XIAMEN UNIVERSITY MALAYSIA

Course Code	:	SWE401
Course Name	:	Programming Elective II (3)
Lecturer	:	Simon Lau Boung Yew
Academic Session	:	2020/09
Assessment Title	:	Wireframe
Submission Due Date	:	2021/1/13

Prepared by :

Student ID	Student Name
SWE1709151	Huang JiaQi
SWE1709204	Li Xia
SWE1709346	Sun RuiTao

Date Received :

Feedback from Lecturer:

Mark:

XIAMEN UNIVERSITY MALAYSIA

Own Work Declaration

I/We hereby understand my/our work would be checked for plagiarism or other misconduct, and the softcopy would be saved for future comparison(s).

I/We hereby confirm that all the references or sources of citations have been correctly listed or presented and I/we clearly understand the serious consequence caused by any intentional or unintentional misconduct.

This work is not made on any work of other students (past or present), and it has not been submitted to any other courses or institutions before.

Signature: Li Xia / Huang JiaQi / Sun RuiTao

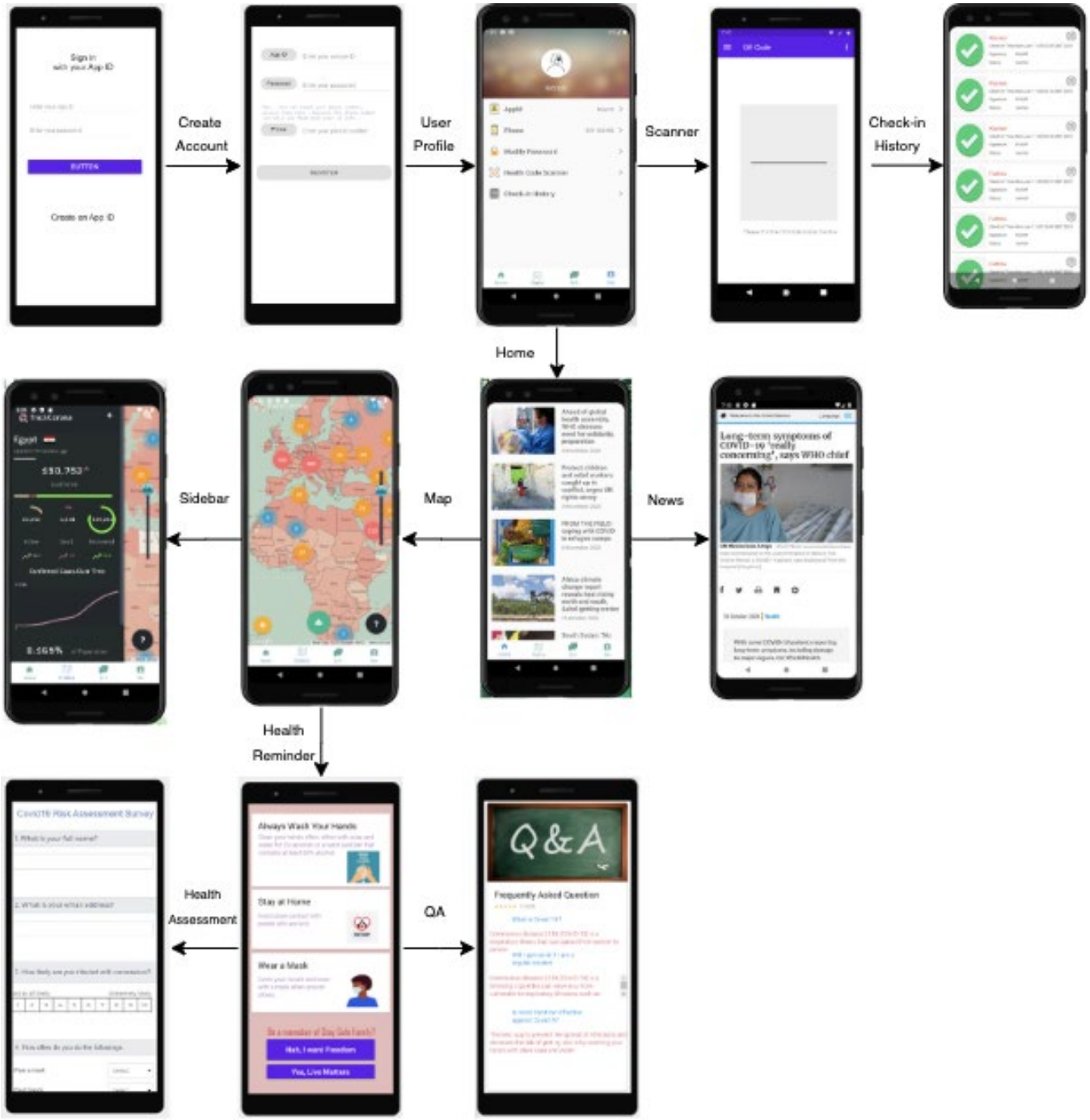
Date: 2021/01/13

TABLE OF CONTENT

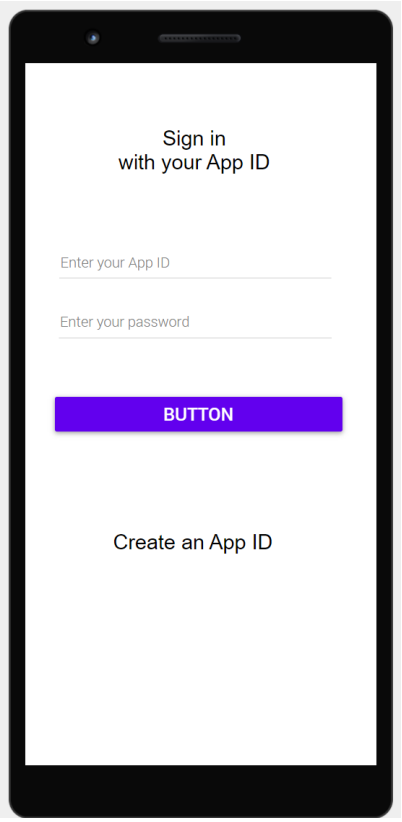
CHAPTER1

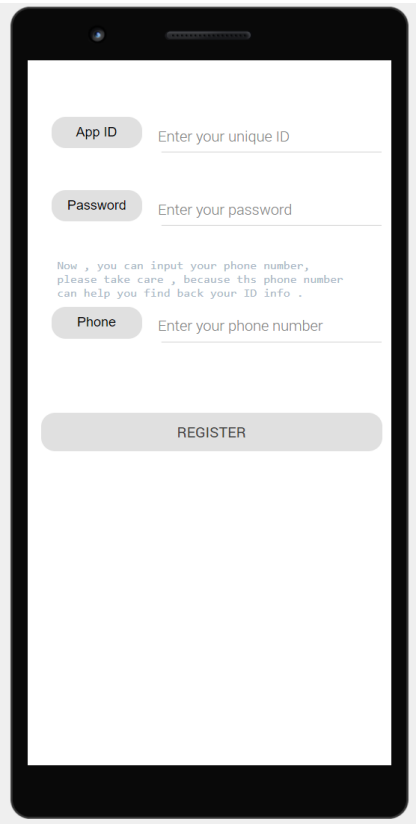
DESIGN

This is the overall flow chart of the Covid-19 App, including a series of pages developed by our team members. Initially, the user can register an account, and then check their personal information in the User