Student Management System

Table of Contents

Introduction

Currently Store application isn't fully working as it doesn't have a server implementation to download from and not working like a store, therefore I propose a solution where teachers can upload study material application and students can download applications from store, use them to learn, record marks, and analyze the marks they scored.

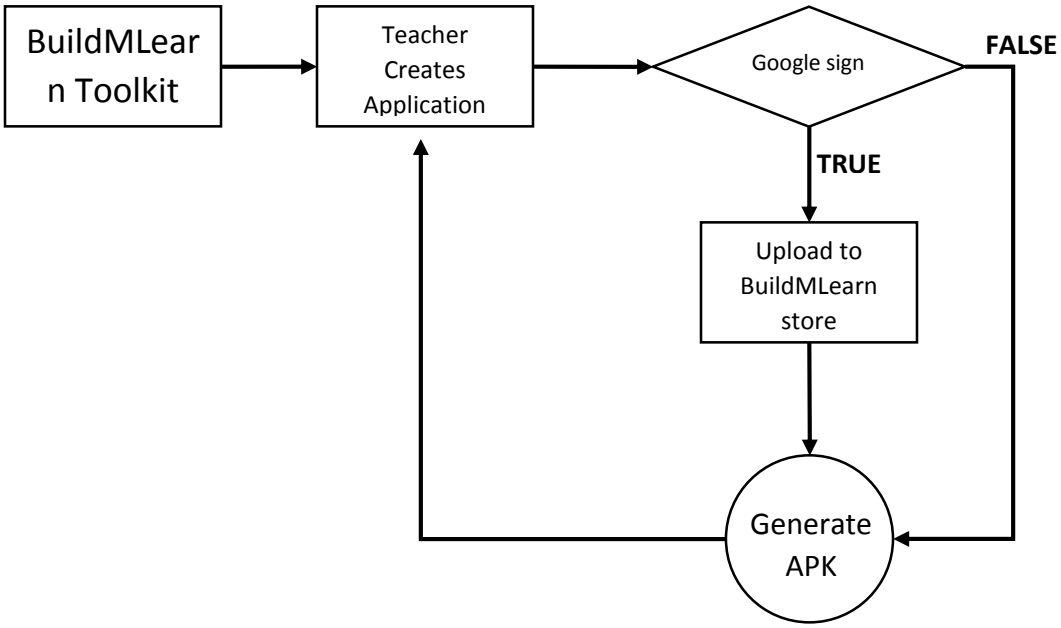
By this a student will be able to analyze and summarize their performance over the time they are using BuildMLearn Application. For this I came up with this idea of a student management system. Where students are shown their reports and easily browse through the study material we have.

Few Keywords in the document-

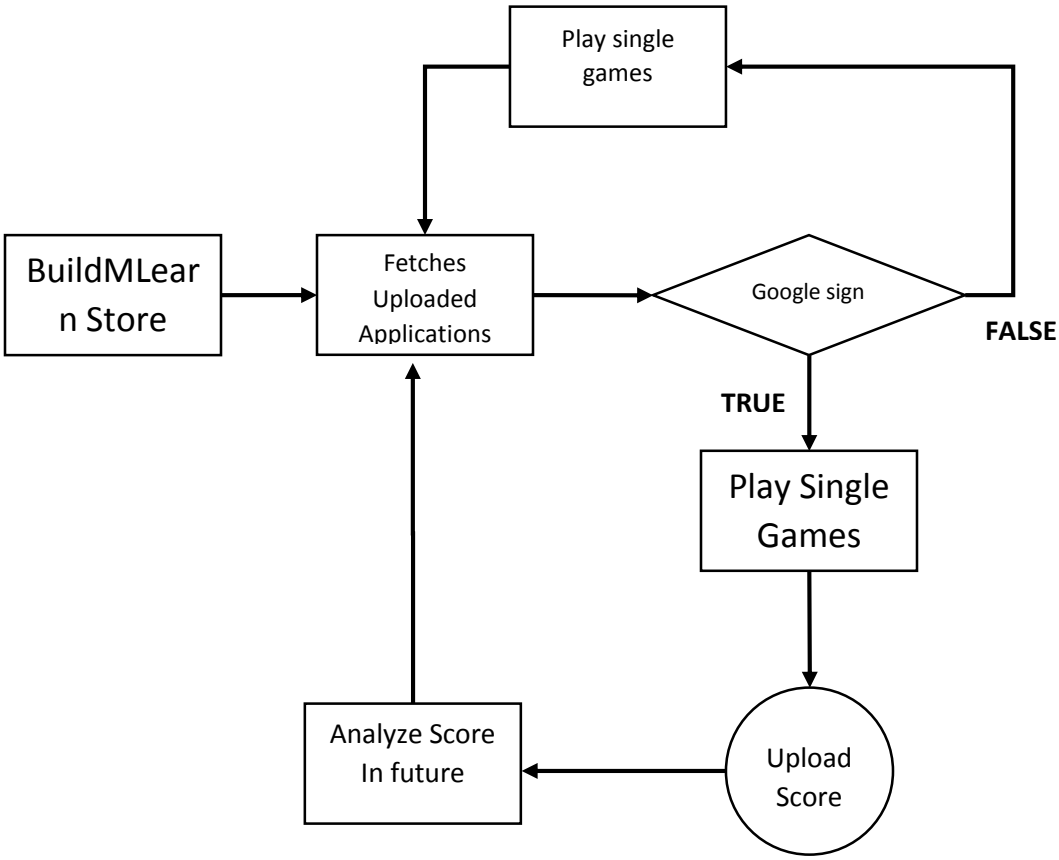
- **Teachers-** They are the people willing to create a **study application** through the BuildMLearn Toolkit using the 8 templates.
- **Students-** They are the people willing to learn through the uploads by teachers.

