Emergency Exit App incorporated with Arduino.

Final Report (semester 2)

Kacper Woloszyn

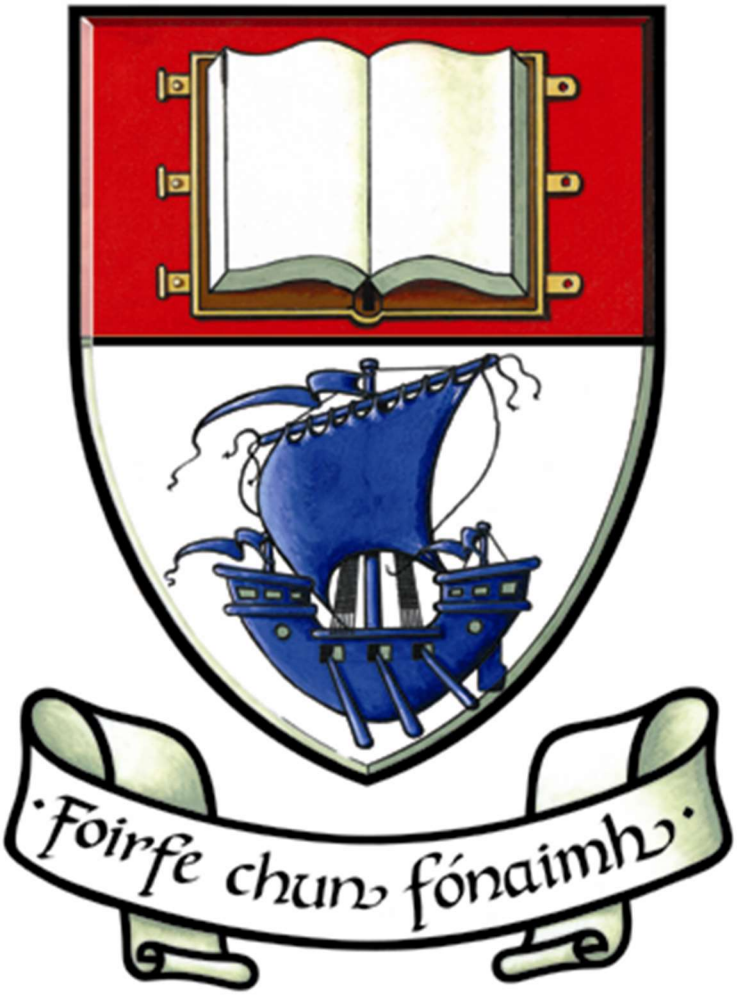
20071494

Supervisor: Caroline Cahill

Second Reader: Lucy White

BSc (Hons) in Applied Computing

Emergency Exit Mobile Application with Arduino



By Kacper Woloszyn, 20071494.

Contents

Introduction

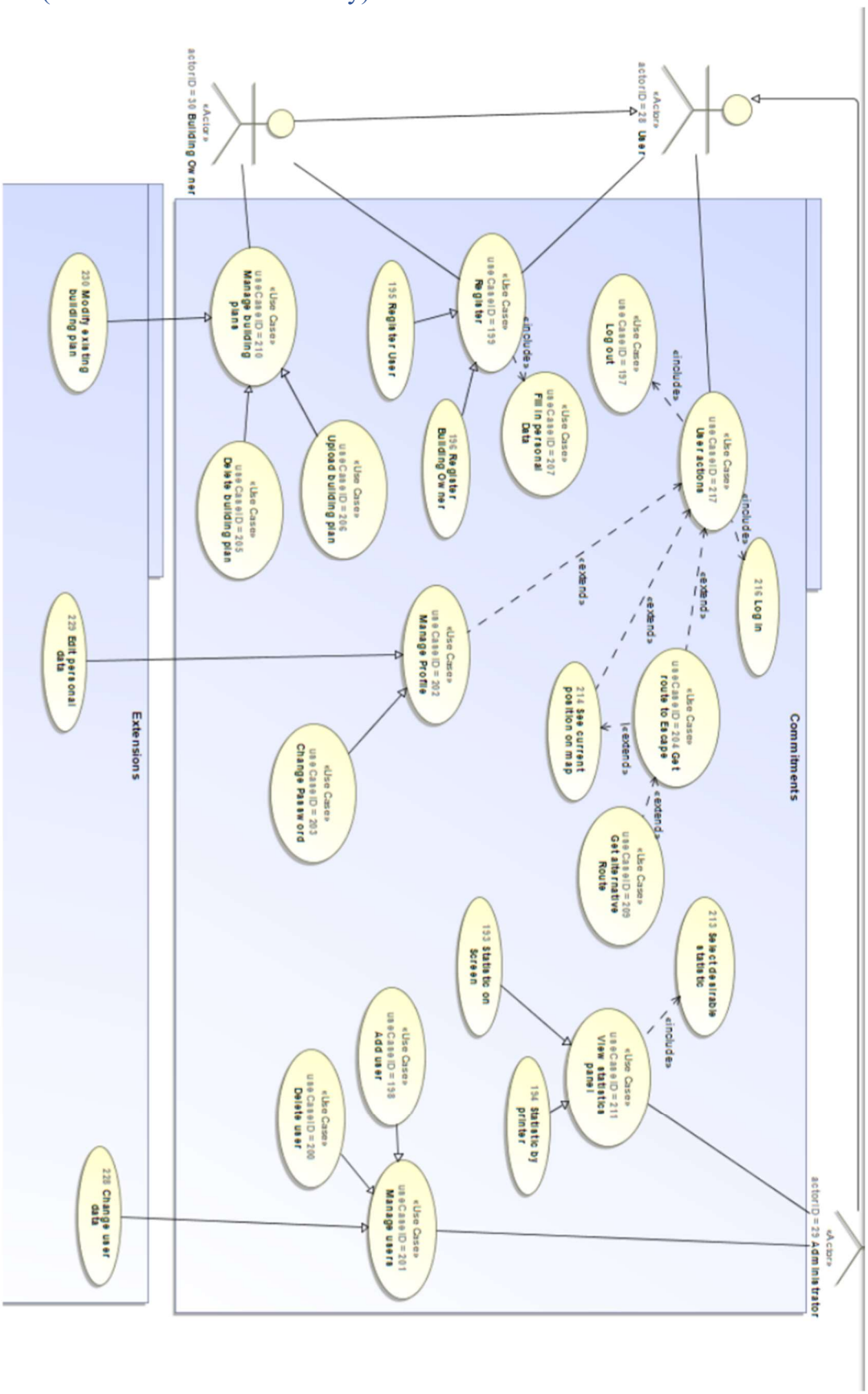
I have decided to go with a mobile application for my final year project. I have developed as a student educationally over the four years in Waterford Institute of Technology and at Universidad de Malaga, to finally be able to put my own project idea combining all the things that I have studied, and the things I would like to learn, not necessarily all the things that I had learned at WIT, but also individually.

