COMP826

Mobile System Development

The application of virtual traffic lights on android-based mobile system

Student Name: Duo Tong

Student ID: 19075095

Lecturer: Roopak Sinha and Chandan Sharma

2nd November 2020

1. Overview

1.1 Virtual traffic light application

Virtual traffic light (VTL) application, which is a significant segment of smart city, is considerably promising since it can control the traffic flow as well as support vehicles cross the conjunctions more effectively and safely without the physical traffic signals. The communication latency could cause severe effect such as traffic accidents and even death, hence it is necessary to achieve real-time intersection among vehicles [1]. [8] pointed out that the VTL application should be implemented with the premise of Dedicated Short Range Communications technology (DSRC), which can meet the criteria of low-latency, efficient and high-quality communication. Reversely, the communicating time is longer in the implementation of APIs because of the necessity to transfer (upload and download) data and the possibility of poor network. Besides, instead of achieving the core algorithm by APIs, the integration of UI, key algorithms and database in the mobile client is much easier for an unexperienced developer. As a result, this project doesn't adopt the APIs but integrate all the components into mobile client and assumes that the collection of vehicle information in terms of location, speed and directions can be achieved by DSRC.

The objective of this project is to construct a virtual traffic light application based on Android platform. It invokes the built-in hardware components including GPS, map, sensors and wifi. The core principle of VTL is that the leader who is the nearest vehicle to the intersection must be elected and instruct other vehicles to pass this conjunction within its leading time when there is any collision.

1.2 Project implementation

1.2.1 Main components

Regarding the outstanding performance and ease of development, a single application, which combines UI, the key algorithm and database, is constructed. The details are illustrated as below：

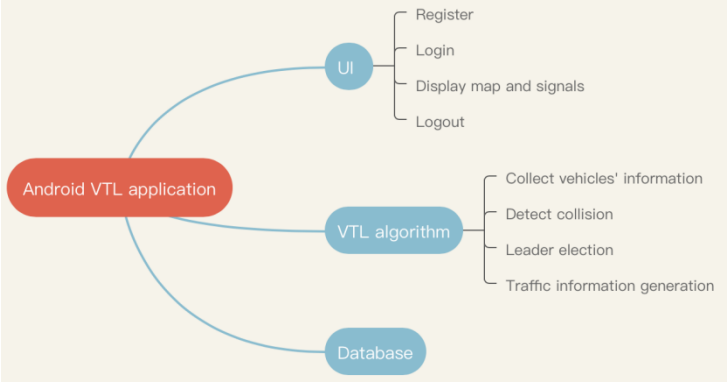


Figure 1.2.1 main components of the application

UI plays a vital part in the project, which is the first sight of the application running on different Android mobile phones and tablets for users. It invokes the Google map and GPS services to obtain the specific location, speed and directions, which is the important parameter in the layout of map and the distance calculation. It involves the function of register, login, data collection, information display and logout.

VTL algorithm. The system first identifies traffic conflict based on the collected data. Then it processes the leader selection and finally the traffic signals are generated by the leader.

Database. In this paper, SQLiteDatabase is applied to warehouse the application data, especially in the scenario of log in and registeration. This part is implemented with minimum effort, which means it can be refactored in the future directions.

1.2.2 Main utilized tools

The option of relevant tools is based on the requirements of the developed system. As mentioned above, the application intends to be implemented in android-based cellphones, hence the main developmental tool is Android studio (version 4.0.1) with java programming language.

2. System artefacts

2.1 Refined design document

File name: Duo_Tong_Assignment1.pdf

Link:https://github.com/MollyTong/VTL_Application/blob/main/Refined%20Design%20Document.pdf

In the amended document, there are mainly three parts refined such as the role in the process of development, the deployment design and evaluation criteria. First and foremost, I focus on system architecture, UI design and testing rather than developing since I have no developing experience, which means an independent task is difficult for me. Another variation is that I do not separate the whole system into mobile client, web server and database but integrate all of them into the client, as it is easier for an unexperienced developer and the communication is much more efficient. Eventually, the number of evaluation criteria is inadequate in the previous version excluding the boundary scenarios, performance and compatibility assessment. In other words, for this project, it is necessary to extend the evaluation conditions, which is demonstrated more specifically in section3.

2.2 Software requirements specification

2.2.1 Introduction

Software requirements specification refers to the description of an application system that is to be constructed. SRS can be considered as a basis for the agreement between suppliers and customers on the features of the product [12]. It is also an evaluation of requirements before system design phase in more details. SRS aims to reduce redesign, provide the basis for expenditure estimation, risks and even schedules by listing sufficient, specific and necessary requirements for the project [9]. It is considerably useful for the workers who devote to the product. If this document is utilized appropriately, it can effectively avoid the failure of the project. SRS usually consists of functional and non-functional requirements [12]:

Functional requirements. It defines the function of an application or its components in which the function refers to the behavior between inputs and outputs.