A flow chart of the BuildMLearn application I'm planning to develop.

A teacher uses the Toolkit and uploads the created study application by logging in with their respective Google ID's. They still can generate APK's and distribute.



A student uses **BuildMLearn Store** to fetch the uploaded applications by teachers and learn through them.



Changes to Be Done in the Existing Applications –

- Changes in the BuildMLearn Toolkit
    - Implement sign in with google option in the navigation drawer.
    - Implement upload option in the template building layouts.
    - Implement share button to share the uploaded study application to Google+ Account.
- Changes in the BuildMLearn Store
    - New User Interface (Shown in Mockups).
    - Implement Sign in with google.
- Changes in the BuildMLearn Toolkits
    - Generate Scores.

- Integrate Google Play services to upload score. (Templates mLearning, Flash cards, Video do not have any score generation but they will be definitely uploaded and meant for basic learning.)
- Integrate the new Templates coming this year.
- Implement reading XML file from the server.

New Additional features-

- Choose Tags in the start of the app, which will show study material applications having these tags in the home page.
- Saving the questions and answers they were not able to answer.
- Sending invitations to your friends.
- **Achievements** to keep excitement going on. These are the 5 initial achievements I'm planning to implement -
  - Mighty Mighty Scholar – Score 100% in 5 games.
  - Social Master – Invite 20 other students.
  - Study Master – Complete 5 quiz's.
  - Games Master – Share 5 games.
  - Topper – Play all subjects Games.
- Links for the Mobile Applications (Label the diagram, Learn from Map, Measure Sensor Data, practice Handwriting, tell the time and the applications selected this year).
- **Analyzing User scores -** I will be using MPAndroidChart for generating Line chart, Bar charts, Pie Charts and Radar charts. These charts will be based upon the marks scored in the games played and categorized according to the subjects.
  Categorized as follows-
  - **Overall –**
    - Marks Scored
    - Games Played

- **Subjects –** These will show complete information of wrong, correct and total scores of the games played in each subject. (Detailed in Mockup's)

As in the database schema I have the information of the wrong questions info of a game ID, I will be able to parse the information at which the graph's will be based on.

Some Technical Specifications

- Scopes used in Login in ToolKit application is Scopes.PROFILE as we need only the basic information like name, Email-address.
- Adding Deep Links for quickly and easily sharing, accessing a Study Application uploaded in the store.
  A Deep link URL will be like – (**buildmlearn.org/store/app?id=123456)**
  123456 referring to the unique ID given to an upload of **study application.**
  There can be two cases -
  - If the URL is opened in a Non-Android device (I will use JavaScript to detect which device) we can give them a message open this link in your android phone to quickly browse to the uploaded study application.
  - If the URL is opened in an Android device then it directly opens up in the BuildMLearn store application by a deeplink like (**buildmlearn://game?id=123456)**
- Implement ButterKnife to annotate fields with @Bind and a view ID for Butter Knife to find and automatically cast the corresponding view in the layout.

- Instead of manual UI testing, implement Espresso Fast Automated Android UI Testing which will help us in better testing of the UI for the bugs.
- Implement ORMLite Database Object Related Mapping.
- Implement Retrofit Network Library in both the applications for connectivity to server.
- Implement Glide Image caching for loading images and gravatars, I prefer Glide over Picasso as it's faster and better image caching.
- I did some test's recently for network connectivity to analyze which would be the best library internet connectivity for an android application. I did these test on a 1.5 Mbps internet connection to fetch from the server having custom headers for authorization.

  Total size to fetch was **487 Kb**
  I did test's for Apache Client (Old), HTTPUrlConnection, Retrofit and volley using the latest version of these libraries.
  I created 4 different projects, each containing one library. And wrote code to connect to server and fetch JSON. I did 5 tests with each of the libraries and I got the following results –

Name	Average Time taken (ms)	Minimal Lines of Code
HTTPURLConnection	3164	28
Retrofit	2831	9
Apache Client	3265	24
Volley	3091	25

  Retrofit being the fastest and the smallest, as it does everything (Sending Request, Reading from the server) on its own.
  Hence I found Retrofit will be the best in writing the network connectivity. And will be using this library.

Server Implementation

As we know that the toolkit builds an apk file which uses an XML as its main source of information. This file will be sent to the server.