I feel the mobile application I did for my final year project may be very important for future generations as each of us nowadays does things on our phone more times than not. I feel there is a need in the market for an application that displays emergency exits to people in public areas and buildings in case of an emergency.

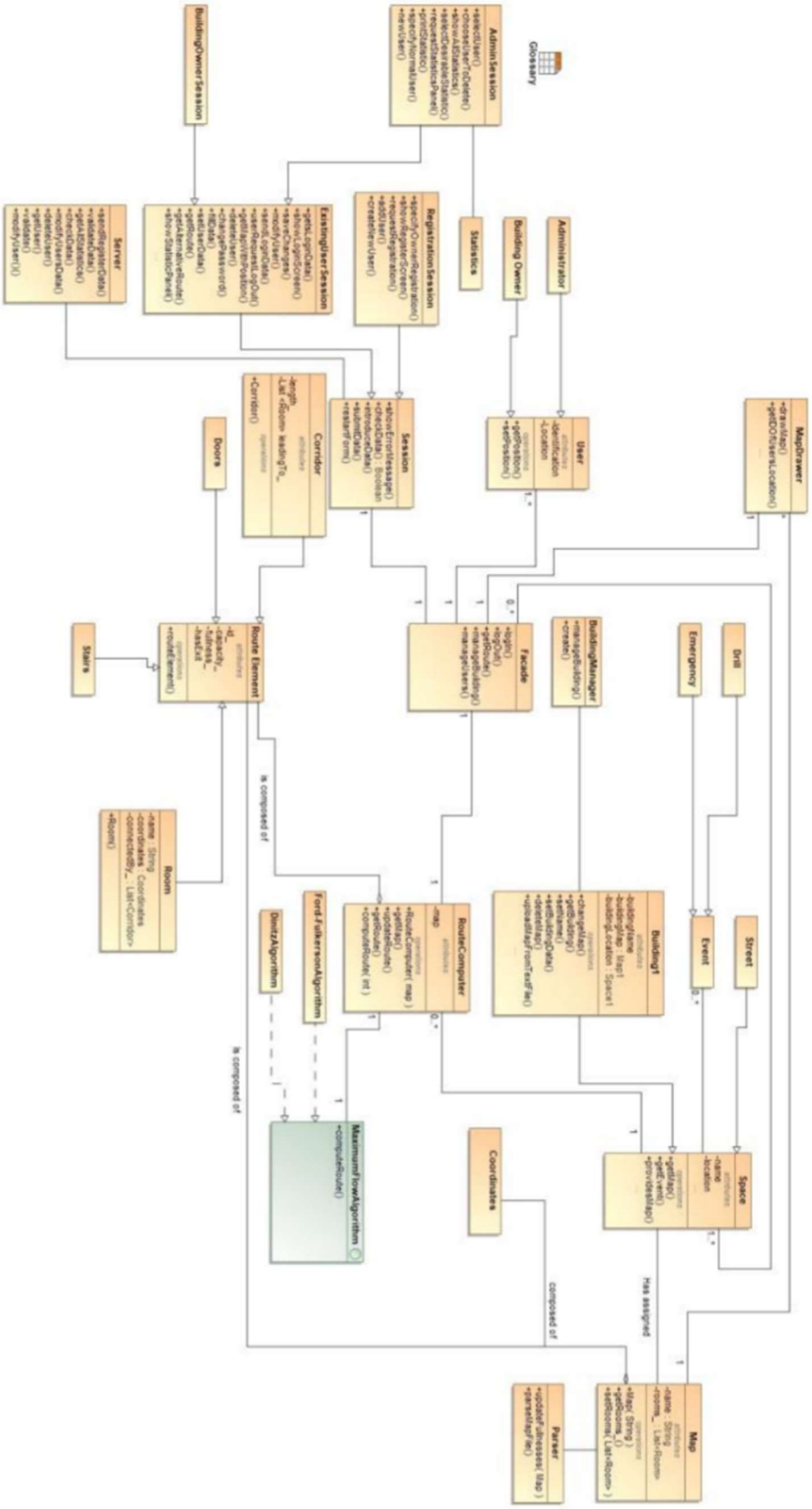
In the project I use an Arduino, the Arduino will be connected to my application and I programmed a buzzer, a sound sensor and the connectivity between Bluetooth in Arduino. The buzzer will sound, and an emergency exit route is displayed on the screen, with some additional information, such as the path length or the time taken to find the route. The application is for Android users, but as the projects developed in the future it could be brought to IOS devices. The application has a database of users that is stored in firebase so a connection to the internet would be a requirement. An algorithm is used in the background to find the quickest route out of the building. The algorithm used is called Dijkstra's. It works on having the start node and the end node and It chooses the unvisited point with the lowest distance, calculates the distance value through it to each unvisited neighbour, and then updates the neighbour's distance if smaller (Wikipedia,2019). I used the knowledge from my Erasmus course in Intelligent Systems, where I learned how to programme route finding algorithms such as A* Algorithms and Dijkstra algorithm or genetic algorithms and neural networks. I take specific measures to make sure the security of the user is not breached, and that the application works as desired, by allowing logins so that I can keep track of who is on the application, and the location is not saved, so just in case someone was trying to steal the data it would be hard.
I have used a tool called magic draw to develop my diagrams which show the functionally of the application.

