

Table of Contents

Database Design:	1
Navigation Design:	2
Feature Name: Covid-19 Information.....	3
Feature Name: Important Notes.....	2
Feature Name: Task to complete	4
Feature name: Check-in	5
Feature name: Covid Hotspot	8
Feature name: Report Rule-Breaking Case.....	14
Feature name: Sign up	17
Feature name: Login	19
Feature name: FAQ.....	21
Feature name: Dashboard	23
Feature name: Change Password	25
Test cases and test results	28

Database Design:

There are six tables implemented as show in the picture below. In this project, local room database is used. MVVM model is mainly implemented also.

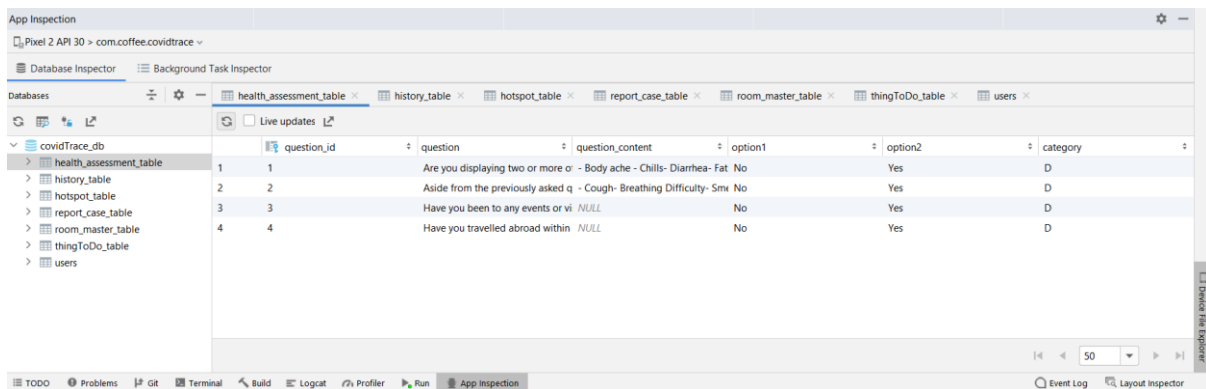
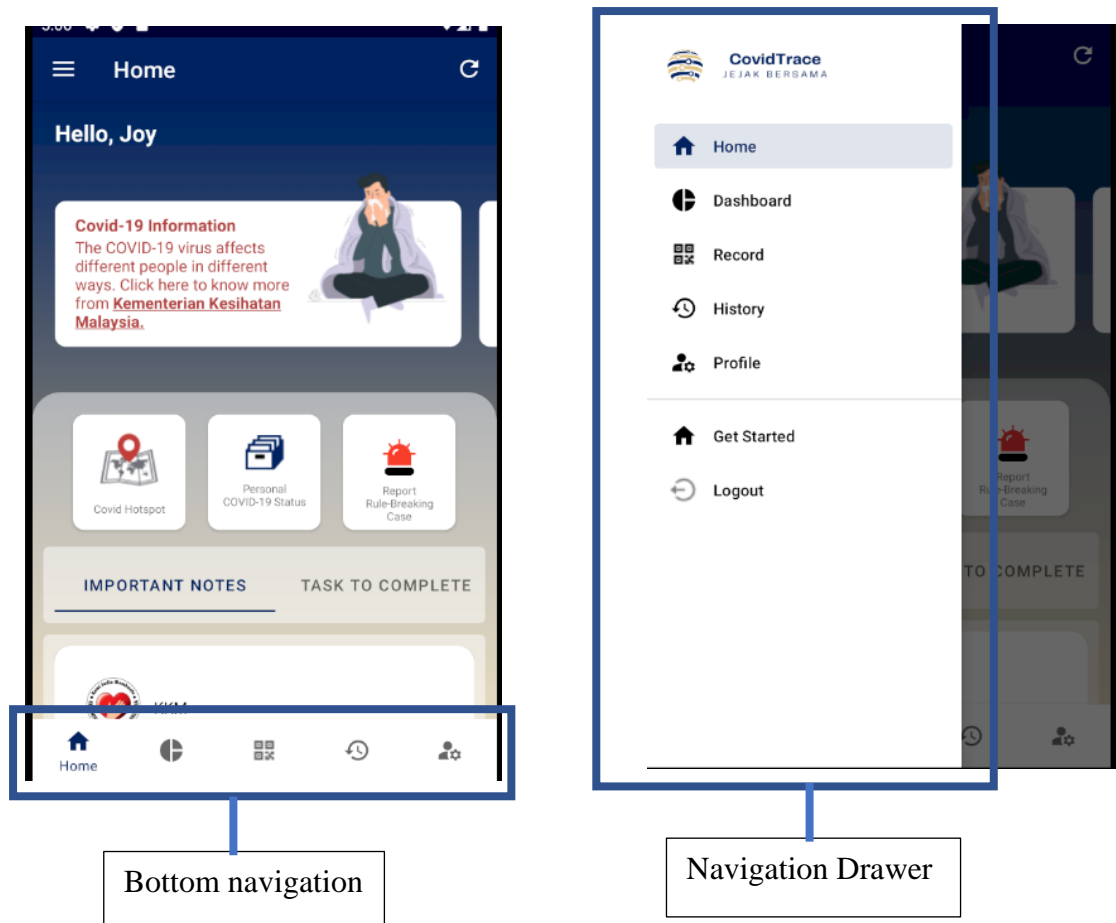


Table	Function
health_assessment_table	Store the questions and the option of the questionnaires used in updating the “Personal COVID-19 Status” .
history_table	Store the check-in history (location, date and time)
hotspot_table	Store the active COVID-19 case numbers of the location (in latitude and longitude)
report_case_table	Store all the reported rule-breaking cases and their status.
thingToDo_table	(Wrongly naming the table) Store the important notes or announcement that the user should know about the COVID-19.
users	Store the information needed for the user including the risk status and the vaccination status of the users.

Navigation Design:

There are two navigation design implemented for the CovidTrace. First, is the bottom navigation and the second navigation is the navigation drawer. The notation of the navigation drawer will change if the bottom navigation change; and also, if the user clicks the “Dashboard” using the navigation drawer built, the bottom navigation will also change automatically to make the logic of the application fluent.



To pass data between the fragments under one activity (MainActivity.java containing the fragment of Home, Dashboard, Record, History, Profile). A shared view model is used to pass the data. The shared view model owner is the MainActivity.java and later on the other fragments share the use of this view model and each fragment has their own view model. The use of the shared view model can be used like a bundle to pass the data.

```

public class SharedViewModel extends ViewModel {

    private final MutableLiveData<UserEntity> current_user = new MutableLiveData<>();
    private final MutableLiveData<String> email = new MutableLiveData<>();
    private final MutableLiveData<String> password = new MutableLiveData<>();

    private UserEntity user;
    private Context context;

    public void setCurrent_user(UserEntity user) { current_user.setValue(user); }

    public LiveData<UserEntity>getCurrent_user(){
        return current_user;
    }

}

```

Figure 1 - SharedViewModel.java

```

//view model
sharedViewModel = new ViewModelProvider( owner: this)
    .get(SharedViewModel.class);

