VIRTUAL LABORATORIES

REVIEW REPORT

Submitted by

Gondala Satvarsh - 18BCE2098

Prepared For

SOFTWARE ENGINEERING (CSE3001) – PROJECT COMPONENT

Submitted To

Dr. Ramanathan L

Assistant Professor

School of Computer Science and Engineering

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Table of Contents

1.0 Introduction

Laboratories have always provided students with proper tools to encourage studying by providing a proper motivation and increasing the retention rate in comparison to other methods as hands-on learning proved time and time again as the ultimate method of learning. Not every institution has the facility to support laboratories be it lack of resources, time or quality. Virtual Laboratories aim to put a stop to these problems by not only enhancing the already existing laboratories, but also providing laboratories of its own by acting as a digital platform that brings forth the efforts of multiple independent parties like students and teachers into one common platform.

Our software is called “VIT LABO”; it is an open-source software that has tools which can be integrated into existing laboratories or can be used to establish newly functional laboratories. Things like independent student performance tracking, anti-cheat systems provide teachers with accurate and genuine efforts that can be used to assess a troubling student. New virtual laboratories like Logical Circuits lab or Engine labs can be added to the existing platform independent of the university. Things can be made public or restricted to a particular university. The main aim of this project is that enthusiastic students and teachers can continue to modify and add things, enhancing theirs and others' experience at the same time. Our project thrives on open collaboration and customer feedback.

The following project concerns the Android version of the software and this can be ported to any other platform given you have a proper license and source. This is an open-source platform.

2.0 Software Requirement Specification

2.1 About the project

VIT LABO is a platform that enables students that are currently studying and curious learns alike to access lessons that are prepared by a community of students/admins/teachers. The main goal of this is to make LABs accessible to students without labs, whilst making current LABs more effective.

This platform can create, modify, delete lessons and modules; Conduct and monitor examinations and quizzes with an anti-cheat system, work alongside university to add authenticity. Data is accessible via cloud and local repository.

2.2 Project Scope

- Establish proper security features
- Define boundaries for all modules/ features
- Gather data from students, teachers, users
- Establish Database requirements for Repository, Cloud
- Establish requirements for users

2.3 Software Requirements

- Established connection to Local/ Cloud
- Login Interface
- Module Interface
- Error Log Interface
- Error Report Interface
- Feedback Interface
- Biometric Authorization
- Facial Recognition
- Admin election system
- Public view
- Private view
- Auto log generator

2.4 Levels of operators/users

The users can be classified into the following Admin, Teacher, Student, Guests.

	Admins	Teachers	Students	Guests
Login required	YES	YES	YES	NO
View Modules	YES	YES	YES	YES
Add Modules	YES	YES	Need Permission	NO
Change Modules	YES	Need Permission	Need Permission	NO
Write Exams	YES	YES	Need Permission	NO
Access Logs	YES	NO	NO	NO
Delete modules	YES	Need Permission	Need Permission	NO
Visit university Portal	YES	YES	YES	NO
Can become admin	NA	Need Permission	Very Rarely	NO
Can access private view	Depends on university	Depends on university	Depends on university	NO

Order of priority is given as:

Admin> Teacher> Student>Guest

Sometimes teachers can act as an admin given the permission. Students may not act as admins, but can be given permission to allow certain functions

2.5 Environment

- Operating System: Android
- Cloud handler: Andromo
- Platform: Any android device with version >= 8.0
- Open-Source Software
- Local/Cloud Storage

2.6 Functional Requirements

- System should request admin for credentials for new users
- System should redirect new users to their respective university
- System should allow existing users to login
- System should recognize the role of the said user (admin, guest)
- System should allow creation, deletion, modification of Modules/Lessons
- System should be connected to local repository at all times
- System should store data in Cloud
- System should generate Logs
- System should allow conduction, attendance of exams
- System should conduct anti-cheat measures as per admin
- System should obey admin
- System should restrict functions accordingly to different roles
- System should take errors, feedback into account

Admins should be granted the following functions:

- Set clear intervals,methods for anti-cheat system
- Delete, modify lessons
- Access Logs
- Share Logs
- Grant, Give permissions
- View all data
- Delete, modify Modules
- Set constraints for exam like time, questions
- Temporarily/ permanently make teachers admin
- Remove a student/Teacher
- Report a student/Teacher
- Set things to private/public

2.7.1 Hardware Specifications

For Mobile Applications

- RAM: 1 GB (Minimum)
- Processor: Any multi core/ Thread Processor
- Hard Disk: 50-100 MB minimum (App + Local Repository)

For Web enabled platforms

- RAM: 1 GB (Minimum)
- Processor: Any processor that supports internet browsing
- Hard Disk: 70 MB minimum (Cache, Cookies)

2.7.2 Software Specifications

For Mobile Applications

- Operating System: Android version 8.0 or above
- Server: Andromo
- Web Technologies: XML, Kotlin, Spring, Spring Boot, Java, Android Studio
- Database: SQL Plus

For Web enabled platforms

- Operating System: Windows, Mac, Linux any browser enabled
- Web Technologies: HTML 5, ArgoUML, Node.js, PHP Laravel
- Database: MySQL
- Server: Apache, LAMP Stack
- Technologies: Web browser, Notepad / Notepad ++, GCC Compiler / Java IDE

2.8 Non-Functional Requirements

1) System should allow concurrent users to access, use its database

2) System should allow concurrent parallel actions like login

3) System should always backup local repositories in Cloud

4) System should support module based system
   - New modules can be added
   - Concurrent modules can run
   - Old modules can be deleted
   - Old modules can be modified

5) System should have a minimalistic UI and must be easy to navigate

6) System should extrapolate user data for optimization purposes

7) System should readily allow integration of new modules

8) Higher level encryption can be provided to cloud data

9) Accessibility options are provided for users suffering from color blindedness, photosynthetic seizures

10) Icons should be bold, visible, simplistic and should stand out

2.8.5 Modules

System has following modules

-

Modules of the System are explained in Implementation along with its functions.

3.0 Software Design Specification

3.1 Introduction

Design covers two main components- Front end, Backend. Both the frontend and backend is powered by Android Studio although with slight modifications and integrations.

The backend is integrated with a cloud platform to extrapolate data from all operating android devices and gather data from a specific university instead of hard coding each and every detail. This is done by utilizing a feature that is already available in default IDE, but could also be done via third party software. Here we used “Andromo” platform to construct the ideal cloud platform.