Diagrams

Use Case (Easier to Read Horizontally)



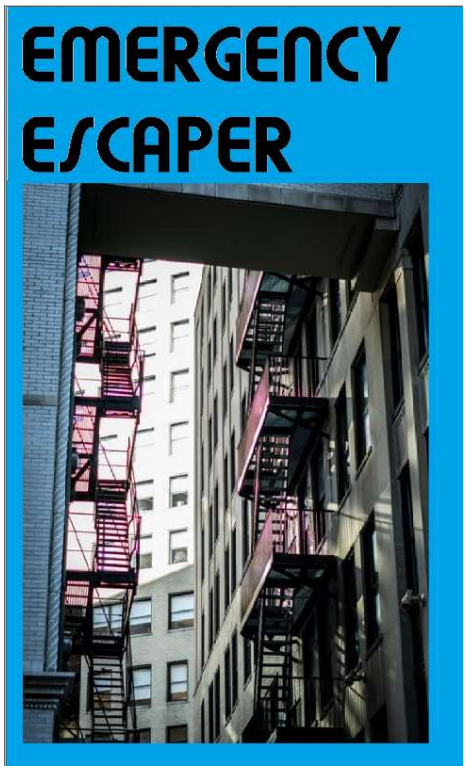
Domain Model



Splash Screen

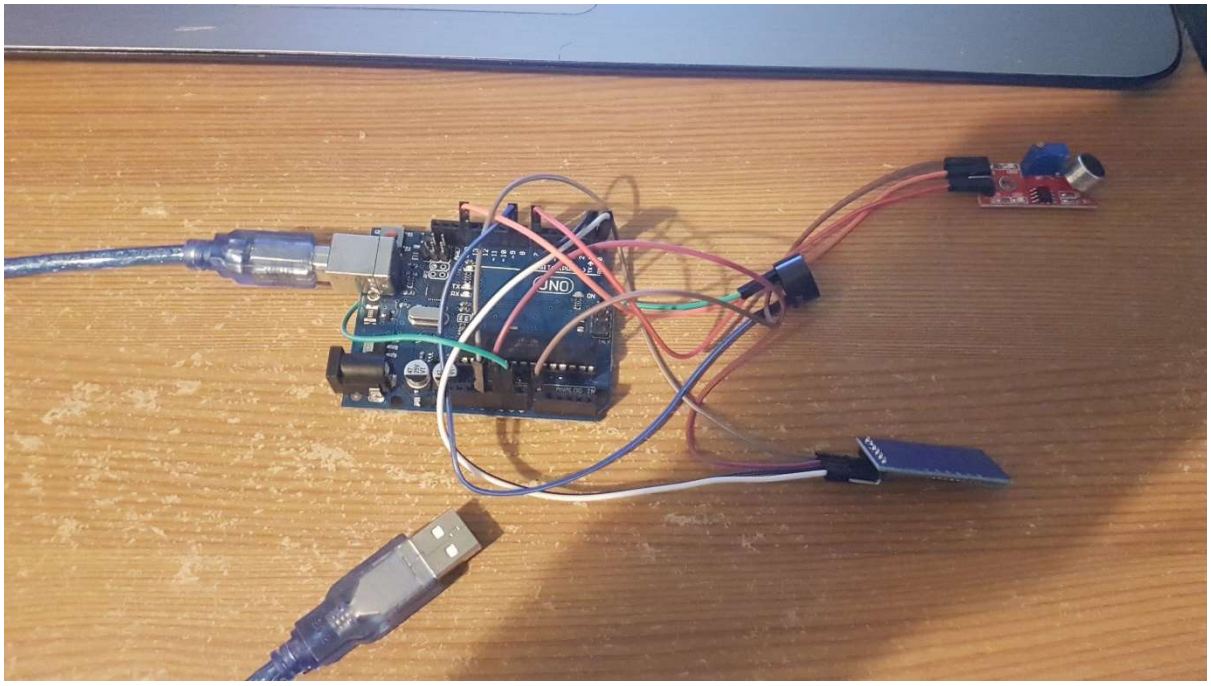
A splash screen is a GUI element consisting of an activity with a containing an image, a logo, and the name of the application in my case. Splash screens have their pros and cons, but since this is a personalised application, I've decided to design my own splash screen. If I were to place this app commercially on the Google Play store or to develop it further, I would not include the splash screen as it steals valuable seconds from a user, to get the results he or she wants. I put in a splash screen, as many commercial applications have one, whilst the application is connecting to the internet. I had developed my own splash screen, and I've set its time to be 3 seconds just for visual purposes (Lifewire,2018).

```
package org.wit.emergencyescape.activities;

import ...

public class SplashScreen extends Activity {

    /** Duration of wait **/
    private final int SPLASH_DISPLAY_LENGTH = 3000; // 3 sec

    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle i) {
        super.onCreate(i);
        setContentView(R.layout.activity_splash);

        new Handler().postDelayed(new Runnable(){
            @Override
            public void run() {
                /* Create an Intent that will start the Menu-Activity. */
                Intent mainIntent = new Intent( packageContext: SplashScreen.this,LogIn.class);
                SplashScreen.this.startActivity(mainIntent);
                SplashScreen.this.finish();
            }
        }, SPLASH_DISPLAY_LENGTH);
    }
}
```



Technology used in my assignment

Arduino



I have an Arduino board with 3 modules I had to obtain separately along with the female-male wires, that did not come with the board. I have connected the Bluetooth module to the 3.5 V entry Ground, RX, and TX as specified in a tutorial online on how to connect and wire up the module. The module I obtained was a cheap one, for this Project, and if I were to do this to a bigger scale, I would get more reliable and more documented hardware, as it was quite a challenge to know what to do with the module once obtained. The buzzer is connected to a digital pin 9, as specified also in the code below and ground. The sound sensor, which is the red item, is connected to ground, Analogue pin 0, as it has to pick up all of the sound ranges and not only 0 and 1's like the buzzer on the digital pin. It also must be connected to a digital pin 7 as there could be a noise signal that was digital perfectly.

```
import ...

public class BluetoothConnector extends Activity implements OnClickListener {

    Button i1;
    TextView t1;
    TextView t2;
    String address = null , name=null;
    BluetoothAdapter myBluetooth = null;
    BluetoothSocket btSocket = null;
    Set<BluetoothDevice> pairedDevices;
    // uuid
    //password for pairing was 123456
    static final UUID myUUID = UUID.fromString("0000FFE7-0000-1000-8000-00805F9B34FB");
```

```
private void bluetooth_connect_device() throws IOException
{
    try
    {
        myBluetooth = BluetoothAdapter.getDefaultAdapter();
        address = myBluetooth.getAddress();
        pairedDevices = myBluetooth.getBondedDevices();
        if (pairedDevices.size()>0)
        {
            for(BluetoothDevice bt : pairedDevices)
            {
                address= bt.getAddress();
                name = bt.getName().toString();
                t1.setText(address);
                t2.setText(name);
                Toast.makeText(getApplicationContext(), text: "Connected", Toast.LENGTH_SHORT).show();
            }
        }
    }
    catch(Exception e){
    }
    myBluetooth = BluetoothAdapter.getDefaultAdapter();//get the mobile bluetooth device
    BluetoothDevice available = myBluetooth.getRemoteDevice(address);//connects to the device's address and checks if it's available
    btSocket = available.createInsecureRfcommSocketToServiceRecord(myUUID);//create a RFCOMM (SPP) connection
    btSocket.connect();
    try { t1.setText("BT Name: "+name+"\nBT Address: "+address); }
    catch(Exception e){}
}
```