I propose the following schema for this purpose –

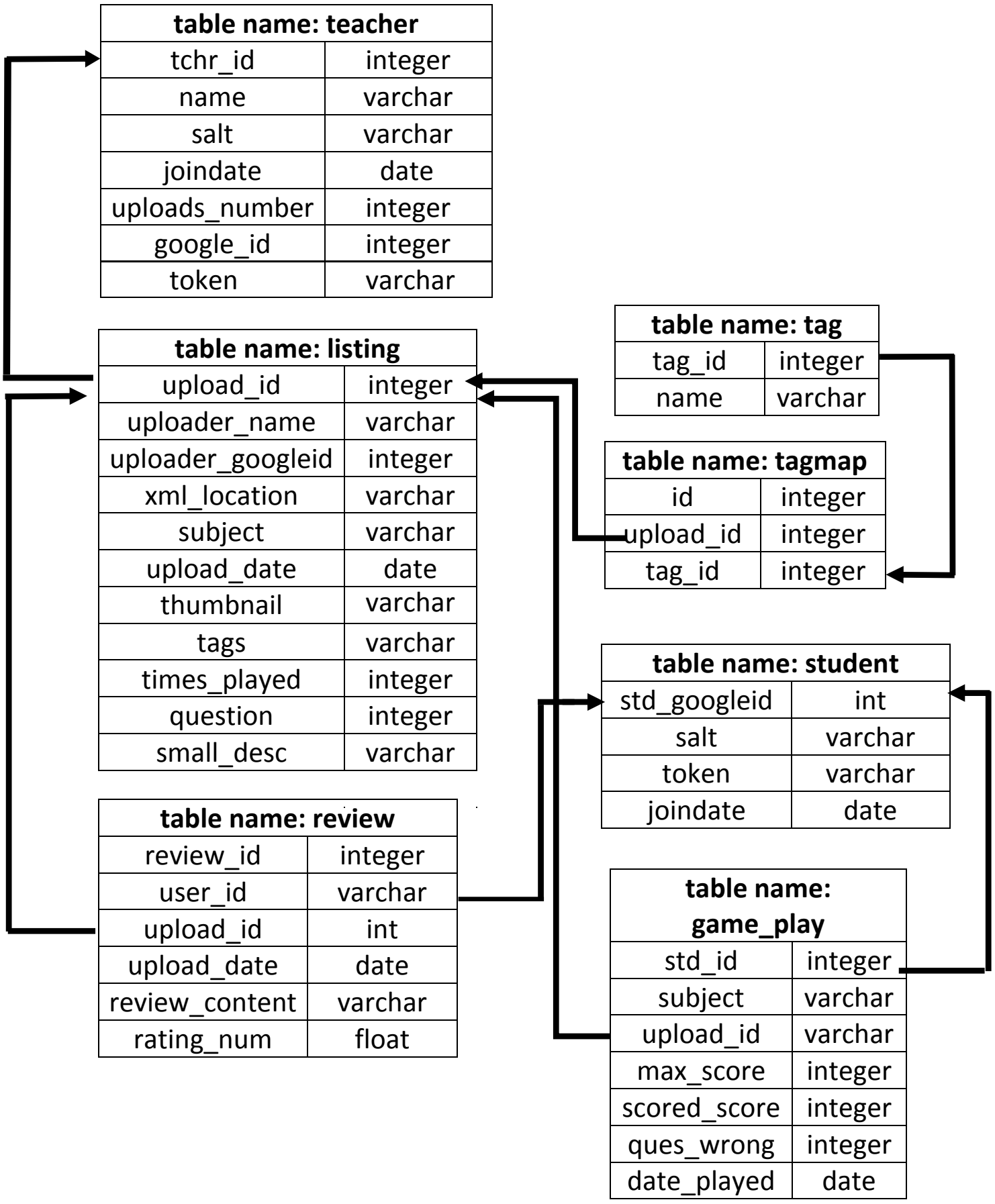


table name: teacher
tchr_id integer
name varchar
salt varchar
joindate date
uploads_number integer
google_id integer
token varchar
table name: listing
upload_id integer
uploader_name varchar
uploader_googleid integer
xml_location varchar
subject varchar
upload_date date
thumbnail varchar
tags varchar
times_played integer
question integer
small_desc varchar
table name: tag
tag_id integer
name varchar
table name: tagmap
id integer
upload_id integer
tag_id integer
table name: student
std_googleid int
salt varchar
token varchar
joindate date
table name: review
review_id integer
user_id varchar
upload_id int
upload_date date
review_content varchar
rating_num float
table name: game_play
std_id integer
subject varchar
upload_id varchar
max_score integer
scored_score integer
ques_wrong integer
date_played date

Uses of these tables:

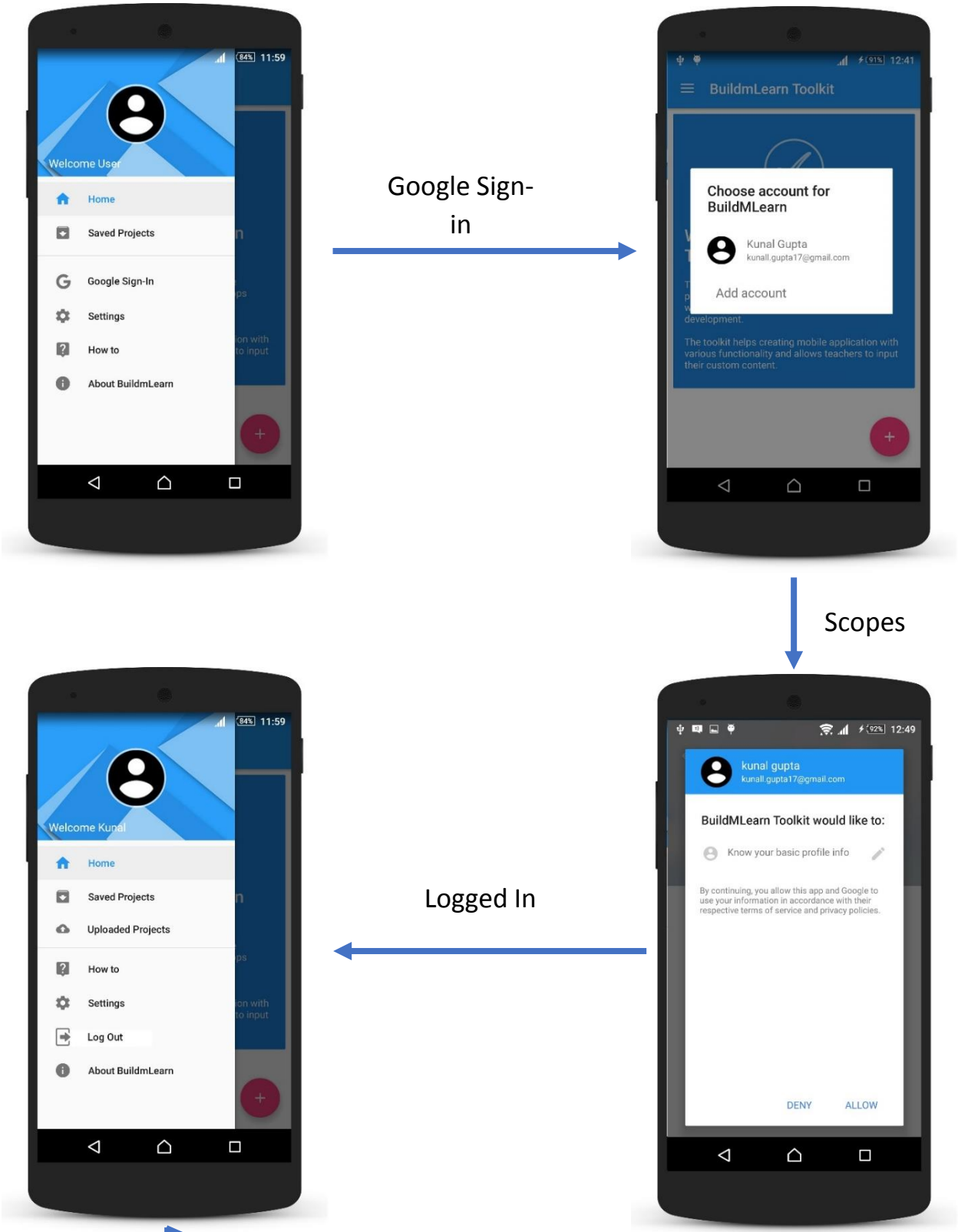
- **Teacher –** When a teacher logs with google in the Toolkit Application i'll add their information in this table.
- **Listing -** When a Teacher in Toolkit application uploads a study material application I will add a new entry in the **listing** table. If no thumbnail is uploaded then I will generate a word Icon (Circle with a letter) at runtime. where xml_location defines the path of the xml file stored in the server.
- **User -** I will be using token authorization in these applications as google stated here. Therefore this table is needed.
- **Tag, tagmap - I'm** using a very popular solution of storing tags in a database called "toxi solution".
- **Review -** If a student has played a particular study material application then only he can give a review about it, on submitting the review, a new entry will be added to the **review** table.
- **Game_play –** This is for storing information of each game played by any user, and storing the questions they weren't able to answer.

From these databases

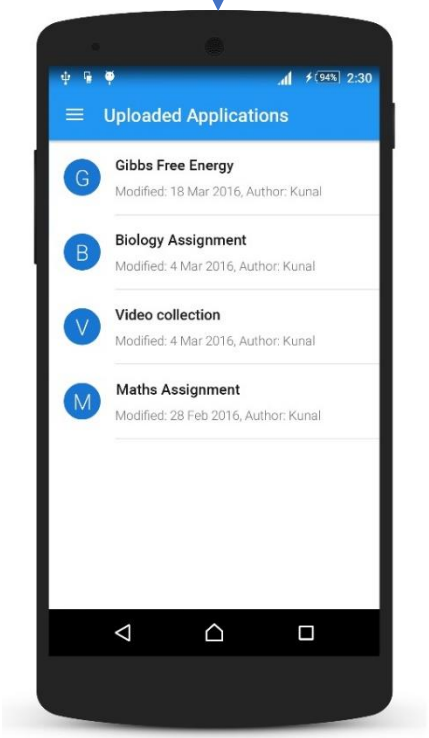
- I will be able to fetch all the files and do the sorting and filtering on the basis of
    - Date Uploaded
    - Uploader name
    - Subject
    - Tags
- Using the **game_play** I will be able to analyze the student's performance over a period of time.