Profile page. Also, there's a health QR code scanner, which is used for tracking user's check-in history. After login, the Home Page shows a list of news related to Covid-19. Clicking on either one of the news items will trigger a web page to be opened in the browser, which contains the specific description of the selected news. By clicking on the bottom navigation bar, users can navigate to the Map page, which display the impact of Covid-19 worldwide in a map. By clicking on specific country in the map will trigger the country's statistics to be opened in the left sidebar. Finally, the Heath Reminder page allows users to learn basic health knowledge and tips. Also, users can choose to open the Health Assessment and Q&A page.


Create Account
User Profile
Scanner
Check-in History
Home
Sidebar
Map
News
Health Reminder
Health Assessment
QA
Q&A
Frequently Asked Question

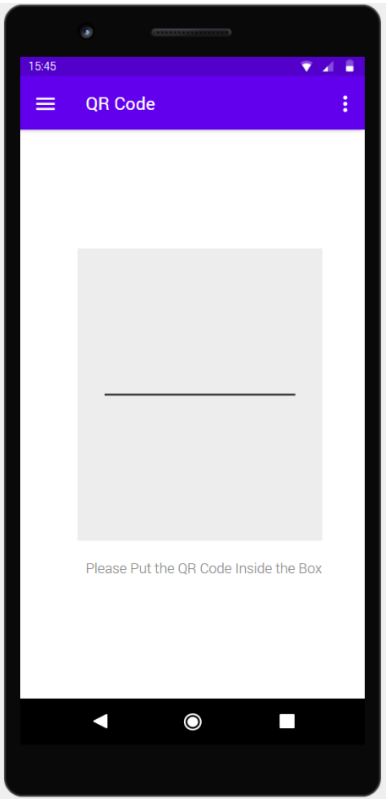
Basic Login Interface:
Press Sign in button to sign in.
Press “Create App ID” button to create new account

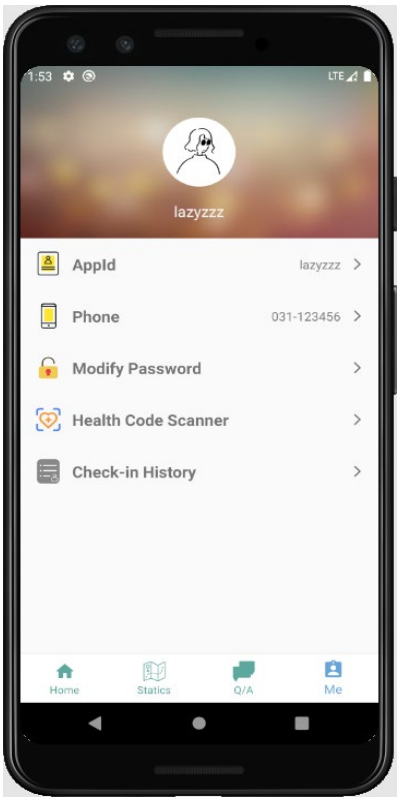




Register Interface:
In this section, Username, Password and Phone number is needed for registration. Particularly, if you forget your password, you can reset it using the phone number provided.

Scanner Interface:
Any QR code inside the scanner box will be detected by the program.