I have a method called Bluetooth_connect_device, that connects to a device, and creates a connection that's stored in a variable. I then set the name and address fields of the application to be that of the device that has been connected. Arduino uses mostly C++ for programming, and I have programmed a connection program, that I flash the board with and a buzzer programme.

```
#include <SoftwareSerial.h>

// Swap RX/TX connections on bluetooth chip
//   Pin 10 --> Bluetooth TX
//   Pin 11 --> Bluetooth RX
SoftwareSerial mySerial(2, 3); // RX, TX

void setup()
{
  Serial.begin(115200);
  while (!Serial) {
    ; // wait for serial port to connect. Needed for native USB port only
  }

  Serial.println("Init complete!");
  mySerial.begin(115200);
  Serial.write("AT+BOUD4"); // set to 9600

  mySerial.write("AT+BOUD4"); // make sure it is set to 9600
  delay(1000);
  while (mySerial.available()) {
      Serial.write(mySerial.read());
   }
}
void loop() {}
```

The programme above basically checks, what is connected to the pins on the Arduino board, and it flashes the board with a programme to allow for the Bluetooth module to work properly. It writes a set baud rate to make sure everything works properly and so that the Arduino board is not overheated and overclocked. To use Bluetooth features in my application, I had to declare two permissions. The first of these is BLUETOOTH. I need this permission to perform any Bluetooth communication, such as requesting a connection, accepting a connection, and transferring data. The other permission that I had to declare is ACCESS_FINE_LOCATION. Location permission is required because Bluetooth scans can be used to gather information about the location of the user.

I use a Bluetooth API, that scans for Bluetooth devices, query's the local Bluetooth adapter and establishes RFCOMM channels.

Arduino is an open-source hardware and software that develops boards and kits for building digital devices and interactive objects that can sense and control from both the physical and digital worlds.

There are DIY kits that I use to make a buzzer sound whenever the application sends out a notification about an emergency. I could use that in the future iterations of the project so that whenever an alarm sounded the application would open, so build a sensor inside the application, but then security comes into play as the application would be listening over time.

A buzzer works like a drum, with a membrane that is struck by a drum stick causing the device to vibrate and produce noise, but in the Arduino, the membrane is an electric current. To connect the device, firstly I connect the positive leg of the buzzer and connect it to a positive digital pin on the board. The negative leg was connected to ground on the Arduino. Tone command plays command, with two parameters, the digital pin, and the frequency of the noise. There is also a no tone command, which I would use in my assignment when there was no clear emergency.

```
char cache;
String inputString="";
//Specify digital pin on the Arduino that the positive lead of piezo buzzer is attached.
int piezoPin = 9;
void setup()
{
  Serial.begin(9600);
  pinMode(9,OUTPUT);
}

void loop()
  {
  if(Serial.available())
    {
    while(Serial.available())
      {
        char inChar = (char)Serial.read();
        inputString += inChar;
      }
    while (Serial.available()>0)
      {
      cache = Serial.read();
      }
    if(inputString == "f")
    {
      tone(piezoPin, 5000, 500);
    }
      else if(inputString == "b")
      {
        noTone(piezoPin);
      }
  inputString = "";
    }
}
```

JavaArduinoLibrary is the library I use which is easy to use with simple methods that allow me to read and write from the serial port. The library has a class that shows the list of all available ports for selection. Overloaded functions for serialWrite () and serialRead () allow multiple methods of communication, I use the button as a serialWrite to the Arduino. (DummyCodes, 2014).

As I developed the application, I noticed that the cheap Bluetooth module had some errors in it, as it would not allow me to write the values and buzz the buzzer properly. The solution for this would be to invest in a better board with a better buzzer, if I were to scale this project.

SoundReceiverBuzzer

```
int buzzerPin=9; //buzzer is on pin 9
int sensorPin=7; // sensor
boolean val =0; // boolean set to off

void setup(){
  pinMode(buzzerPin, OUTPUT);
  pinMode(sensorPin, INPUT);
  Serial.begin (9600);
}

void loop (){
  val =digitalRead(sensorPin);
  Serial.println (val);
  // when the sensor detects a signal above the threshold value, Buzzer sounds
  if (val==HIGH) {
      tone(buzzerPin, 1000, 10000000);
  }
  else {
   noTone(buzzerPin);
  }
}
```

I've also a Sound Receiver Buzzer, which I do not use in the assignment, but I did code, so basically, it's a Boolean and if the value of the sensor pin is high, I tone the buzzer, else the buzzer is not run. This could be changed to an LED, to visualise the alarm, for the people that have hearing problems.

Firebase – Log In / Register / Change Password / Deactivate

Databases for mobile applications need to be:

- Small in size- since mobile phones have many apps to cater form.
- In a form a dependency so that it can be used when needed
- Fast
- Secure
- Easy to code and to include an option to make it private or shared with other apps/users.
- Low power consumption.

Firebase DB is a mobile app and web app development platform developed by Firebase in 2011, and it has been bought by Google in 2014. I would consider using it as it has been bought by Google, so it would have many advantages, such as easier access to the Google play store. Firebase is used by more than 1.5 million applications (Firebase.Inc, 2018).

Firebase is a free database, hence why it has a big advantage, as probably the application could be free. "Firebase Auth" is a service that can validate users using client-side code, so it wouldn't need the internet to log the user in, just would need it to pull an emergency. The company provides client libraries that enable integration with Android / Java and Kotlin, hence why I would consider using it in my assignment.

Firebase Storage provides secure file uploads and downloads for mobile applications. That is done without a good network quality. The user can use it to store images, video, or other user-generated content, so if I were to store a Map of a building, it would be very secure, also takin in the fact that internet connection does not have to be amazing to access the service (FirebaseStorage, 2016).