On the frontend we used Android Studio as the visual forefront for previewing, creating and testing the app by creating a virtual device. The coding is implemented via XML and Java.

3.1.1 Dependencies

The implemented application works on any android device with version 8.0 (Oreo) or above. The working and testing of the app depends on the accessibility of the aforementioned device by users.

Possible changes in functionality based on feedback in the future:

- Creating a web based application using spring or spring boot framework.
- Better system to establish admins preferably using a voting system of sorts.
- Option to vote kick an admin.
- Configuring interval times for biometric verification
- Choosing the type of biometric mainly facial
- Asking a custom question as per admin on specific times
- Accompany for further increase in application scope
- Data mining to observe and optimize users actions
- Voting system within the feedback system

All these changes can be done in the app provided there is permission to do so.

3.1.2 General Constraints

1. This heavily limits the percentage of available users as only 12.9% of android devices are of 8.0 or above versions.

2. Application currently only supports a single login from a specific device

3. Only admins can delete, change or approve created modules

4. Each module requires a source, which needs to be active at all times

3.1.3 Goals and Guidelines

This project strives on solving the following goals:

- Intuitive
- User Friendly
- Minimal UI
- Secure
- Open-source license
- Simple to use
- Accessibility

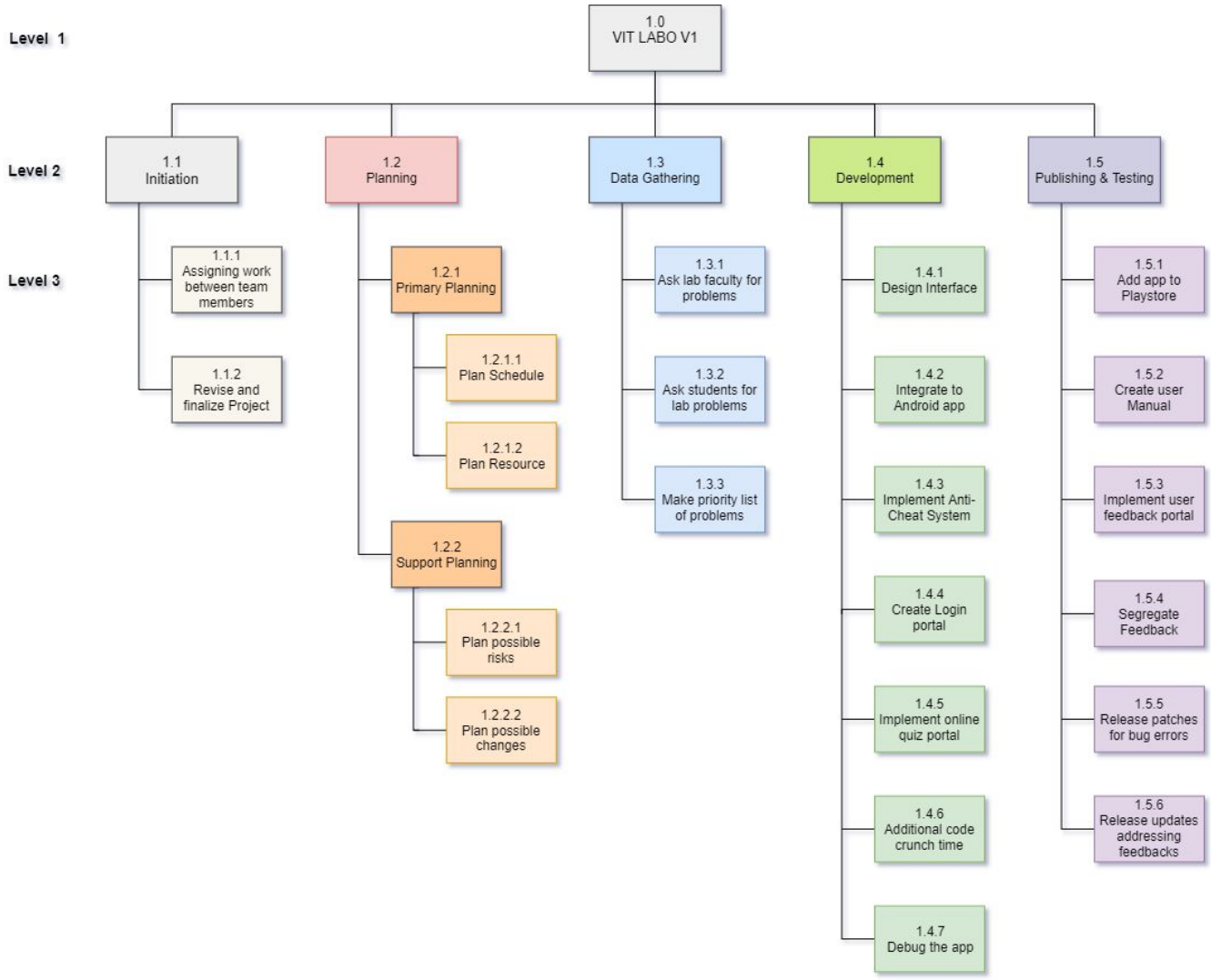
3.1.4 Exercised Strategies

- Easy Indexing
- Cloud Storage
- Minimalistic UI
- Expandable skeletal structure
- Module based integration (similar to Visual Code)
- Extrapolate user data

3.2 System Architecture

This project thrives on the integration of modules that can be accessed and modified by users that have the privileges.

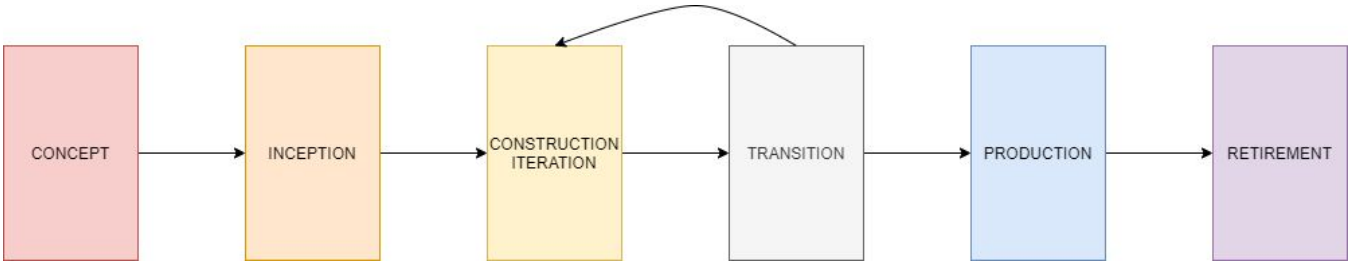
1) Work Breakdown Structure



1.3,1.4,1.5 are heavily stringent on feedback and licensing.

with the priority order being : 1.5>1.3>1.4 (Feedback)
1.4>1.5>1.3(QOL improvements)

2) Software Model Used: Agile Model



Agile is the most suitable as our project is heavily tailored by customer feedback. There is no monetary gain so the only measurable gain is the number of users. This is a project with many ideas so it's only natural for new things to be added or deleted and this inflexibility is accounted for in the Agile model.