```

Figure 2 - MainActivity.java

```

public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
    super.onViewCreated(view, savedInstanceState);
    mViewModel = new ViewModelProvider( owner: this).get(HomeViewModel.class);
    sharedViewModel = new ViewModelProvider(requireActivity()).get(SharedViewModel.class);
}

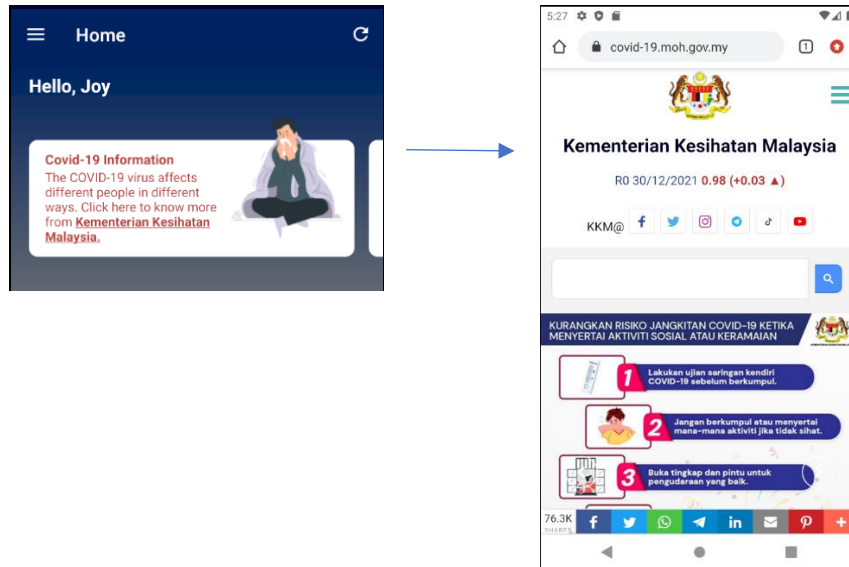
```

Figure 3 - HomeFragment.java

mViewModel is the java of the HomeFragment. sharedViewModel is getting the view model that have been shared.

Feature Name: Covid-19 Information

Function: Implicit intent that leads the user to the official website of the Ministry of Health of Malaysia.



Feature Name: Important Notes

Function: Make the users to know the latest information that they should be informed.

The function is completed by using the thingToDo_table, then an adapter “NotesAdapter.java” is implemented for the recycler view. Once the row of the data is increased, the date and time is automatically recorded for that row.

```
RecordFragment.java x ThingsAdapter.java x ThingsAnnouncement.java x HealthAssessment.java x activity_main_drawer.xml x
29
30 @ColumnInfo(name = "date", defaultValue = "CURRENT_DATE")
31 private String date;
32
33 @ColumnInfo(name = "time", defaultValue = "CURRENT_TIME")
34 private String time;
35
```

The admin can choose to add text announcement or only picture announcement or both.



p/s: the font type here is the font type of the phone used, not the font set in the code.

Then, Glide is used to get the image stored into the database as shown in the figure below.

```
Glide.with(holder.itemView.getContext())
    .load(current.getPic_authorities()) // set the img Url
    .apply(new RequestOptions().transform(new CenterCrop(), new RoundedCorners( roundingRadius: 16)))
    .into(holder.authorities_logo); // destination path
```

Then, ImpNotesViewModel is used to retrieve the live announcement data.

```
mViewModel = new ViewModelProvider( owner: this, ViewModelProvider
    .AndroidViewModelFactory
    .getInstance(getActivity()
        .getApplication()))
    .get(ImpNotesViewModel.class);
```

Figure 4 – ImpNotesFragment.java

```
public class ImpNotesViewModel extends AndroidViewModel {
    // TODO: Implement the ViewModel
    private ThingsAnnouncementRepository repository;
    private LiveData<List<ThingsAnnouncement>> allAccouncement;

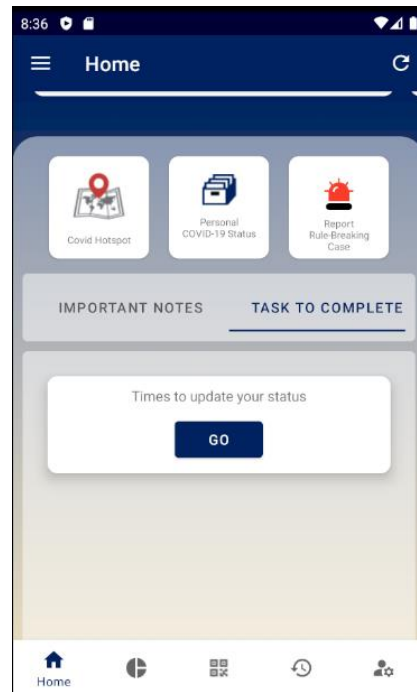
    public ImpNotesViewModel(@NonNull Application application) {
        super(application);
        repository = new ThingsAnnouncementRepository(application);
        allAccouncement = repository.getAllAnnouncement();
    }
    // pass the all announcement list to the activity that invokes it
    public LiveData<List<ThingsAnnouncement>> getAllAnnouncement() { return allAccouncement; }
}
```

Figure 5 – ImpNotesViewModel.java

The announcements are passed in the form of the live data list.

Feature Name: Task to complete

Function: Aimed to remind the user to complete some tasks, for example. complete the profile information or update the user's status by answering the health assessment questionnaire. As the local room database is used in this project, it is static page now.



Feature name: Check-in

Function: As COVID-19 can be spread easily in wide area, then, record of the place user visited is becoming important. This function can scan the QR code to record the location into the history list of the user.

```
lastestHistory.addLast(history);

cv_scan.setOnClickListener(v -> scanCustomScanner(recordFragment));
cv_scan.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

        IntentIntegrator integrator = IntentIntegrator.forSupportFragment(RecordFragment.this);
        //set the properties of the scan
        integrator.setDesiredBarcodeFormats(IntentIntegrator.QR_CODE);
        integrator.setPrompt("Scan");
        integrator.setCameraId(0);
        integrator.setBeepEnabled(true);
        integrator.setBarcodeImageEnabled(false);
        integrator.setOrientationLocked(true);
        integrator.initiateScan();
        scanCustomScanner(view);
    }
});
```

Figure 6 - RecordFragment.java

This allows the button to access the API to allow scanning, however, the result will return a horizontal scanning function, to make the scanning vertical, the following code was required.

```
<activity
    android:name="com.journeyapps.barcodescanner.CaptureActivity"
    android:screenOrientation="fullSensor"
    android:theme="@style/zxing_CaptureTheme"
    tools:replace="screenOrientation" />
```

Figure 7 - AndroidManifest.xml

After scanning, the content of the QR code was stored into the database and using `.now()` to get the scanning time, the time and date are given format desired. The date and time are recorded separately to pass them as intent and also stored into the database.

```

//formatting the date and time
DateTimeFormatter dateFormat = DateTimeFormatter.ofPattern("dd-MM-yyyy");
LocalDate dateNow = LocalDate.now();
String current_date = dateNow.format(dateFormat);

//formatting the date and time
DateTimeFormatter timeFormat = DateTimeFormatter.ofPattern("HH:mm:ss");
LocalTime timeNow = LocalTime.now();
String current_time = timeNow.format(timeFormat);

sharedViewModel = new ViewModelProvider(requireActivity()).get(SharedViewModel.class);

if (sharedViewModel.getCurrent_user().getValue() != null){
    UserEntity userEntity = sharedViewModel.getCurrent_user().getValue();
    Log.d( tag: "shared view model, Record fragment", userEntity.getName());

    int user_id = userEntity.getId();

    History history = new History(scan_location, current_date, user_id, current_time);
    mViewModel.insert(history);

    Intent intent = new Intent(getActivity(), SuccessCheckInActivity.class);
    intent.putExtra( name: "user", userEntity);

    if (history != null){
        intent.putExtra( name: "SCAN_RESULTS", history);
    }
}

```

Figure 8 - RecordFragment.java

The sharedViewModel is used to access the data of the user from the MainActivity.java because RecordFragment.java is one of the fragments of the MainActivity.java; hence shared view model is applied to make the data passing like bundle.

Then, history is stored using *insert* of the HistoryViewModel.java.

After scanning, the history was updated in the HistoryFragment.java and the latest history was retrieved at the RecordFragment.java.