Firebase Notifications is a service that enables targeted user notifications for mobile app developers at no cost, so the user would get notified about an emergency without having to have the app opened.

In my mobile application, I use firebase auth mostly, as I do not want to store the user's location since it would be unnecessary and could lead to very dramatic consequences if someone would hack the application. Firebase auth, basically allows users to log in and register. I do this to make sure I, the administrator of the application, know who is currently using the mobile application to avoid security breaches. Firebase stores the passwords in a salted and hashed manner, so even I do not know what a user's password is, which is good, as it would be hard for a security breach to happen. Firebase Auth makes my application very secure, so that I know who is using the application in all times.

```
private void registerUser() {
    String email = registeremail.getText().toString();
    String password = registerpassword.getText().toString();

    if (email.isEmpty()) {
        registeremail.setError(" Please enter an email ");
        registeremail.requestFocus();
        return;
    }
    //matcher compares typical emails to email inputed
    if (!Patterns.EMAIL_ADDRESS.matcher(email).matches()){
        registeremail.setError(" Please enter a valid email ");
        registeremail.requestFocus();
        return;
    }
    if (password.isEmpty()){
        registerpassword.setError(" Password is required ");
        registerpassword.requestFocus();
        return;
    }
    if (password.length()<6){
        registerpassword.setError(" Password must contain more than 6 characters ");
        registerpassword.requestFocus();
        return;
    }
    //progress bar
    progressBar.setVisibility(VISIBLE);

    mAuth.createUserWithEmailAndPassword(email,password).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
        @Override
        //Annonymous class to see if reg is completed successfully
        public void onComplete(@NonNull Task<AuthResult> task) {
            progressBar.setVisibility(GONE);
            if(task.isSuccessful()){
                Toast.makeText(getApplicationContext(), text: "User Registered Successfully", Toast.LENGTH_SHORT).show();
                //goes back to login screen after completed
                startActivity(new Intent( packageContext: Register.this,LogIn.class));
            }
```

This above is the register method, I check if the email is of the correct format, and I check if the password is not empty and it has more than 6 characters, as specified by the firebase authentication.

```
    public void onComplete(@NonNull Task<AuthResult> task) {
        progressBar.setVisibility(GONE);
        if(task.isSuccessful()){
            Toast.makeText(getApplicationContext(), text: "User Registered Successfully", Toast.LENGTH_SHORT).show();
            //goes back to login screen after completed
            startActivity(new Intent( packageContext: Register.this,LogIn.class));
        }
        else{
            if(task.getException() instanceof FirebaseAuthUserCollisionException){
                Toast.makeText(getApplicationContext(), text: "You already registered", Toast.LENGTH_SHORT).show();
            }
            else{
                Toast.makeText(getApplicationContext(),task.getException().getMessage(),Toast.LENGTH_SHORT).show();
            }
        }
    }
});
```

The on Complete is an anonymous class to see if the registration occurred without any errors, if no errors, the user goes to the login screen, if an error with the email already in use, the user gets a message to the screen specifying that he or she is already registered. If firebase error I display the exception.

```
private void loginUser() {

    String email = loginemail.getText().toString();
    String password = loginpassword.getText().toString();

    if (email.isEmpty()) {
        loginemail.setError(" Please insert an email ");
        loginemail.requestFocus();
        return;
    }

    if (!Patterns.EMAIL_ADDRESS.matcher(email).matches()) {
        loginemail.setError(" Please enter a valid email ");
        loginemail.requestFocus();
        return;
    }
    if (password.isEmpty()) {
        loginpassword.setError(" Password is required ");
        loginpassword.requestFocus();
        return;
    }
    //progress bar
    progressBar.setVisibility(VISIBLE);

    mAuth.signInWithEmailAndPassword(email,password).addOnCompleteListener(new OnCompleteListener<AuthResult>() {
        @Override
        public void onComplete(@NonNull Task<AuthResult> task) {
            progressBar.setVisibility(GONE);
            if(task.isSuccessful()){
                Intent intent= (new Intent ( packageContext: LogIn.this, Home.class));
                //clears activity from stack so when back is pressed the user is not log out
                intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                startActivity(intent);
            }
            else{
                Toast.makeText(getApplicationContext(),task.getException().getMessage(),Toast.LENGTH_SHORT).show();
            }
        }
    });
```

This is the login method, it checks if the email is empty, also if the email is not valid and if the password is empty. The on Complete checks if the email is correct and if the password is correct, if so it brings the user to the home page, which contains the 3 buttons. If incorrect it gets the exception error from firebase.

As seen below, firebase allows users to also change the password. This is mainly completed in the backend. I let the user see his or her current email, and I've a textbox to allow for a new password to get input. The change method gets the current user, and if the password input is longer than 6 characters, i.e. it satisfies the constraints of firebase auth, then the user is logged out and brought back to the login screen with a message saying that his or her password is changed correctly, otherwise the password stays the same.

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_changepassword);
    Toolbar toolbar = (Toolbar) findViewById(R.id.toolbarSettings);
    setSupportActionBar(toolbar);
    dialog = new ProgressDialog( context: this);
    password = (EditText) findViewById(R.id.settings_password);
    email = (TextView)findViewById(R.id.settings_email);
    auth = FirebaseAuth.getInstance();
    email.setText(FirebaseAuth.getInstance().getCurrentUser().getEmail());
}

public void change(View V){
    FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
    if (user!=null){
        dialog.setMessage("Changing password, please wait");
        dialog.show();
        user.updatePassword(password.getText().toString())
                .addOnCompleteListener((task) -> {
                    if(task.isSuccessful()){
                        dialog.dismiss();
                        Toast.makeText(getApplicationContext(), text: "Your password has been changed",Toast.LENGTH_SHORT).show();
                        auth.signOut();
                        finish();
                        Intent l = new Intent( packageContext: ChangePassword.this, LogIn.class);
                        startActivity(l);
                    }

                    else{
                        dialog.dismiss();
                        Toast.makeText(getApplicationContext(), text: "Password inputed incorrectly",Toast.LENGTH_SHORT).show();
                    }
                });
    }
}
```

For a user to deactivate his or her account, they press a button, that has a delete Account method that is on Click. The current user is obtained, thanks to firebaseAuth.getInstance.getCurrentUser(). Firebase auth has a method that allows the users to be deleted easily by just calling user.delete() method, which deletes all the user's entries in the authentication database, and there are no details left about the user.