Target deliverables for each phase:

a) Concept: Selection of ideas and features
   - Formulate general features, idea
   - Formulate a priority list/ main features
   - Formulate financial, marketing model for project.

b) Inception: Initiating the project
   - Build team
   - Gather initial requirements
   - Envision initial architecture, design
   - Stakeholder gathering

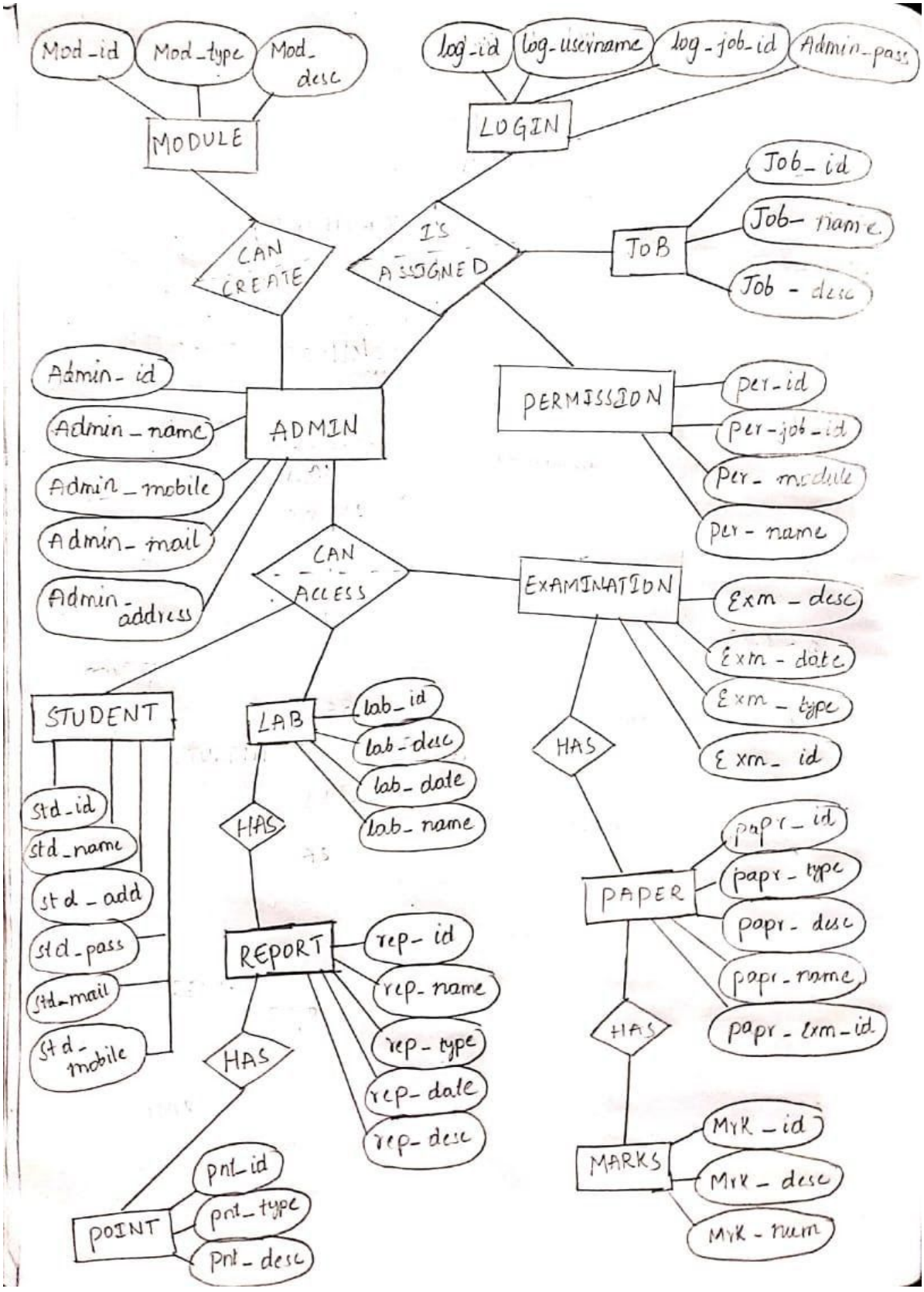
c) Construction Iteration: Working model creation that forms as a base
   - Evolve documentation
   - Internally deploy software
   - Perform internal testing
   - Stakeholder participation

d) Transition: Nth release (End Release)
   - Final system testing
   - Final acceptance testing
   - Train end users
   - Deploy product to all staff/ Stakeholders
   - Active Stakeholder participation

e) Production: Operate and support end release
   - Operate & Support the system
   - Identify problems and enhancements

f) Retirement: Stop the production system
   - Migrate users
   - Provide future models/ upgrades

3.3 ER Diagram



3.3.1 Tables and Constraints (Database)

- TABLE NAME - ADMIN

Attribute	Domain type	Constraint
admin_id	Number(10)	Primary key
admin_name	Varchar(40)	Not null
admin_mobile	Number(10)	Not null
admin_mail	Varchar(50)	Not null
admin_address	Varchar(150)	Not null

- TABLE NAME- MODULE

Attribute	Domain type	Constraint
Module_id	Number(10)	Primary key
Module_type	Varchar(10)	Not null
Module_desc	Varchar(150)	Not null

- TABLE NAME-LOGIN

Attribute	Domain type	Constraint
Login_id	Number(10)	Primary key
Login_username	Varchar(30)	Not null
Login_job_id	Number(10)	Not null
Admin_password	Varchar(30)	Not null

- TABLE NAME-JOB

Attribute	Domain type	Constraint
Job_id	Number(10)	Primary key
Job_name	Varchar(40)	Not null
Job_desc	Varchar(150)	Not null