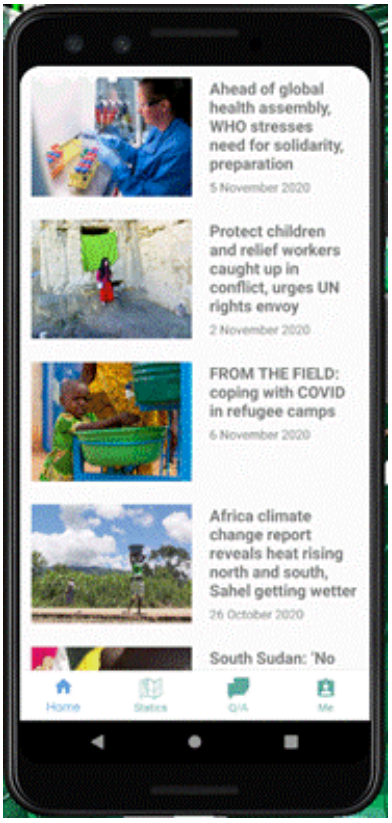




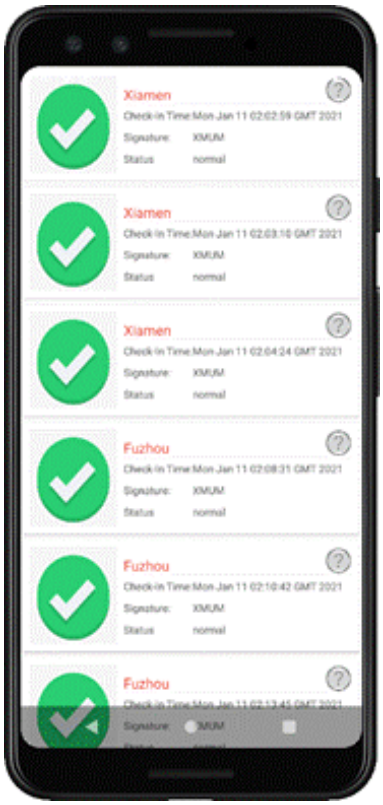
Profile Interface:
In this part, you can modify your account information such as:

- Username
- Password
- Phone Number
- Avatar

Home:
In this page, users can scroll down to read news related to the Covid-19. Besides, clicking on news item will open a new web page in the browser with detailed information.



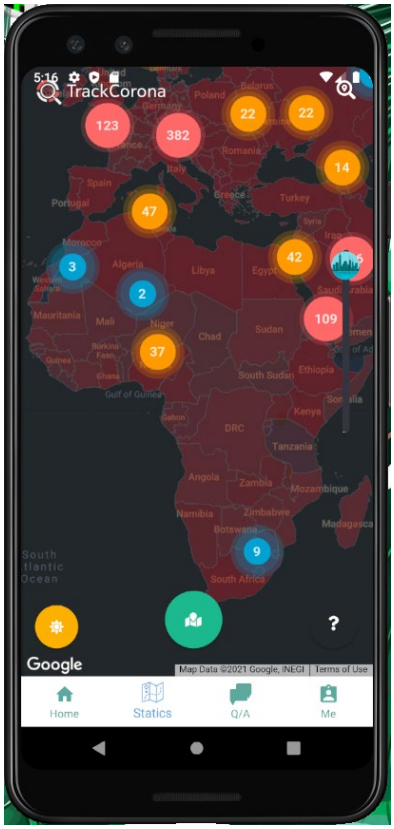
News:
This page will be opened in the browser when users click on the news item on the Home page. It is linked to the United Nations official website, since the news data come from the Covid-19 news section of the United Nations' official website.



Check-In History:
In this part, you can check your check-in history ordered by check-in time.
Basic information like
- Check-In Location
- Time
- Condition
- Location

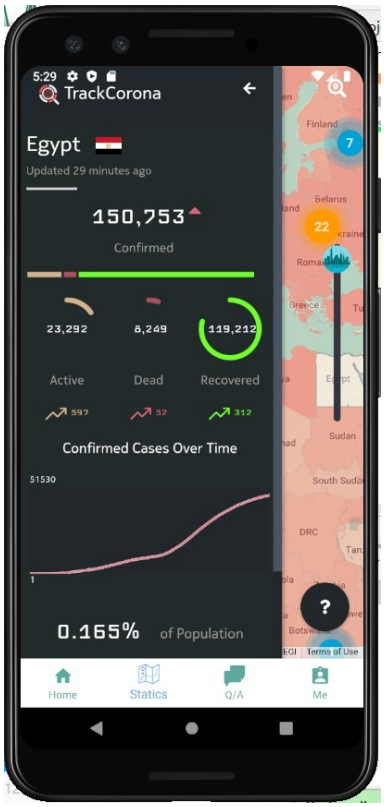
Will be provided in the list.
User can also report a wrong record by pressing the "question mark" button

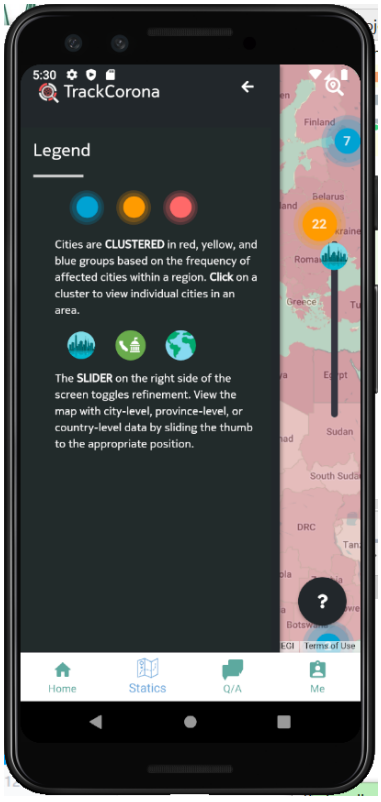
Statistics:
This page shows the current impact of the coronavirus pandemic throughout the world. It is implemented through a WebView that links to the Track Corona website.