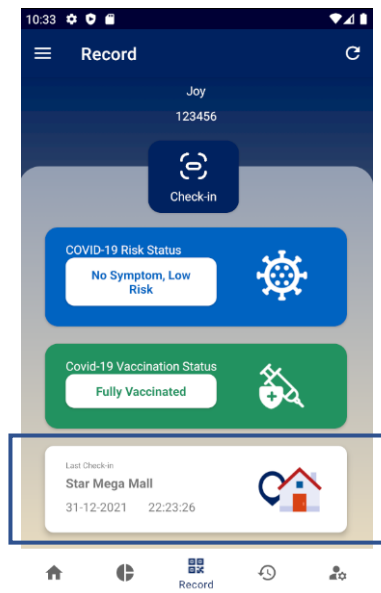


Figure 10 -

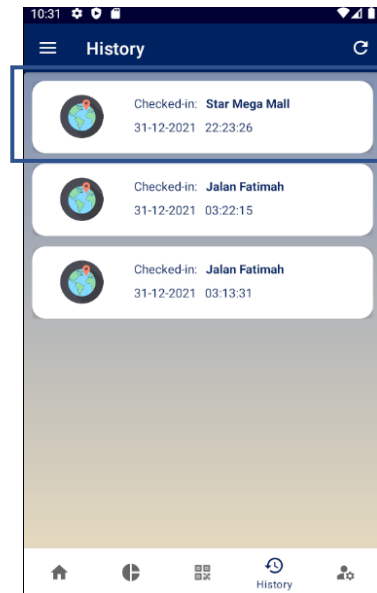


Figure 9 - HistoryFragment.java

To allow multiple users, the user id will be stored also to make retrieve the data based on the user id.

```
@JvmOverloads
@Query("SELECT * FROM history_table " +
        "WHERE history_table.user_id == :id " +
        "ORDER BY id DESC")
LiveData<List<History>> getAllHistory(int id);

@Query("SELECT * FROM history_table " +
        "WHERE id = (SELECT MAX(id) FROM history_table) AND history_table.user_id == :id")
LiveData<List<History>> getLatestHistory(int id);
```

Figure 11 - HistoryDao.java

First query is used to retrieve all the history using the user id and used in HistoryFragment.java. Next query is used to select the latest history using the user id to show the data in RecordFragment.java.

Feature name: Covid Hotspot

Function: This feature was built to let the user have prior knowledge on the reported Covid-19 cases at certain places. Hence, it can assist the user in deciding to go that place or not.

The user can search the place by entering the name of the place or just click a button on the screen to auto focus his or her current location.

```
private void getCurrentLoc() {

    progressBar.setVisibility(View.VISIBLE);
    LocationRequest locationRequest = new LocationRequest();
    locationRequest.setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
    locationRequest.setInterval(UPDATE_INTERVAL);
    locationRequest.setFastestInterval(FATEST_INTERVAL);
    if (ActivityCompat.checkSelfPermission(context: this,
        Manifest.permission.ACCESS_FINE_LOCATION) != PackageManager.PERMISSION_GRANTED &&
        ActivityCompat.checkSelfPermission(context: this, Manifest.permission.ACCESS_COARSE_LOCATION) != PackageManager.PERMISSION_GRANTED) {
        ActivityCompat.requestPermissions(activity: this, new String[]
            { Manifest.permission.ACCESS_FINE_LOCATION}, REQUEST_CODE);
        return;
    }

    LocationServices.getFusedLocationProviderClient(activity: HotspotActivity.this)
        .requestLocationUpdates(locationRequest, new LocationCallback(){
            @Override
            public void onLocationResult(LocationResult locationResult) {
                super.onLocationResult(locationResult);
                LocationServices.getFusedLocationProviderClient(activity: HotspotActivity.this)
                    .removeLocationUpdates(locationCallback: this);
                if (locationResult!=null && locationResult.getLocations().size()>0){
                    int latestLocationIndex = locationResult.getLocations().size()-1;

                    //get current latitude and longitude of the user
                    double latitude = locationResult.getLocations().get(latestLocationIndex).getLatitude();
                    double longitude = locationResult.getLocations().get(latestLocationIndex).getLongitude();

                    Location location = new Location(provider: "providerNA");
                    location.setLatitude(latitude);
                    location.setLongitude(longitude);
                    fetchAddressFromLatlng(location);
                }
                else{
                    progressBar.setVisibility(View.GONE);
                }
            }
        }, Looper.getMainLooper());

    Task<Location> task = mLocationClient.getLastLocation();
    task.addOnSuccessListener(new OnSuccessListener<Location>() {
        @Override
        public void onSuccess(Location location) {
            //when success
            if (location!=null){
                currentLocation = location;
                Toast.makeText(getApplicationContext(), text: currentLocation.getLatitude()
                    + " " + currentLocation.getLongitude(), Toast.LENGTH_SHORT).show();
                Log.d(TAG, msg: "onSuccess: latitude, longitude" + currentLocation.getLatitude() + " " + currentLocation.getLongitude());
                mMapVew.getMapAsync(new OnMapReadyCallback() {
                    @Override
                    public void onMapReady(@NonNull GoogleMap googleMap) {

                        //initialize lat lng
                        goToLocation(location.getLatitude(), location.getLongitude(), String.valueOf(location.getAccuracy()));
                    }
                });
            }
        }
    });
}
```

Figure 12 - HotspotActivity.java

The function *getCurrentLocation()* is used to auto focus the user current location.

```

private void goToLocation(double latitude, double longitude, String location) {
    googleMap.clear();
    progressBar.setVisibility(View.VISIBLE);
    Log.d(TAG, msg: "goToLocation: latitude longitude " + latitude + " " + longitude );
    Log.d(TAG, msg: "goToLocation: abs latitude longitude " + Math.round(latitude) + " " + Math.round(longitude) );
    hotspotViewModel.getAllHotspot(Math.round(latitude), Math.round(longitude)).observe( owner: this, new Observer<List<HotSpot>>() {
        @SuppressWarnings("ResourceAsColor", "SetTextI18n")
        @Override
        public void onChanged(List<HotSpot> hotSpots) {
            cv_cases_no.setVisibility(View.VISIBLE);
            cv_cone_area.setVisibility(View.VISIBLE);
            if (!hotSpots.isEmpty()){
                progressBar.setVisibility(View.GONE);
                hotSpot = hotSpots.get(0);
                tv_case_no.setText("In the past 14 days, within a 1km radius from your sear..." + " " + hotSpot.getCases() + " " + "reported case(s) of COVID-19.");
                Log.d(TAG, msg: "onChanged: text" + "In the past 14 days, within a 1km radius from your sear...");
                if (hotSpot.getCases() >= 15){
                    cv_cone_area.setCardBackgroundColor(ContextCompat.getColor(getApplicationContext(), R.color.red177));
                    tv_cone_area.setText("Red Zone Area");
                }else if (hotSpot.getCases() >= 10){
                    cv_cone_area.setCardBackgroundColor(ContextCompat.getColor(getApplicationContext(), R.color.darkYellow));
                    tv_cone_area.setText("Yellow Zone Area");
                }else if (hotSpot.getCases() < 10){
                    cv_cone_area.setCardBackgroundColor(ContextCompat.getColor(getApplicationContext(), R.color.grassGreen));
                    tv_cone_area.setText("Green Zone Area");
                }
            }else{
                progressBar.setVisibility(View.GONE);
                tv_case_no.setText("No data for the current region. Please come back later. Thanks");
                cv_cone_area.setCardBackgroundColor(R.color.darkGrey);
                tv_cone_area.setText("Unknown Zone Area");
            }
        }
    });
    LatLng latLng = new LatLng(latitude, longitude);
    MarkerOptions markerOptions = new MarkerOptions()
        .position(latLng).title(location)
        .draggable(false).visible(true);

    googleMap.addMarker(new MarkerOptions());
    googleMap.animateCamera(CameraUpdateFactory.newLatLngZoom(latLng, zoom: 45));
    googleMap.addMarker(markerOptions);
    googleMap.setMapType(GoogleMap.MAP_TYPE_NORMAL);
}

