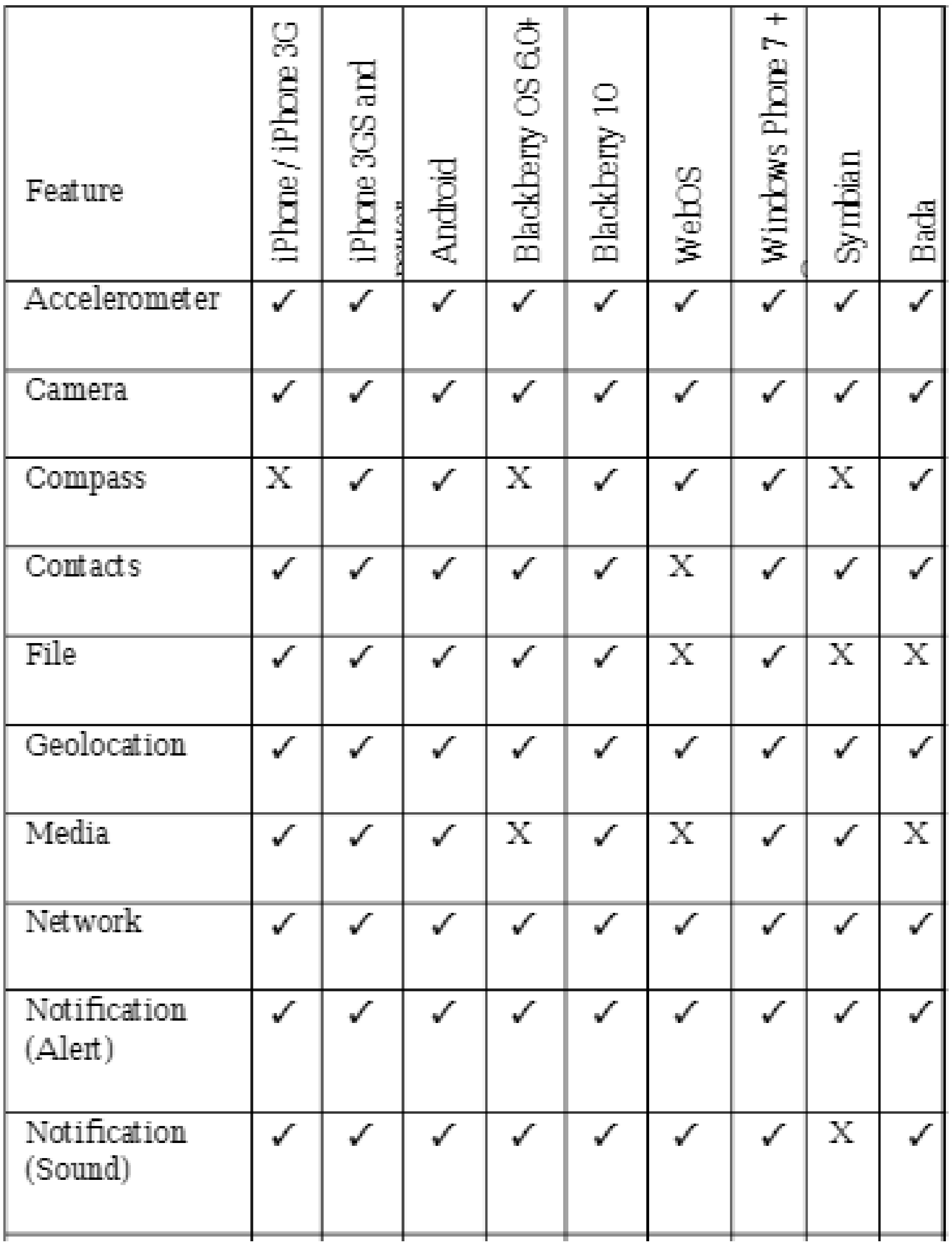
At the point when the application starts, it stacks the web application's startup page into the web view and afterward passes control to the web view to enable the client to communicate with the web application.

As the client communicates with the application, JavaScript code inside the application can stack other substance from inside the asset documents bundled with this application or can contact the system and draw content down from a web or application server.



Mobile OS	Operating System	Software/IDEs	Programming Language
iOS	Mac only	Xcode	Objective C
Android	Windows/ Mac/ Linux	Elipse/ Java/ ADT	Java
BlackBerry	Windows mainly	Eclipse/ JDE, Java	Java
Symbian	Windows/ Mac/ Linux	Carbide.c++	C++
WebOS	Windows/ Mac/ Linux	Eclipse/ WebOS plugin	HTML/ JavaScript/ C++
Windows 7 Phone	Windows mainly	Visual Studio 2010	C#, .NET, Silverlight or WPF

Table 2.1: Development requirements for various mobile platforms



Feature	iPhone / iPhone 3G	iPhone 3GS and	Android	Blackberry OS 6.0+	Blackberry 10	WebOS	Windows Phone 7 +	Symbian	Bada
Accelerometer	✓	✓	✓	✓	✓	✓	✓	✓	✓
Camera	✓	✓	✓	✓	✓	✓	✓	✓	✓
Compass	X	✓	✓	X	✓	✓	✓	X	✓
Contacts	✓	✓	✓	✓	✓	X	✓	✓	✓
File	✓	✓	✓	✓	✓	X	✓	X	X
Geolocation	✓	✓	✓	✓	✓	✓	✓	✓	✓
Media	✓	✓	✓	X	✓	X	✓	✓	X
Network	✓	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Alert)	✓	✓	✓	✓	✓	✓	✓	✓	✓
Notification (Sound)	✓	✓	✓	✓	✓	✓	✓	X	✓

Table 2.2: Native device services, offered by PhoneGap

Review of PhoneGap APIs Accessing the Native Mobile Platform APIs
Niyigena Jean Pierre and Mukiza Octavien
Lecture Notes on Software Engineering
4. 12-15. 10.7763/LNSE.2016.V4.216

2.5 Firebase Cloud Messaging (Android)

From this we learnt about firebase which is a real time database that can send and receive messages explicitly from the client. When java script object notion data is saved to Firebase, all users immediately receive the changes along with web and mobile.

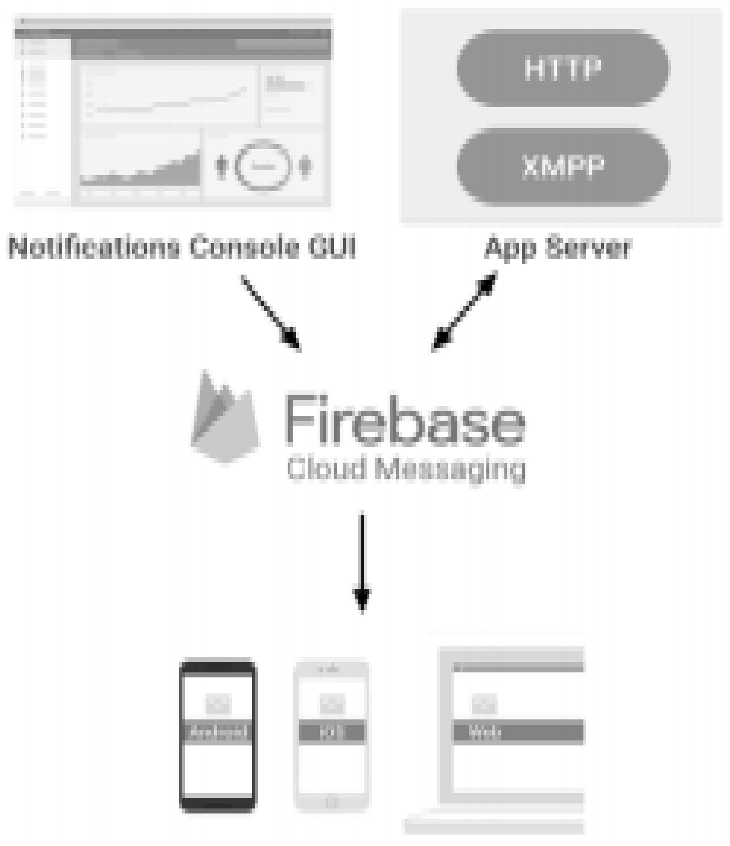


Figure 2.3: Organization of Firebase Cloud Messaging

It covers all that mobile developers need to build, maintain and scale a mobile app on all platforms from Storage and databases and other tools. Some characteristic of firebase are -

- **Real time Database:** A schema less and cloud hosted database that is structured in Java Script Object Notation
- **Authentication**: Authentication in firebase provides a very good service of authenticating all the users without
- even using any client-side coding.
- **Cloud messaging**: Firebase cloud messaging allows messages to be delivered from any platform other than the one on which it is developed in a reliable manner.
- **Cloud Storage**: files can be transferred and downloaded in a protected way without being influenced by communication medium service.
- **Secure Hosting:** applications can be statically hosted in a efficient way.
- **Remote config:** This feature allows to update the user application without the need of deploying the current updated app.
- **Test labs:** It involves testing of applications with different configurations and with variety of devices.
- **Crash reporting:** This feature is used to prepare a report of all the app crashes and errors in application.

Firebase Cloud Messaging (Android)

Neha Srivastava, Uma Shree, Nupa Ram Chauhan, Dinesh Kumar Tiwari

IJIRSET

Vol. 6, Special Issue 9, May 2017

2.6 Working of the OAuth Protocol

In the traditional client-server authentication model, the client requests an access restricted resource (protected resource) on the server by authenticating with the server using the resource owner's credentials. In order to provide third-party applications access to restricted resources, the resource owner shares its credentials with the third-party. This creates several problems and limitations:

1. Third-party applications are required to store the resource owner's credentials for future use, typically a password in clear-text.
2. Servers are required to support password authentication, despite the security weaknesses inherent in passwords.
3. Third-party applications gain overly broad access to the resource owner's protected resources, leaving resource owners without any ability to restrict duration or access to a limited subset of resources.
4. Resource owners cannot revoke access to an individual third-party without revoking access to all third-parties and must do so by changing their password.

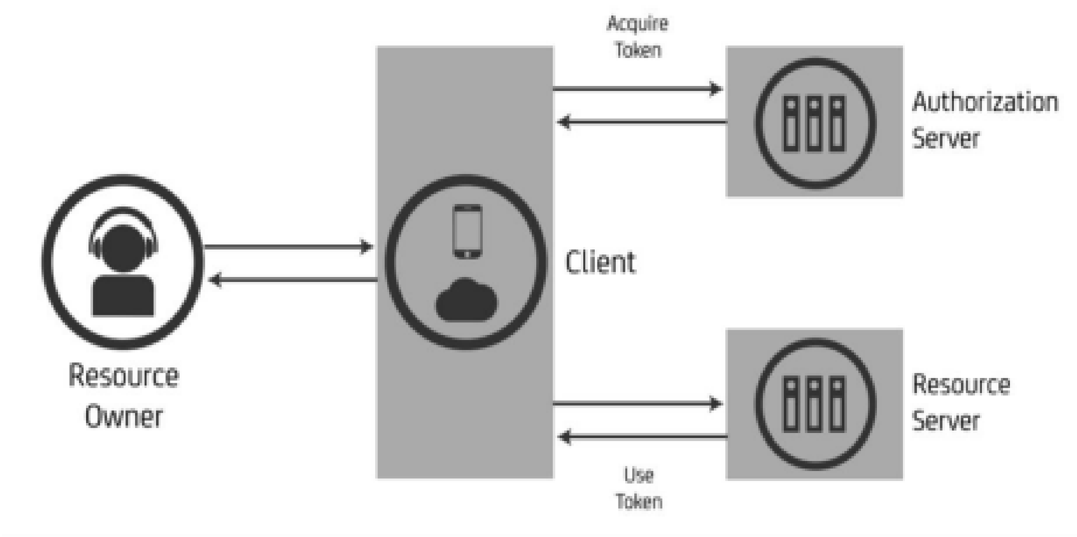


Figure 2.4: Working of OAuth 2.0 Mechanism

Compromise of any third-party application results in compromise of the end user's password and all of the data protected by that password. OAuth 2.0 (Open Authentication) Protocol addresses these issues by introducing an authorization layer and separating the role of the client from that of the resource owner. In OAuth, the client requests access to resources controlled by the resource owner and hosted by the resource server and is issued a different set of credentials than those of the resource owner. Instead of using the resource owner's credentials to access protected resources, the client obtains an access token - a string denoting a specific scope, lifetime, and other access attributes. Access tokens are issued to third-party clients by an authorization server with the approval of the resource owner.

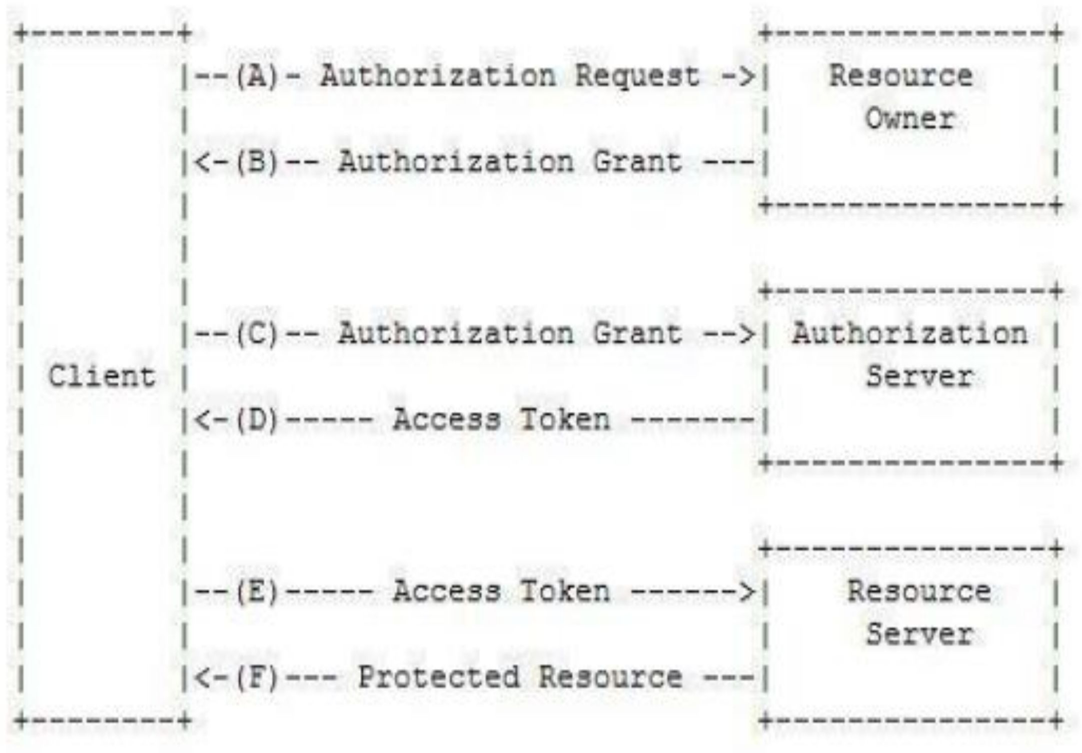

Client
(A)- Authorization Request
Resource Owner
(B)-- Authorization Grant
(C)-- Authorization Grant
Authorization Server
(D)----- Access Token
(E)----- Access Token
Resource Server
(F)--- Protected Resource

Figure 2.5: Client Authorization process

The client uses the access token to access the protected resources hosted by the resource server. For example, an end-user (resource owner) can grant a printing service (client) access to her protected photos stored at a photo sharing service (resource server), without sharing her username and password with the printing service. Instead, she authenticates directly with a server trusted by the photo sharing service (authorization server) which issues the printing service delegation-specific credentials (access token).