Statistics (Dark):
This is dark view of the Statistics page, which will be updated automatically according to the time of the day.

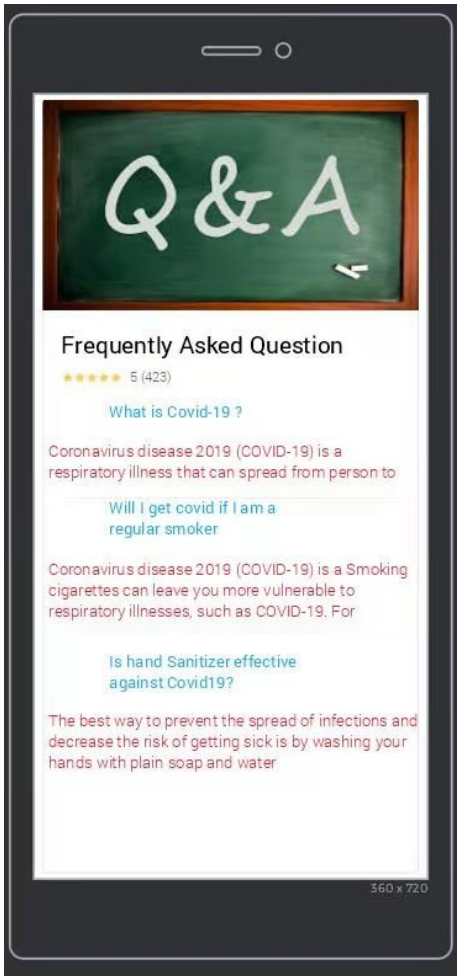
Sidebar:
The sidebar shows the detailed statistics of the country clicked by the user. It will slide from the left once user clicks on a country in the map.



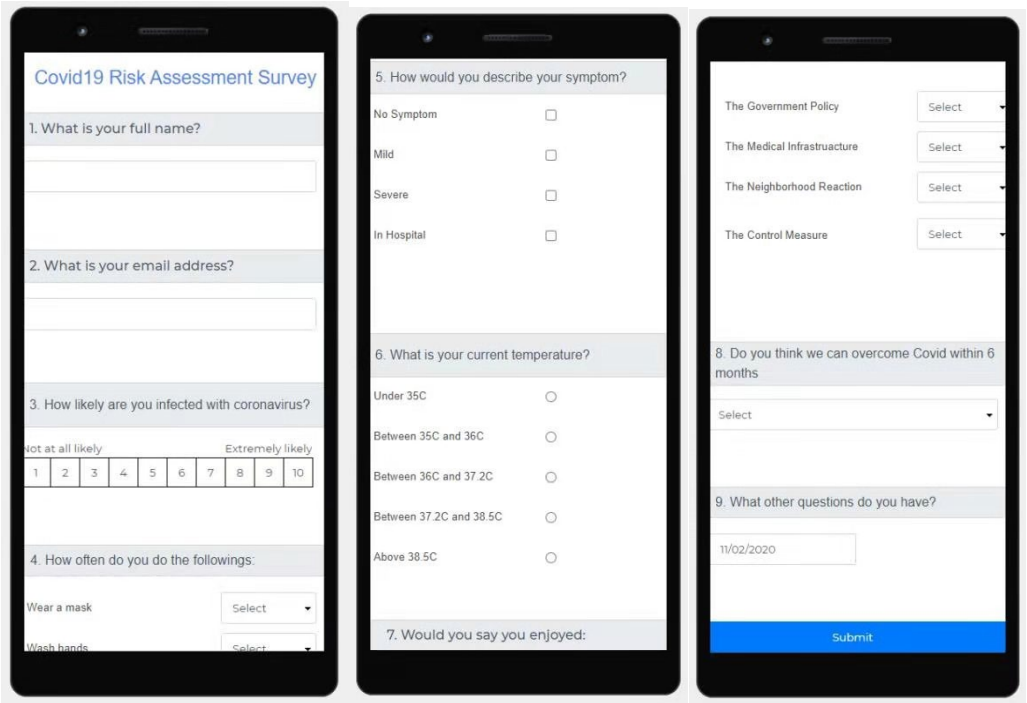


Legend Info:
This sidebar will appear when users clicks on the question mark button on the bottom right corner of the page, which describes the Legend information.

Q&A Interface:
In this interface, you can see some common ask questions and answers



Survey Interface:
The survey measures user behavior during pandemic and gives a feedback accordingly.



Knowledge Interface:
In this interface, you can see some common methods to stay away from corona virus. By pressing No button you will go to Survey Page. Clicking yes, you will go to Q&A page.

CHAPTER2

CODING

2.1 Server Side

In our project, we deploy our server based on a popular framework called **Spring Boot**(http://spring.io) based on Java language. Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that you can "just run".