UI DESIGN/WIREFRAMES

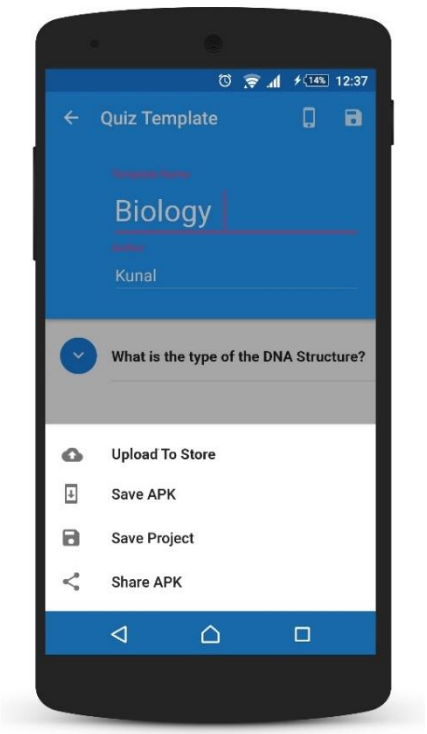
TOOLKIT



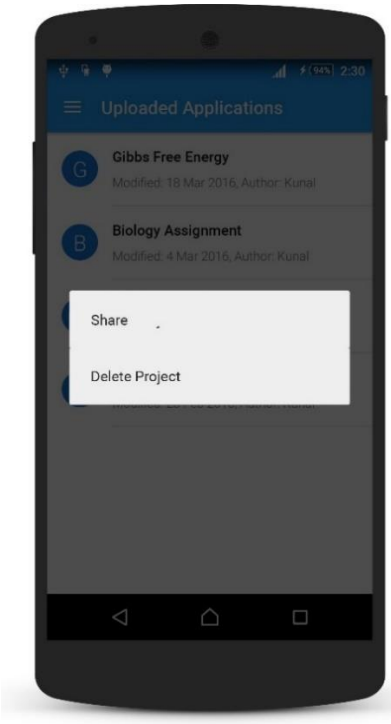
Uploaded Applications



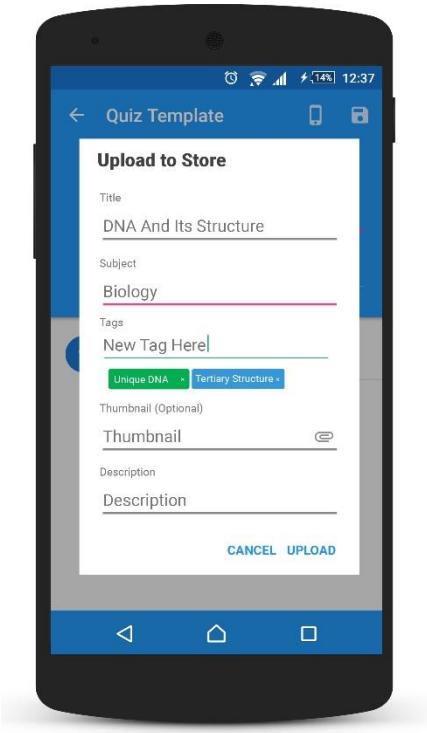
Upload to Store



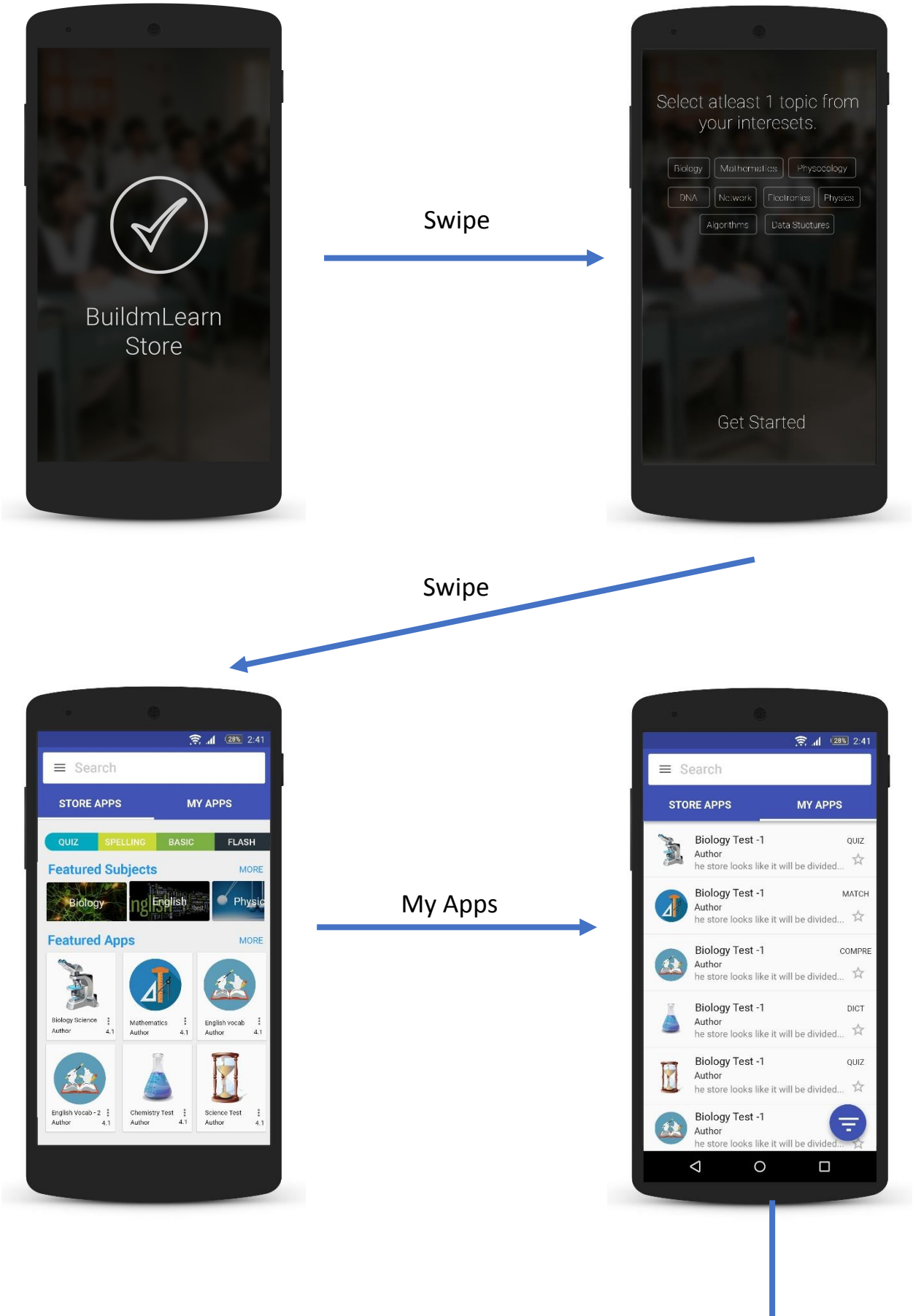
Context Menu



Upload To Store

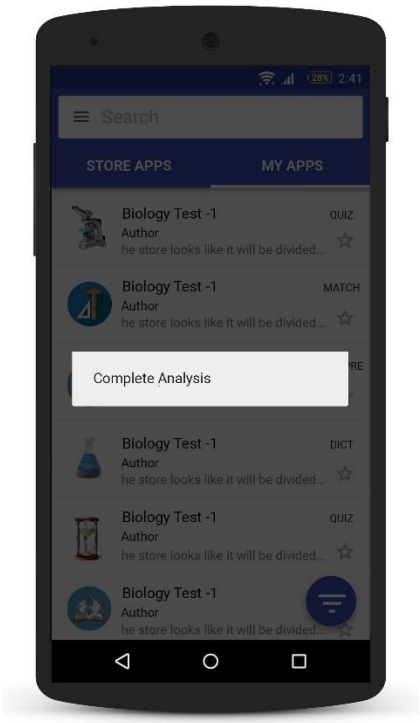


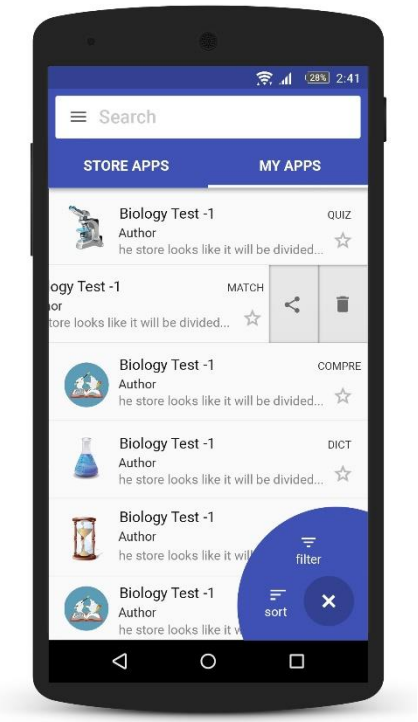
BUILDMLEARN STORE