A. The client requests authorization from the resource owner. The authorization request can be made directly to the resource owner (as shown), or preferably indirectly via the authorization server as an intermediary.
B. The client receives an authorization grant, which is a credential representing the resource owner's authorization, expressed using one of four grant types defined in this specification or using an extension grant type. The authorization grant type depends on the method used by the client to request authorization and the types supported by the authorization server.
C. The client requests an access token by authenticating with the authorization server and presenting the authorization grant.
D. The authorization server authenticates the client and validates the authorization grant, and if valid, issues an access token.
E. The client requests the protected resource from the resource server and authenticates by presenting the access token.
F. The resource server validates the access token, and if valid, serves the request.

3. SYSTEM DEVELOPMENT

The project has been broadly organized in three phases of development, followed by a deep study on the Access Token's cryptanalysis. The major phases of our project development are as follows

3.1 REQUIREMENT SPECIFICATIONS

The requirements for the cloud based smart notice delivery system are as follows:

3.1.1 Purpose

The main objective of this project is to build an online system to manage and deliver public notices of the university direct to all the students. To reduce the paper wastage while also provide a quick and genuine source for the notices or information.

3.1.2 Intended Audience & Reading Suggestions

This project is a prototype for the notice delivery system and it is mainly targeted for the college students and faculties. To deliver public notices, on the go, directly to the user's device, is the main purpose to deal with. This project is useful for the college administration and as well as to the students.

3.1.3 Project Scope

The purpose of the smart notice delivery system is to efficiently deliver notices and to build a convenient & easy-to-use mobile application for students, usually navigate notice boards for daily updates. The whole mechanism is based on a NoSQL database, mainly FireStore database, with its notices storage and notification delivery functions. Moreover, we target to provide an impressive user experience along with the enhanced device capabilities.

3.1.4 Performance Matrices

Performance analysis plays a very significant role to measure the acceptance of the product by the end users. This project is purely users (students) oriented and thus the various parameters has to be checked from time to time and should be focused onto enhance the overall performance of the product. We had conducted various tests and analysis on the actual software, based on the various parameters, such as:

- Server Response Time
- Throughput
- Capacity
- Administration response
- Memory utilization
- Internet Utilization
- Scalability

3.1.5 Functional Description

The entire notice delivery system is having several major modules, which empowers the application to be a smart solution. These functions are as follows:

- ✓ Authentication
- ✓ Realtime Synchronization
- ✓ Quick Search
- ✓ Time Since (Notice Published)
- ✓ Drop Zone to upload file to the cloud storage
- ✓ Image Compression
- ✓ OCR to extract useful information
- ✓ Automated notifications
- ✓ Offline persistence and cache
- ✓ User Friendly Interface

3.2 ARCHITECTURAL DESIGN

These designs were made before the actual starting of the project implementation. They helped a lot to shape the actual product and were very useful.

3.2.1 Basic Users Modelling:

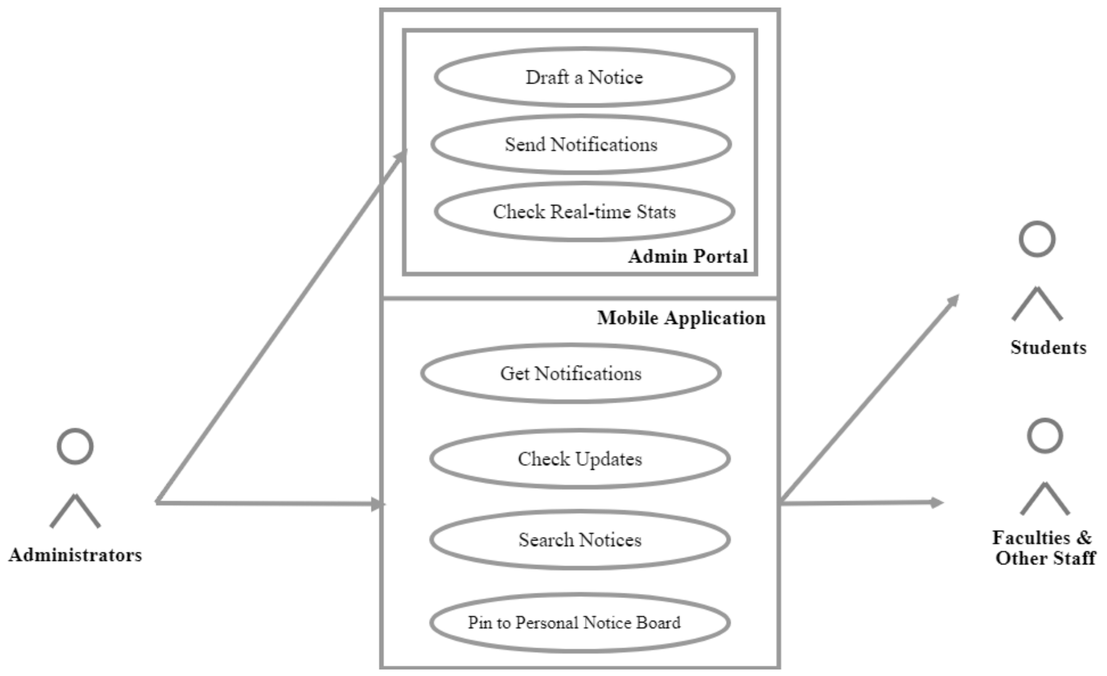


Figure 3.1: Simple use case scenario at the university level

3.2.2 Functional Modelling:

The whole application works on the three simultaneous sub-projects, which are highly integrated with the firebase cloud services. The flow of process in all these sub-projects has been discussed as on the figure: 3 .2

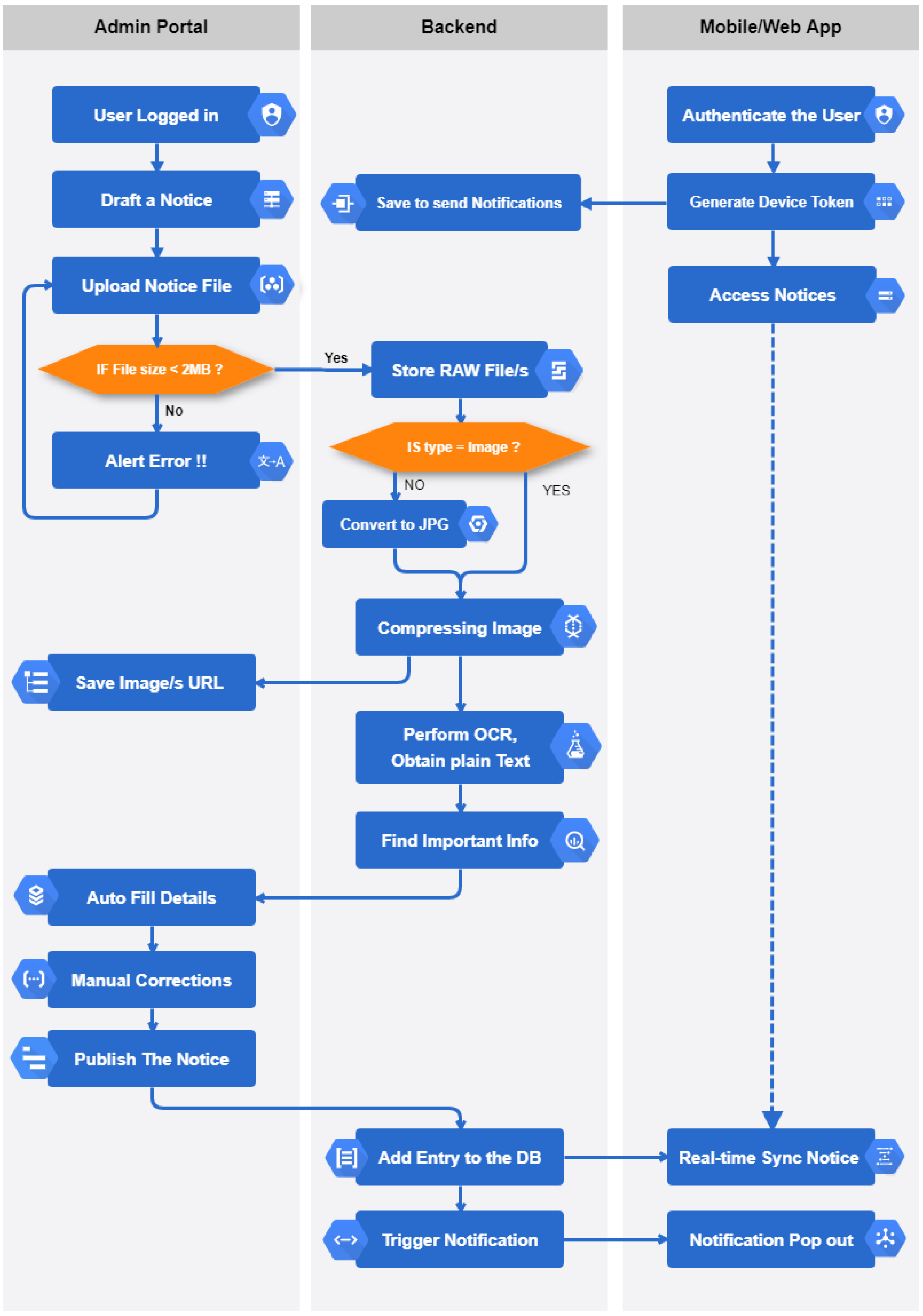


Figure 3 .2: Functional Flow diagram for the whole, notice delivery mechanism

The process initializes when an admin logs into his account in admin portal using credentials that are his email id and password.
Then he clicks on the button “Draft A Notice” and the notice is uploaded on the server and the size of the uploaded file is checked if it is more than 2MB then the admin is prompted with an alert message or if the size is acceptable then the uploaded file moves to backend in a temporary location on Google cloud storage for further work.
Now it is checked whether the uploaded type is of image or not. If the condition does not match that of image then it is converted to JPG and we move onto image compression phase otherwise directly to image compression phase. Also, before moving to next phase the URL of image is sent to the admin portal where it is stored for further needs.

In backend the OCR i.e., optical content recognition is performed and plain text is collected and from this we scrutinize the important information which feeds into the admin portal’s AutoFill detail part and after this some manual corrections if required are done the notice is finally published. In Backend a corresponding entry is made into database and a notification is triggered in the user’s device.

When the user wishes to use our application or is prompted by a notification from our side, we first authenticate the user by his credentials and a device token is generated which is sent to our backend where it is stored as it is needed for sending notifications and the client is granted to access notice. Whenever a new entry is added into database the notice is real time synchronised from database to client and a notification pops out on the client side.

3.3 TOOLS AND TECHNOLOGIES

- **NodeJS** along with **NPM** for managing dependencies and other packages.
- NoSQL based Realtime **FireStore Database**
- **Firebase Authentication** along with google cloud storage for save the notices on the cloud.
- **Firebase Push Notifications** and **Firebase Function** as a central backend.
- **Cordova** along with **PhoneGap** Utility to build up the client application.
- **Framework7** for the basic wireframe
- **Angular5** & **VueJS** for the Frontend framework.
- **Material** (Native Angular) UI.
- Secured **Azure Cloud Hosting**.
- **GitHub** for code synchronization and Version Controlling.

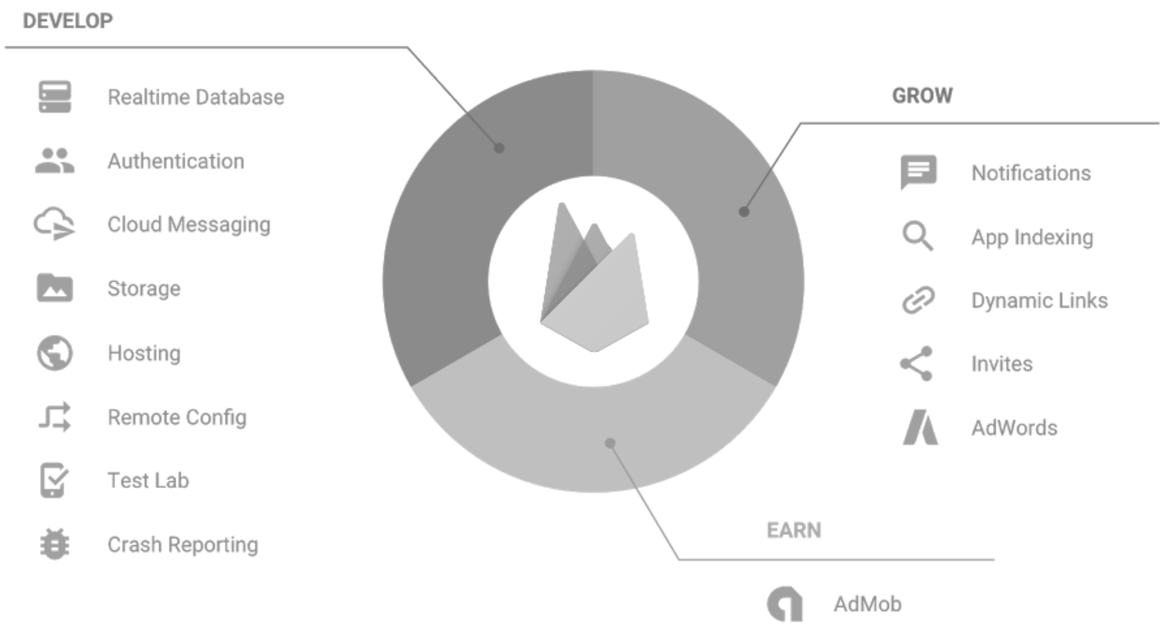


Figure 3.3: Various Firebase Cloud Services, used to support the project

3.4 PHASES AND DELIVERABLES

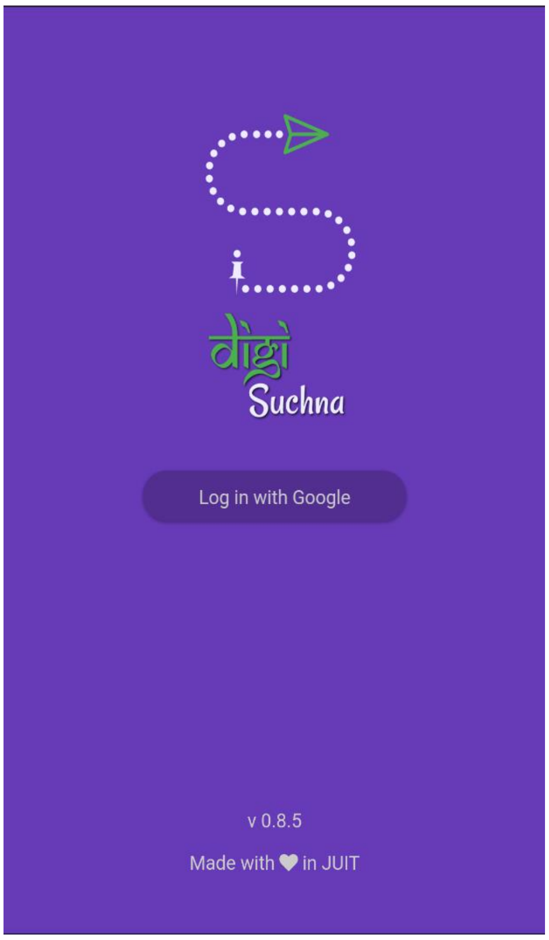
The project has been broadly organized in three phases of development, followed by a deep study on the image compression and optical character recognition to extract important information from the notices. The major phases of our project development are as follows:

3.4.1 The Client Application (DIGI Suchna Mobile App)

A Cross platform mobile application to digitalise the frequently used content and to outreach the information with all the students. We come up with an optimized solution, basically an android app, that delivers the public notices digitally and also notify end users as soon as a notice being published. digitalize the public notices of the university. This app will provide and target all the public notices, highlights their important facts and also provides important links at the same place.