Non-functional requirements (NFRs). It can be defined as the criteria that specify the quality of the operation of a system, involving security, reliability, performance, maintainability, compatibility, localization and usability. Additionally, they are the attributes which considerably affects the functionality of the system and user experience. Hence it is significant to acquire NFRs accurately to

ensure the sustainability of functioning well. The definition of appropriate NFRs plays an important role since they are contributing factors of project achievement. Under-specifying NFRs will result in an insufficient system, whereas over specifying will put massive pressure on the viability and price of the system. In this project, SRS can be separated into four segments including introduction, users, functional and non-functional requirements, which can be seen in detail in the following link.

2.2.2 Deliverables

File name: Requirement specification for VTL.docx

Link:https://github.com/MollyTong/VTL_Application/blob/main/Requirement%20Specification%20for%20VTL.pdf

2.3 Artefacts of mobile app for Android

The artefacts in this fraction can be divided into three parts involving the overall structure of the mobile application, the approach to achieving MVVM design pattern and the implementation of the VTL algorithm. The programming code can be found in the following website.

The link: https://github.com/MollyTong/VTL_Application

2.3.1 Structure

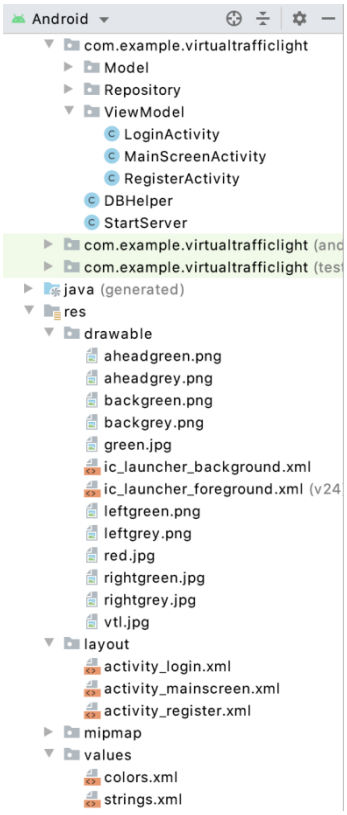


Figure 2.3.1.1 the structure of front-end

There are six main folders utilized when designing the UI presentation in Android Studio including drawable, layout and values in the path of 'res', the activity classes, AndroidManifest.xml and build.Gradle (Module:app) file.

- **Drawable** is used to place the related pictures, which are necessary in the display of UI.
- **Layout** composites of different files programming with Extensible Markup Language (XML). It defines all the components or views in the display of pages. To ensure the compatibility in different size screen, the UI design in this application applies the constraint layout with guidelines and wrapping content.
- **Values** is used to define the content of each component, which assists to remain their independence and achieve the support of various languages.
- **Activity classes** defines the corresponding reaction of components such as jumping to the main screen after clicking log in button.
- **AndroidManifest.xml** declares all the activities.
- **The buid.Gradle file** is utilized to state all the dependencies in terms of constraint layout and the implementation of google map.

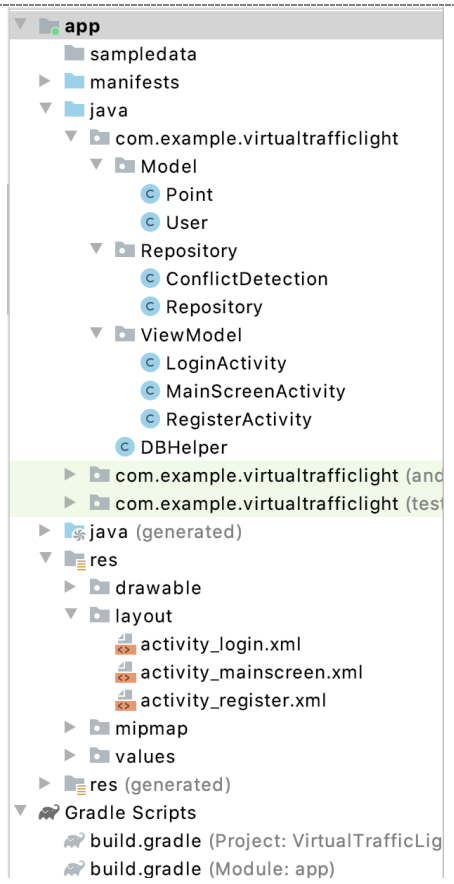


Figure 2.3.1.2 the overall structure

Architectural pattern

This mobile application applies the architectural pattern called Model-View-View Model (MVVM). It is beneficial to decouple the data and view due to its feature of two-way data binding, which means the page will be automatically updated as long as the data model is changed [13].

- **View** also named layout describes the components in UI. In this project, only three pages are defined including registering, log in and main screen.
- **View model** functions as the bridge between view and repository, which is used to acquire the data from view, transfer information to repository and define the navigational logic to the next page.
- **Repository** is used to achieve the core algorithm in the VTL application including collision detection, repository (leader election and signal generation). It is the medium to connect the view model and model.
- **Model** is utilized to define the information of vehicles in terms of the speed and location and user data including email and password.
- **DB helper** establishes the user table in the database by applying SQLiteDatabse.