As for the database part, for the purpose of demonstration and simplicity, we use an embedded light database called H2 instead of databases such as MySQL that requires additional deployments and maintenances. The performance of H2 is acceptable while under low concurrency circumstances.

Additionally, for the ORM framework, we choose JPA as our main ORM framework because it can automatically create tables and generate SQL based on a java entity class. We only need to create the corresponding entity classes that mapped to the database then we can just manipulate the entity class only without writing any lines of SQL. It's very convenient for a small project.

Finally, we deployed our project to Alibaba Cloud in China. Although it only has 2 cores with 4G of RAM, it's enough for our project to do a demonstration.

A document with clear annotations of our server API is generated using Swagger, you can click the link to see more details of our API (http://47.112.193.148:8080/swagger-ui.html).

As a big android project, we use Git to do the version control and we share the project on Gitee which is a Chinese version of GitHub that we can access it without worry about the network issues.

2.2 Network Communication Tools:

In order to communicate with our server, we combined two libraries which are RxJava(*ReactiveX/RxJava*, 2013/2021) and Retrofit (*square/retrofit*, 2010/2021). They provide us an asynchronous type-safe HTTP access to our server based on observer pattern. Here, we'll show the usage of these library with the example of login and register function in Figure 1 & Figure 2.

The logic of login and register is showed in the following figure. Invalid username or password will be rejected by the server. If user doesn't have an account, they'll need to first register one. Noted that the username is unique and duplicated one will not be allowed for registration. If user successfully login into the system, an access ***token*** will be given from the server for future work that need user's authorization.

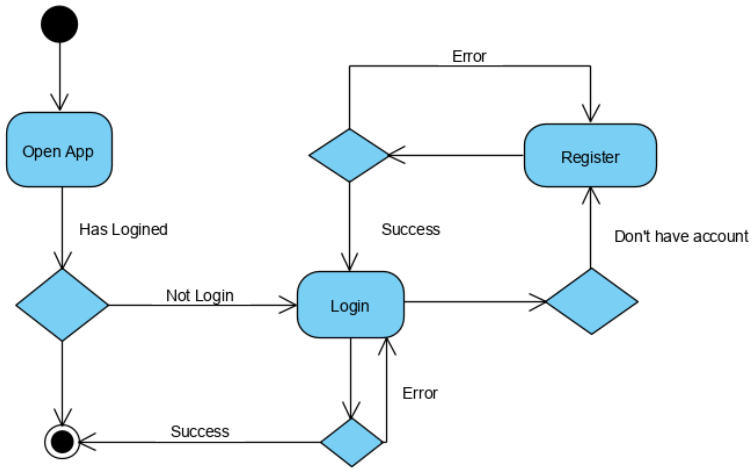


Figure 1. UML of Login & Register Process

```
userService.register(user)
        .subscribeOn(Schedulers.io())
        .observeOn(AndroidSchedulers.mainThread())
        .subscribe(new IObserver<IResponse<Void>>( context: this) {
            @Override
            public void onNext(IResponse<Void> response) {
                if (response.isSuccess()) {
                    Toast.makeText( context: RegisterActivity.this,  text: "Register Successfully", Toast.LENGTH_SHORT).show();
                    startActivity(new Intent( packageContext: RegisterActivity.this, LoginActivity.class));
                } else {
                    Toast.makeText( context: RegisterActivity.this,  text: "Username already exists!", Toast.LENGTH_SHORT).show();
                }
            }
        });
```

Figure 2. Sample of using RxJava + Retrofit for User Registration

2.3 Home Page:

In the home page (in Figure 1), a list of Covid-19 news items is displayed. This is implemented based on Recycler View. As shown in code below, three mandatory components are required to make the Recycler View work, including an adapter, a layout manager and a view holder. The adapter is used to bind the Recycler View with a list of news data. While the layout manager is used to correctly position all the data in the list. Finally, the view holder is used to visually represent an element in the data list in the Recycler View.

Also, the Swipe Refresh Layout widget is employed to achieve refreshing of news data. It allows users to trigger an update with a vertical swipe. In this case, the refresh behavior is implemented to get the next batch of news data.

```
public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle sa
    View view = inflater.inflate(R.layout.activity_news, container,  attachToRoot: false);
    random = new Random();

    news_list = new ArrayList<>();
    readJSON( ...strings: "un-news.json");

    recyclerView = view.findViewById(R.id.news_recyclerview);

    NewsRecyclerViewAdapter viewAdapter = new NewsRecyclerViewAdapter(view.getContext(), getNextBatch());
    recyclerView.setLayoutManager(new LinearLayoutManager(view.getContext()));
    recyclerView.setAdapter(viewAdapter);

    SwipeRefreshLayout refreshLayout = view.findViewById(R.id.swipe_news_layout);
    refreshLayout.setOnRefreshListener(() -> {
        viewAdapter.setmData(getNextBatch());
        viewAdapter.notifyDataSetChanged();
        refreshLayout.setRefreshing(false);
    });

    return view;
}
```

Figure 3. Recycler View for Home Page

2.4 QR-Code Scanning:

In this part, we used a third-party library called ***zxing-android-embedded*** (https://github.com/journeyapps/zxing-android-embedded) which is an embedded version of ***zxing*** library that used for various scanning task such bar code and QR-code to taggle the tasks. Particularly, we replaced the original UI of zxing refer to the scanning function in WeChat.