```

Figure 13 - HotspotActivity.java

goToLocation() is the function to set the map marker and zoom the Google map. The latitude and longitude of the place is retrieved. If there is the data (number of cases) of the location (latitude and longitude) in the database, the number of cases is retrieved and to indicate the dangerousness of that zone area (red, blue, yellow).

```

searchView.setOnQueryTextListener(new SearchView.OnQueryTextListener() {
    @Override
    public boolean onQueryTextSubmit(String query) {
        String location = searchView.getQuery().toString();
        List<Address> addressList = null;

        if (location != null || !location.equals("")) {
            Geocoder geocoder = new Geocoder( context: HotspotActivity.this);
            try {
                addressList = geocoder.getFromLocationName(location, maxResults: 10);
            } catch (IOException e) {
                e.printStackTrace();
            }
            assert addressList != null;
            Address address = addressList.get(0);

            txtTitleLocation.setVisibility(View.VISIBLE);
            txtUserCurrentLocation.setVisibility(View.VISIBLE);
            txtUserCurrentLocation.setText(address.getAddressLine( index: 0));

            Log.d(TAG, msg: "Search view: latitude, longitude" + address.getLatitude() + " " + address.getLongitude());
            goToLocation(address.getLatitude(), address.getLongitude(), location);
        }
        return false;
    }
});
@Override
public boolean onQueryTextChange(String newText) { return false; }
}

```

Figure 14 - HotspotActivity.java

Figure 11 is the code to realize the search function using GMS.

```
//Select all items
@Query("SELECT * FROM hotspot_table " +
        "WHERE latitude=(:latitude) OR longitude=(:longitude)")
LiveData<List<HotSpot>> getAllHotSpot(double latitude, double longitude);
```

Figure 15 – HotSpotDao.java

Query to retrieve the number of the cases using latitude and longitude. The latitude and longitude of the location which are very précised, were rounded to whole number. Hence, it is stated the number of the cases is the case number within certain km of area as shown in the figure below.

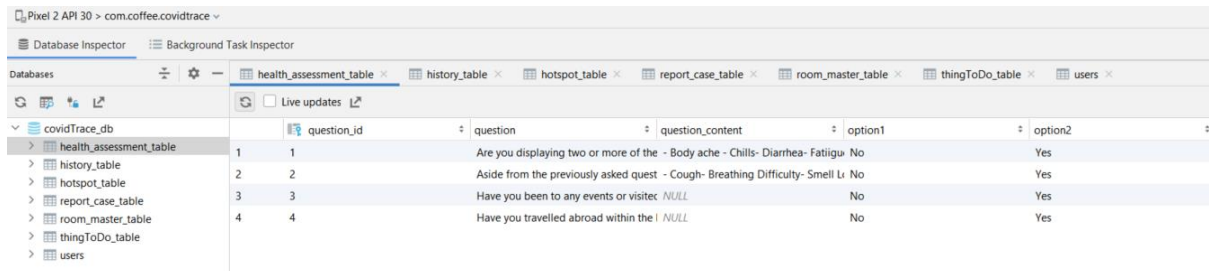


Figure 16 – Number of cases retrieved.

Feature Name: Personal COVID-19 Status

Function: To update the users' status.

The questions and option of health assessment questionnaires were stored in the room database also.



question_id	question	question_content	option1	option2
1	Are you displaying two or more of the	- Body ache - Chills- Diarrhea- Fatigue	No	Yes
2	Aside from the previously asked quest	- Cough- Breathing Difficulty- Smell Li	No	Yes
3	Have you been to any events or visitc	NULL	No	Yes
4	Have you travelled abroad within the l	NULL	No	Yes

Then, in the HealthAssessmentDao.java, the assessment questions were retrieved using the query in the figure below.

```
@Dao
public interface HealthAssessmentDao {

    @Query("SELECT * FROM health_assessment_table " +
            "ORDER BY question_id ASC")
    LiveData<List<HealthAssessment>> getAllAssessment();
}
```

Figure 17 – HealthAssessmentDao.java

```
questionViewModel.getAllAssessment().observe( owner: this, new Observer<List<HealthAssessment>>() {
    @SuppressWarnings("SetTextI18n")
    @Override
    public void onChanged(List<HealthAssessment> assessmentList) {
        questionCountTotal = assessmentList.size();
        currentQuestion = assessmentList.get(questionCounter);

        Log.d(TAG, msg: "showNextQuestion: Total: " + questionCountTotal);
        Log.d(TAG, msg: "showNextQuestion: Every: " + questionCounter);
    }
})
```

Figure 18 – QuestionActivity.java

The live data of assessment question list is retrieved using the questionViewModel. The number of the question is retrieved.


```

        questionCounter = currentQuestion.getQuestion_id();
    if have questions to answer
        if (questionCountTotal == 0) {
            Toast.makeText(
                context: QuestionActivity.this,
                text: "No question. Please come back latter. Thanks.",
                Toast.LENGTH_LONG
            ).show();
            finish();
        } else if (questionCounter < questionCountTotal) {
            currentQuestion = questionList.get(questionCounter);
            currentQuestion = assessmentList.get(questionCounter);

            tvQuestion.setText(currentQuestion.getQuestion());
            tvQuestionCount.setText("Question " + (currentQuestion.getQuestion_id()));
            tvQuestionContent.setText(currentQuestion.getQuestion_content());

            rb1.setText(currentQuestion.getOption1());
            rb2.setText(currentQuestion.getOption2());

            answered = false;
        } else {
            submitAssessment();
        }
    }
}

```

Figure 19 – QuestionActivity.java

If there are no question in the database (act as an error handling), a toast message is used to notice the user to be come back to this function later. If there is question in the database, the question text will change using the index. When user click the button “Next”, the index will increase change the question.

```

private void checkAnswer() {
    //in the case of radio button of "no" or "yes" is chosen
    if (rb1.isChecked()) {
        //1 equals to agree
        Log.d(TAG, msg: "Selected No"); //text show in console to double check
    } else if (rb2.isChecked()) {
        Log.d(TAG, msg: "Selected Yes");
        counter ++;
        Log.d(TAG, msg: "Counter: " + counter);
    }
}
}

```

Figure 20 – QuestionActivity.java

Each question has two options for this project, the above code (Figure 17) will check the user choice and the amount of the counter was passed to the SubmitActivity.java to update the user’s status. If the user choose yes, the counter will increase by one.


```

        userDao userDao,

        int low = 1;
        int medium = 2;
        int high = 3;
        private SubmitViewModel submitViewModel;

        case R.id.btn_submit:
            intent = new Intent( packageContext, this.MainActivity.class);
            if (counter == 0){
                submitViewModel.insert(low, currentUser.getId());
            } else if (counter >= 2){
                submitViewModel.insert(high, currentUser.getId());
            } else{
                submitViewModel.insert(medium, currentUser.getId());
            }

            progressBar.setVisibility(View.VISIBLE);
            Handler handler = new Handler();
            handler.postDelayed(new Runnable() {
                @Override
                public void run() {
                    intent.putExtra( name: "user", currentUser);
                    Toast.makeText( context: SubmitActivity.this, text: "Status Submitted", Toast.LENGTH_SHORT).show();
                    startActivity(intent);
                }
            }, loading_time);
            break;
    }
}

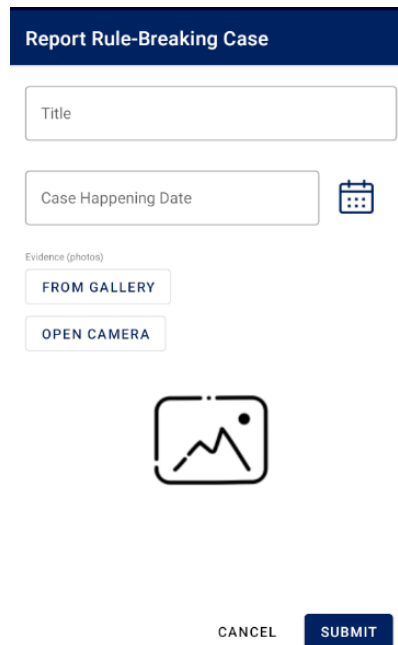
```

Figure 21 - SubmitActivity.java

The user status was updated if he or she press the submit button, the amount of the counter is passed from the QuestionActivity.java.

Feature name: Report Rule-Breaking Case

Function: Allow user to report the rule-breaking cases to the authorities.