Key Features:

- ✓ Authentication with Google OAuth
- ✓ Sorted list view of Notices
- ✓ User Profile
- ✓ Notice Browser
- ✓ Realtime searching
- ✓ Notifications on new notice or update
- ✓ Organizers based categorization
- ✓ Multiple Attachments
- ✓ Related Links for each Notice
- ✓ Attractive placeholder for loading Interfaces
- ✓ Add Notices to remember list



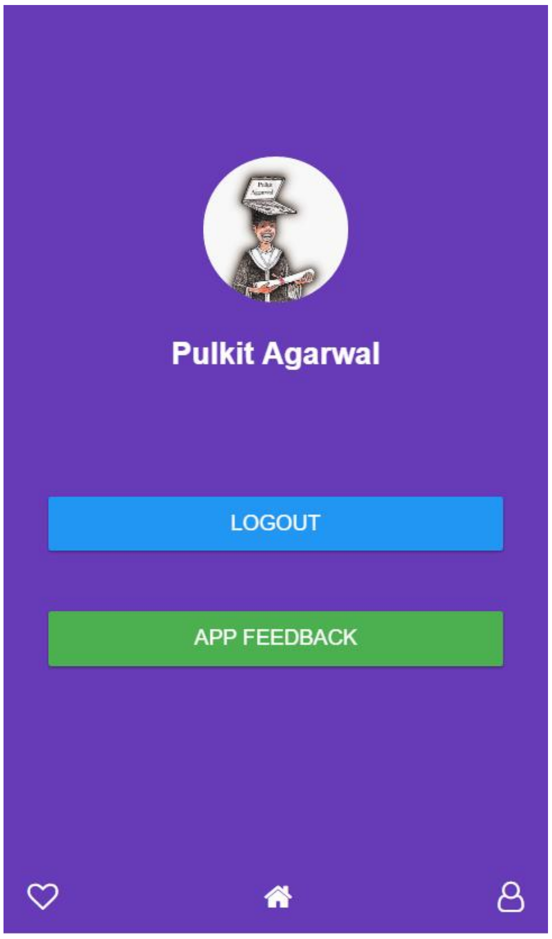
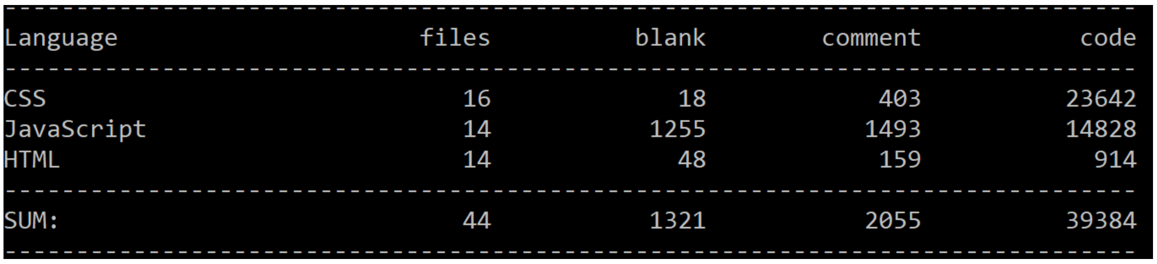
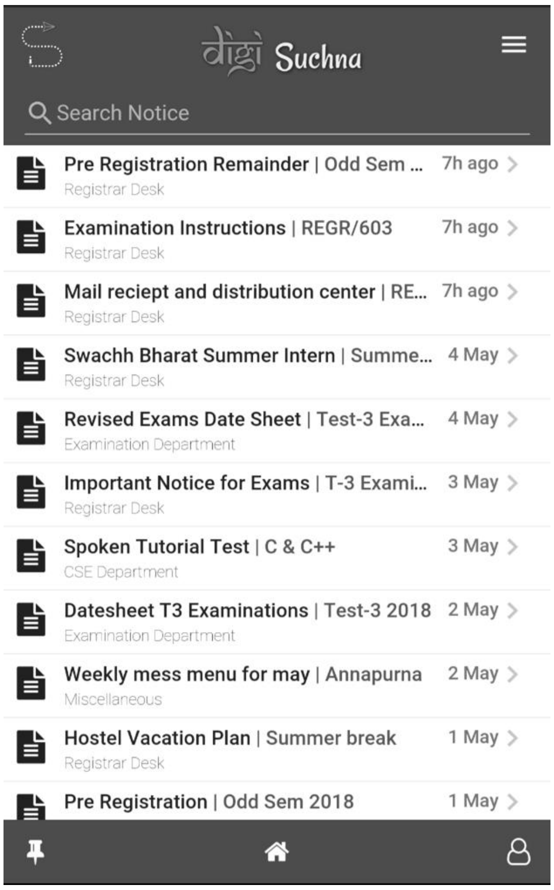


Figure 3.4 App screenshot: Login Screen & Profile Screen



```
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
CSS                             16             18            403          23642
JavaScript                      14           1255           1493          14828
HTML                            14             48            159            914
-------------------------------------------------------------------------------
SUM:                            44           1321           2055          39384
-------------------------------------------------------------------------------
```

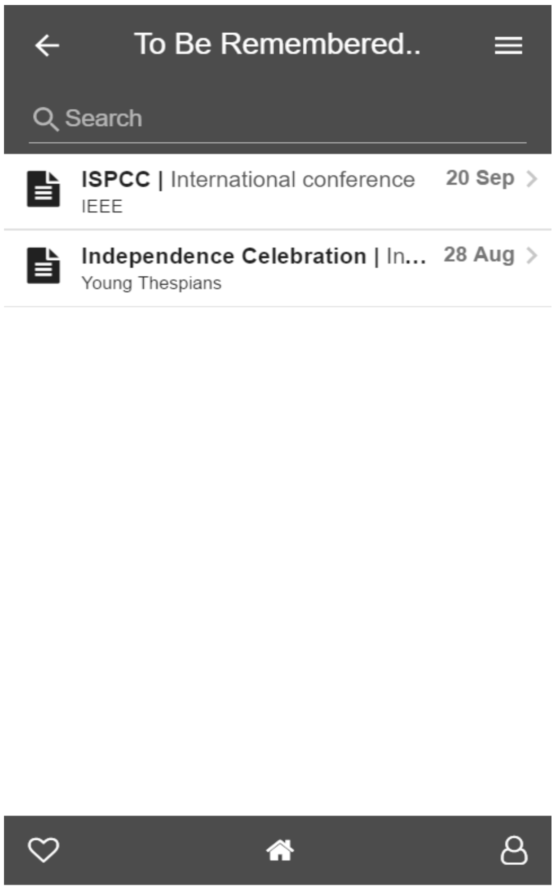
Figure 3.5: Brief count of lines of code for mobile application



digi Suchna

Search Notice

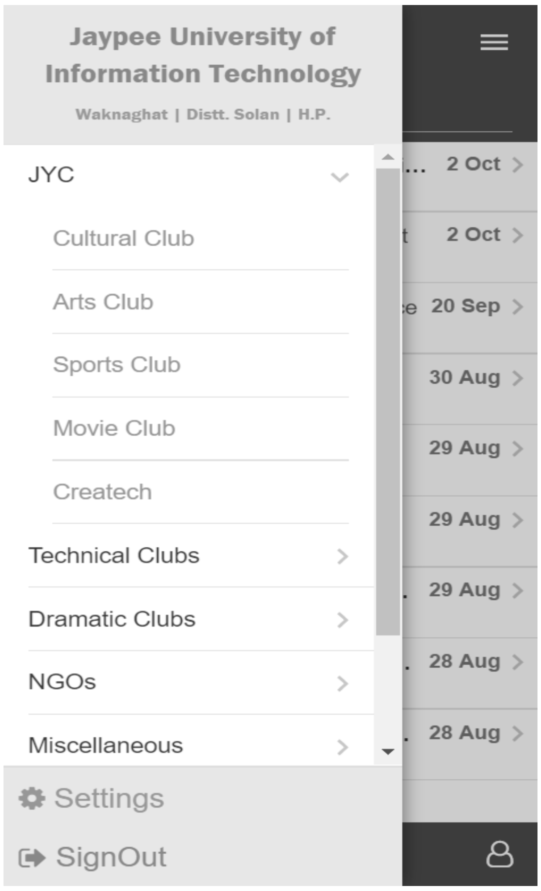
- **Pre Registration Remainder** | Odd Sem ... 7h ago — Registrar Desk
- **Examination Instructions** | REGR/603 7h ago — Registrar Desk
- **Mail reciept and distribution center** | RE... 7h ago — Registrar Desk
- **Swachh Bharat Summer Intern** | Summe... 4 May — Registrar Desk
- **Revised Exams Date Sheet** | Test-3 Exa... 4 May — Examination Department
- **Important Notice for Exams** | T-3 Exami... 3 May — Registrar Desk
- **Spoken Tutorial Test** | C & C++ 3 May — CSE Department
- **Datesheet T3 Examinations** | Test-3 2018 2 May — Examination Department
- **Weekly mess menu for may** | Annapurna 2 May — Miscellaneous
- **Hostel Vacation Plan** | Summer break 1 May — Registrar Desk
- **Pre Registration** | Odd Sem 2018 1 May



To Be Remembered..

Search

- **ISPCC** | International conference 20 Sep — IEEE
- **Independence Celebration** | In... 28 Aug — Young Thespians



Jaypee University of Information Technology

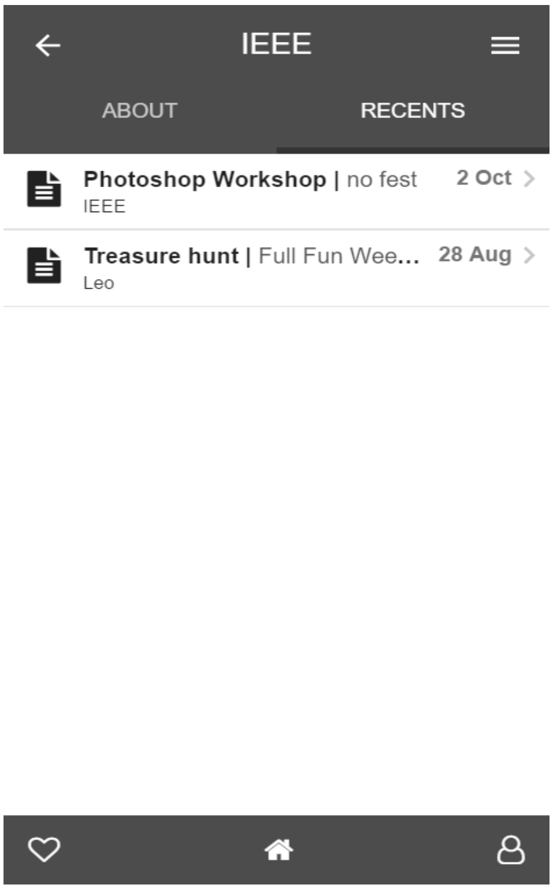
Waknaghat | Distt. Solan | H.P.

- JYC
  - Cultural Club
  - Arts Club
  - Sports Club
  - Movie Club
  - Createch
- Technical Clubs
- Dramatic Clubs
- NGOs
- Miscellaneous

Settings

SignOut

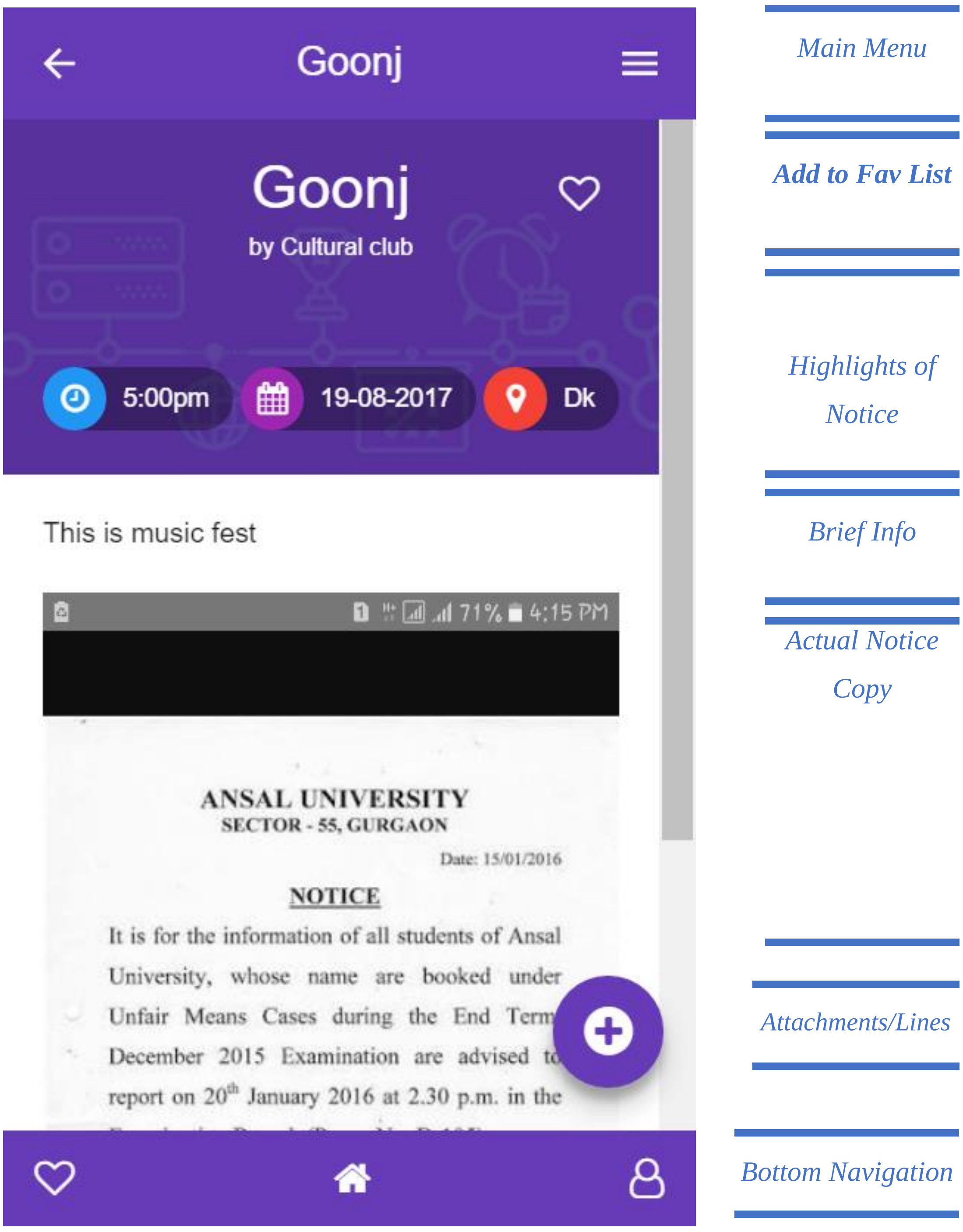
2 Oct, 2 Oct, 20 Sep, 30 Aug, 29 Aug, 29 Aug, 29 Aug, 28 Aug, 28 Aug