2.3.2 The screenshots of the achivement of MVVM

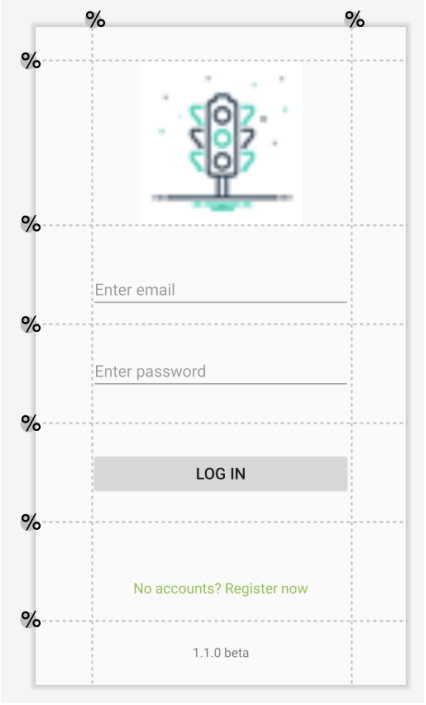


Figure 2.3.2.1 constraint layout

```
<androidx.constraintlayout.widget.Guideline
    android:id="@+id/guideline8"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    app:layout_constraintGuide_percent="0.85" />

<ImageView
    android:id="@+id/imageView"
    android:layout_width="0dp"
    android:layout_height="0dp"
    app:layout_constraintBottom_toTopOf="@+id/guideline2"
    app:layout_constraintEnd_toStartOf="@+id/guideline8"
    app:layout_constraintStart_toStartOf="@+id/guideline7"
    app:layout_constraintTop_toTopOf="@+id/guideline"
    app:layout_constraintVertical_bias="0.469"
    app:srcCompat="@drawable/vtl"
    android:contentDescription="TODO" />

<EditText
    android:id="@+id/editTextEmail"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:autofillHints=""
    android:ems="10"
    android:hint="Enter email"
    android:inputType="textEmailAddress"
    app:layout_constraintBottom_toTopOf="@+id/guideline4"
    app:layout_constraintEnd_toStartOf="@+id/guideline8"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintStart_toStartOf="@+id/guideline7"
    app:layout_constraintTop_toTopOf="@+id/guideline2"
    app:layout_constraintVertical_bias="0.769" />
```

Figure 2.3.2.2 Activity

```
public class LoginActivity extends AppCompatActivity {
    private EditText emailEditText,passwordEditText;
    private TextView registerText;
    private Button loginButton;
    DBHelper db;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        loginButton = findViewById(R.id.Login);
        registerText = findViewById(R.id.registertextView);
        db = new DBHelper( context: this);
        loginButton.setOnClickListener((v) → {
                emailEditText = (EditText) findViewById(R.id.editTextEmail);
                passwordEditText = (EditText) findViewById(R.id.editTextPassword);
                String uemail = emailEditText.getText().toString();
                String uPassword = passwordEditText.getText().toString();
                System.out.print("onclick");
                if (uemail.equals("")) {
                    displayToastMessages("Enter email").show();
                } else if (uPassword.isEmpty()) {
                    displayToastMessages("Enter password").show();
                } else {
                    boolean checkuser = db.checkuser(uemail,uPassword);
                    if(checkuser==true){
                        displayToastMessages("Log in successfully").show();
                        Intent mainIntent = new Intent( packageContext: LoginActivity.this, MainScre
                        startActivity(mainIntent);
                    } else {
                        displayToastMessages("Please register").show();
                    }
                }
        });
        registerText.setOnClickListener((v) → {
                Intent register = new Intent( packageContext: LoginActivity.this,RegisterActivity.cl
                startActivity(register);
        });
    }
```

Figure 2.3.2.3 view model

```
public class ConflictDetection {
    private static final  double EARTH_RADIUS = 6378137;
    private static final double toDistance = 500;

    public static boolean checkRange(Point point, Point intersection){
        double distance = getDistance(point, intersection);
        return distance <= toDistance;
    }

    public static double getDistance(Point point, Point intersection){

        double lon1 = point.getLongitude();
        double lat1 = point.getLatitude();
        double lon2 = intersection.getLongitude();
        double lat2 = intersection.getLatitude();
        double radLat1 = Math.toRadians(lat1);
        double radLat2 = Math.toRadians(lat2);
        double a = radLat1 - radLat2;
        double b = Math.toRadians(lon1-lon2);
        double s = 2 * Math.asin(Math.sqrt(Math.pow(Math.sin(a / 2), 2)
                + Math.cos(radLat1) * Math.cos(radLat2)
                * Math.pow(Math.sin(b / 2), 2)));
        s = s * EARTH_RADIUS;
        s = (double)Math.round(s * 10000) / 10000;
        s = s * 10000/ 10000;
        return s * 1000d;
    }

    public static double getbearing(Point point, Point intersection){
        double lat1 = point.getLatitude();
        double lon1 = point.getLongitude();
        double lat2 = intersection.getLatitude();
        double lon2 = intersection.getLongitude();
        double longitude1 = lon1;
        double longitude2 = lon2;
        double latitude1 = Math.toRadians(lat1);
        double latitude2 = Math.toRadians(lat2);
```