- TABLE NAME-PERMISSION

Attribute	Domain type	Constraint
Per_ id	Number(10)	Primary key
Per_ job_id	Number(10)	Candidate Key
Per_ module	Varchar(20)	Not null
Per_ name	Varchar(40)	Not null

- TABLE NAME-STUDENT

Attribute	Domain type	Constraint
stu_id	Number(10)	Primary key
stu_add	Varchar(150)	Not null
stu_name	Varchar(40)	Not null
stu_mobile	Number(10)	Not null
stu_email	Varchar(50)	Not null
stu_pass	Varchar(30)	Not null

- TABLE NAME-LAB

Attribute	Domain type	Constraint
Lab_id	Number(10)	Primary key
Lab_name	Varchar(40)	Not null
Lab_desc	Varchar(150)	Not null
Lab_date	Date	Not null

- TABLE NAME-EXAMINATION

Attribute	Domain type	Constraint
Exm_id	Number(10)	Primary key
Exm_desc	Varchar(150)	Not null
Exm_date	Date	Not null
Exm_type	Varchar(10)	Not null

- TABLE NAME-PAPER

Attribute	Domain type	Constraint
Papr_id	Number(10)	Primary key
Papr_exm_id	Number(10)	Candidate Key
Papr_name	Varchar(40)	Not null
Papr_type	Varchar(10)	Not null
Papr_desc	Varchar(150)	Not null

- TABLE NAME-REPORT

Attribute	Domain type	Constraint
Rep_id	Number(10)	Primary key
Rep_desc	Varchar(150)	Not null
Rep_type	Varchar(10)	Not null
Rep_date	Date	Not null
Rep_name	Varchar(40)	Not null

- TABLE NAME-POINT

Attribute	Domain type	Constraint
Pnt_id	Number(10)	Primary key
Pnt_type	Varchar(40)	Not nulll
Pnt_desc	Varchar(150)	Not null

- TABLE NAME-MARKS

Attribute	Domain type	Constraint
Mrk_id	Number(10)	Primary key
Mrk_desc	Varchar(150)	Not null
Mrk_num	Number(10)	Candidate Key

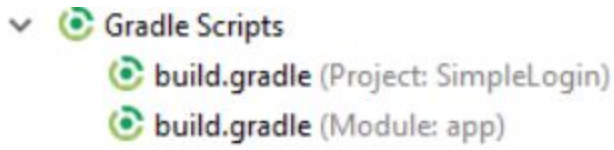
3.4 Queries Implementation/Source Code

Credentials - Repository (Tray)(Gradle Dependencies)

```
apply plugin: 'com.android.VITLABO'

android {

   compileSdkVersion 25

   defaultConfig {

      applicationId "com.VITLABO.simplelogin"

      minSdkVersion 23

      targetSdkVersion 25

      versionCode 1

      versionName "1.1"

      testInstrumentationRunner "android.support.runner.AndroidJUnitRunner"

   }

   buildTypes {

      release {

         minifyEnabled false

         proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'

  } }}
```

CORE REPOSITORY (UNDER Dependencies)

```
androidTestImplementation 'com.android.VITLABO.test.espresso:espresso-core:3.0.1'
```



Feedback Repository (Tray)

1. Intent email = new Intent(Intent.ACTION_SEND);
2. email.putExtra(Intent.EXTRA_EMAIL, new String[]{ to:'VITLABO@Gmail.com'});
3. email.putExtra(Intent.EXTRA_SUBJECT, subject_Feedback_VITLABO);
4. email.putExtra(Intent.EXTRA_TEXT, message_VITLABO);

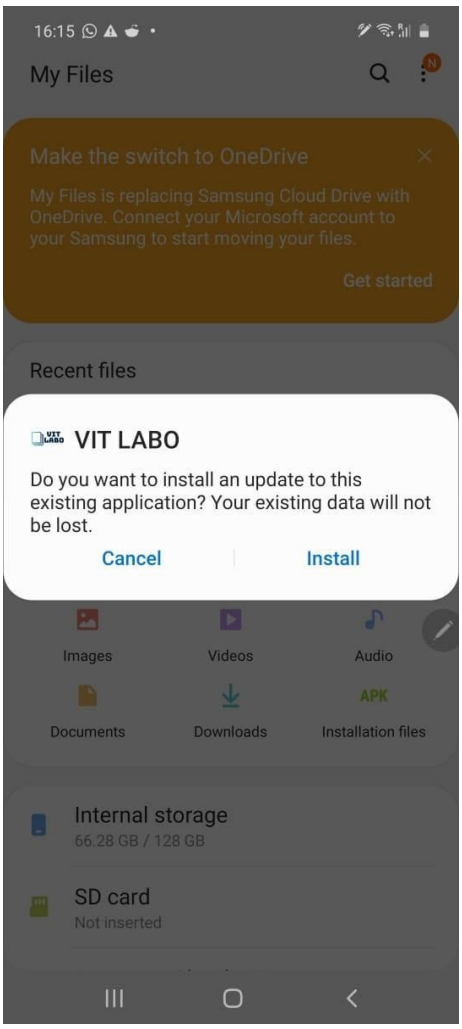
Locational Repository(Portal Identification/Assignment)