IEEE

ABOUT RECENTS

- **Photoshop Workshop** | no fest 2 Oct — IEEE
- **Treasure hunt** | Full Fun Wee... 28 Aug — Leo



This is music fest

ANSAL UNIVERSITY
SECTOR - 55, GURGAON

Date: 15/01/2016

NOTICE

It is for the information of all students of Ansal University, whose name are booked under Unfair Means Cases during the End Term December 2015 Examination are advised to report on 20th January 2016 at 2.30 p.m. in the

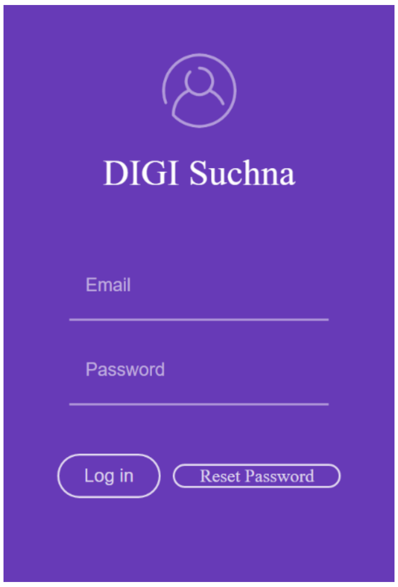
Figure 3.6: Detailed elements of a Notice in the Application

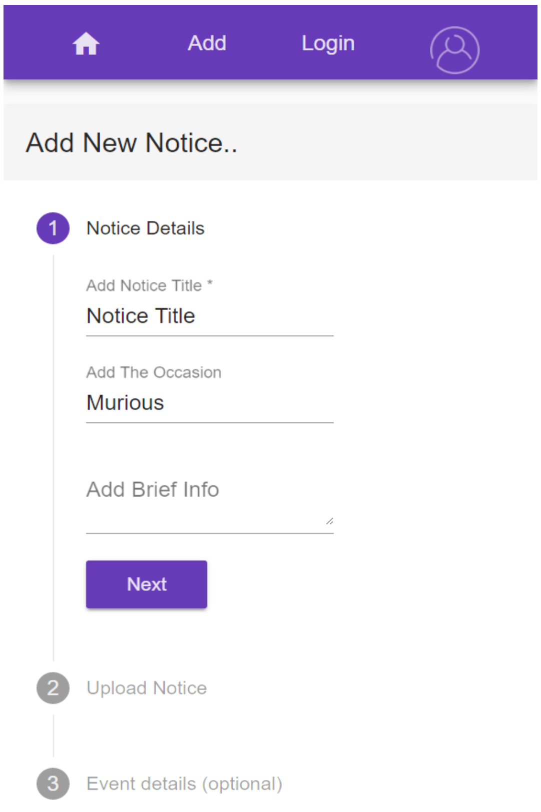
3.4.2 The Admin Portal

A separate web portal connected with the firebase cloud services to provide privileges to the administrator. It grants authorized person to draft a new notice that is to be broadcast via the android application.

Features

- ✓ Organizer Login with email and password.
- ✓ Add new Notice
- ✓ Attach multiple attachments
- ✓ Register / More Info Links
- ✓ Mention Highlights
- ✓ Realtime Broadcast/ Publish Notice
- ✓ Realtime Update Previous Notice
- ✓ Realtime Update Organizer Info

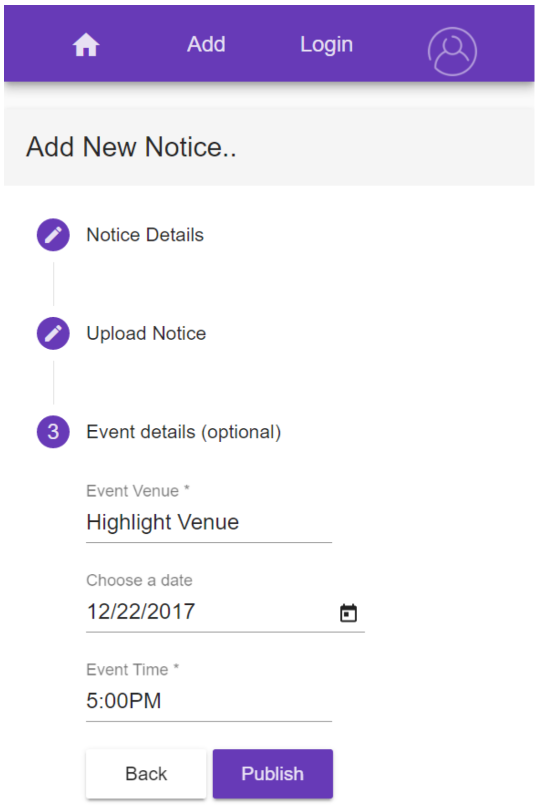




Add Notice Title *
Notice Title

Add The Occasion
Murious

Add Brief Info



Event Venue *
Highlight Venue

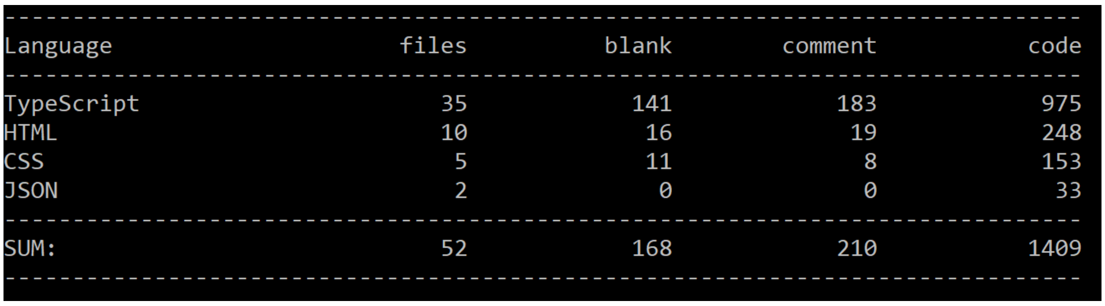
Choose a date
12/22/2017

Event Time *
5:00PM

Figure 3.7: Admin Portal: Draft a Notice

3.4.3 Cloud Backend (Firebase Functions)

It is the backbone of the whole system and plays a central role to connect the mobile application, database and other third-party services.



```
-------------------------------------------------------------------------------
Language                     files          blank        comment           code
-------------------------------------------------------------------------------
TypeScript                      35            141            183            975
HTML                            10             16             19            248
CSS                              5             11              8            153
JSON                             2              0              0             33
-------------------------------------------------------------------------------
SUM:                            52            168            210           1409
-------------------------------------------------------------------------------
```

Figure 3.8: Brief of Lines of Code for the Cloud Backend.

The Cloud backend is based on the NodeJS and ExpressJS frameworks, to deliver a high-performance backend. It starts with importing all the supporting modules and firebase initialization.

```
const functions = require('firebase-functions');
var gcs = require('@google-cloud/storage')({ keyFilename: 'noticeapp-
8a030-firebase-adminsdk-z06mo-43332e65ee.json' });

const admin = require('firebase-admin');
admin.initializeApp(functions.config().firebase);
var bucket = functions.config().firebase.storageBucket;
var db = admin.firestore();

exports.initUser = functions .auth .user() .onCreate(e => {
    var user = e.data; // The Firebase user.
    console.log(JSON.stringify(user));
    admin.firestore().collection('Users').doc(user.uid).set({
        userName: user.displayName,
        verified: 'NO',
        fav: []
    }).catch((err) => {
        console.log('Error creating user', err);
    });
});
```

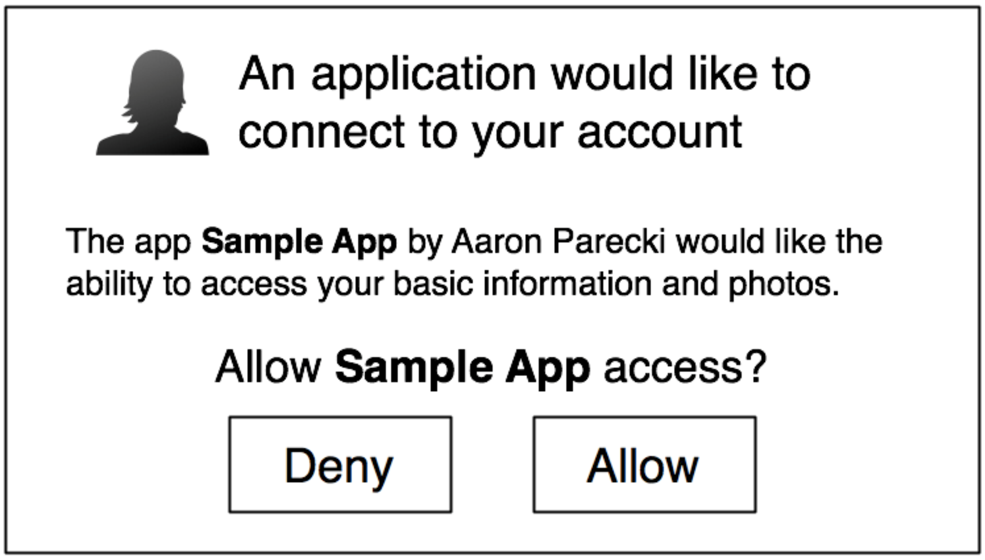
3.5 MODULES

The entire notice delivery system is build up with eight major modules, which empowers the application to be a smart solution. These modules are as follows:

3.5.1 Authentication

User Authorization is the very prime objective of each and every mobile application. This project also enables the user authorization in a very simple and flexible way. By using the firebase authorization mechanism, based on the O-Auth 2.0 protocol, the application authenticates the user via google email ID. The module of authentication is itself quite critical.

OAuth2.0 mechanism, is fairly secured. It provides REST APIs to communicate with the client and thus authorizes users for the third-party client applications. The client will directly connect to resource server, by means of encrypted access token.



An application would like to connect to your account

The app **Sample App** by Aaron Parecki would like the ability to access your basic information and photos.

Allow **Sample App** access?

Figure 3.9: OAuth's user permission to access the information.

The authentication is being provided by the Firebase Authentication. It provides a simple and end-to-end identity authentication solution, supporting phone auth, email-password credentials, and login with Google ID

```
var provider = new firebase .auth .GoogleAuthProvider();

firebase.auth().languageCode = 'en';

provider .addScope('https://www.googleapis.com/auth/contacts ');

firebase .auth() .signInWithRedirect(provider);

firebase .auth() .getRedirectResult() .then(function(user) {
  if (user .credential) {
    // User details on redirection
    var token = user .credential .accessToken;

  }
  // The user info.
  var User = user.user;
}).catch(function(err) {
  // Handling errors.
  var Code = err .code;
  var Message = err .message;
  //
  var Email = err .email;
  // The Authorization type of user.
  var Cred = err .credential;
});
```

3.5.2 Realtime Synchronization

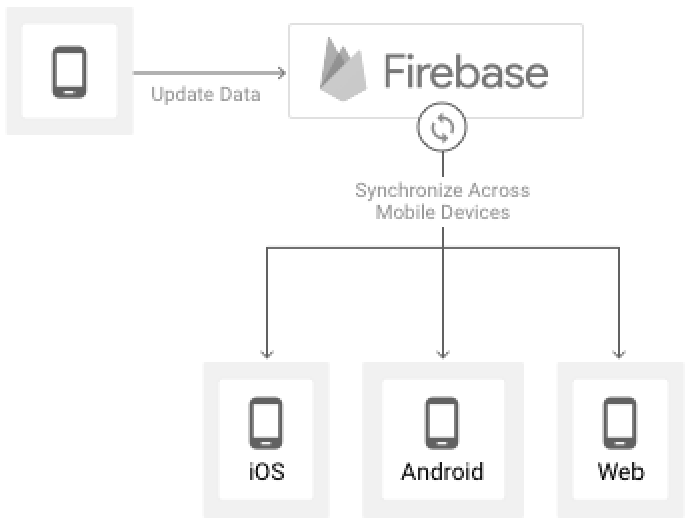


Figure 3.10: Firebase Realtime synchronization of database

We can *listen* to a document with the *onSnapshot()* method. An initial call using the callback us provide creates a document snapshot immediately with the current contents of the single document. Then, each time the contents change, another call updates the document snapshot.

```
db.collection("Production/JUIT/Notices")
      .onSnapshot({ includeQueryMetadataChanges: true },
         function(snapshot) {
            snapshot.docChanges.forEach(function(change) {
                if (change.type === "added") {
                    console.log("New notice: ", change.doc.data());
                    var notifyHTML =
                      Template7.templates.notifyTemp(change.doc.data());
                    $$(".loadingContents").hide();
                    $$('#noticeListfav').append(notifyHTML);
                }
            });
        });
```

3.5.3 Quick Search

A quick search panel is given in the application on the top of home page from where user can easily search the notices that they want. The quick search is one of the best features of the application.

The filteredList is being calculated dynamically and will get updated for every search query in a Realtime scenario. The pseudocode for the same is as follows:

```
  computed: {
    filteredList() {
      return this.notices.filter(notice => {
        return
notice.title.toLowerCase().includes(this.searchTerm.toLowerCase())
            ||
notice.fest.toLowerCase().includes(this.searchTerm.toLowerCase())
      })
    }
  },
```

3.5.4 Time Since (notice published)

This is a very small utility and a sub-module of the mobile application. It return the time in human readable form, while taking the UNIX timestamp as the input.