Figure 2.3.2.4 repository-conflict detection

```java
package com.example.virtualtrafficlight.Model;

public class User {
    private String email, password;

    public User() {
    }

    public User(String email, String password) {
        this.email = email;
        this.password = password;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

Figure 2.3.2.5 Model of user

```java
public class DBHelper extends SQLiteOpenHelper {

    public static final String DATABASE_NAME = "user.db";
    public static final String TABLE_NAME = "user_table";

    public DBHelper(@Nullable Context context){
        super(context, DATABASE_NAME, factory: null, version: 1);
        SQLiteDatabase db = this.getWritableDatabase();
    }

    public void onCreate(SQLiteDatabase db){
        db.execSQL("CREATE TABLE " + TABLE_NAME + "(email TEXT PRIMARY KEY,password TEXT)");
    }

    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion){
        db.execSQL("DROP TABLE IF EXISTS " + TABLE_NAME);
    }

    public boolean insert(String email,String password){
        SQLiteDatabase db = this.getWritableDatabase();
        ContentValues contentValues = new ContentValues();
        contentValues.put("email",email);
        contentValues.put("password",password);
        long ins = db.insert(TABLE_NAME, nullColumnHack: null,contentValues);
        if (ins == -1) return false;
        else return true;
    }

    public boolean chkemail(String email){
        SQLiteDatabase db = this.getWritableDatabase();
        Cursor cursor = db.rawQuery( sql: "select * from " + TABLE_NAME + " where email=?",new St
        if(cursor.getCount()>0) return false;
        else return true;
    }

    public boolean checkuser(String email,String password){
        SQLiteDatabase db = this.getWritableDatabase();
```

Figure 2.3.2.6 DB Helper

2.3.3 VTL algorithm

The VTL algorithm is implemented in the repository class. In this project, the algorithm is achieved with the premise of DSRC to acquire information of other vehicles and the defaulting location of intersection. The principle and implementation of VTL algorithm are depicted as Figure 2.3.3.1 and Figure 2.3.3.2, respectively.

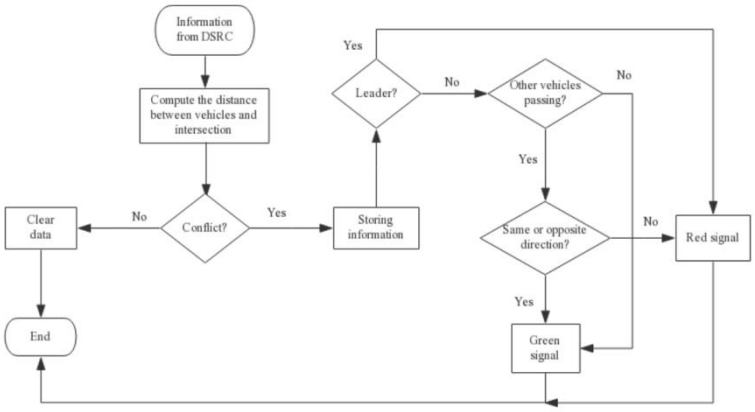
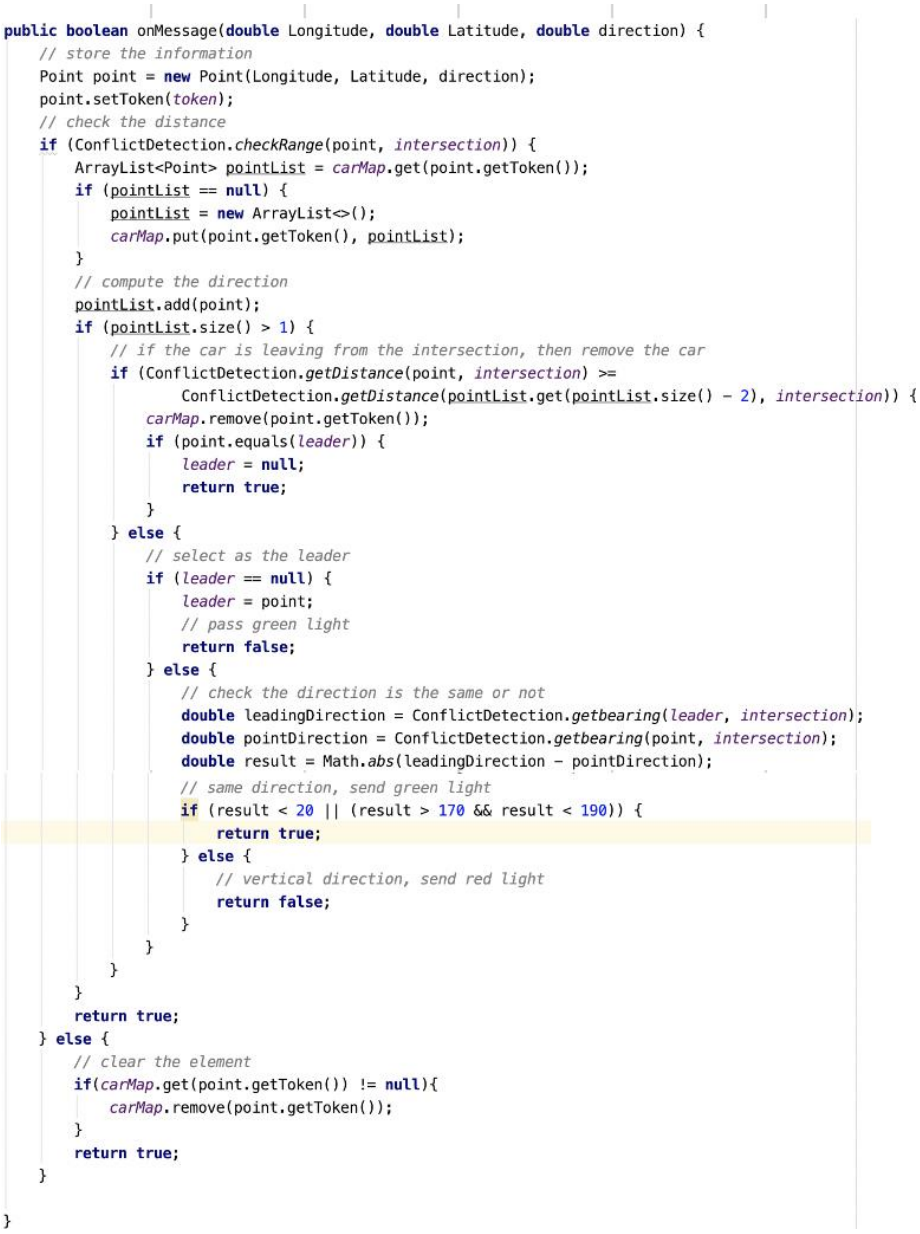


Figure 2.3.3.1 the workflow of VTL algorithm