The QR-Code scanning function works with two parts which are user side and administrator side. In the end-user side, it's pretty easy to use it by simply scanning the QR-Code posted by the administrators, for example schools. The server will automatically record your attendance with your physical location and health status. For the administrators' side, they will need to generate the QR-Code using the tools that we provided by filling in the location and issuer. More specifically, the QR-Code contains the information that needed for the attendance. These parameters will be collected and then send to the server for recording.

```
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    IntentResult result = IntentIntegrator.parseActivityResult(requestCode, resultCode, data);
    if (result != null) {
        if (result.getContents() == null) {
            Toast.makeText(context, text: "Cancelled", Toast.LENGTH_SHORT).show();
        } else {
            Gson gson = new Gson();
            Attendance attendance = gson.fromJson(result.getContents(), Attendance.class);
            attendance.setUserId(user.getId());
            attendance.setHealthy(true);
            attendance.setCreateTime(new Date());
            attendanceService.singUp(token, attendance)
                    .subscribeOn(Schedulers.io())
                    .observeOn(AndroidSchedulers.mainThread())
                    .subscribe(new IObserver<IResponse<Void>>(context) {
                        @Override
                        public void onNext(IResponse<Void> response) {
                            if (response.isSuccess()) {
                                Toast.makeText(context, text: "Sign In Successful", Toast.LENGTH_SHORT).show();
                            }
                        }
                    });
        }
    } else {
        super.onActivityResult(requestCode, resultCode, data);
    }
}
```

Figure 4. Parse the scanned QR-Code and Send the attendance to the server

2.5 Health Risk Assessment:

The logic behind health assessment is firstly by listing down a couple of common sense to test user's health awareness, the use will be redirect to a survey asking him a few questions.

```
public class AssessmentActivity extends AppCompatActivity {

    public static final String EXTRA_NUMBER = "com.example.application.example.EXTRA_NUMBER";
    public static final String EXTRA_NUMBER_1 = "com.example.application.example.EXTRA_NUMBER_1";
    public static final String EXTRA_NUMBER_2 = "com.example.application.example.EXTRA_NUMBER_2";
    public static final String EXTRA_NUMBER_3 = "com.example.application.example.EXTRA_NUMBER_3";
    public static final String EXTRA_NUMBER_4 = "com.example.application.example.EXTRA_NUMBER_4";
    public static final String EXTRA_NUMBER_5 = "com.example.application.example.EXTRA_NUMBER_5";

    @BindView(R.id.editName)
    EditText editName;
    @BindView(R.id.editEmail)
    EditText editEmail;
    @BindView(R.id.ratingLikely)
    RatingBar ratingLikely;
    @BindView(R.id.ratingMask)
    RatingBar ratingMask;
    @BindView(R.id.ratingHand)
    RatingBar ratingHand;
    @BindView(R.id.ratingSleep)
    RatingBar ratingSleep;
    @BindView(R.id.ratingHome)
    RatingBar ratingHome;
    @BindView(R.id.radioGroupSymptom)
    RadioGroup radioGroupSymptom;
    @BindView(R.id.radioGroupTemperature)
    RadioGroup radioGroupTemperature;
    @BindView(R.id.btn_submit_form)
    Button submitForm;
```

```
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_assessment);
        ButterKnife.bind( target: this);
    }

    @OnClick(R.id.btn_submit_form)
    public void onClick() {
        String name = editName.getText().toString();
        String email = editEmail.getText().toString();
        float ratingLikelyFloat = ratingLikely.getRating();

        RatingBar[] ratings = new RatingBar[]{ratingMask, ratingHand, ratingSleep, ratingHome};
        float ratingSum = 0.0f;
        for (RatingBar bar : ratings) {
            ratingSum += bar.getRating();
        }

        RadioButton symptomSelected = findViewById(radioGroupSymptom.getCheckedRadioButtonId());
        RadioButton temperatureSelect = findViewById(radioGroupTemperature.getCheckedRadioButtonId());

        Intent intent = new Intent( packageContext: this, SummaryActivity.class);
        intent.putExtra(EXTRA_NUMBER, ratingLikelyFloat);
        intent.putExtra(EXTRA_NUMBER_1, ratingSum);
        intent.putExtra(EXTRA_NUMBER_2, name);
        intent.putExtra(EXTRA_NUMBER_3, email);
        intent.putExtra(EXTRA_NUMBER_4, symptomSelected.getText().toString());
        intent.putExtra(EXTRA_NUMBER_5, temperatureSelect.getText().toString());

        startActivity(intent);
```

Figure 5. Health Risk Assessment code

The question form consists of some typical android components like rating bar, radio button, common button, etc. the data collected in this page will be passed to next page, aka, summary activity. In the summary activity, there are some mathematical logics to decide if you are risky to COVID-19. Based on the result of your survey, the sum of rating bar value will be stored. Then accordingly to the logic above, the system will decide how likely you are infected with the coronavirus.

CHAPTER3

TESTING

3.1 UI-Testing

In the project, we use the UI testing framework called expresso to the UI testing. As a result, four UIs were tested which are login, register, news and survey.