```
public class Deactivate extends AppCompatActivity {

    TextView information;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_deactivate);
        information = (TextView)findViewById(R.id.info);
        FirebaseUser auth = FirebaseAuth.getInstance().getCurrentUser();
        information.setText(auth.getEmail());
    }

    public void deleteAcc (View v){
        FirebaseUser user = FirebaseAuth.getInstance().getCurrentUser();
        if(user!=null){
            user.delete().addOnCompleteListener((task) -> {
                    if(task.isSuccessful())
                    {
                        Toast.makeText(getApplicationContext(), text: "ACCOUNT DELETED \n   HOPE TO SEE YOU SOON",Toast.LENGTH_LONG).show();
                        finish();
                        Intent i = new Intent( packageContext: Deactivate.this,LogIn.class);
                        startActivity(i);
                    }
                    else
                    {
                        Toast.makeText(getApplicationContext(), text: "Account Could not be deactivated",Toast.LENGTH_SHORT).show();
                    }
            });
        }
    }

}
```

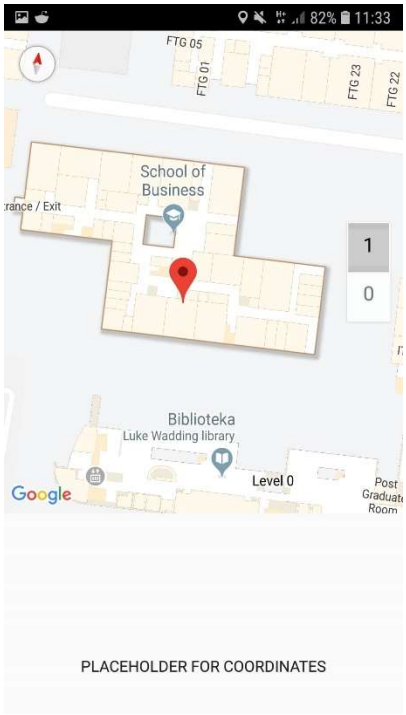
Google Maps API

An API (application programming interface) is a framework that I use to write the Google Maps. An API contains classes that the developer, me uses to avoid re-doing and recoding the low-level code for displaying the map and drawing a new layer on it.

The Google Maps API allows for the embedding of Google Maps onto the screen of my application. It is designed to work on both mobile devices as well as desktops, so if I were to develop a web app I would use this API also. The API includes language localization for over 50 languages, region localization and geocoding, I use the Google Maps API in Polish as it's the language I set my phone to be. It has mechanisms for enterprise developers who want to utilize the Google Maps API within an intranet. The API HTTP services can be accessed over a secure (HTTPS) connection by Google Maps API Premier customers (programmableweb,2018).

```
// Obtain the SupportMapFragment and get notified when the map is ready to be used.
SupportMapFragment mapFragment = (SupportMapFragment) getSupportFragmentManager()
        .findFragmentById(R.id.map);
mapFragment.getMapAsync(this);
locationManager = (LocationManager) getSystemService(LOCATION_SERVICE);

txtview = (TextView) findViewById(R.id.latlng);
locationListener = new LocationListener() {
    @Override
    public void onLocationChanged(Location location) {
        double latitude = location.getLatitude();
        double longitude = location.getLongitude();
        txtview.setText("Latitude = " + latitude + ", Longitude = " + longitude);
        //get the location name from latitude and longitude
        Geocoder geocoder = new Geocoder(getApplicationContext());
        try {
            List<Address> addresses =
                    geocoder.getFromLocation(latitude, longitude, 1);
            String result = addresses.get(0).getLocality() + ":";
            result += addresses.get(0).getCountryName();
            LatLng latLng = new LatLng(latitude, longitude);
            if (marker != null) {
                marker.remove();
                marker = mMap.addMarker(new MarkerOptions().position(latLng).title(result));
                mMap.setMaxZoomPreference(20);
                mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(latLng, 19.0f));
            } else {
                marker = mMap.addMarker(new MarkerOptions().position(latLng).title(result));
                mMap.setMaxZoomPreference(20);
                mMap.moveCamera(CameraUpdateFactory.newLatLngZoom(latLng, 19.0f));
            }


        } catch (IOException e) {
            e.printStackTrace();
        }
```

I have a class called Maps Activity that contains and holds the map. The user of the application must give permissions for the application to get his or her current location. I get the location to be very accurate, as that is important for the user, to use when they want to escape from a specific room, to know exactly which room they are in. Thankfully Google has a very accurate map, allowing the user to know which room he or she is at when he or she specifies the floor they are at. I tried this feature in WIT and the results I obtained were very good. In the code, I get the location from the get System Service Location Service method. I have an On Location Changed method. This is to make sure that the map updates in real-time with the user. If a user moves, the application will know and present a new location. I have added a coordinate block too, in case the user had to specify where he currently is to emergency services and to make the user more aware of the surroundings.



This screenshot was taken before I implemented the code, showing the correct coordinates, as I will show in the demo. The user would need this information to specify his/her location on the graph where they start and where they end. I decided not to store this information, as it would be taking security breach risks to a high level.

I had problems with API keys, as I've created a number of projects previously and had to contact Google to get more keys available for my projects, this was a bit of a challenge and a pause in my application development as without the location the user would not know where they are and where to set the start points of the escape to be correctly. The keys were then working but I had to add a package on the google maps API website, as before that, I was just getting a blank map and it was not working correctly. After I completed the map, which was a vital part of my assignment, I decided to design and do out the algorithm, Dijkstra's and started to look up ways of implementing it. I will talk about this in the next part of the report. Location was in the end obtained accurately and very precisely for the user to use. Not in the video, or the screenshot above, I've added a button to save the user from going back to the main screen to get a building plan, and instead they are brought to the screen directly from the Maps Activity. This is a user experience addition, as the user would have fresh in his/her mind where they were last time.

Route finding – Dijkstra's algorithm implementation

The user is given a route to escape, but how is the route computed. I use one of the algorithms that I learned during Erasmus called Dijkstra which works by starting at one vertex point and exploring nodes nearby until the destination node is reached. Dijkstra algorithm works by having a start node and a set of candidate nodes to possibly use. At each iteration, the node in the open set with the lowest distance from the start is looked at. The node is marked "closed" and visited, and all nodes adjacent to it are added to the open set if they have not been examined.

Building View Class Explained – Dijkstra's algorithm