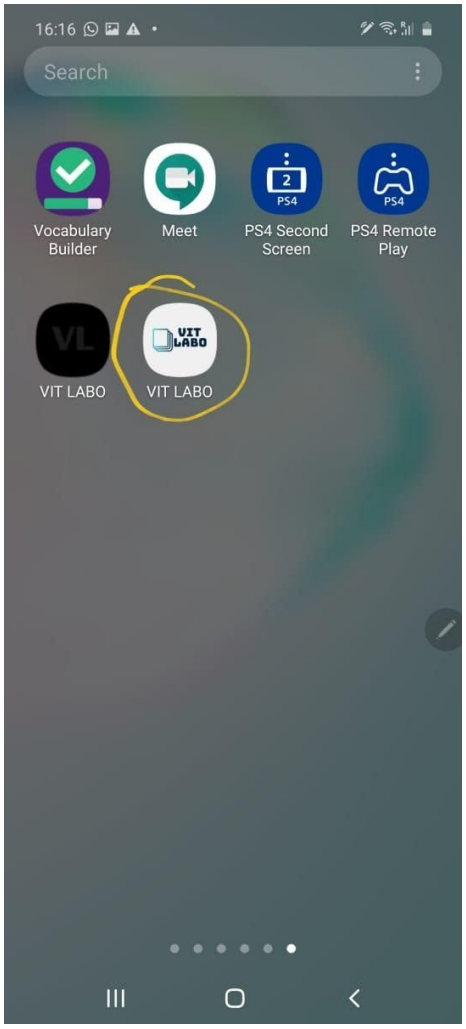
```
package com.journaldev.gpslocationtracking;

import android.annotation.TargetApi;

import android.content.DialogInterface;

import android.content.pm.PackageManager;

import android.os.Build;

import android.support.v7.app.AlertDialog;

import android.support.v7.app.AppCompatActivity;

 @Override

   protected void onCreate(Bundle savedInstanceState) {

       super.onCreate(savedInstanceState);

       setContentView(R.layout.activity_main);

   permissions.add(ACCESS_FINE_LOCATION);

    permissions.add(ACCESS_COARSE_LOCATION);

  permissionsToRequest = findUnAskedPermissions(permissions)

   if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {

         if (permissionsToRequest.size() > 0)

            requestPermissions(permissionsToRequest.toArray(new
String[permissionsToRequest.size()]), ALL_PERMISSIONS_RESULT);   }
```

```
Button btn = (Button) findViewById(R.id.btn);

 btn.setOnClickListener(new View.OnClickListener() {

 @Override

          public void onClick(View view) {

             locationTrack = new LocationTrack(MainActivity.this);

             if (locationTrack.canGetLocation()) {

                double longitude = locationTrack.getLongitude();

                double latitude = locationTrack.getLatitude();

                Toast.makeText(getApplicationContext(), "Longitude:" + Double.toString(longitude) +
"\nLatitude:" + Double.toString(latitude), Toast.LENGTH_SHORT).show();

             } else {

                locationTrack.showSettingsAlert();

             } } });  }


 @TargetApi(Build.VERSION_CODES.M)

   @Override

    public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults) {

       switch (requestCode) {

          case ALL_PERMISSIONS_RESULT:

             for (String perms : permissionsToRequest) {

                if (!hasPermission(perms)) {

                   permissionsRejected.add(perms);

                } }
```

4.0 Implementation (Captured in NOTE 10 Lite)(Not Virtual Device)

4.1 Package Installation (APK)



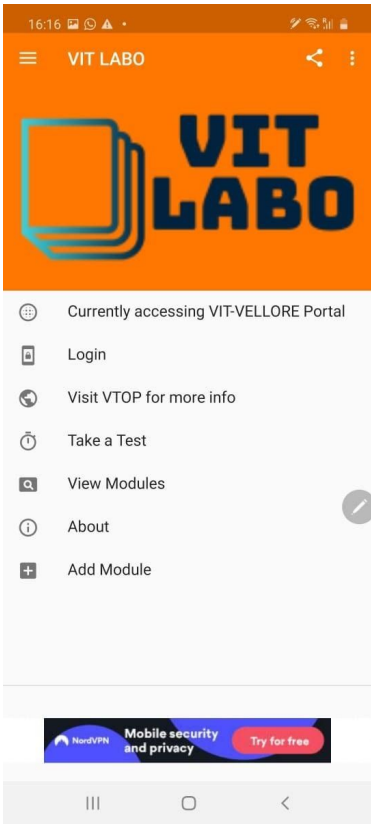


Following Things are to be noted:

- Running on android version 10
- App only available on local storage and cloud
- App not available on Play Store
- App size is 6 MB, but repository is located in Cloud
- Cloud integration is implemented via Anromo platform

Video demo/proof for the app:
https://drive.google.com/open?id=1zhe-d3nJOsYo1rA4lDNOULh2Rq3eZx6H
(Audio is due to background noise please IGNORE it, recorded in phone via Xrecorder)

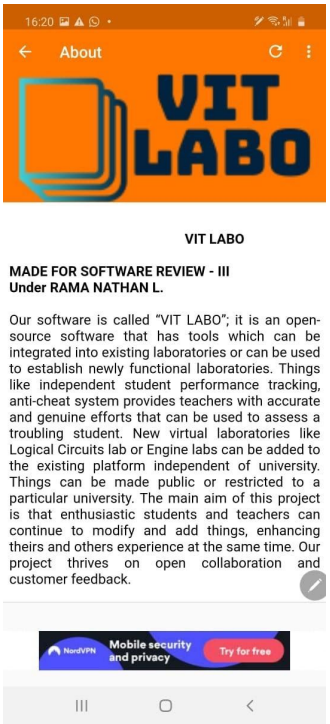
4.1.1 Home Page



This is the homepage of the app, each menu is further explained in detail in the upcoming pages.

Index:

- Top Bar always shows your university portal
- Login……....4.2
- Visit VTOP for more info……4.2
- Take a test……4.5,4.5.1
- View Modules……4.3,4.3.1
- About……4.1.1
- Add Module……4.4