For the login test, only correct username and password will be allowed to login. Incorrect username or password should be rejected. As a result, user login successfully with correct username and password and rejected with invalid username and password.

```
@RunWith(AndroidJUnit4.class)
public class LoginActivityTest {

    @Rule
    public ActivityTestRule<LoginActivity> rule = new ActivityTestRule<>(LoginActivity.class);

    @Test
    public void loginTest() {
        onView(ViewMatchers.withId(R.id.username_et))
                .perform(ViewActions.clearText(), typeText( stringToBeTyped: "lazyzzz"), ViewActions.closeSoftKeyboard());
        onView(ViewMatchers.withId(R.id.password_et))
                .perform(ViewActions.clearText(), typeText( stringToBeTyped: "123"), ViewActions.closeSoftKeyboard());
        onView(ViewMatchers.withId(R.id.login_bt))
                .perform(click());
    }
}
```

Figure 1. Login Test

For the register test, user should fill in their username, password and phone number to complete the registration. As a result, register successful once all fields are filled in. Any missing fields will lead to failure.

```
@RunWith(AndroidJUnit4.class)
public class RegisterActivityTest {

    @Rule
    public ActivityTestRule<RegisterActivity> rule = new ActivityTestRule<>(RegisterActivity.class);

    @Test
    public void registerTest() {
        onView(ViewMatchers.withId(R.id.register_username_et))
                .perform(typeText( stringToBeTyped: "Test"), ViewActions.closeSoftKeyboard());
        onView(ViewMatchers.withId(R.id.register_password_et))
                .perform(typeText( stringToBeTyped: "Test"), ViewActions.closeSoftKeyboard());
        onView(ViewMatchers.withId(R.id.register_register_btn))
                .perform(click());
    }
}
```

Figure 2. Registration Test

For the testing the navigation, once the Main Activity is launched, if the Map button is clicked in the bottom navigation, the Map page is opened automatically.

```
@RunWith(AndroidJUnit4.class)
public class NewsTest {

    @Rule
    public ActivityTestRule<MainActivity> rule = new ActivityTestRule<>(MainActivity.class);

    @Test
    public void testEventFragment() {
        onView(ViewMatchers.withId(R.id.nav_page_map))
                .perform(click());
    }
}
```

Figure 6. Navigation Test

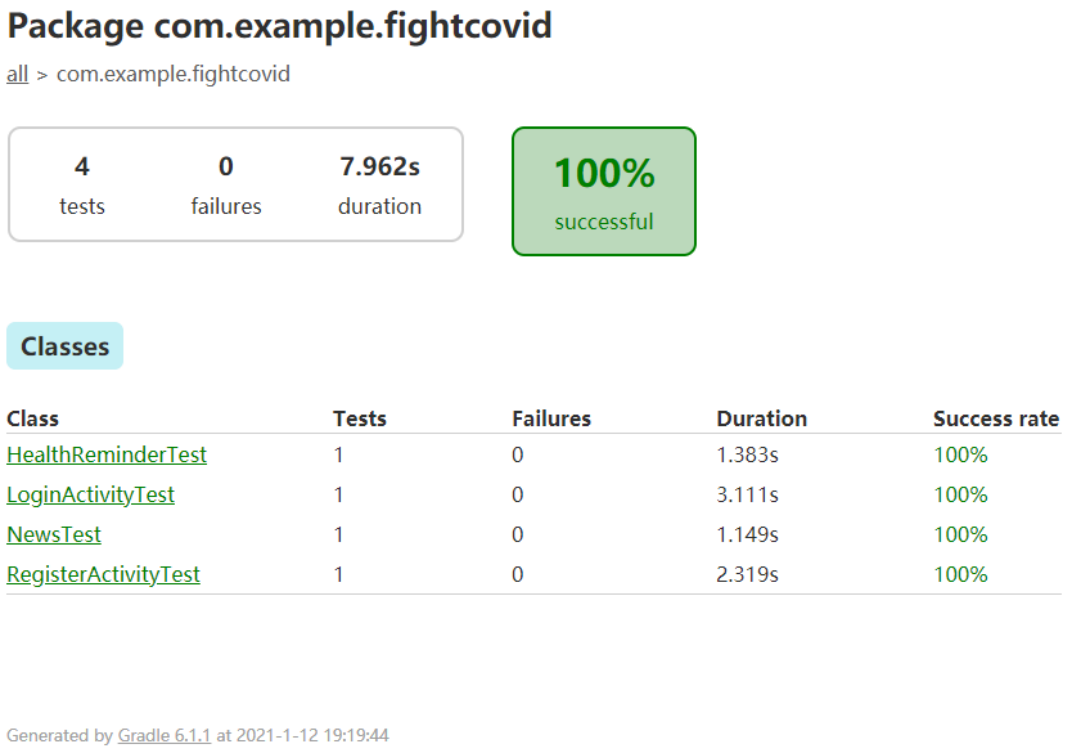
The testing for health reminder is to verify whether the button works fine or not. The final result matches the expected result where the page switch is successful.

```
@RunWith(AndroidJUnit4.class)
public class HealthReminderTest {
    @Rule
    public ActivityTestRule<HealthReminder> rule = new ActivityTestRule<>(HealthReminder.class);
    @Test
    public void JumpTest() {
        onView(ViewMatchers.withId(R.id.btn_reminder_no))
                .perform(click());
    }
}
```

Figure 7. Health Reminder Test

3.2 Test Result

Eventually, we tested all our four tests and then generated the test report automatically by Android Studio which indicates that all of our tests are successfully and take 8 seconds to finished the test.