It reflects the time, since a notice has been published. It also helps in sorting the notices and information, based on the time, they have been published.

The pseudocode the timeSince function is as follows:

```
function timeSince(timeStamp) {
    var time = new Date(timeStamp);
    var ago = " ago";
    var now = new Date(),
        secondsPast = (now.getTime() - time.getTime()) / 1000;
    if (secondsPast < 60) {
        return parseInt(secondsPast) + 's' + ago;
    }
    if (secondsPast < 3600) {
        return parseInt(secondsPast / 60) + 'm' + ago;
    }
    if (secondsPast <= 86400) {
        return parseInt(secondsPast / 3600) + 'h' + ago;
    }
    if (secondsPast > 86400) {
      day = time.getDate();
      month = time.toDateString().match(/ [a-zA-Z]*/)[0].replace(" ", "");
      year = time.getFullYear() == now.getFullYear() ? "":" " +
time.getFullYear();
        return day + " " + month + year;
    }
}
```

3.5.5 Upload to Cloud storage

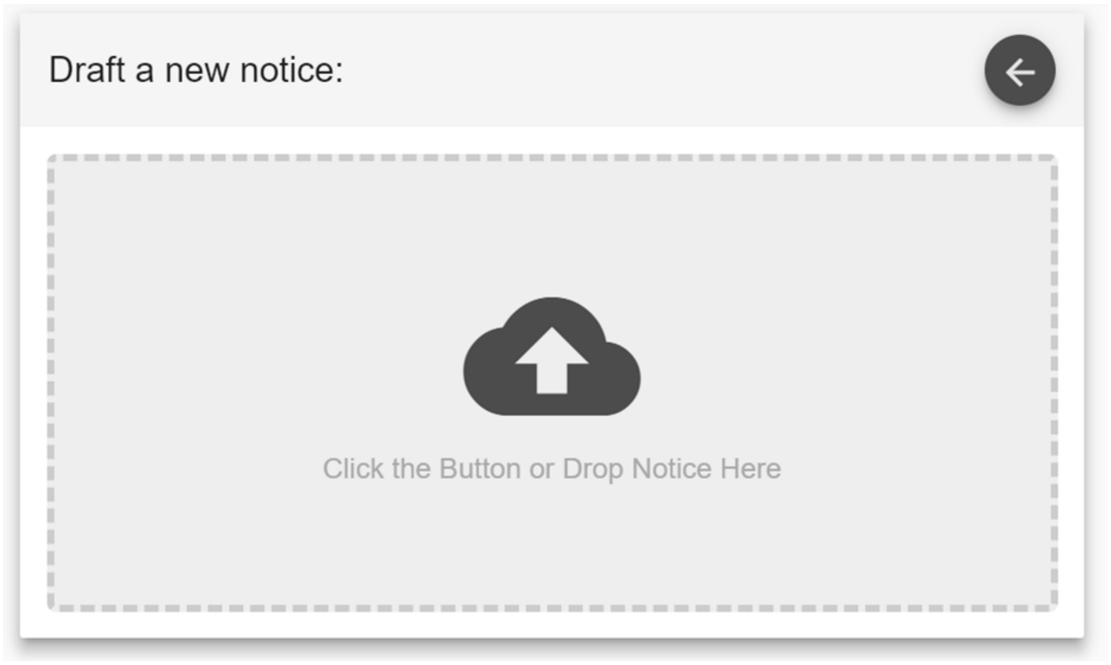


Figure 3.61: DropZone to draft a new Notice.

```
    <div class="cardContent" [hidden]="isUploaded">
        <div class="dropzone" dropZone (hovered)="toggleHover($event)"
(dropped)="startUpload($event)" [class.hovering]="isHovering">
            <!-- <h3>Upload a Notice</h3> -->
            <div class="file-upload">
                <label class="file-main-label">
                <input class="file-main-input" type="file" multiple
(change)="startUpload($event.target.files)" style="display: none">
                <div class="file-upload-area">
                  <div class="file-upload-icon">
                    <i icons class ="material-icons">cloud_upload </i>
                  </div>
                  <div class="file-label">
                    Click the Button or Drop Notice Here
                  </div>
                </div>
              </label>
            </div>
        </div>
    </div>
```

3.5.6 Image Compression

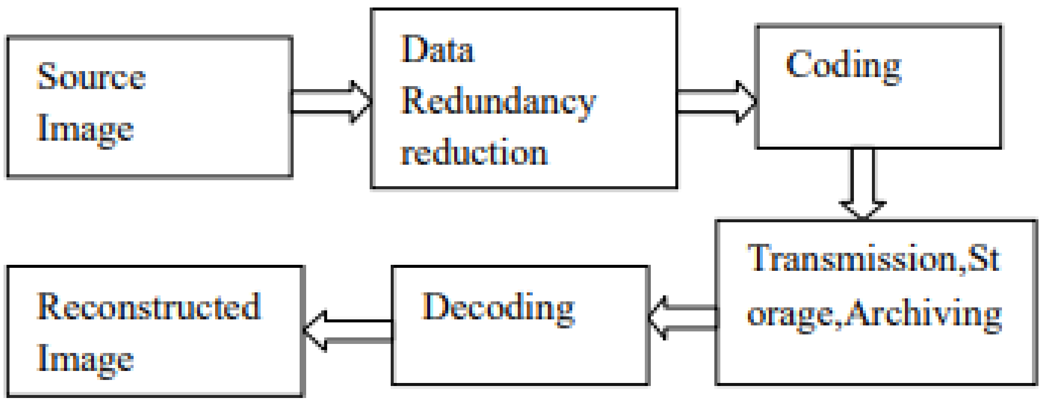


Figure 3.72: Image compression process

The Reducer API by shortPixel allows us to compress an image based on the public URL of the image. The image should be available online (in the cloud storage) in order to be compressed via this API. we can trigger it from our backend that grant us to send a POST request over HTTP.

```
$.post( 'https://api.shortspixel.com/ v2/ reducer.php ',
   JSON.stringify({
    "key": "<<OUR_KEY>>",
    "lossyy": 1,
    "resiize": 1,
    "cmyktorgb": 1,
    "pluginVersion": "JS123",
    "refreshRate": 0,
    "urlsList": ["http://digisuchna.com/assets/logos/JUIT.png",
        "http://www.digisuchna.com/assets/logos/JYC.png"
    ]
}), function(data) {
    alert('success');
});
```

The compression is being done in bunch of steps. Firstly converting an image from CMYK to RGB standards, Then it is being resized to max width of 900 px. After this step image will get compressed around to 60%. To reduce size of image more, it’d pixel density decreased to 72dpi, results into almost 80% of the image compression.

3.5.7 OCR to Extract useful information

This feature returns information about visual content found in an image. Use tagging, descriptions and domain-specific models to identify content and label it with confidence. Apply the adult/racy settings to enable automated restriction of adult content. Identify image types and color schemes in pictures.

OCR technology detects text content in an image and extracts the identified text into a machine-readable character stream. We used the result for search and numerous other purposes. It automatically detects the language. OCR saves time and provides convenience for users by allowing them to take photos of text instead of transcribing the text.

If needed, OCR corrects the rotation of the recognized text, in degrees, around the horizontal image axis. OCR provides the frame coordinates of each word as seen in below illustration.

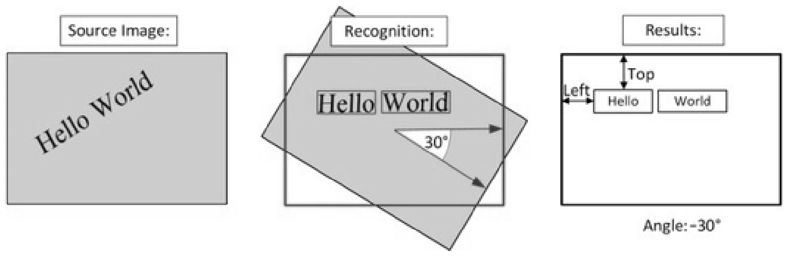


Figure 3.83: Character recognition from an image

Requirements for OCR:

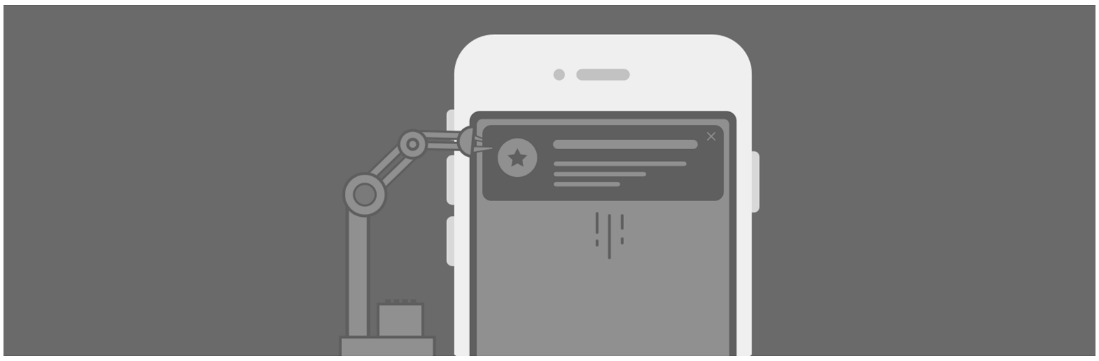
- ➢ The size of the input image must be between 40 x 40 and 3200 x 3200 pixels.
- ➢ The image cannot be bigger than 10 megapixels.

The input image can be rotated by any multiple of 90 degrees plus a small angle of up to '40 degrees.

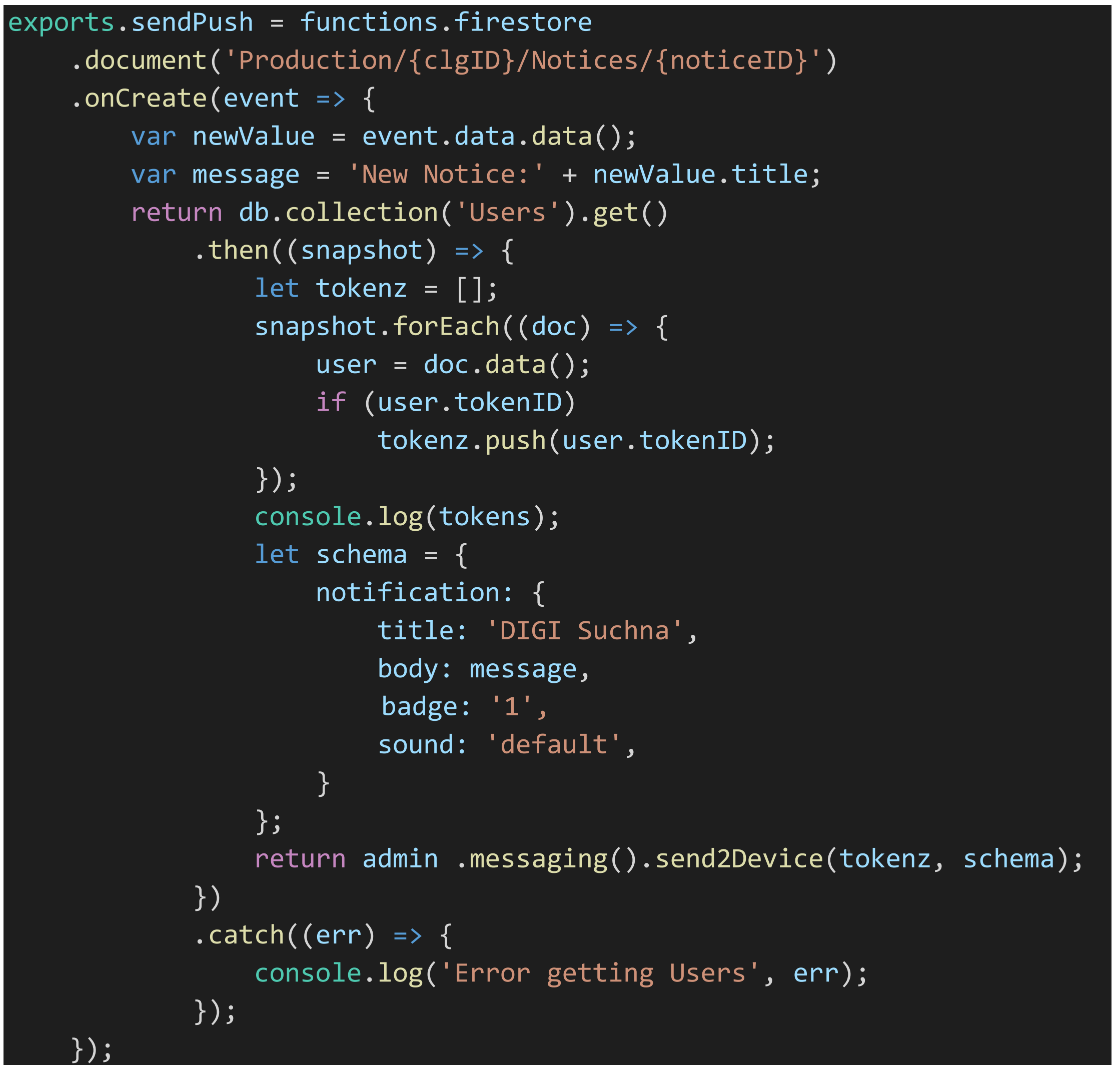
The accuracy of text recognition depends on the quality of the image.

3.5.8 Automated Notifications

A very nice feature of the application is that all the notices will reaches with a notification at the time of publishing Notices.



Digi Suchna delivers the most important updates and notify you for major upcoming events. Firebase Cloud Messaging (FCM) gives a solid and battery-productive association between the server and gadgets that enables us to deliver and get messages and notices on iOS, Android, and the web at no cost.



```
exports.sendPush = functions.firestore
    .document('Production/{clgID}/Notices/{noticeID}')
    .onCreate(event => {
        var newValue = event.data.data();
        var message = 'New Notice:' + newValue.title;
        return db.collection('Users').get()
            .then((snapshot) => {
                let tokenz = [];
                snapshot.forEach((doc) => {
                    user = doc.data();
                    if (user.tokenID)
                        tokenz.push(user.tokenID);
                });
                console.log(tokens);
                let schema = {
                    notification: {
                        title: 'DIGI Suchna',
                        body: message,
                        badge: '1',
                        sound: 'default',
                    }
                };
                return admin .messaging().send2Device(tokenz, schema);
            })
            .catch((err) => {
                console.log('Error getting Users', err);
            });
    });
```

3.5.9 Offline persistence

Cloud Firestore supports offline data persistence. This feature caches a copy of the Cloud Firestore information that is being actively used by the app, so the mobile application can access the data when the device is offline. We can read, write, listen to, and query the cached data, whenever the device reach online. Cloud Firestore can synchronizes all the local changes made by the app to the data stored remotely in Cloud Firestore.

```
firebase.firestore().enablePersistence()
  .then(function() {
      // Initialize Cloud Firestore through firebase
      var db = firebase.firestore();
  })
  .catch(function(err) {
      if (err.code == 'failed-precondition') {
          // Multiple tabs open, persistence can only be enabled
          // in one tab at a a time.
          // ...
      } else if (err.code == 'unimplemented') {
          // The current browser does not support all of the
          // features required to enable persistence
          // ...
      }
  });
```

3.6 Deployment Lifecycle

When we build the application for deployment to an emulator, simulator, physical device or the Cordova Simulate browser, we're building a version of the application specifically crafted for local testing of the app. The app is usually built with debug information packaged into the executable, and the app is signed with a signing key which allows it to work on your local device, not on any device.