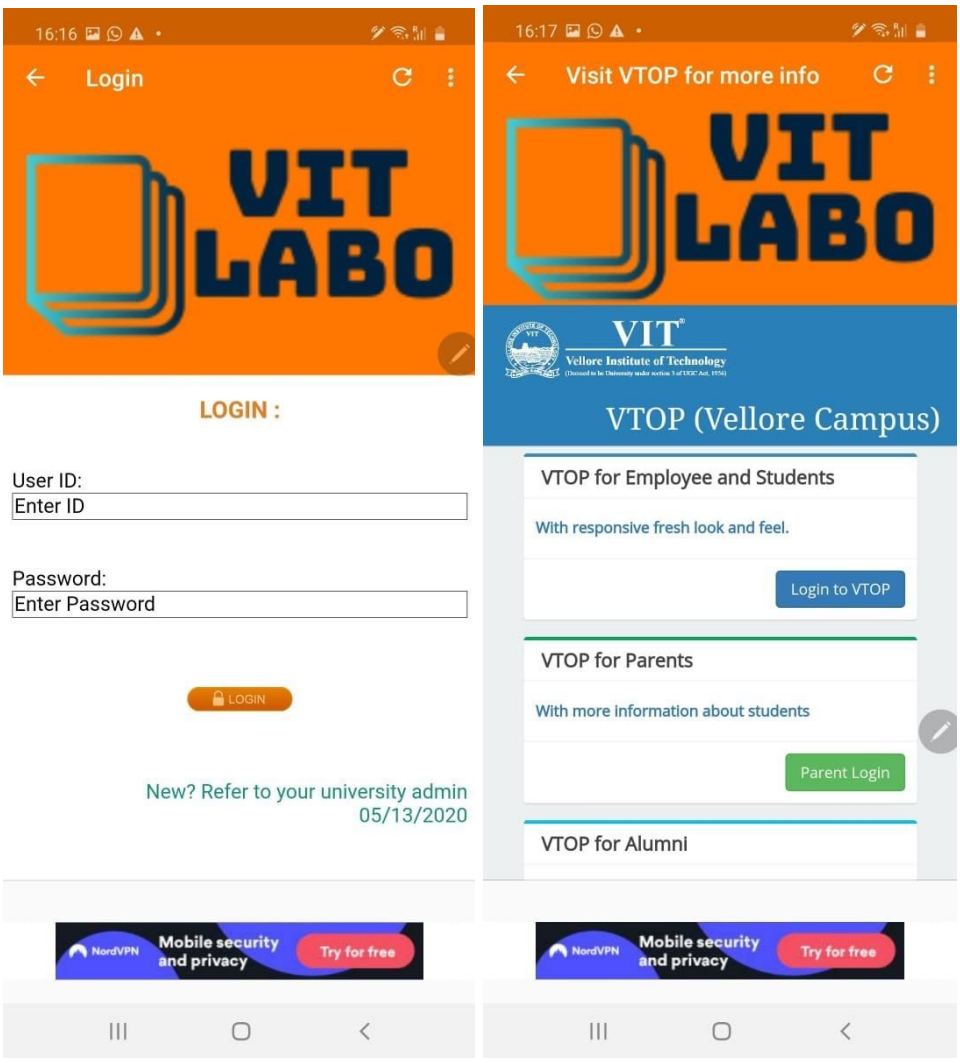


About Screen is just information that people who access your app will see. It could be changed according to:

- University
- Admin
- App Creator

Currently this only displays about the current software final review

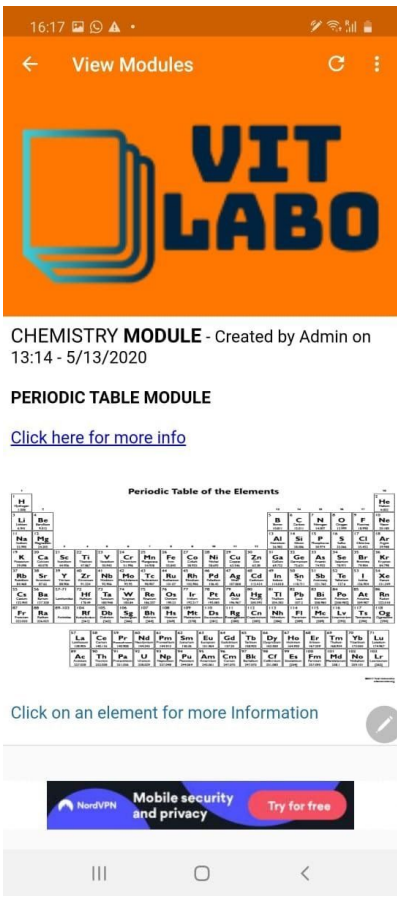
4.2 Login Methods

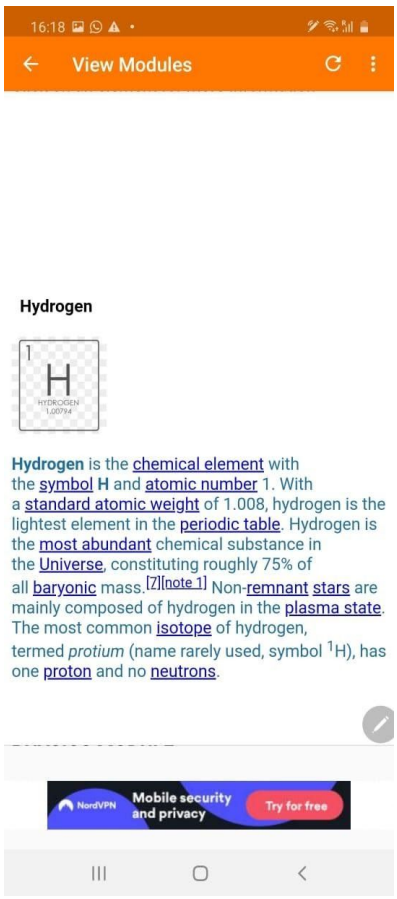


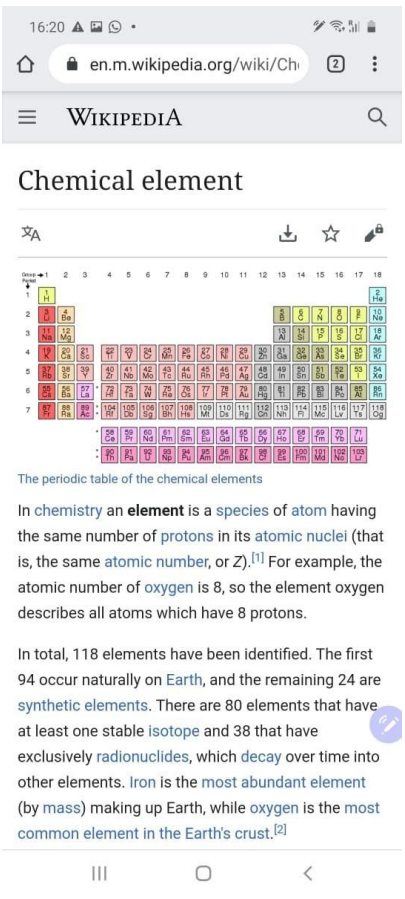
Login can be done via: Locally or VTOP

- Here VTOP can be any domain that belongs to your university
- Local login requires the credentials to be stored in cloud
- Admin privileges can ONLY be granted locally
- Admins are elected WITHIN the app
- New users can not sign up WITHIN the app
- Guest privileges are granted by default if user is not logged in
- Students MUST take admin permission to use sign in for the first time
- Guests can not write exams, make/modify modules
- Guests have unrestricted access to VIEW modules