Package com.example.fightcovid

all > com.example.fightcovid

4 tests	0 failures	7.962s duration	100% successful

Classes

Class	Tests	Failures	Duration	Success rate
HealthReminderTest	1	0	1.383s	100%
LoginActivityTest	1	0	3.111s	100%
NewsTest	1	0	1.149s	100%
RegisterActivityTest	1	0	2.319s	100%

Generated by Gradle 6.1.1 at 2021-1-12 19:19:44

Figure 8. Test Result

REFERENCE

ReactiveX/RxJava. (2021). [Java]. ReactiveX. https://github.com/ReactiveX/RxJava (Original work published 2013)

Square/retrofit. (2021). [Java]. Square. https://github.com/square/retrofit (Original work published 2010)

Marking Rubric

No.		Poor	Adequate	Proficient	Mark
Component 1: Design (Lab Assignment 1)					
1	Navigation design	No Navigation design	Navigation flow which is unreasonable	Navigation design which is reasonable and follow convention	
		0	**1**	**2**	/2
2	UI layout	Poor UI layout design without following good design principles	Reasonably sound UI layout design	Follow good Google Material design principles Good look and feel	
		0-1	**2**	**3**	/3
				Subtotal	**/5**
Component 2: Coding					
1	Lab assignment 2: App pages and navigation	The app pages (at least 5 pages) and navigation are not completed and not functional.	The app pages (at least 5 pages) and navigation are partially completed and functional.	The app pages (at least 5 pages) and navigation are fully completed	
		0-2	**3**	**4-5**	/5
2	Lab assignment 3: Listview	The listview is not completed	The listview is partially completed with dummy data	The listview is fully completed with live/real data	
		0-2	**3**	**4-5**	/5
3	QR code scanner for check in and history listview	The QR code scanner and history listview do not function Poor code quality No comments	The QR code scanner and history listview function partially with dummy data	The QR code scanner and history listview funtions fully with data retrieved from database Good code quality Well commented	
		0 - 3	**4 - 7**	**8 - 10**	/10
4	User profile (including sign up and sign in)	User profile, sign up and sign in pages do not function fully Poor code quality	User profile, sign up and sign in pages function partially	User profile, sign up and sign in pages function fully Good code quality	/5

		No comments		Well commented	
		0-1	**2-3**	**4-5**	
5	Health awareness reminder page Frequently asked questions and answers page	Both of the pages do not function fully Poor code quality No comments	Only one of the pages function fully	Both of the pages function fully Good code quality Well commented	
		0-2	**3-4**	**5**	/5
6	Bonus and/or server-side features	No bonus features and no server-side	1 bonus feature with partial server-side	More than 1 bonus feature with full server-side	
		0	**1 - 8**	**9 - 10**	/10
				Subtotal	**/40**
Component 3: Testing					
1	Test case and test scenario	No test case and test scenario	Test case and test scenario are partially complete and correct.	Test case and test scenario are complete and correct.	/2
		0	**1**	**2**	
2	Test results	No test results	Partially complete test results	Complete test results (with at least functional and usability test)	/3
		0	**1 - 2**	**3**	
				Subtotal	**/5**
Component 4: Project Report					
1	Documentation of Design	Poor design graphics quality Poor writing quality Poor or no formatting	Satisfactory quality design graphics Satisfactory writing quality, grammar and flow Formatted	High quality design graphics and good resolution Good writing quality, grammar and flow Well formatted	/10
		0 - 3	**4 - 7**	**8 - 10**	
2	Documentation of Coding	Poor writing quality Lack of technical detail on coding Poor or no formatting	Satisfactory writing quality, grammar and flow Formatted Demonstrate good	Good writing quality, grammar and flow Well formatted and good presentation	

			technical know-how	Demonstrate excellent technical know-how	/10
		0 - 3	**4 - 7**	**8 - 10**	
3	Documentation of test case, test scenario and test results	Poor writing quality Poor or no formatting / presentation Lack of credible test results	Satisfactory writing quality, grammar and flow Formatted Satisfactory test results and reasoning	Good writing quality, grammar and flow Well formatted and good presentation Demonstrate excellent technical know-how with comprehensive test results and reasoning	
		0 - 3	**4 - 7**	**8 - 10**	/10
				Subtotal	**/30**
Component 5: Presentation and Demonstration					
1	Presentation	Poor quality slides Poor time management Speech that is unclear	Satisfactory quality slides Speech that is satisfactory and understandable	High quality slides Good time management Speech that is clear and impactful	
		0 - 3	**4 - 7**	**8 - 10**	/10
2	Demonstration	Poor demonstration that is unclear The prototype is generally not functioning	Satisfactory demonstration The prototype is generally functioning	Good demonstration The prototype is fully functioning and impactful	
		0 - 3	**4 - 7**	**8 - 10**	/10
				Subtotal	**/20**
				Grand Total	**100**

XIAMEN UNIVERSITY MALAYSIA

Project / Assignment

Course Code: SWE401

Course Name: Embedded Programming

Lecturer: Dr. Simon Lau Boung Yew

Academic Session: 2020/09

Title:

Student ID	Student Name
SWE1709204	LI XIA
SWE1709151	HUANG JIAQI
SWE1709346	SUN RUITAO

	Marks
Component 1 (5%)	
Component 2 (40%)	
Component 3 (5%)	
Component 4 (30%)	
Component 5 (20%)	
Total	/100