```
public boolean onMessage(double Longitude, double Latitude, double direction) {
    // store the information
    Point point = new Point(Longitude, Latitude, direction);
    point.setToken(token);
    // check the distance
    if (ConflictDetection.checkRange(point, intersection)) {
        ArrayList<Point> pointList = carMap.get(point.getToken());
        if (pointList == null) {
            pointList = new ArrayList<>();
            carMap.put(point.getToken(), pointList);
        }
        // compute the direction
        pointList.add(point);
        if (pointList.size() > 1) {
            // if the car is leaving from the intersection, then remove the car
            if (ConflictDetection.getDistance(point, intersection) >=
                    ConflictDetection.getDistance(pointList.get(pointList.size() - 2), intersection)) {
                carMap.remove(point.getToken());
                if (point.equals(leader)) {
                    leader = null;
                    return true;
                }
            } else {
                // select as the leader
                if (leader == null) {
                    leader = point;
                    // pass green light
                    return false;
                } else {
                    // check the direction is the same or not
                    double leadingDirection = ConflictDetection.getbearing(leader, intersection);
                    double pointDirection = ConflictDetection.getbearing(point, intersection);
                    double result = Math.abs(leadingDirection - pointDirection);
                    // same direction, send green light
                    if (result < 20 || (result > 170 && result < 190)) {
                        return true;
                    } else {
                        // vertical direction, send red light
                        return false;
                    }
                }
            }
        }
        return true;
    } else {
        // clear the element
        if(carMap.get(point.getToken()) != null){
            carMap.remove(point.getToken());
        }
        return true;
    }