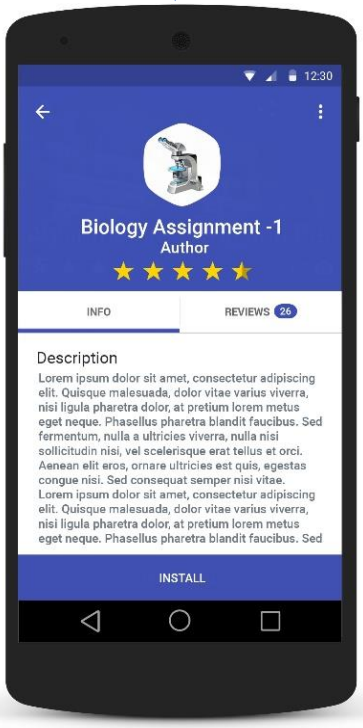
Swipe App or Floating Action Button

Context Menu

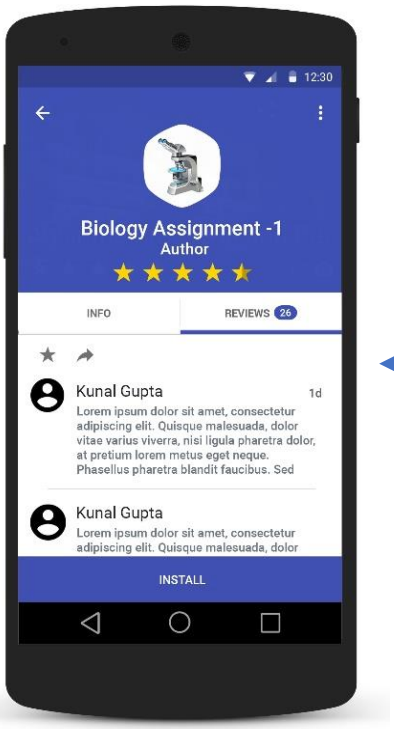




Open study Application

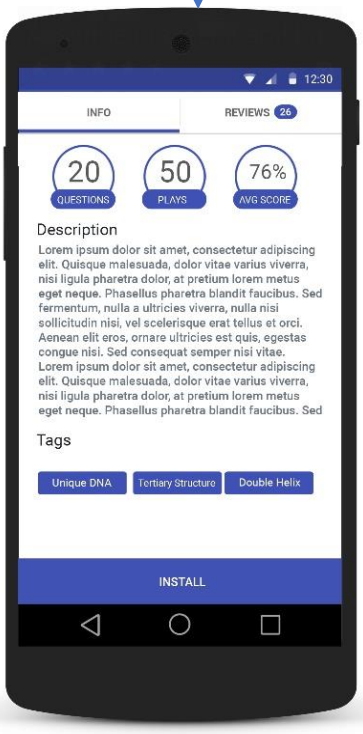


Reviews Tab

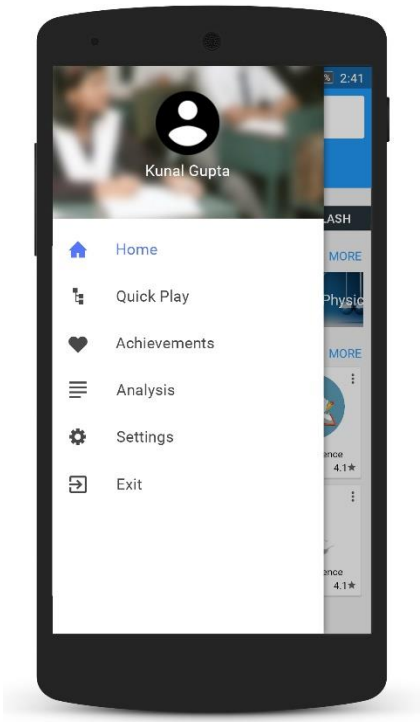


Scroll in Reviews Tab

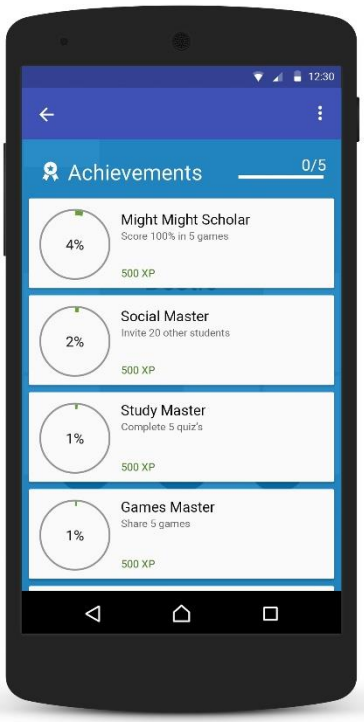
Scroll in Info Tab



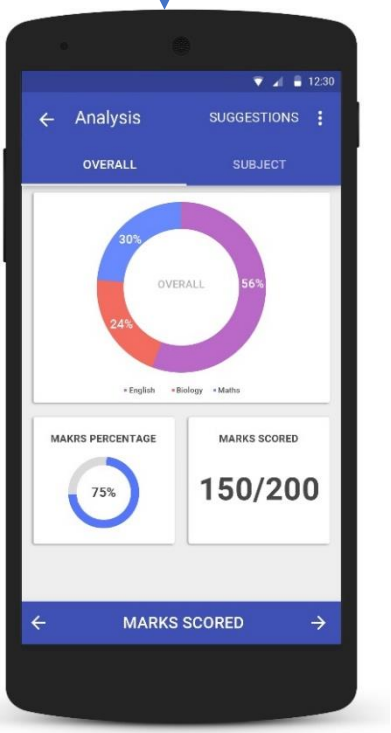
NAVIGATION DRAWER OF APP STORE



Achievements



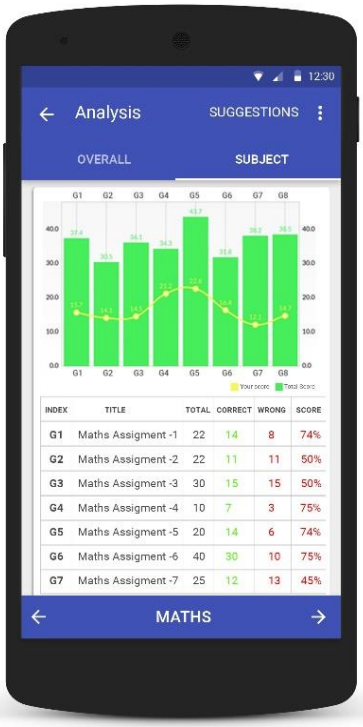
ANALYSIS



Win an Achievement

Change Graph

Analysis Subject Wise



[Full Resolution Mockups]

Road-map (Timeline)

GSOC goes for about 13 weeks in which

- 1 month (4 weeks) for Community Bonding period.
- 1 month before Mid Term Evaluations.
- Approximately 2 months after the Mid Term Evaluations.