3.7 Further Development

The 'DigiSuchna' app has a lot of potential and scope in the present scenario where almost every institution relies majorly on print media to circulate official notices. In the first step, we plan to take our services to other JUIT sister institutes in Noida, Guna, Anoopshehar and nearby universities in Himachal Pradesh. Here we will have our experience in running the service in JUIT as an added benefit. Further we also plan to take our service to various government organizations.

4. PERFORMANCE ANALYSIS

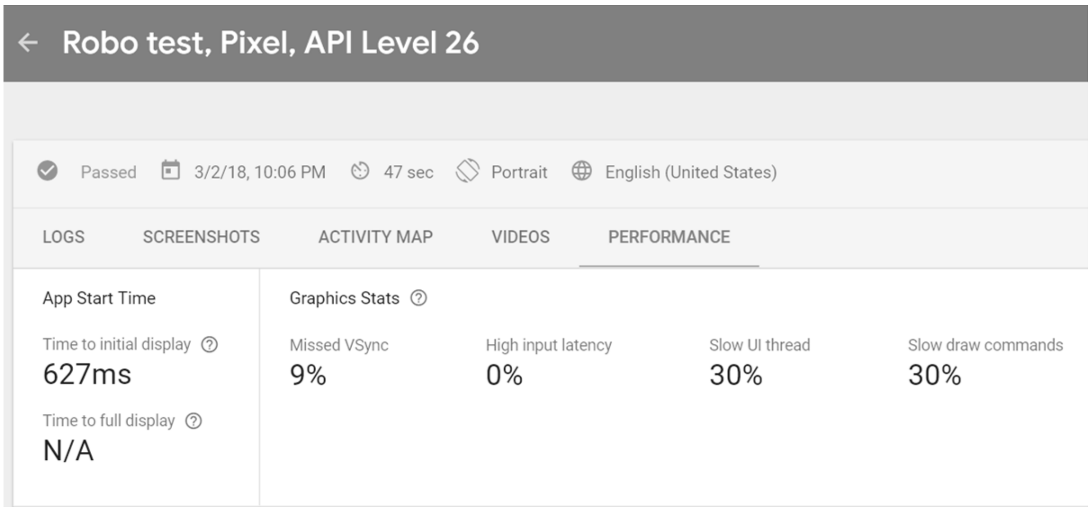
Performance analysis plays a very significant role to measure the acceptance of the product by the end users. This project is purely users (students) oriented and thus the various parameters has to be checked from time to time and should be focused onto enhance the overall performance of the product. We had conducted various tests and analysis on the actual software and the results for same are as followed:

4.1 REALTIME ANALYSIS (on virtual Device)

An Automated Robo Test of the mobile application has been done using the **firebase test-lab**, to test on various performance parameters. To ensure the app quality, the Test Lab for android gives us with virtual and physical devices that permit us simulate actual resource utilization.

4.1.1 Initial Robo Test (3/2/18, 10:06 PM)

The initial test was conducted on Google Pixel device with **API Level 26** in the portrait mode**.** The total duration of the initial test was 47 seconds and the results are as followed:



Robo test, Pixel, API Level 26

Passed | 3/2/18, 10:06 PM | 47 sec | Portrait | English (United States)

App Start Time

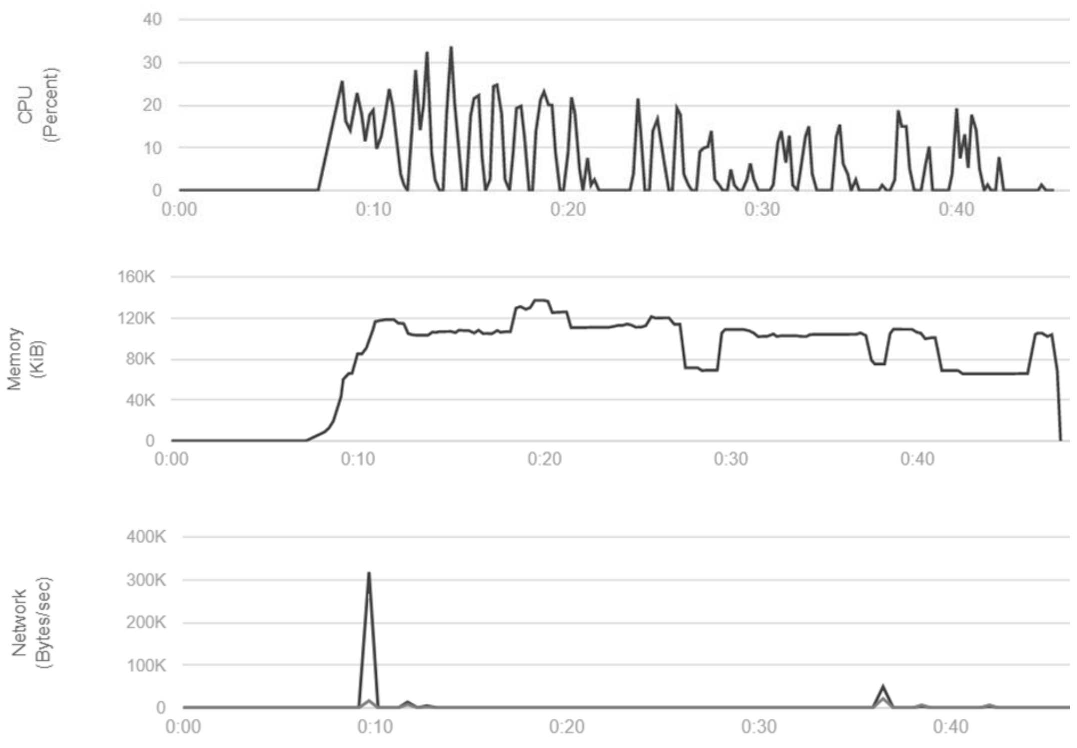
Time to initial display: 627ms

Time to full display: N/A

Graphics Stats

Missed VSync	High input latency	Slow UI thread	Slow draw commands
9%	0%	30%	30%

The major outcome of the test shows that the initial delay is of only **627ms**. The delay is expected and but still can be improved to run out an android application. Other parameters are also quite acceptable and doesn’t affects much on the performance of the running of the application. The test by Firebase test-lab concludes on some other parameters, which are being executed in a Realtime scenario. It computes the CPU Usage, Memory (Primary) utilization and the Network usage (for both, data received and the data being sent over network). The Realtime analysis was as followed:



Graph 4.1: Robo Test 1 on Google Pixel, API Level 26

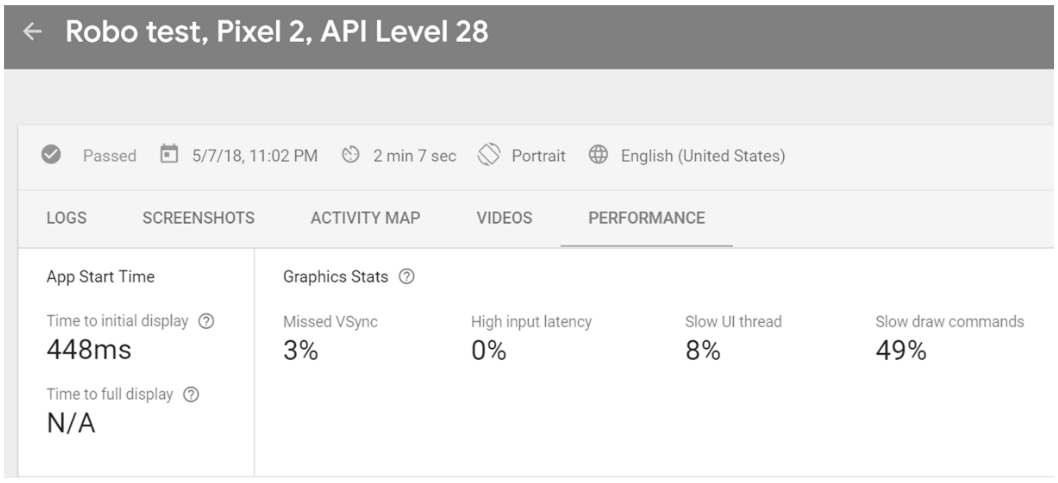
Test Outcome:

The defined test of 47 seconds represents a very clear image on the all basic parameters of an android device. The average CPU usage comes out to be **13%** while it varies for action to action. The CPU usage is still a challenging part for us to be optimised more. Next comes the Memory utilization for the whole application. The application is crafted so light weight, that the whole **package size is itself 5.3 MB only** and the runtime memory intake is also seeming to be quite impressive. The app consumes on an average **947 KB of the primary memory** for the execution of the all processes.

The network usage is also highly optimized with the support of the offline caching and thus the test results represents. The app hardly consumes 300 KB of mobile data, that's only to synchronize the list of updated notices.

4.1.2 Final Robo Test (5/7/18, 11:02 PM)

The major test with stable release (v 0.8.5) was conducted on **Google Pixel 2** device with **API Level 28** in the portrait mode**.** The total duration of the major test was 2 min 7 seconds and the results are as followed:



Robo test, Pixel 2, API Level 28

Passed | 5/7/18, 11:02 PM | 2 min 7 sec | Portrait | English (United States)

LOGS | SCREENSHOTS | ACTIVITY MAP | VIDEOS | PERFORMANCE

App Start Time

Time to initial display
448ms

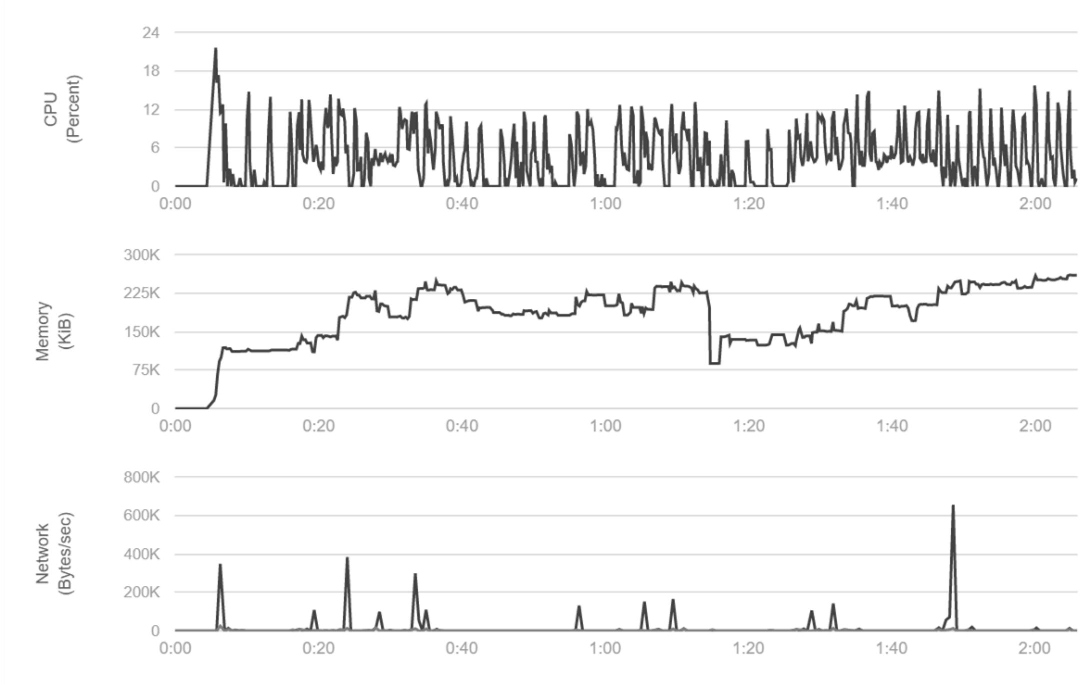
Time to full display
N/A

Graphics Stats

Missed VSync	High input latency	Slow UI thread	Slow draw commands
3%	0%	8%	49%

The major outcome of the test shows that the initial delay is of **only 448ms**. The delay has been highly optimized from the previous test, which was around 627ms. The initial delay of 448ms is quite acceptable as per the industry standards. Other parameters are also get better in terms of stats and the performance. However, there is a slight increase in the Slow draw commands, from 30 % to 49%. Still, not a big deal. it can be more improved in the upcoming updates.

The test by Firebase test-lab also concludes on some other parameters, which are being executed in a Realtime scenario. It computes the CPU Usage, Memory (Primary) utilization and the Network usage (for both, data received and the data being sent over network). The Realtime analysis was as followed:



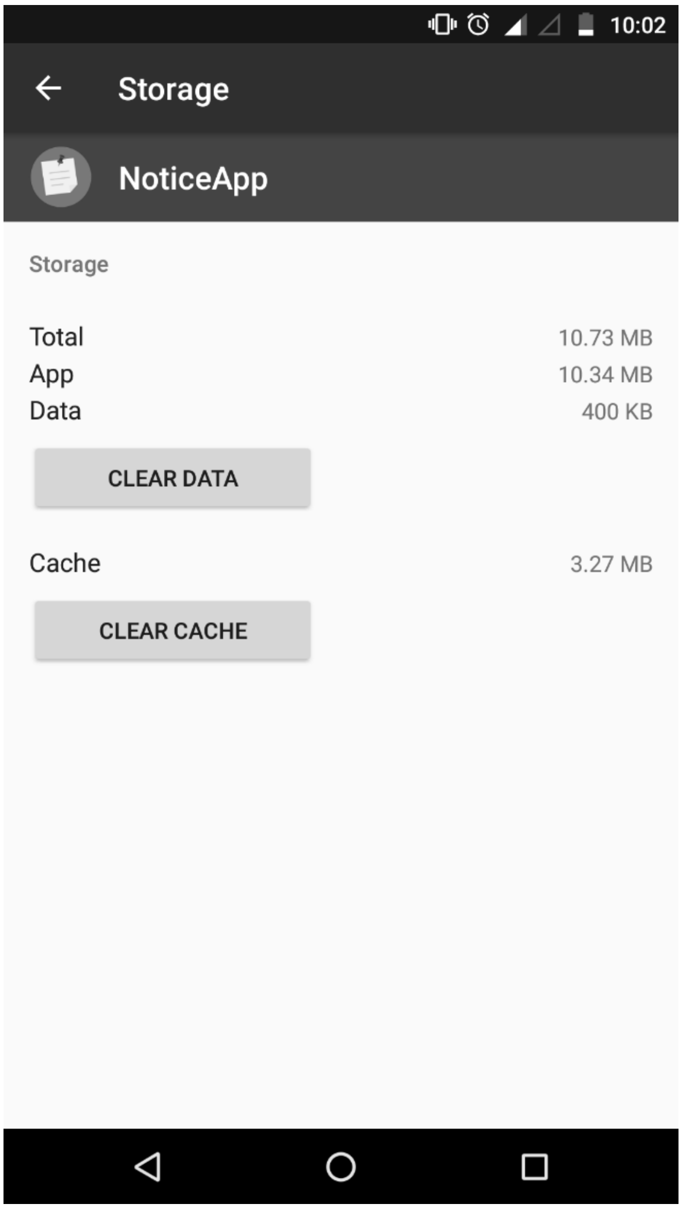
Graph 4.2: Robo Test 2 on Google Pixel 2, API Level 28

Test Outcome:

The defined test of **2 min 7 seconds** represents a very clear image on the all basic parameters of an android device. The average CPU usage comes out to be **5%** which was a highly reduced from 13% of the previous test. This reduction was mainly due to the enhanced logics and algorithms.