```
public class BuildingPlanView extends View{
    int rows=30;
    int cols=30;
    float tileWidth;
    float tileHeight;
    float width;
    float height;
    int[][] grid = new int[rows][cols]; //0-empty 1-start 2-end 3-hurdle 4-located in set 5-located on queue 6-tile path
    int startClicked = 0;
    int start_x,start_y;
    int stopClicked = 0;
    int stop_x,stop_y;
    final int animationtime=01;
    Graph graph;
    Paint paint = new Paint();
    async dijkstrathread=new async();
    Bitmap bmp;

// initialization of canvas
    public void init(AttributeSet attrs,int defStyle){
        paint.setColor(Color.BLACK);
        paint.setAntiAlias(true);
        bmp = BitmapFactory.decodeResource(getResources(), R.drawable.second);
        }

    public BuildingPlanView(Context context){
        super(context);
        init( attrs: null, defStyle: 0);
    }

    public BuildingPlanView(Context context, AttributeSet attrs) {
        super(context, attrs);
        init(attrs, defStyle: 0);
    }
```

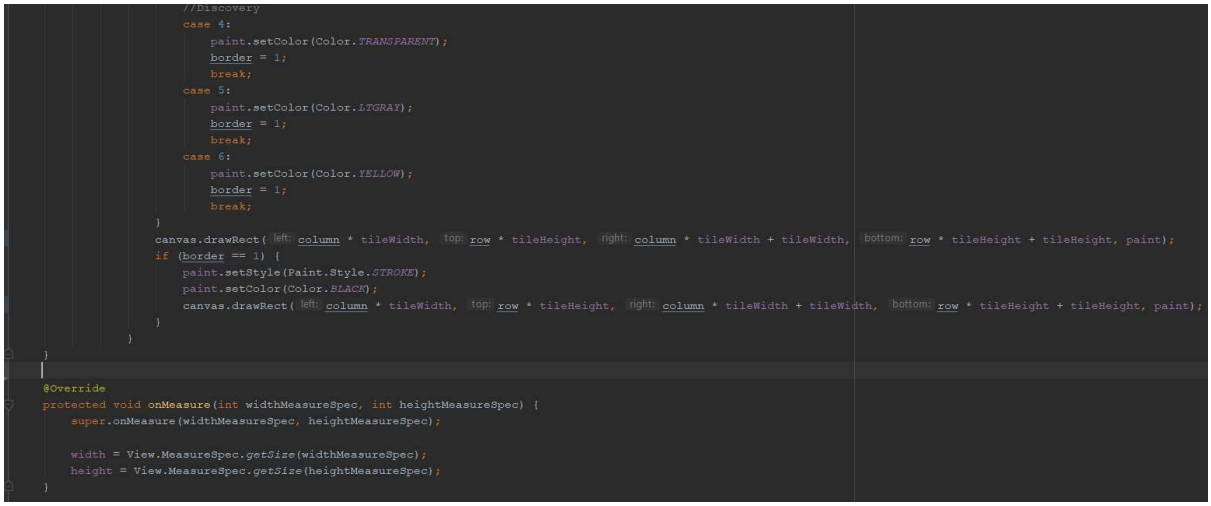
Above I state all the fields I will use, such as the rows and columns value, the higher the number, the smaller the grids get as more columns and rows are added decreasing the size. Tile Width and height are used to draw out the rows and columns on the canvas. Width and height are the width and height of the screen and that is to make sure nothing is created outside of the boundaries. Grid is an array of arrays, used to store rows and columns. Comment specifies the values of grid available. Start Clicked is 0 and 1, to specify where the user presses start, then when that happens, Grid start_x and start_y is set to be an empty grid. I then let grid[row][column] to be 1, and I let the start_x to be column and start_y is row. Stop Clicked is set to be 0 by default, stop_x and stop_y are variables for column and row when stop is clicked. Animation time is the time for the graph to be displayed, I have it as 01 as it is relatively quick, but could be set lower, this is purely for visuals during the demonstration. Graph is there to get the Vertex start and vertex destination in the async method. This gets the start_x and start_y, and the stop_x and stop_y and sets one as start and the other one as destination. Paint variable is calling a Paint() constructor, that creates a new paint with default settings. Later, in this project, I manipulate this method to make sure the Canvas paints what I want to paint, i.e. Start, Route, End. Async enables an easy use of the UI thread. This class allows me to perform background operations and publish results on the UI thread without having to manipulate threads and/or handlers. I use this to print out the results of the compilation of my algorithm. Bitmap is used to load the saved image if needed onto the canvas, which could be developed further if my assignment developed commercially.

The constructors init and BuildingPlanView are both important so no exception occurs. The init initializes the application. The Building plan context is used when instantiating Views programmatically. The BuildingPlanView context attrs is used by the layout inflater to apply XML attributes.

```
//canvas drawing
    public void onDraw(Canvas canvas){


        super.onDraw(canvas);
        //canvas.setBitmap(bmp);
        tileWidth = width / cols;
        tileHeight = height / rows;
        int border = 0;

        // Stroke
        paint.setStyle(Paint.Style.STROKE);
        paint.setColor(Color.BLACK);
        canvas.drawBitmap(bmp, left: 0, top: 0, paint: null);

        //
        // draw each tile according to grid matrix content
        for (int row = 0; row < rows; row++)
            for (int column = 0; column < cols; column++) {
                paint.setStyle(Paint.Style.FILL);
                int value = grid[row][column];
                switch (value) {
                    case 0:
                        paint.setStyle(Paint.Style.STROKE);
                        paint.setColor(Color.BLACK);
                        break;
                    case 1:
                        paint.setColor(Color.BLUE);
                        break;
                    case 2:
                        paint.setColor(Color.GREEN);
                        break;
                    case 3:
                        paint.setColor(Color.RED);
                        break;
                    //Discovery
```

Having both Canvas and Paint objects will allow me to draw anything I need -> in my case rectangles in the grid

A Canvas instance comes as on Draw parameter, it basically responds for drawing a different shape, while Paint object defines that colour that shape will get. I get the tile width and height to be half of the width divided by columns and likewise with the height, being height divided by rows. The stroke and the colour are set to default values, and a bitmap is created. I have a nested for loop to draw and iterate through each row and column of the grid. The value is the grid[row][column], if it's the start, set the colour to be blue. The end node is green, the Red is the fire and that's case3, where a user paints onto the canvas where the fire currently is. This updates on a touch method which I will describe in a later part of this report.



The screenshot above is a continuation of the on-draw method. I set the colour to be transparent, in the poster I had a light blue colour set, for this option. This is basically all the discovered nodes that the algorithm has processed. The light grey, which is case 5, is the nodes that are currently being processed and case 6 is the final stage, which is the yellow path that is drawn.

For columns the draw rectangle method is called, with the grid values and the current paint depending on the case. If border is one, i.e. the discovery or discovered nodes then draw in black, this method is only used if an error occurs of some sort with the border value.