The UI for the 'Report Rule-Breaking Case' function consists of a dark blue header bar with the title 'Report Rule-Breaking Case'. Below the header, there is a white input field for 'Title'. Underneath that is another white input field for 'Case Happening Date', accompanied by a calendar icon. Below the date field, there is a label 'Evidence (photos)' and two buttons: 'FROM GALLERY' and 'OPEN CAMERA'. Below these buttons is a large icon representing a photo gallery. At the bottom of the form, there are two buttons: 'CANCEL' and 'SUBMIT'.

Figure 22 – The UI of the function

```
public void openGallery(View view) {
    Intent intent = new Intent();
    intent.setType("image/*");
    intent.setAction(Intent.ACTION_GET_CONTENT);
    startActivityForResult(Intent.createChooser(intent, "Select Picture"), GALLERY_REQUEST_CODE);
}

public void openCamera(View view) {
    Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
        startActivityForResult(takePictureIntent, CAMERA_REQUEST_CODE);
    } else {
        {
            Toast.makeText(
                getApplicationContext(),
                "Reopen camera open.",
                Toast.LENGTH_LONG).show();
        }
    }
}

public void cancel(View view) { finish(); }
```

Figure 23 – ReportCaseActivity.java

The functions above are functioning to open the gallery, open the camera and cancel submitting.

```

public void save(View view) {
    Bitmap image = ((BitmapDrawable)selectedImageView.getDrawable()).getBitmap();
    reportCase = new ReportCase(new_case_title.getText().toString(), image, txShowDateButton.getText().toString(), currentUser.getId());
    if (!new_case_title.getText().toString().equals("") && !txShowDateButton.getText().toString().equals("") && !image.toString().equals(""))
    {
        new ReportCaseRepository(getApplication()).insert(reportCase);
        finish();
    }
    else
    {
        Toast.makeText(
            getApplicationContext(),
            text: "Please enter the information required",
            Toast.LENGTH_LONG).show();
    }
}
}

```

Figure 24 – ReportCaseActivity.java

This is an error handling to make sure that the user is entering all the necessary information. The image, title and the date are the necessary things. The image is important to act as evidence. The image is captured in Bitmap format, then it is converted to String to store into the Database.

```

gridView.setLayoutManager(new GridLayoutManager( context: this, spanCount: 2));
gridView.setAdapter(adapter);
int reportListSize = reportList.size();

reportList.addLast(reportCase);
gridView.getAdapter().notifyDataSetChanged();
// Scroll to the bottom.
gridView.smoothScrollToPosition(reportListSize);

// Populate the recyclerview with list from database
mViewModel.getAllCases(currentUser.getId()).observe( owner: this, new Observer<List<ReportCase>>() {
    @Override
    public void onChanged(List<ReportCase> reportCasesList) {
        if (reportCasesList.isEmpty()){
            gridView.setVisibility(View.GONE);
            activity_main_empty_view.setVisibility(View.VISIBLE);
        }
        else{
            gridView.setVisibility(View.VISIBLE);
            activity_main_empty_view.setVisibility(View.GONE);
            adapter.submitList(reportCasesList);
        }
    }
}
}

```

Figure 25 – ReportCaseMainActivity.java

After the user report the case, the ReportCaseMainActivity.java were retrieved data from the database. Recycler view is applied but it is programmatically set to the grid view with span count of two.

```

public void onBindViewHolder(@NonNull RecordViewHolder holder, int position) {
    Context context = holder.cv_report_status.getContext();
    ReportCase current = getItem(position);
    holder.titleTextView.setText(current.getTitle());
    holder.imageView.setImageBitmap(current.getImage());
    holder.tv_date.setText(current.getDate());

    if (current.getStatus() == 0){
        holder.cv_report_status.setCardBackgroundColor(ContextCompat.getColor(context,R.color.darkGrey));
        holder.tv_report_status.setText("Reported and get to make sure soon.");
    }else if (current.getStatus()==1){
        holder.cv_report_status.setCardBackgroundColor(ContextCompat.getColor(context,R.color.skyBlue));
        holder.tv_report_status.setText("Come to the case now");
    }else if (current.getStatus() == 2){
        holder.cv_report_status.setCardBackgroundColor(ContextCompat.getColor(context, R.color.grassGreen));
        holder.tv_report_status.setText("Action taken");
    }else if (current.getStatus() == 3){
        holder.cv_report_status.setCardBackgroundColor(ContextCompat.getColor(context, R.color.red177));
        holder.tv_report_status.setText("Report Rejected");
    }
}

```

Figure 26 - ReportAdapter.java

The status of each report was changed here.

Feature name: Sign up

Function: Allow user to create a new account.

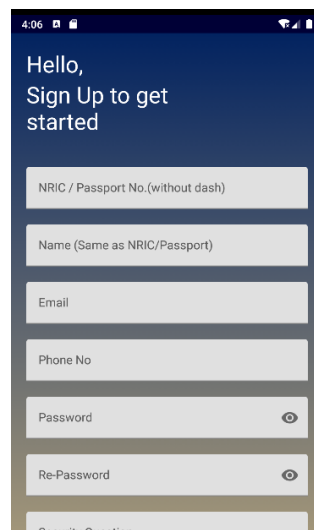


Figure 24 – UI of sign up

```
if (validateInput(userEntity)){
    if (password.getText().toString().equals(Objects.requireNonNull(repass.getText()).toString())){
        //insert
        Database database = Database.getDatabase(getApplicationContext());
        final UserDao userDao = database.userDao();
        UserEntity userEntity1 = userDao.uniqueEmail(email.getText().toString());
        if (userEntity1 == null){
            new Thread(new Runnable() {
                @Override
                public void run() {
                    //register user
                    userDao.registerUser(userEntity);
                    runOnUiThread(new Runnable() {
                        @Override
                        public void run() {
                            Toast.makeText(getApplicationContext(), text: "User Registered", Toast.LENGTH_SHORT).show();
                            Intent intent = new Intent(getApplicationContext(), LoginActivity.class);
                            intent.putExtra();
                            startActivity(intent);
                        }
                    });
                }
            }).start();
        }else {
            Toast.makeText(getApplicationContext(), text: "Email has been used", Toast.LENGTH_SHORT).show();
        }
    }else{
        Toast.makeText(getApplicationContext(), text: "Password not match", Toast.LENGTH_SHORT).show();
    }
} else{
    Toast.makeText(getApplicationContext(), text: "Please fill all fields", Toast.LENGTH_SHORT).show();
}
```

Figure 25 - SignUpActivity.java

The functions above are used to check the input of the user and put the data into the database. There are three IF statements, these are used to check the validation of the input (whether the input is null), to confirm the password, and to confirm the uniqueness of the email (one email can only link to one account). Call *userDao.registerUser()* to store the data into the database.

```
private Boolean validateInput(UserEntity userEntity){  
    if (userEntity.getEmail().isEmpty() ||  
        userEntity.getName().isEmpty() ||  
        userEntity.getNric().isEmpty() ||  
        userEntity.getPassword().isEmpty() ||  
        userEntity.getQuestion().isEmpty() ||  
        userEntity.getAnswer().isEmpty()){  
        return false;  
    }  
    return true;  
}
```

Figure 26 - SignUpActivity.java

In this Boolean function, we check the input of the user, if the user forgets to enter any data, the function will return false.

Feature name: Login

Function: Allow the user to login to the system.