Task Name	Start Date	End Date	Time (Days)
Google Summer Of Code	**25th April**	**23rd August**	**116**
Community Bonding	**22nd April**	**22nd May**	**30**
Gather Information	22nd April	25th April	3
Documentation	25th April	1st May	7
Make rest of UI in Photoshop	1st May	11th May	10
Convert PSD to XML	11th May	21st May	10
Modify App Store Dashboard	21st May	25th May	2
Implement App Store design	25th May	27th May	5
Implement Install screen design	27th May	29th May	5
Development of Server	29th May	5th June	10
Writing POJOS for the Apps	5th June	8th June	4
Implement Google Authenticate in Toolkit	8th June	11h June	1
Implement Upload to store in Toolkit	11th June	21st June	3

Midterm Evaluations	**21st June**	**28th June**	**7**
Writing TODOS for the left part of the app	21st June	22nd June	1
Testing and Submitting the app for the midterm Evaluations.	22nd June	28th June	6
Fetch Uploaded apps in store	28th June	30th June	2
Modify quiz and Flash cards	30th June	2nd July	3
Uploading Scores	2nd July	4th July	2
Analyze (Graphs)	4th July	12th July	8
Achievements	12th July	17th July	5
Implement Deep links	17th July	19th July	2
Send Invitations and share	19th July	20th July	1
Implement Dictation Template	20th July	25th July	5
Implement Video Template	25th July	30th July	5
Implement Match Template	30th July	4th August	5
Implement Comprehension Template	4th August	9th August	5
Completing any left part	9th August	13th August	4
Implement Expresso Testing	13th August	14th August	1
Thorough Testing of the App	14th August	18th August	4
Completion of the documentation part.	18th August	20th August	2

Submission of the Final application	20th August

Note – I am hoping BuildMLearn organization will select one proposal for the Comprehension, Dictation, Video and Match templates.

If any problem arises I will re-schedule timeline of mine to match the other developer's timeline.

What to expect at the end

- Great increase in the number of students using the App.
- Better retention of the users.
- Students will be more benefited analyzing their performance and week areas.
- And last but not the least a collective set of study tools deployed in the play store.

About me

I am **Kunal Gupta**, an engineering undergraduate pursuing Information Technology at Manipal Institute of Technology.

I've been developing for android past 2 years, have made contributions to the BuildMLearn Toolkit as well.

Here are some of my applications-

- **Crowdspell**- A game built for Android/Windows Phone
- **EntryScreenManager** - Built a library for importing Intro/Entry screens posted at materialup.com as well!
- **TextBuilder**- Built a Standalone App/Plugin for IntelliJ(Android studio) for generating automated text, Details.
- **IONAutoLogin**- Android app to automatically login to my college Wi-Fi.

Platforms/Languages I have knowledge of - Android, Windows Phone, Web (HTML/CSS), Server Dev (Jersey), C#, JAVA.

I have also worked as a freelancer for Android Projects, also worked as a Web-Developer in a Website Dev company!

Contact Information

Email – kunall.gupta17@gmail.com

Phone No. - +91 9795197787

Some Quick Questions

Q. Why will I be the best suited for this project?

A. Firstly, I am an open source lover and love to contribute for the project. I do not have any other projects to work in the coming summer. I do have experience with android and professional development as well, I have done some freelancing projects.

Q. How much time will I be able to give per week?

A. I am into writing code past 5-6 years. So I have the stamina to work at least 10 hours a day. So I can give about 40+ hours per week.

Q. Am I comfortable working independently under a supervisor or mentor who is just available virtually?

A. Yes, I am absolutely comfortable with both the ways. As I have worked with bosses over the internet (As freelancer).

Q. Have I submitted proposals for any other organizations except BuildMLearn? If yes, which orgs have I applied for?

A. Yes, I have submitted a proposal for Zulip organization as well.

Q. Have I participated in any previous GSoC programs? If yes, tell BuildMLearn about the experience briefly and if I completed successfully.
A. No, This is my first time applying for GSOC.

Q. What is my primary motivation to submit a proposal for this project? Am I interested because I am getting a stipend as a part of GSoC or would I be interested to develop this project for BuildMLearn even outside GSoC?
A. Honestly, I am working for GSOC for the stipend and for the Certification from Google. Plus I'll be learning so many new things.

Yeah, I would definitely like to contribute if I don't have any current projects of mine going on as I like the idea of the application and have an optimism that this application will be very popular for learning once it has been fully published and advertised.