}
```

Figure 2.3.3.2 the code of VTL algorithm (repository)

3. System evaluation

System evaluation refers to the process of assessing the consistency between the product and its functional and non-functional requirements through testing activities, which aims to ensure the software quality and measure the standard [5]. The assessment of mobile application requires to take a variety of perspectives into consideration including functionality, usability, compatibility, security, performance and network testing [7]. The details are demonstrated as below:

Testing types	Executor	Specification	Objectives	Executing time	Opacity
Unit testing	Developer	By running script to ensure the test from failure to pass	To check the structure of app code	When writing code	White box testing
Fuctional testing	Developer	Verify the context, which is displayed on the mobile devices	To check the functionality consistent with requirement or not	During and after development stage	White/Black box testing
Usability testing	Client, users	Refers to the effectiveness, efficiency and satisfication	To check the ease of using the app for users	After completing the functionality testing	Black box testing
Compatibility testing	Tester	The ability to be used in different operation systems and devices	To validate the compability	After development and before deliverable	White/Black box testing
Security testing	Developer	To protect the sensitive information such as ID and password	To guarantee the information preservation	At the end of development	White/Black box testing
Performance testing	Tester	Relates to the speed, fluency and responsiveness (loading and stress testing)	To ensure the effciency	From the beginning of the development to deployment	Black box testing
Acceptance testing	Client	To evaluate the app from the user viewpoints	To check the app consistent with requirements/specification or not	When the user acceptance criteria met with the requirements	White/Black box testing
Network testing	Developer	The ability to be used in different network (wifi, 4G and 5G)	To check the strength/weakness of apps' connection	Before deliverable	White box testing

Figure 3 The introduction of testing types

In the business version of any mobile application, it is indispensable to deploy all of the above testing to comprehensively ensure the software quality [6]. However, due to the time limitation, only acceptance testing is performed, which will be introduced in section3.2.

3.1 Refined evaluation criteria

File name: Duo_Tong_Assignment1.pdf

Link:https://github.com/MollyTong/VTL_Application/blob/main/Refined%20Design%20Document.pdf

Compared to the old version, the refined evaluation criteria are extended from eight to thirteen, which includes the assessment of registering, login, compatibility and performance. In addition, the abnormal and boundary scenarios are covered in the following acceptance testing, which is presented in Figure 3.1.2.

3.2 Acceptance testing

Acceptance testing always behaves the interests of customers, which strengthens their confidence that the application has the needed features and they respond correctly [10]. Distinguishing from unit testing, acceptance testing is written by customers to inspect whether the system behaves accurately. For

instance, when a user interface is constructed to display the products in an order, an acceptance testing must include the remove of items and check its response. Acceptance testing functions as a “contract” between programmers and customers [9]. It is vital to prove the consistency with the contract by the preservation of those tests, frequent execution and the continuous amendment with the changes of requirements. Its devotion for the developing teams can be listed as follows:

- It captures user requirements and evaluates the consistency with requirements.
- It exposes issues, which might be missed in unit tests.
- It defines the standards of a finished system.

As a consequence, not only can acceptance testing continuously capture the requirements as the system evolve, but also can effectively validate them in a direct way.

3.2.1 How to conduct acceptance testing

Premise:

- Requirements must be available and explicit.
- Both system and regression testing have been completed and signed off.
- Acceptance test bed has been clarified as well as the high-level check should be confirmed.

Procedure:

The process to conduct an acceptance testing can be divided into six phases in a sequence, including requirements analysis, design test plan, design acceptance test, the definition of test bed, the construction of test data and execution [10].

3.2.2 The execution of acceptance testing

Testing environment:

- Laptop (Macbook Pro with the system version of 10.15.7)
- Android Studio (v4.0.1)
- Simulator (Pixel 2 API 29 and 3.4 WQVGA API 23 with wifi)

Test cases and outcomes:

Test Module	Case ID	Preconditions	Description	Expected Result	Actual Result	Status
Register	register_01	Have accounts	Input an existing account information to register	Can't register	Can't register	Pass
	register_02	No accounts	Step1: nothing is input and click "register" E2: any input box isn't be entered then click "register" E3: the two passwords are different then click "register"	Can't register	Can't register	Pass
	register_03	No accounts	Step1: input name Step2: input information not an email and enter password Step3: click "register"	Can't register	Can't register	Pass
	register_04	No accounts	Step1: input name and email Step2: input password combined with number, characters and special signals Step3: click "register"	Can register	Can register	Pass
Log In	Login_01	No accounts	Input a unregistered account and click "login"	Can't log in	Can't log in	Pass
	Login_02	Have accounts	Input a registered account (email or password) incorrectly and click "login"	Can't log in	Can't log in	Pass
	Login_03	Have accounts	Step1: nothing is input and click "login" E2: only input email then click "login" E3: only input password then click "login"	Can't log in	Can't log in	Pass