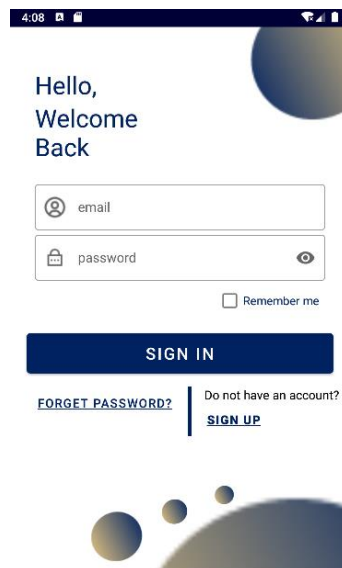


Figure 27 – UI of login

```
if (emailText.isEmpty() || passwordText.isEmpty()){
    Toast.makeText(getApplicationContext(), text: "Please fill all fields", Toast.LENGTH_SHORT).show();
}else{
    //query
    database = Database.getDatabase(getApplicationContext());
    UserDao userDao = database.userDao();
    new Thread(new Runnable() {
        @Override
        public void run() {
            UserEntity userEntity = userDao.login(emailText, passwordText);
            if (userEntity == null){
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        Toast.makeText(getApplicationContext(), text: "Invalid Credentials!", Toast.LENGTH_SHORT).show();
                    }
                });
            }else{
                currentUser = userEntity;
                Log.d( tag: "login", String.valueOf(currentUser.getName()));
                Log.d( tag: "login", String.valueOf(currentUser.getId()));
                Intent intent = new Intent(getApplicationContext(), MainActivity.class);
                //intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK|Intent.FLAG_ACTIVITY_NEW_TASK);
                intent.putExtra( name: "user", currentUser);
                startActivity(intent);
            }
        }
    }).start();
}
```

Figure 28 - LoginActivity.java

The functions above are used to authenticated the user. Retrieve the email and password in the room database. For the email entered by the user, if the password is the same as the one in the database with that particular email. Then the user is allowed to login to the system.

```
@Query("SELECT * FROM users WHERE email=:email and password=:password ")
UserEntity login(String email, String password);
default LiveData<List<UserEntity>> getAllUsers() { return mUserEntity; };
```

Figure 29 - UserDao.java

In figure 26 is how we do the authentication of the user.

Feature name: FAQ

Function: Help user to get more related information about the COVID-19.



Figure 30 – UI of FAQ

```
boolean isExpendable = faqList.get(position).isExpendable();
holder.relativeLayout.setVisibility(isExpendable ? View.VISIBLE : View.GONE);

if(position == 0){
    holder.linearLayout.setBackgroundColor(Color.parseColor( colorString: "#FF0000"));
    holder.linearLayout.getBackground().setAlpha(50);
    holder.relativeLayout.setBackgroundColor(Color.parseColor( colorString: "#FFFFFF"));
    holder.relativeLayout.getBackground().setAlpha(150);}

else if(position == 1){
    holder.linearLayout.setBackgroundColor(Color.parseColor( colorString: "#FC4C00"));
    holder.linearLayout.getBackground().setAlpha(50);
    holder.relativeLayout.setBackgroundColor(Color.parseColor( colorString: "#FFFFFF"));
    holder.relativeLayout.getBackground().setAlpha(150);}

else if(position == 2){
    holder.linearLayout.setBackgroundColor(Color.parseColor( colorString: "#FC4000"));
    holder.linearLayout.getBackground().setAlpha(50);
    holder.relativeLayout.setBackgroundColor(Color.parseColor( colorString: "#FFFFFF"));
    holder.relativeLayout.getBackground().setAlpha(150);}

else if(position == 3){
    holder.linearLayout.setBackgroundColor(Color.parseColor( colorString: "#8DFC00"));
    holder.linearLayout.getBackground().setAlpha(50);
    holder.relativeLayout.setBackgroundColor(Color.parseColor( colorString: "#FFFFFF"));
    holder.relativeLayout.getBackground().setAlpha(150);}

else if(position == 4){
    holder.linearLayout.setBackgroundColor(Color.parseColor( colorString: "#00FCA1"));
    holder.linearLayout.getBackground().setAlpha(50);
    holder.relativeLayout.setBackgroundColor(Color.parseColor( colorString: "#FFFFFF"));
    holder.relativeLayout.getBackground().setAlpha(150);}

else if(position == 5){
    holder.linearLayout.setBackgroundColor(Color.parseColor( colorString: "#0092FC"));
    holder.linearLayout.getBackground().setAlpha(50);
    holder.relativeLayout.setBackgroundColor(Color.parseColor( colorString: "#FFFFFF"));
    holder.relativeLayout.getBackground().setAlpha(150);}
```

```
linearLayout.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        faqs faqs = faqList.get(getAdapterPosition());
        faqs.setExpendable(!faqs.isExpendable());
        notifyItemChanged(getAdapterPosition());
    }
});
```

Figure 31 – `faq_name_adapter.java`

Set the parameter *isExpendable* and set the visibility of the relative layout which carries the details of the FAQ. The IF statement and ELSE IF statements are used to set the colour of the relative layout. So that the colour of every FAQ element is different.

```
private void initData() {
    faqList = new ArrayList<>();

    faqList.add(new faqs( faq_name: "IS COVID-19 WORSE THAN FLU?", faq_detail: "COVID-19 has a higher rate of severe disease and mortality in nearly every age group, compared to John Hopkins Bloomberg School of Public Health reported that "COVID-19 has killed more people in the US than flu has in the last five years."
    "\n\n" +
    "In the UK, data revealed that between January-August, COVID-19 killed 3.4 times as many people in 2020 than flu and pneumonia combined.\n\n" +
    "The US Centers for Disease Control and Prevention (CDC) estimate that in the United States flu has resulted in "between 12,000-61,000 deaths annually since 2
    faqList.add(new faqs( faq_name: "HOW DOES COVID-19 COMPARE TO OTHER PUBLIC HEALTH THREATS?", faq_detail: "This depends very much on where people live and how old they are,
    "Even though COVID-19 appears to directly affect children less than adults, it can have indirect impacts. Modelling studies have predicted that COVID-19 could
    "When COVID cases peaked in Europe, it killed almost twice as many Europeans as cancer which is the biggest killer under normal circumstances. ");

    faqList.add(new faqs( faq_name: "HOW EFFECTIVE ARE MASKS AND DO THEY ALSO NEED TO COVER MY NOSE?", faq_detail: "Mask wearing is a very simple and effective way to reduce 1
    "Masks alone are not enough to fully protect someone from the virus, but they still have an essential role to play in protecting high risk individuals, such a
    "COVID-19 is predominantly spread via respiratory droplets that are released when someone that has been infected coughs, sneezes or talks. Breathing those dro
    "Using a mask to just cover the mouth and not the nose only addresses part of risk, and so could defeat the whole purpose of wearing one in the first place. S
    faqList.add(new faqs( faq_name: "WHY DO GOVERNMENTS BENEFIT FROM HELPING TO ENSURE OTHER COUNTRIES ACCESS VACCINES?", faq_detail: "Ensuring COVID-19 vaccines reach people
    "Until every country has access to vaccines and they have been administered to a high enough proportion of the population to considerably reduce death and sev
    "Countries hoarding vaccines is known as vaccine nationalism, which is what happened with the 2009 H1N1 swine flu pandemic. If this happens with COVID-19 vacc
    "The quickest way to end the acute phase of this pandemic and limit the economic damage is for all countries to have equitable access to and effective deliver
    faqList.add(new faqs( faq_name: "WOULD IT BE POSSIBLE TO ACHIEVE HERD IMMUNITY WITHOUT VACCINES?", faq_detail: "Historically, the most successful way to achieve herd immu
    "Attempting to achieve natural herd immunity through infection will lead to a rise in cases, and a surge in the number of deaths. Moreover, natural immunity t
    faqList.add(new faqs( faq_name: "HOW CAN WE TRUST VACCINES THAT HAVE BEEN DEVELOPED SO FAST?", faq_detail: "COVID-19 vaccines have been developed and produced in record ti
    "The COVAX Facility - co-led by Gavi, the Coalition for Epidemic Preparedness Innovations (CEPI) and the World Health Organization (WHO) - has incentivised va
    faqList.add(new faqs( faq_name: "WHAT ARE THE RISKS OF RE-INFECTION?", faq_detail: "As with other coronaviruses, such as those that cause the common cold, SARS-CoV-2 can
    "Even if you have had a test that detects antibodies, reinfection is still possible, although less likely.\n\n" +
    "Antibodies are only one part of the vast, complex immune response that is set in motion when a person becomes infected with a virus like SARS-CoV-2 so having
    "In many people, infection with SARS-CoV-2 can lead to so-called 'long COVID' - where people suffer long-term complications resulting from the virus. This can
    faqList.add(new faqs( faq_name: "IS REGULARLY WASHING MY HANDS ENOUGH TO PROTECT ME?", faq_detail: "Even though regularly washing your hands is very important, as with mai
    "Keep washing your hands for at least 20 seconds each time, keep carrying antibacterial gel when you're travelling and remember to wear a mask and practice ph
    faqList.add(new faqs( faq_name: "DO I STILL NEED TO WORRY ABOUT INFECTION EVEN THOUGH I'M FIT AND HEALTHY?", faq_detail: "Search for keywords like 'fit,' 'healthy' and 'CI
    "A research paper on COVID-19-related mortality among US adults aged 25-44 revealed that in July 2020 almost 12,000 more deaths were recorded than normally ex
    "Even if you are fit and healthy, you could spread the infection to vulnerable people, such as older people or those with compromised immune systems or diabet
    faqList.add(new faqs( faq_name: "SHOULD I BE CONCERNED THAT THE SAMPLE SIZES IN VACCINE CLINICAL TRIALS WEREN'T BIGGER?", faq_detail: "A small sample size in clinical tri
    "COVID-19 is spreading so rapidly that needlessly delaying vaccine production in favour of having a larger clinical trial would put more people at risk. With
}
```