Next comes the Memory utilization for the whole application. The application is crafted so light weight, that the whole **package size is itself 5.3 MB only** and the runtime memory intake is also seeming to be quite impressive. The app consumes on hardly 250 **KB of the primary memory** for the execution of the all processes. Again, it was a great reduction since the initial test. Almost 70% of the memory utilization has been reduced.

The network usage is also highly optimized with the support of the offline caching and thus the test results represents. The app hardly consumes 100-300 KB of mobile data, that's only to synchronize the list of updated notices.


10:02

Storage

NoticeApp

Storage

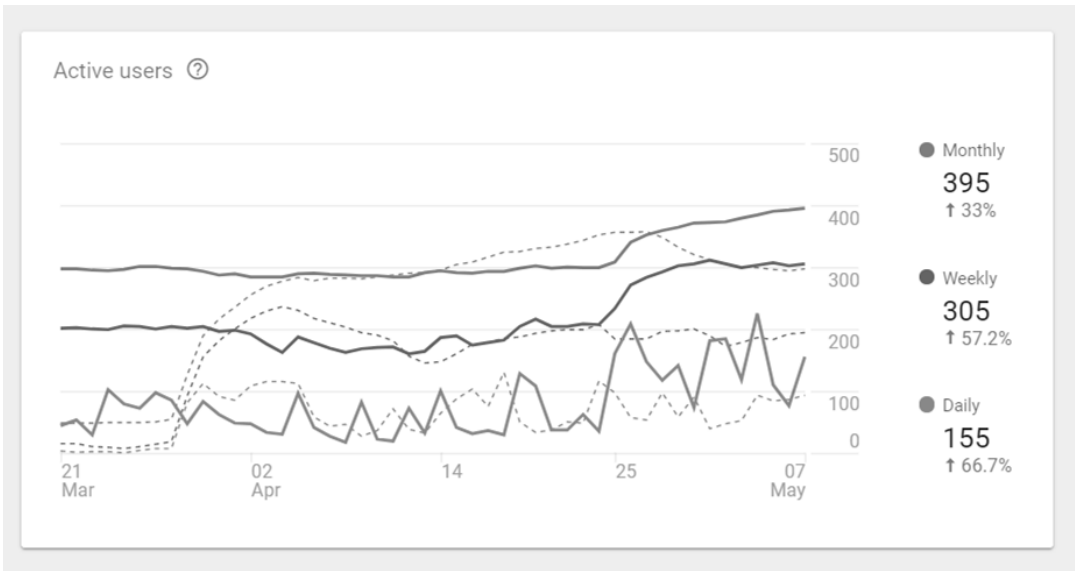
Total	10.73 MB
App	10.34 MB
Data	400 KB

Cache	3.27 MB

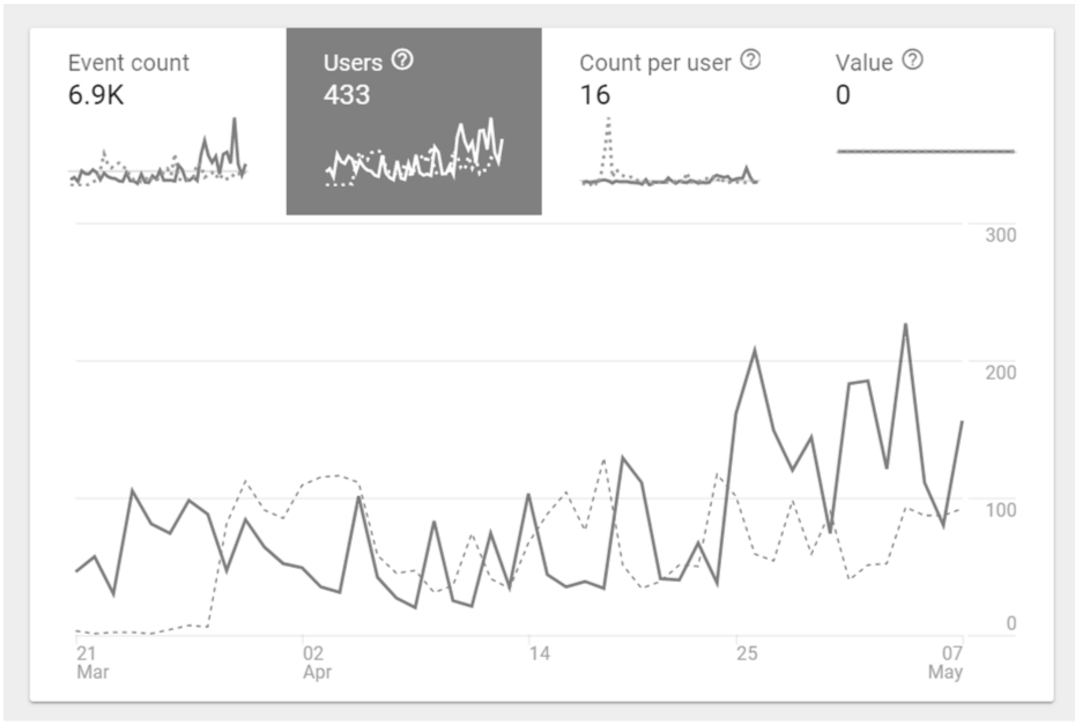
Figure 4.1: Memory consumption of the application

The app's actual usage on the real device is as mentioned in the figure. Total app size is 10.34 MB with Additional data of 400 KBs. The app is capable to store cache for fast accessing and currently stores an amount of 3.27 MB in the client's individual device.

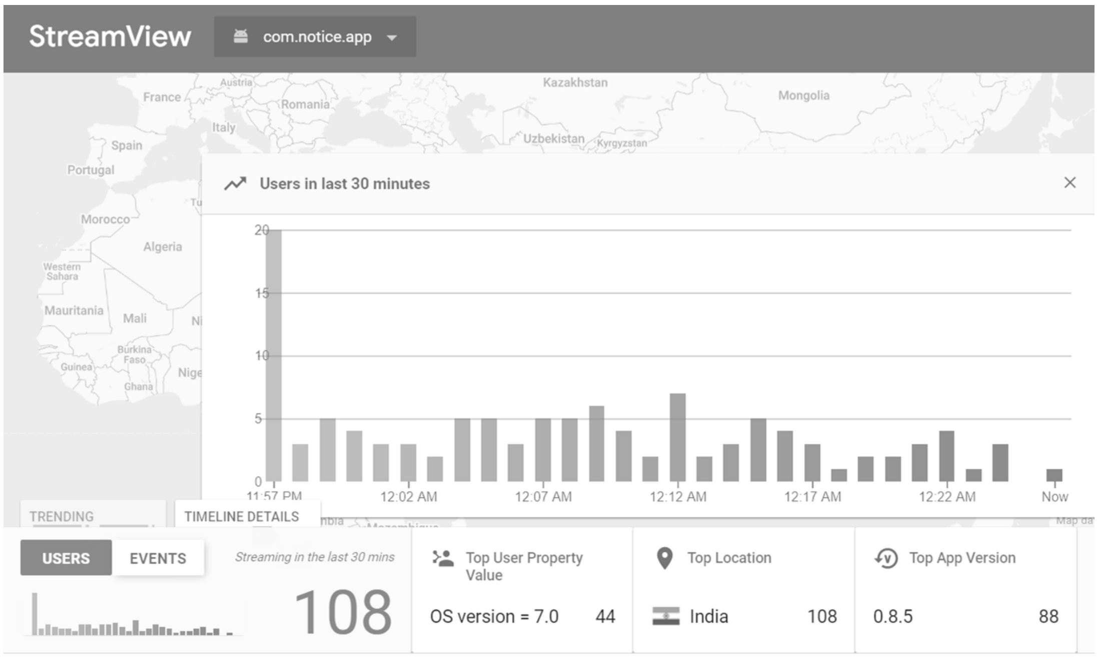
4.2 USER INTERACTIONS

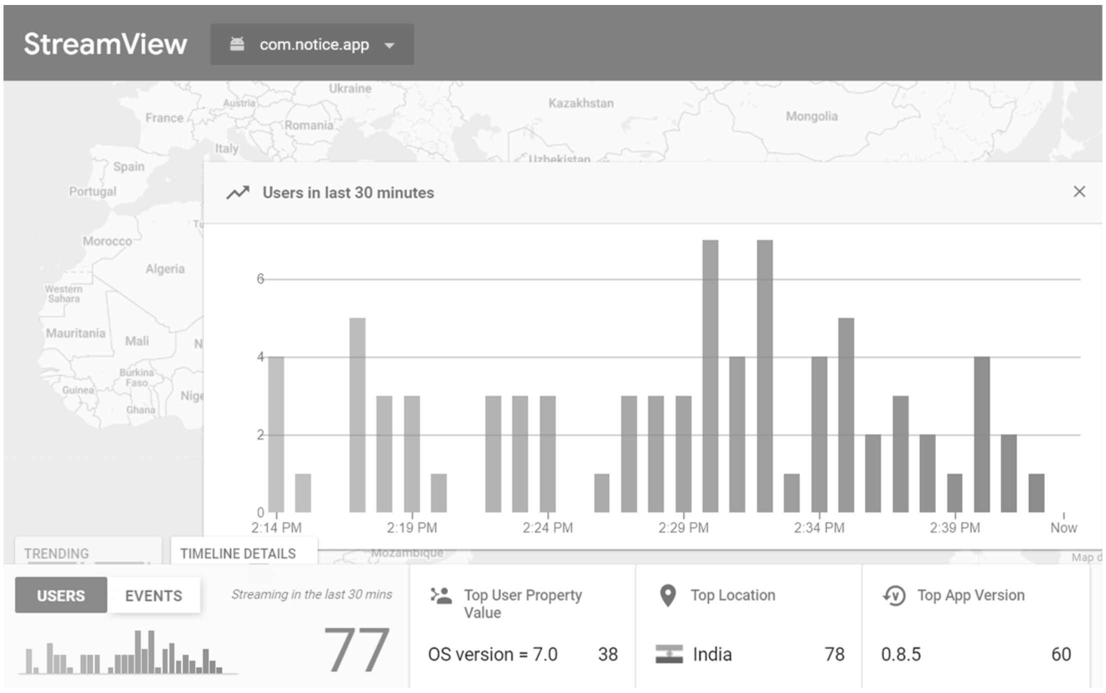


Graph 4.3: Users engagement from 27 MAR to 07 MAY 18 ***(compared with last cycle)***



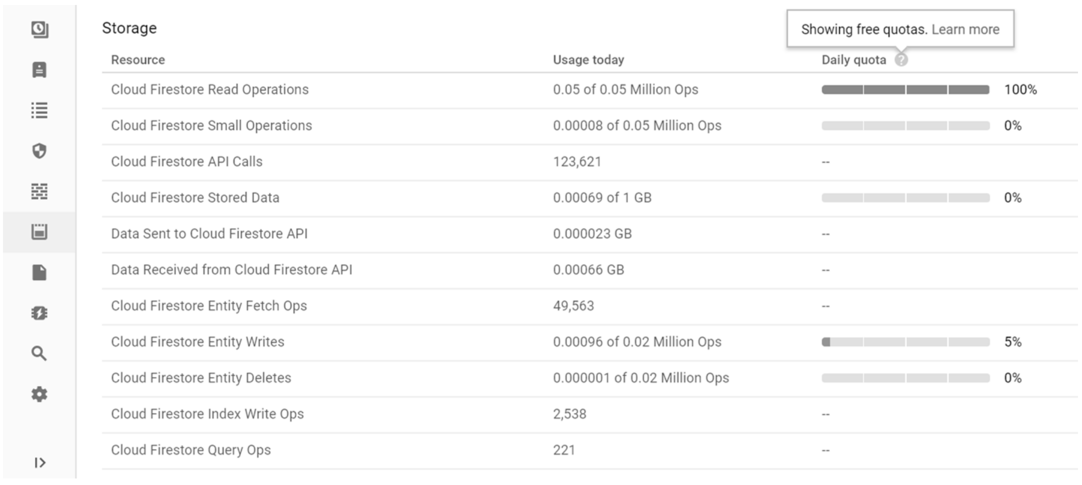
Graph 4.4: App's Event counts for all users.





Graph 4.5: Live user's StreamView for 30 minutes of a) Publishing a notice; b) Random Involvement.

4.3 SCALABILITY ANALYSIS


Storage

Showing free quotas. Learn more

Resource	Usage today	Daily quota
Cloud Firestore Read Operations	0.05 of 0.05 Million Ops	100%
Cloud Firestore Small Operations	0.00008 of 0.05 Million Ops	0%
Cloud Firestore API Calls	123,621	--
Cloud Firestore Stored Data	0.00069 of 1 GB	0%
Data Sent to Cloud Firestore API	0.000023 GB	--
Data Received from Cloud Firestore API	0.00066 GB	--
Cloud Firestore Entity Fetch Ops	49,563	--
Cloud Firestore Entity Writes	0.00096 of 0.02 Million Ops	5%
Cloud Firestore Entity Deletes	0.000001 of 0.02 Million Ops	0%
Cloud Firestore Index Write Ops	2,538	--
Cloud Firestore Query Ops	221	--

Figure 4.2: Database scalability issue with daily quota (on 26 APR 18)

On the analysis and based on the actual stats, it seems like the system is not ready to be scalable to reach all the students. We are limited with upto 50,000 database read daily, which isn't quite enough. We have already crossed this quota twice in the last month. The given screenshot is of 26 APR, when on publishing six notices, our servers get crashed, and thus needed to be optimised highly.