	Login_04	Have accounts	Input a registered account correctly and click "login"	Transfer to the main screen	Transfer to the main screen	Pass
VTL information generation	VTL_01	No intersection ahead	There is no conjunction within 500 meters	No signals	No signals	Pass
	VTL_02	Have intersection ahead	There is a conjunction within 501 meters	No signals	No signals	Pass
	VTL_03	Have intersection ahead and no conflict	There is a conjunction within 499 meters	No signals	No signals	Pass
	VTL_04	Have intersection ahead and with conflict	There is a conjunction within 499 meters and there is any other car in the different direction except the opposite	Red light	Red light	Pass
	VTL_05	Have intersection ahead but all the cars are in the same and opposite direction	Have intersection ahead within 500 meters but all the cars are in the same and opposite direction	No signals	No signals	Pass
	VTL_06	Have intersection ahead with collision	Have intersection ahead within 500 meters and there is any other car in the different direction except the opposite	Red light	Red light	Pass
Leader	Leader_01	Leadership within two minutes and no more conflicts	There is no vehicles in the different lanes apart from the oppostie direction	Green light	Green light	Pass
	Leader_02	The time of leadership have exactly reached to two minutes and conflict detected	The time of leadership have exactly reached to two minutes and conflict detected	the light of all the vehicle in the same and opposite lane with the leader turn into green,while others change to red	the light of all the vehicle in the same and opposite lane with the leader turn into green,while others change to red	Pass
	Leader_03	Continue to Leader_02	After two minutes there are some vehicles with green have not passed, check the vehicle status with red light.	The green light turns to red but red transfer to green	The green light turns to red but red transfer to green	Pass

Figure 3.2.2.1 test cases and its outcomes

Note: 500 meters and two minutes are defined as the boundary in SRS

The screenshots of partial testing results:

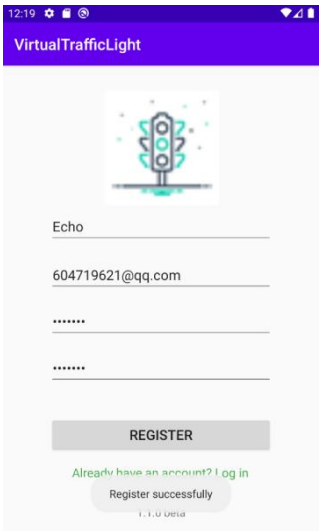


Figure 3.2.2.2 register

Figure 3.2.2.3 login

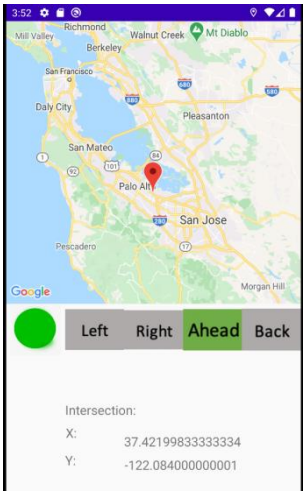


Figure 3.2.2.4 green light without collision

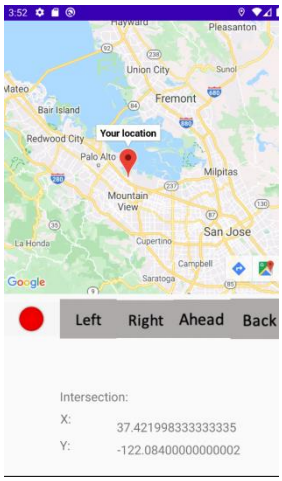


Figure 3.2.2.5 red light with collision

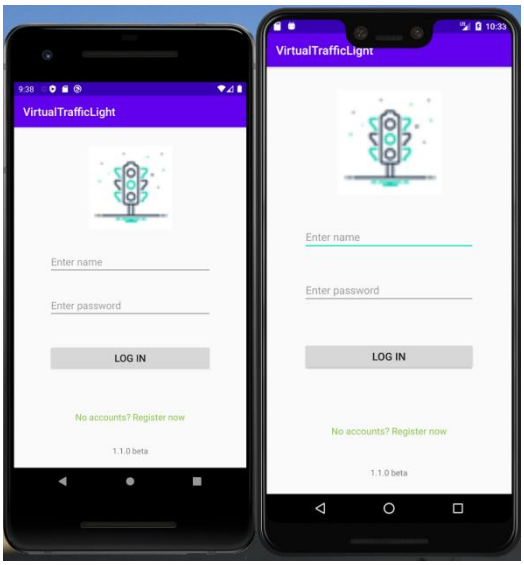


Figure 3.2.2.6 compatibility in different size screen

3.2.3 Evaluation analysis

In summary, the evaluation is performed twice. Firstly, the testing is executed by the developer during the procedure of development to verify the achievement of a specific function. Then, the acceptance testing is implemented by the tester in order to ensure the quality of the application in terms of the accordance with the software requirements.

During the process of the development, there are totally three bugs fixed. The first one is the failure of displaying on various size screen, which means the presentation of layout is disordered such as overlap and omission. By utilizing the guidelines and wrapping the content of each component in the design of constraint layout, it is finally addressed. The further issue is that the status of the signal is always changing and flashing between green and red, which results from the default value of 'min_dis' in 'LocationManager'. It is preseted as zero at the beginning, which means the function 'onLactionChanged' will be triggered in each second even though the distance is not updated. The corresponding solution is modify the default value to ten. The final bug is related to the logic in core algorithm where the instructive information is green light when a conflict is detected. This severe bug is caused by the wrong logic and eventually solved in the second modification. After that, the regression testing is performed with the same test cases and all of them are executed successfully.

The evaluation only covers the acceptance and compatibility testing. However, this is obviously insufficient to ensure the software quality. First and foremost, the performance should be considered and assessed in the VTL project since poor performance could engender traffic congestion, accidents and even death. This project, nonetheless, fails to cover the performance evaluation due to lack of time. Besides, the abnormal cases are inadequate, either. For instance, the disconnection of network is excluded in the testing cases. Consequently, it is necessary to cover more cases in the future to guarantee the quality of the application.

4. Recommendations

4.1 Process

In terms of developing an application, the selection of the development process is one of the most important stages. On the milestone1, scrum is opted to be the optimal one for the VTL project, which includes the analysis of requirements, system design, development, testing and deployment. In each stage, there are corresponding deliverables utilized to measure the product in terms of requirement specifications, design patterns, code review, testing cases and evaluation reports.

During the development process, the most significant benefit is the ability to prioritize the importance of tasks and distribute them into different sprints [11]. Because of the time constraint, only the minimum viable product was constructed including login and the process of VTL information generation. First and foremost, the UI pages are generated since they are the direct interface presenting to users. The design of database is the second priority due to it being indispensable part of storing user information. Finally, the core algorithm from intersection identification to the generation of signal information is programmed. The prioritized order and scheduled timeline enable the development process more effectively and efficiently.