Figure 32 – FAQActivity.java

Function *InitData()* is used to initiate the content of the FAQ list. Use *add()* to add information into the array list.

Feature name: Dashboard

Function: Get COVID-19 information from the website use API.

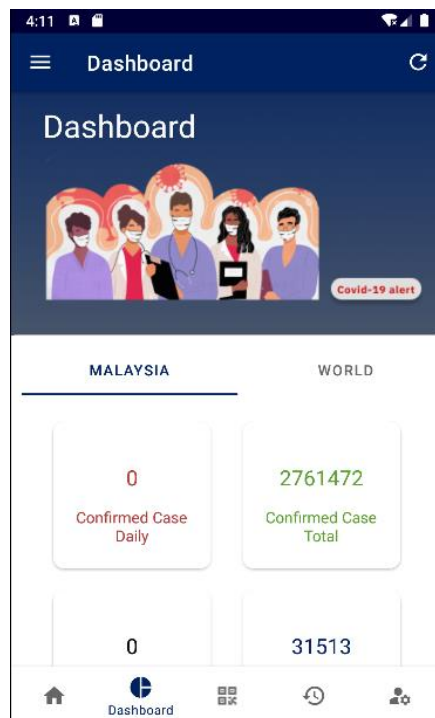


Figure 33 – UI of dashboard

```
<uses-permission android:name="android.permission.INTERNET" />
```

Figure 34 – AndroidManifest.xml

The code above is used to allow the device to connect to the Internet. Because later it needs to get the data from the internet.

```

private void fetchAPIUsingVolley(){
    RequestQueue requestQueue = Volley.newRequestQueue(this.getContext());
    String url = "https://disease.sh/v3/covid-19/countries/malaysia";

    StringRequest request = new StringRequest(
        Request.Method.GET,
        url,
        new Response.Listener<String>() {
            @Override
            public void onResponse(String response) {
                try {
                    JSONObject jsonObject = new JSONObject(response.toString());
                    m_daily_confirmed.setText(jsonObject.getString( name: "todayCases"));
                    m_total_confirmed.setText(jsonObject.getString( name: "cases"));
                    m_daily_death.setText(jsonObject.getString( name: "todayDeaths"));
                    m_total_death.setText(jsonObject.getString( name: "deaths"));
                    m_daily_recovered.setText(jsonObject.getString( name: "todayRecovered"));
                    m_total_recovered.setText(jsonObject.getString( name: "recovered"));

                    String current_timestamp = jsonObject.getString( name: "updated");
                    Timestamp ts = new Timestamp(Long.parseLong(current_timestamp));
                    Date date = new Date(ts.getTime());

                    m_datetime.setText(date.toString());
                }
                catch (JSONException e){
                    e.printStackTrace();
                }
            }
        },
        new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                Toast.makeText(getContext(),error.getMessage(),Toast.LENGTH_SHORT).show();
            }
        }
    );

    requestQueue.add(request);
}

```

Figure 35 – MalaysiaFragment.java

Use Volley to fetch the API. Declare a RequestQueue using volley, initiate the url in which the data we need is stored. Use JSONObject to get the information from the website and set all the information into the corresponding TextView in our screen. For the time,

Specially need to transform from the number into the time stamp before set to the TextView.

Feature name: Change Password

Function: Allow user to change password by correctly answer their security question.

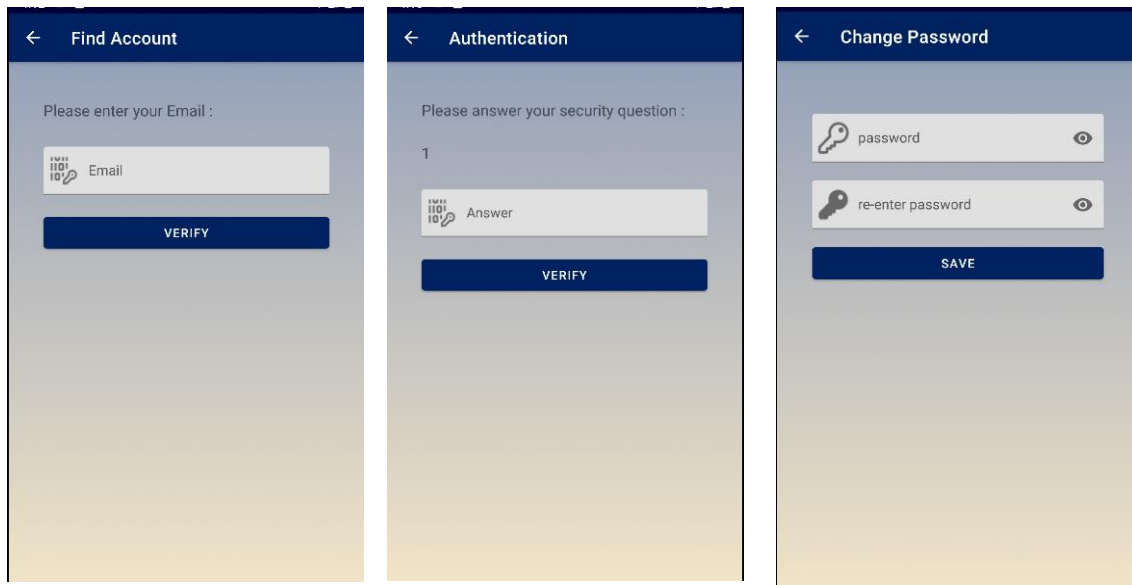


Figure 36 – UI of change password

```
String email_txt = email.getText().toString();
if (email_txt.isEmpty()){
    Toast.makeText(getApplicationContext(), text: "Please enter your email", Toast.LENGTH_SHORT).show();
}else{
    database = Database.getDatabase(getApplicationContext());
    UserDao userDao = database.userDao();
    new Thread(new Runnable() {
        @Override
        public void run() {
            UserEntity userEntity = userDao.findAccount(email_txt);
            if (userEntity == null){
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        Toast.makeText(getApplicationContext(), text: "Invalid Credentials!", Toast.LENGTH_SHORT).show();
                    }
                });
            }else{
                currentUser = userEntity;
                Log.d( tag: "login", String.valueOf(currentUser.getName()));
                Log.d( tag: "login", String.valueOf(currentUser.getId()));
                Intent intent = new Intent(getApplicationContext(), AuthenticationActivity.class);
                //intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK|Intent.FLAG_ACTIVITY_NEW_TASK);
                intent.putExtra( name: "user", currentUser);
                intent.putExtra( name: "email", email_txt);
                startActivity(intent);
            }
        }
    }).start();
}
```