4.3 Modules Showcase (Lessons are same as modules)



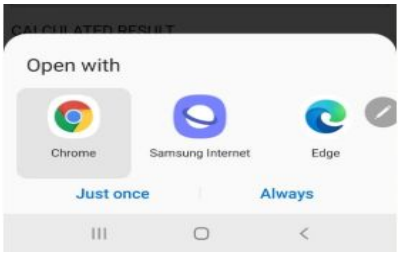




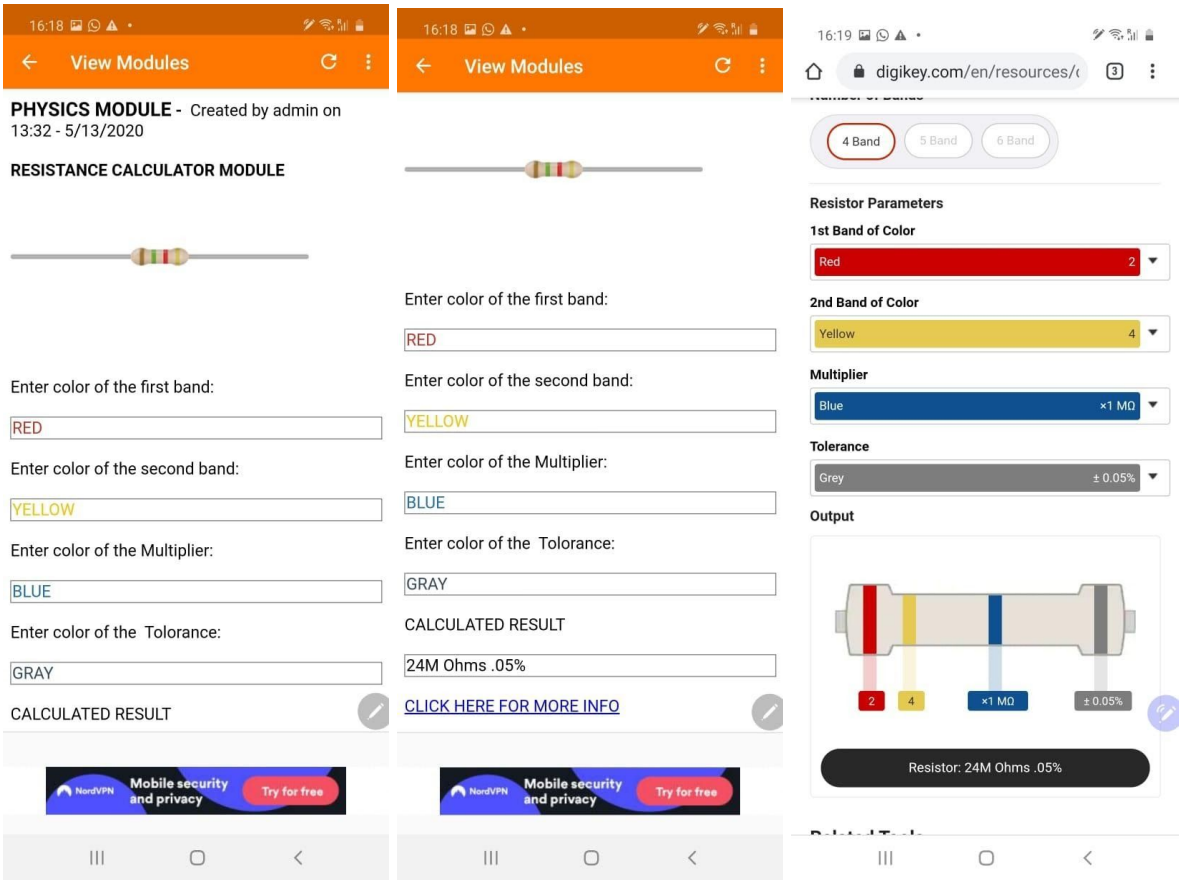
The following example is periodic table module, where you can click on an element in periodic table to get more information on it WITHIN the app.

a) Each module MUST require a source for it to function here the source is linked which takes us to the THIRD IMAGE.
b) Each of the purple links in SECOND IMAGE takes us to the wikipedia page of the respective keyword (You can directly import hyperlinks from wikipedia into this app)

You can do both a,b within the app or on external window



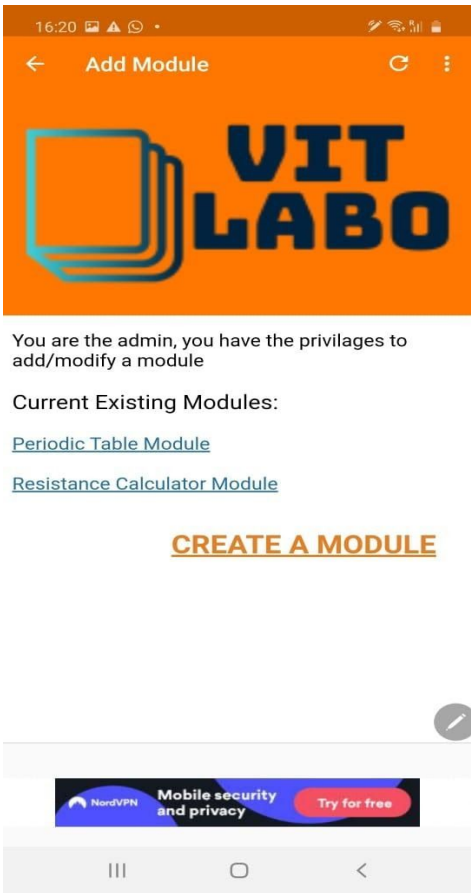
4.3.1 Other example modules



The following is the Resistance Calculator Module, where you can enter the colors for each band in the resistor and it calculates the final resistance of the band.

- FIRST IMAGE states the time and user of the created module, you can INPUT data locally without relying on external windows.
- SECOND IMAGE gives the result from source or local repository, blue link is for source
- THIRD IMAGE is the source in this example both the source and local repository shows the SAME value meaning that this app is RELIABLE and doesn't deviate from the source code.
- Depending on the situation you might need permission from the source. You can NOT link paid software like matlab as source, ONLY open sources are eligible

4.4 Module Creation



This can be found in the "Add Module" option.

- In this example we are using this as an admin.
- Admin always can post/create/delete modules without permission
- Students can create module, but need admin approval
- Faculty has the same privileges as the student when it comes to modules
- Guests can only view modules
- You can also view list of existing modules
- Blue links redirect you to existing modules
- Orange link lets you create a module

You need the following to create a module:

- Permission*
- Title*
- Content/Text*
- A source of Input
- A source of Output
- Source Code

Things with * are mandatory.