In principle, scrum has a number of strengths such as higher quality, excellent productivity, the ability to embrace changes and better user satisfaction [11]. However, in reality, since there is only one person responsible for all roles during development, its advantages are not apparent and prominent. For instance, it is difficult to achieve the proximity to users (user satisfaction) as the same individual takes charge of the requirement analysis, design, coding and even testing, which means there is no interaction

with users or team members and no opinion collection in this project. As a consequence, one of the weakness during the development of VTL is the feature of individual project. If the requirement is ambiguous and implicit, there is no opportunity to clarify it by consulting with others. Reversely, it will be time-effective when the requirement is clear. Additionally, the user feedback that is one of the most important characteristics in scrum is not obtained in this project because of the restriction of time and no user participation. It could affect user satisfaction to some extent, as well.

In the future direction, the participation of users and the communication among related roles will be taken into consideration in a team project. It is beneficial to produce a high-quality and satisfying product. Nonetheless, in an individual work, the requirement clarification should be emphasized since the initial explicit requirement will avoid repeat work and significantly save time. In addition, the prioritization and allocation of tasks into each sprint also play a key role in development, which is similar to the process in this project.

4.2 Technology

In this project, a native platform named Android studio is applied. The layout is easy to be constructed with this technology because of its capability to directly drag the UI components into a view page, which significantly provides convenience for front-end developers. Besides, it can invoke all the functionality of local devices in the easiest approach such as GPS and sensor, which are necessary components in this project [2]. In addition, the MVVM design pattern is employed in the project of VTL mobile application in order to remain the consistency with data, low coupling and independent development among the design of UI, data and business logic. The developer only needs to concentrate on the main business logic without the consideration of data processing in MVVM.

Nevertheless, there are still several constraints in the adopted technology. The first one is the developed application can be only deployed in the terminals based on Android platforms, which is not conductive to code reuse. The second shortcoming is that it is difficult to apply the customed framework in Android Studio. Another weakness is the difficulty to locate bugs in MVVM development pattern as data binding results in the transmission of bugs [13]. The further disadvantage is that data binding in two-directions is not beneficial to code reuse where each view is bound to a model. Furthermore,

MVVM could cause more memory for calls without releasing memory even if it is convenient to utilize and easy to ensure the accordance of data [4].

Although the applied technology and selected design pattern have a number of limitations, their strengths apparently outperform the shortcomings because of the superiority of performance and the importance of low coupling in this project [4]. Therefore, I will employ the same technique for the VTL application in the future to achieve the goal of performance and the independence of development. However, there are several parts required to be promoted in the future projects. First and foremost, the concept of development based on components should be implemented to further achieve the lowing coupling among different modules, which is one of the main contributing factors of the success for an application. Moreover, the continuous releasing memory should also be taken into consideration to reduce the utilization of memory so that the performance can be significantly enhanced.

5. References

[1] A. Bazzi, A. Zanella, B. M. Masini, and G. Pasolini, “A distributed algorithm for virtual traffic lights with IEEE 802.11 p,” In 2014 European Conferenceon Networks and Communications, pp. 1–5, 2014.

[2] B. Eisenman, “Learning react native: Building native mobile apps with JavaScript,” O’Reilly Media, 2015.

[3] D. Franke, and C. Weise, “Providing a software quality framework for testing of mobile applications,” In 2011 fourth IEEE international conference on software testing, pp. 431–434, 2011.

[4] G. Medina, J. Menéndez, and J. Vallejo, “Comparative Study of Performance and Productivity of MVC and MVVM design patterns,” KnE Engineering, pp. 241–252, 2018.

[5] J. Gao, X. Bai, W. T. Tsai, and T. Uehara, “Mobile application testing: a tutorial,” Computer, vol. 47, pp. 46–55, 2014.

[6] K. Holl, and F. Elberzhager, “Mobile Application Quality Assurance: Reading Scenarios as Inspection and Testing Support,” In 2016 42th Euromicro Conference on Software Engineering and Advanced Applications, pp. 245–249, 2016.

[7] K. S. Arif, and U. Ali, “Mobile Application testing tools and their challenges: A comparative study,” Mathematics and Engineering Technologies, pp.1–6, 2019.

[8] M. Nakamurakare, W. Viriyasitavat, and O. K. Tonguz, “A prototype of virtual traffic lights on android-based smartphones,” In 2013 IEEE InternationalConference on Sensing, Communications and Networking, pp. 236–238, 2013.

[9] P. Hsia, D. Kung, and C. Sell, “Software requirements and acceptance testing,” Annals of software Engineering, vol. 3, pp. 291–317, 1997.

[10] R. Miller, and C. T. Collins, “Acceptance testing,” XPUniverse, p. 238, 2001.

[11] T. Dingsyr, S. Nerur, V. Balijepally, and N. B. Moe, “A decade of agile methodologies: Towards explaining agile software development,” 2012.

[12] T. Hovorushchenko, “Information technology for assurance of veracity of quality information in the software requirements specification,” In Conference on Computer Science and Information Technologies, pp. 166–185, 2017.

[13] W. Sun, H. Chen, and W. Yu, “The Exploration and Practice of MVVM Pattern on Android Platform,” Materials and Information Technology Applications, 2017.