Figure 37 – FindAccountActivity.java

The function above is used to check whether the user is existed. Firstly, check whether the email is null, the user must enter the email to verify. Then retrieve the email from the database, if the email exists, then go to the next step.

```
database = Database.getDatabase(getApplicationContext());
UserDao userDao = database.userDao();
String question_txt = userDao.findQuestion(email_txt);
question.setText(question_txt);

String answer_txt = answer.getText().toString();
if (answer_txt.isEmpty()){
    Toast.makeText(getApplicationContext(), text: "Please enter your answer", Toast.LENGTH_SHORT).show();
}else{
    //database = Database.getDatabase(getApplicationContext());
    //UserDao userDao = database.userDao();
    new Thread(new Runnable() {
        @Override
        public void run() {
            UserEntity userEntity = userDao.authentication(email_txt, answer_txt);
            if (userEntity == null){
                runOnUiThread(new Runnable() {
                    @Override
                    public void run() {
                        Toast.makeText(getApplicationContext(), text: "Wrong Answer!", Toast.LENGTH_SHORT).show();
                    }
                });
            }else{
                currentUser = userEntity;
                Intent intent = new Intent(getApplicationContext(), ChangePasswordActivity.class);
                intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TASK|Intent.FLAG_ACTIVITY_NEW_TASK);
                intent.putExtra(name: "user", currentUser);
                intent.putExtra(name: "email", email_txt);
                startActivity(intent);
            }
        }
    }).start();
}
```

Figure 38 – AuthenticationActivity.java

Call *userDao.findQuestion()* to find the security question of the particular email. Then the question will be shown in the TextView. Then call *userDao.authentication()* to check whether the answer is correct.

```

if (password.getText().toString().equals(Objects.requireNonNull(repassword.getText()).toString())){
    database = Database.getDatabase(getApplicationContext());
    UserDao userDao = database.userDao();
    new Thread(new Runnable() {
        @Override
        public void run() {
            String newPass = password.getText().toString();
            userDao.updatePassword(newPass, email_txt);
            runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    Toast.makeText(getApplicationContext(), text: "Password Successfully Changed", Toast.LENGTH_SHORT).show();
                    Intent intent = new Intent(getApplicationContext(), LoginActivity.class);
                    intent.putExtra();
                    startActivity(intent);
                }
            });
        }
    }).start();
}else{
    Toast.makeText(getApplicationContext(), text: "Password not match", Toast.LENGTH_SHORT).show();
}
}

```

Figure 39 – ChangePasswordActivity.java

The function above is used to change the password of the user. Call *userDao.updatePassword()* to update the data in the room database.

```

@Query("SELECT * FROM users WHERE email = (:user_email)")
UserEntity findAccount(String user_email);

@Query("SELECT * FROM users WHERE email=(:email) and answer = (:user_answer)")
UserEntity authentication(String email, String user_answer);

@Query("SELECT question FROM users WHERE email=(:email)")
String findQuestion(String email);

@Query("UPDATE users SET password = (:password) WHERE email = (:email)")
void updatePassword(String password, String email);

```

Figure 40 – UserDao.java

These four Query are used in the change password feature.

Test cases and test results

No	Scenario	Test Case Steps	Expected Outcome	Pass/Fail	If fail, what was observed
1	User read the important notes and task to complete.	1. User choose the home page and pages of important notes and task to complete. 2. User click the image of the announcement.	1. The announcement show on the page. 2. The image is enlarged.	Pass	
2	User check-in their location using the application.	1. User clicks the check-in button. 2. User scans the QR code.	1. The information of the latest location and the scanning time are stored and show on the UI. 2. The list of the history location also shown in descending order (from latest to old).	Pass	
3	User checks the reported COVID-19 cases of certain area.	1. User enters the location name. 2. User clicks the auto focus bottom.	1. The location can be search. 2. The auto focus of the area is accurate. 3. The number of cases of certain area is retrieved and the dangerousness of the area is showed.	Pass	

4	User update the status.	<ol style="list-style-type: none"> 1. User clicks the radio button. 2. User submits his or her questionnaires. 	<ol style="list-style-type: none"> 1. The question can be retrieved correctly. 2. After answering all the question, they can be submitted. 	Pass	
5	User report the rule-breaking cases.	<ol style="list-style-type: none"> 1. User types the text inside the "Title" textbox/. 2. User enters the date by clicking the button with calendar icon. 3. User uploads the image. 4. User submit the reports. 	<ol style="list-style-type: none"> 1. The text can be entered correctly. 2. The date can be chosen from the date picker. 3. The image can be uploaded through the local storage of the phone or the image capture spontaneously. 4. The report appear at the frontier page, and the status of the report is shown. 	Pass	<p>Failed at first, due to the condition set for error handling: The user does not enter the title or the date or not upload the image as evidence.</p> <p>Fixed by setting all of the title, date and the image must have value.</p>
6	User register	<ol style="list-style-type: none"> 1. User types in the NRIC, name, email, phone number, password (twice), security question, answer to the security question. 2. User tap the button "SIGN UP" 	<ol style="list-style-type: none"> 1. The text can be entered correctly. 2. The system shows the message "User Registered" 	Pass	

7	User login with correct password.	<ol style="list-style-type: none"> 1. User types in the email and correct password. 2. User tap the button “SIGN IN” 	<ol style="list-style-type: none"> 1. The Text can be entered correctly. 2. User enter the Homepage. 	Pass	
8	User login with wrong password.	<ol style="list-style-type: none"> 1. User types in the email and wrong password. 2. User tap the button “SIGN IN” 	<ol style="list-style-type: none"> 1. The Text can be entered correctly. 2. System shows the message “Invalid Credential” 	Pass	
9	User change password	<ol style="list-style-type: none"> 1. User tap “FORGET PASSWORD” in the Login page. 2. User enter the email. 3. User tap “VERIFY” button. 4. User enter correct answer. 5. User tap “VERIFY” button. 6. User enter new password twice. 	<ol style="list-style-type: none"> 1. All text can be entered correctly. 2. System shows the message “Password Successfully changed” 	Pass	

		7. User tap “SAVE” button.			
10	User change password with invalid email	<ol style="list-style-type: none"> 1. User tap “FORGET PASSWORD” in the Login page. 2. User enter the email. 3. User tap “VERIFY” button. 	<ol style="list-style-type: none"> 1. All text can be entered correctly. 2. System shows the message “Invalid Credentials!” 	Pass	
11	User change password with wrong answer	<ol style="list-style-type: none"> 1. User tap “FORGET PASSWORD” in the Login page. 2. User enter the email. 3. User tap “VERIFY” button. 4. User enter correct answer. 5. User tap “VERIFY” button. 	<ol style="list-style-type: none"> 1. All text can be entered correctly. 2. System shows the message “Wrong Answer” 	Pass	

12	User read Dashboard	<ol style="list-style-type: none"> 1. User login to the system. 2. User tap “Dashboard” from the bottom navigation. 3. User tap the “World” button. 	<ol style="list-style-type: none"> 1. The bottom navigation can correctly turn the page into the dashboard page. 2. The six COVID-19 related data of Malaysia and world can be shown correctly. And the update date in the bottom is correct. 	Pass	
13	User read profile	<ol style="list-style-type: none"> 1. User login to the system. 2. User tap “Profile” from the bottom navigation. 	<ol style="list-style-type: none"> 1. The bottom navigation can correctly turn the page into the profile page. 2. The email, phone, and IC number shown in the profile is correct. (the same as what the user entered when registered.) 		
14	User read FAQ	<ol style="list-style-type: none"> 1. User login to the system. 2. User scroll the card and tap the second card. 3. User tap one of the FAQ titles. 	<ol style="list-style-type: none"> 1. The Card in the homepage can be scrolled. 2. After tap the FAQ title, that one can expanded and more details can be shown. 		

Project Name	CovidTrace
Testing Start Date / Time	31/12/2021 1.00 p.m.
Testing End Date / Time	31/12/2021 1.30 p.m.
Name of Tester/s:	Fatimah binti Abdullah