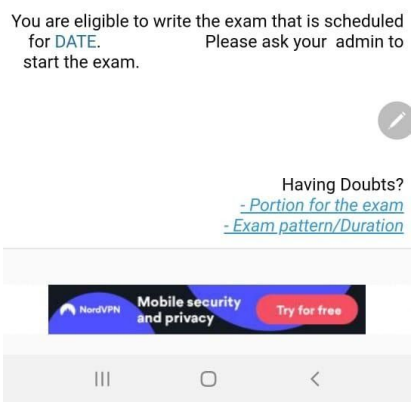
- Source code is a must if there is an output as source code stores the output.
- Source code is not necessary for input if there is no output (Periodic Table, 4.3)
- Source code is must if there is input with outputs (Resistance Calculator, 4.3.1)
- Source code has to be of open license

4.5 Examinations/Quizzes

When your biometric is AUTHORIZED, you are eligible to write the exam. Biometric could be verified in 2 ways.

1) From your records in VTOP (Hard, Cloud data)
2) From your phone lock screen (Simple easy)

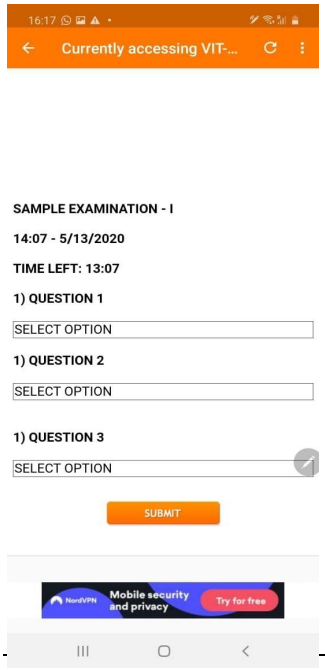
Second method is easy compared to the first method ,but it is also exploitable. People can add their friends biometric on their phone. The First one is simple, but is not practical as there could be mapping issues.



The current implemented method is the second method, which is simple and easy. Admin can also request users/ students for biometric at specific intervals during examination like, every 15 min or so.

Admins, Students can write the examination.
Errors generated are sent to log(More detail in 4.5.1)

Students can also check the exam portion, model by clicking the blue links.



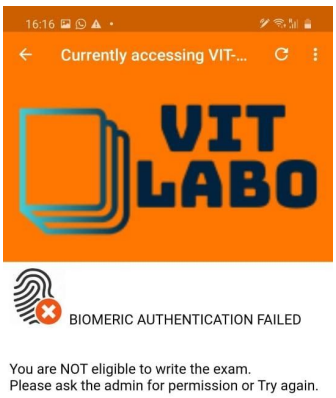
This is an SAMPLE EXAMINATION. Admin has control over:

- Duration
- Questions
- Answers
- Grading for questions
- Managing students permissions
- Give the exam

Attenders of the exam will have access to:
i.e Students, Admins

- Time Left
- List of questions/Answers
- Submitting a complaint
- Submitting the exam

4.5.1 Error Logs/ Anti cheat



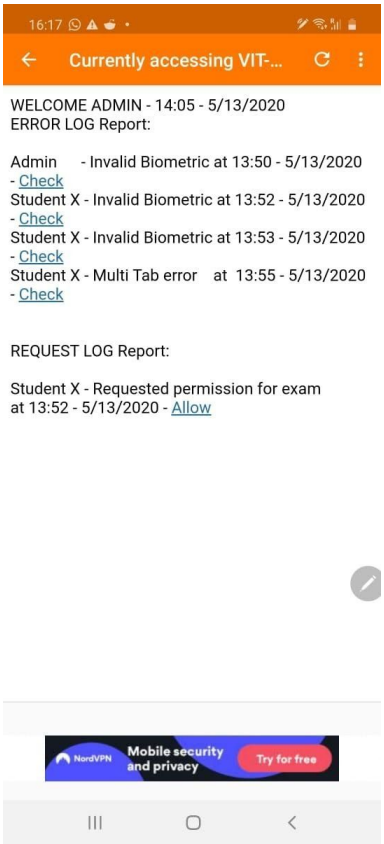
If a student fails to give proper biometric during/before examination. Admin can set the following rules:

- Students will be redirected out of the exam
- Students will stay in exam without any punishment

Regardless of what method the admin chooses an ERROR is always generated in the LOG(s). Admin/Teachers can:

- View the errors/logs
- Check the error in detail
- Grant student permission if they're redirected out of exam
- Send/share errors to others

Errors are NOT the only things generated in logs more can be seen below:



LOGs are generated in the following manner:

Errors: Invalid Biometric, Multitab error, exiting

Admins/ Teachers can click the blue link to check the error in more detail

Requests: When students request permission of admins for things like

- Approving a module
- Unable to write the exam
- Caught cheating in the exam

Admin/ Teacher can click on the allow link to allow/give permission to the student like, allowing student to write an exam

All LOGs have name, role, date,time,error type, available

actions.

5.0 Referrals

ADDITIONAL FILES FOR VITLABO

All High Resolution Screenshots, Assets, Logos used for VIT LABO:
https://drive.google.com/open?id=1ZnPLVl_cZkQurarwpn5s3mVWwAE3EnQB

Video Demo of VIT LABO:
https://drive.google.com/open?id=1zhe-d3nJOsYo1rA4lDNOULh2Rq3eZx6H

DEVELOPMENT/BACK END

1) https://www.journaldev.com/13325/android-location-api-tracking-gps
2) https://developer.android.com/studio
3) https://developer.android.com/training/beam-files
4) http://www.presentationeze.com/presentations/software-validation/software-validation-full-details/software-design-specification/
5) https://en.wikipedia.org/wiki/Comparison_of_smartphones
6) https://builder.andromo.com/#/project/737313/activities/
7) https://www.javatpoint.com/how-to-send-email-in-android-using-intent
8) https://developer.android.com/guide/topics/data
9) https://developer.android.com/training/search/search
10) https://www.onlinetutorialspoint.com/android/basic-android-login-form-example.html
11) https://www.tutorialspoint.com/android/android_login_screen.html

FRONT END

1) https://www.freelogoservices.com/step1
2) https://editor.freelogodesign.org/?lang=en&logo=7e895572-60f1-4bcd-87b0-6e403d37b553
3) https://abhiandroid.com/ui/button
4) https://www.tutorialspoint.com/android/android_acitivities.htm
5) https://www.freepik.com/free-vectors/buttons
6) https://www.vecteezy.com/free-vector/app-button

MODULES

1) https://www.ptable.com/
2) https://www.digikey.com/en/resources/conversion-calculators/conversion-calculator-resistor-color-code-4-band