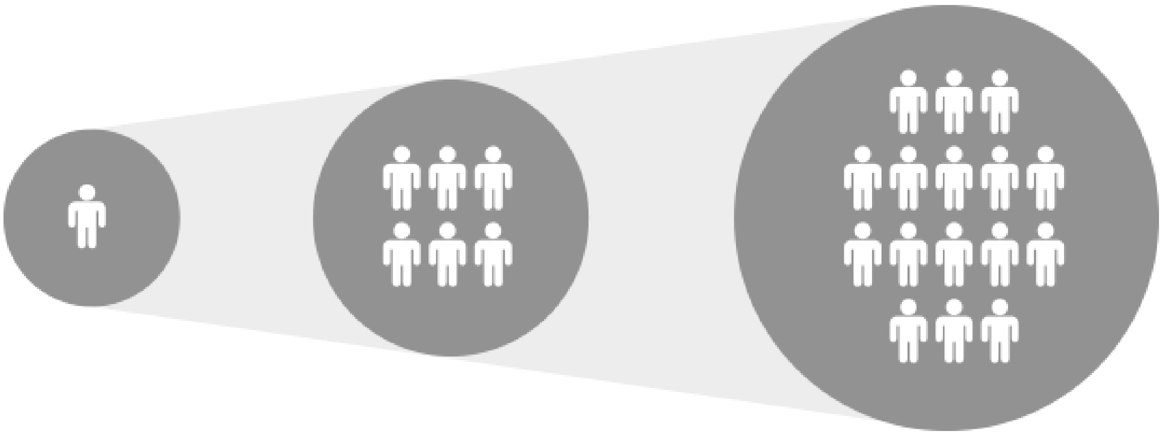


Figure 4.3: Users growth rate: Eight users with every single notice

4.4 S.W.O.T. ANALYSIS

The SWOT Analysis is very useful analysis to understand our basic Strengths and manipulate Weaknesses. It also help us to identify both the Opportunities favourable to us and to work around the Threats we usually do face.

4.4.1 Strengths

- ✓ Quick source of information with instant notification.
- ✓ Receive the information directly at anywhere across the globe without any duplicity.

4.4.2 Weaknesses

- ➢ Difficult to expand
- ➢ Audience is less
- ➢ Team is a Small for market

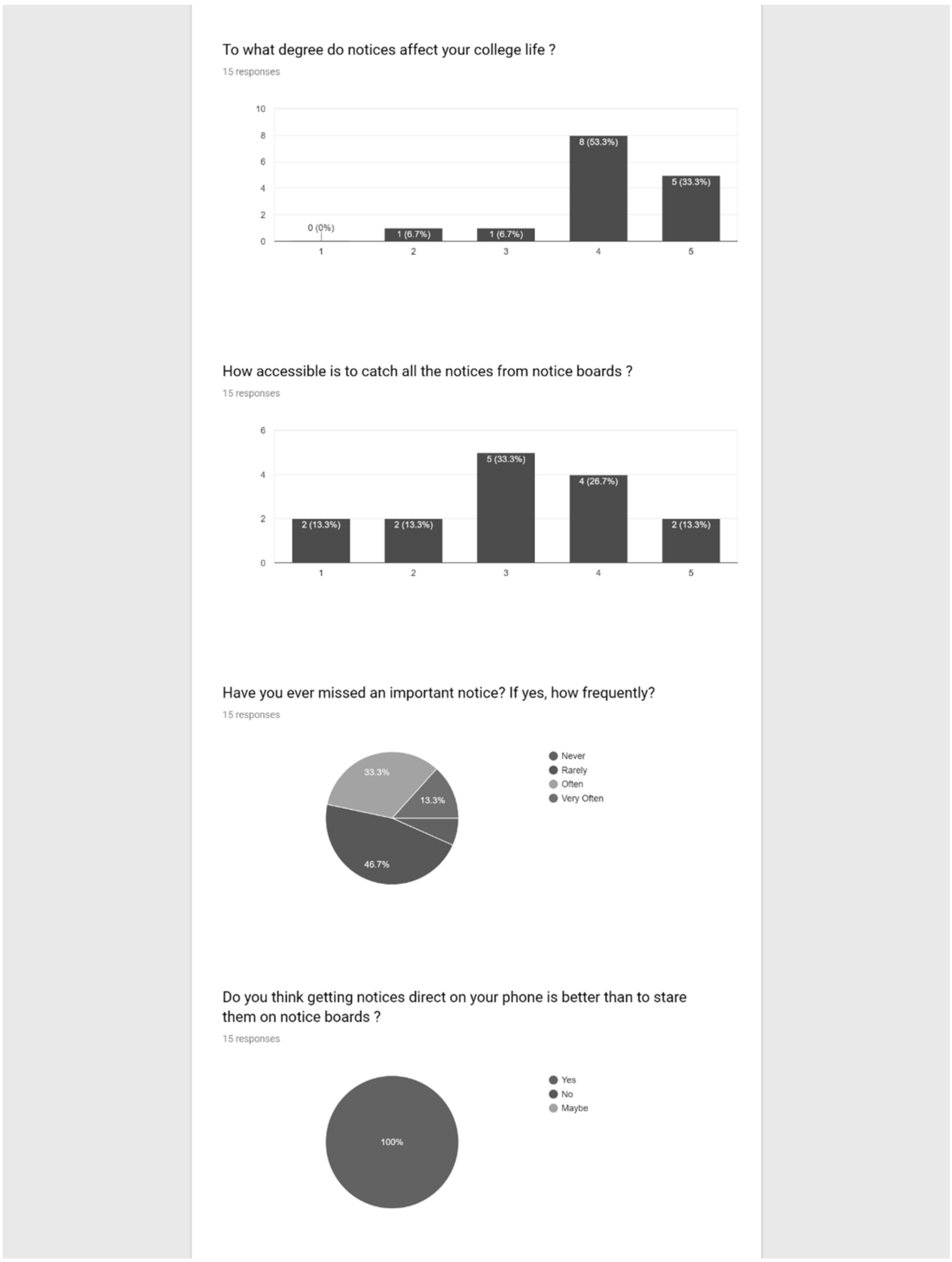
4.4.3 Opportunities

- ➢ Maybe create a difference
- ➢ Environmental issues to resolve

4.4.4 Threats

- ➢ little bugs in application to resolve

4.5 STATISTICAL SURVEY



To what degree do notices affect your college life ?

15 responses

How accessible is to catch all the notices from notice boards ?

15 responses

Have you ever missed an important notice? If yes, how frequently?

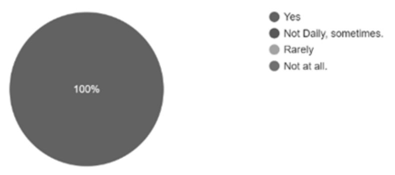
15 responses

Do you think getting notices direct on your phone is better than to stare them on notice boards ?

15 responses

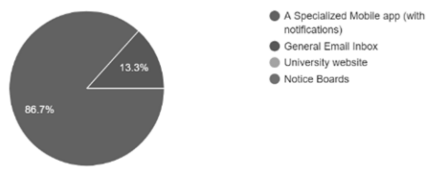
Do you access the internet daily on your cell phone?

15 responses



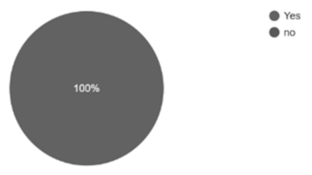
Whats your preferred mode to get all the University notices?

15 responses



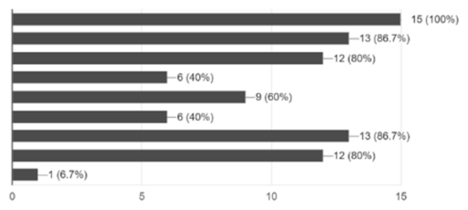
Are you using Digi Suchna ?

15 responses



Which notices you are looking for the most :

15 responses



Graph 4.6: Statistical survey on the actual users.

5. CONCLUSION

As a result, we are able to work around an android application that authenticates users, using the OAuth mechanism & thus provides REST APIs to third-party clients as well. The Authorization server is being hosted on the Azure Cloud.

We currently have deployed the DIGI Suchna app in JUIT, campus under beta testing phase and got a fruitful response from the JUIT Students with over 300 registered users on the DIGI Suchna.

5.1 Observe the notice delivery across devices

We observed a very clear intimation that people use such apps across all the devices. A user can regularly visit the app on their mobile device, can also open it in their desktops, and usually recheck it later on their tablet device. The project is a hybrid application works across all the major devices.

5.2 Get Notifications on the go

Digi Suchna delivers the most important updates and notify you for major upcoming events. Firebase Cloud Messaging (FCM) gives a solid and battery-productive association between the server and gadgets that enables us to deliver and get messages and notices on iOS, Android, and the web at no cost.

5.3 Save Papers

The project delivers best in service to Stay updated and create a step a towards Green India as we mentioned also that we are saving 8760 trees a year.

REFERENCES:

[1] *Kitty Arora, Manshi Shukla* "A Comprehensive Review of Image Compression Techniques"(IJCSIT) International Journal of Computer Science and Information

[2] *Niyigena Jean Pierre and Mukiza Octavie* "Review of PhoneGap APIs Accessing the Native Mobile Platform APIs" Lecture Notes on Software Engineering

[3] *Neha Srivastava, Uma Shree, Nupa Ram Chauhan, Dinesh Kumar Tiwari* "Firebase Cloud Messaging (Android)" IJIRSET

[4] *Gurleen Kaur & Deepak Aggarwal, BBSBEC Fatehgarh Sahib(Punjab)* "A Survey Paper on Social Sign-On Protocol OAuth 2.0" Journal of Engineering, Computers & Applied Sciences (JEC&AS)

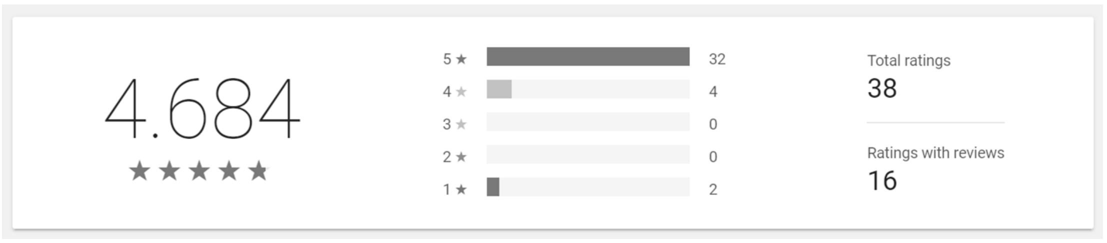
[5] N*ajib Ali Mohamed Isheawy And Habibul* Hasan "Optical Character Recognition (OCR) System "IOSR Journal of Computer Engineering (IOSR-JCE)

[6] Hamad, Karez & Kaya, Mehmet "A Detailed Analysis of Optical Character recognition"

APPENDICES:

# User Acceptance & Reviews:

The application is being highly supported by the actual users and the actual ratings on the play store are mentioned below:



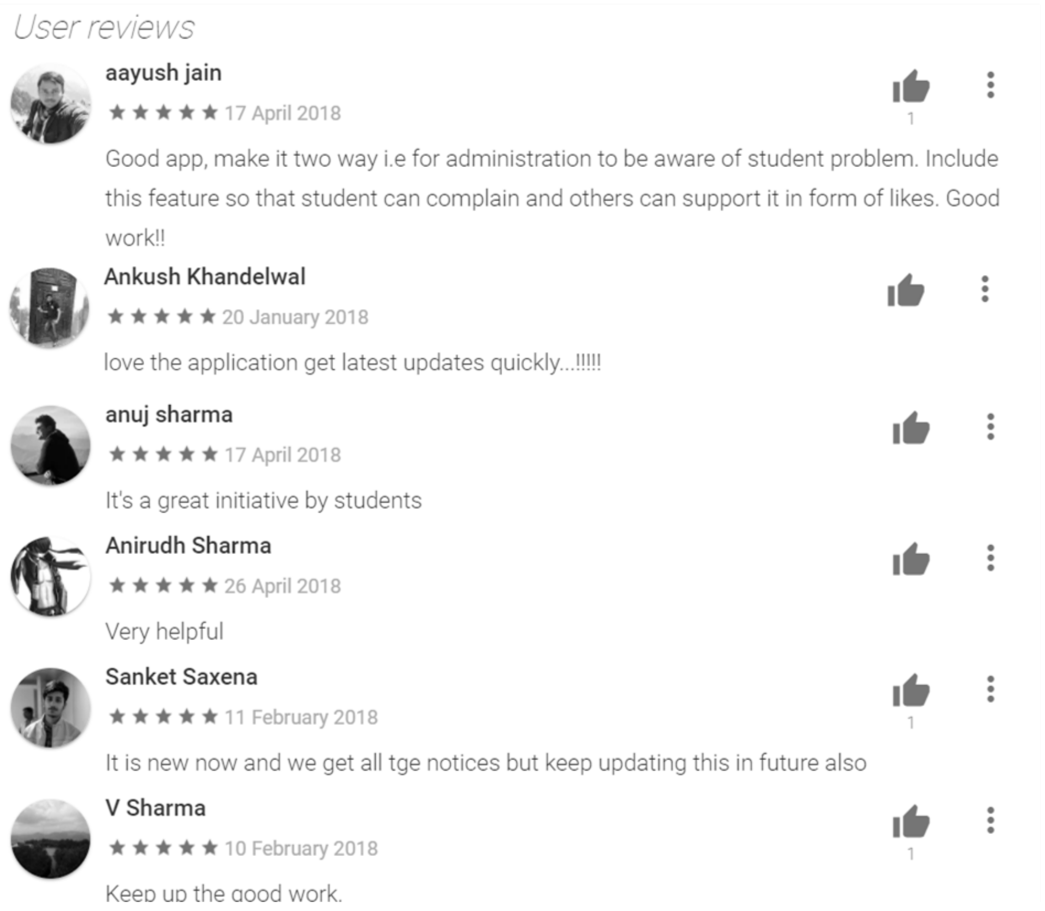
4.684
★★★★★

5 ★	▇	32
4 ★	▇	4
3 ★		0
2 ★		0
1 ★	▇	2

Total ratings
38

Ratings with reviews
16

Figure: Users Ratings and Reviews



User reviews

aayush jain
★★★★★ 17 April 2018
Good app, make it two way i.e for administration to be aware of student problem. Include this feature so that student can complain and others can support it in form of likes. Good work!!
1

Ankush Khandelwal
★★★★★ 20 January 2018
love the application get latest updates quickly...!!!!!

anuj sharma
★★★★★ 17 April 2018
It's a great initiative by students

Anirudh Sharma
★★★★★ 26 April 2018
Very helpful

Sanket Saxena
★★★★★ 11 February 2018
It is new now and we get all tge notices but keep updating this in future also
1

V Sharma
★★★★★ 10 February 2018
Keep up the good work.
1

Figure: Positive Reviews by the users.