```
//method when tile is touched
public boolean onTouchEvent(MotionEvent motionEvent){
    if(motionEvent.getAction()==MotionEvent.ACTION_DOWN) {
        float x = motionEvent.getX();
        float y = motionEvent.getY();
        int column = (int) (x / tileWidth);
        int row = (int) (y / tileHeight);
        if(startClicked==1){
            grid[start_y][start_x]=0; // set start to be empty grid
            grid[row][column]=1;
            start_x=column;
            start_y=row;
            startClicked=0;
        }
        else if(stopClicked==1){
            grid[stop_y][stop_x]=0;
            grid[row][column]=2;
            stop_x=column;
            stop_y=row;
            stopClicked=0;
        }
        else if (grid[row][column]!=1&&grid[row][column]!=2) {
            if (grid[row][column] != 3)
                grid[row][column] = 3;
            else
                grid[row][column] = 0;
        }
    }
    if(motionEvent.getAction()==MotionEvent.ACTION_MOVE){
        float x = motionEvent.getX();
        float y = motionEvent.getY();
        int column = (int) (x / tileWidth);
        int row = (int) (y / tileHeight);
        if(x>width || y>height||x<0||y<0)
            return false; // false if out of bounds
```

```
        if(grid[row][column]!=1&&grid[row][column]!=2)
            if (grid[row][column] != 3)
                grid[row][column] = 3;
    }
    invalidate();
    return true;
}
```

The OnTouchEvent method is used for when the user pressed down on a specific tile. The float values of x and y are the values of the screen x and y positions the user has pressed on. Column is x divided by Width and row is y divided by height.

When the start button is clicked, the y and x values are 0 in the array of arrays value and row and column values are 1, meaning that the user specifies a start node.

When the stop button is clicked, the y and x values are 0 in the array of arrays value and row and column values are 2, meaning that the user specifies an end node.

When the user did not press the button the row and column array of arrays value is 3, meaning that it is the red colour, of fire, or 0 meaning black, which is a default I set as an error checking value.

I also check that the width and height are in the bounds of the screen and aren't pressed in a way I did not expect, so I make the on touch method very predictable to return a Boolean value of false, if the press was out of bounds, and then I check if the user pressed the start or the end node on the buttons and if not I set the colour to be 3.

I then call an invalidate method, this is an android method used because the system handles resizing, hiding, showing, but it sometimes has had problems if the underlying buffer for drawn pixels or backing data has changed, for example I swap the image resource on a View or the raw dataset changes- when drawing the Canvas bitmap. This occurs because there is no way that the Android system can know that the data changed in the specific manner that it did.

Invalidate() means 'redraw on screen' and afterward a call of the view's onDraw() method, so if something changed in the visuals and it needs to be reflected on screen, I need to call invalidate().

The return true is regarding the fact that the press, with the x and y coordinates, was within the boundaries and is a correct press on the screen.

```
//responsible for the results printed above the graph
public String Dijkstra(){
    System.gc();
    getGraph();
    dijkstrathread = new async();
    dijkstrathread.execute();
    getGraph();
    String text=graph.Dijkstra(graph.getV(start_x, start_y), graph.getV(stop_x, stop_y)); // text for statistics above
    return text;
}


public void getGraph(){
    //we get grid, compute graph, remove those tiles that have been processed by any of the algorithms before
    Graph g = new Graph();
    int counter=0;
    for(int i=0;i<rows;i++) {
        for (int j = 0; j < cols; j++) {
            if(grid[i][j]!=3)
                g.addVertex(counter, j, i);
            if(grid[i][j]==4||grid[i][j]==5||grid[i][j]==6)
                grid[i][j]=0;
            // increment counter
            counter++;
        }
    }

    // x y
    int x1,x2,x3,x4,x5,x6,x7,x8;
    int y1,y2,y3,y4,y5,y6,y7,y8;
```

```
for(int i=0;i<rows;i++) {
    for (int j = 0; j < cols; j++){
        if(grid[i][j]==3)
            continue;
        x1=j-1;y1=i-1;
        x2=j;y2=i-1;
        x3=j+1;y3=i-1;
        x4=j-1;y4=i;
        x5=j+1;y5=i;
        x6=j-1;y6=i+1;
        x7=j;y7=i+1;
        x8=j+1;y8=i+1;
        if(x1>=0&&x1<cols&&y1>=0&&y1<rows)
            if(grid[y1][x1]!=3)
                g.addEdgeGrid(g.getV(j,i),g.getV(x1,y1), weight: 1.4);
        if(x2>=0&&x2<cols&&y2>=0&&y2<rows)
            if(grid[y2][x2]!=3)
                g.addEdgeGrid(g.getV(j,i),g.getV(x2,y2), weight: 1);
        if(x3>=0&&x3<cols&&y3>=0&&y3<rows)
            if(grid[y3][x3]!=3)
                g.addEdgeGrid(g.getV(j,i),g.getV(x3,y3), weight: 1.4);
        if(x4>=0&&x4<cols&&y4>=0&&y4<rows)
            if(grid[y4][x4]!=3)
                g.addEdgeGrid(g.getV(j,i),g.getV(x4,y4), weight: 1);
        if(x5>=0&&x5<cols&&y5>=0&&y5<rows)
            if(grid[y5][x5]!=3)
                g.addEdgeGrid(g.getV(j,i),g.getV(x5,y5), weight: 1);
        if(x6>=0&&x6<cols&&y6>=0&&y6<rows)
            if(grid[y6][x6]!=3)
                g.addEdgeGrid(g.getV(j,i),g.getV(x6,y6), weight: 1.4);
        if(x7>=0&&x7<cols&&y7>=0&&y7<rows)
            if(grid[y7][x7]!=3)
                g.addEdgeGrid(g.getV(j,i),g.getV(x7,y7), weight: 1);
        if(x8>=0&&x8<cols&&y8>=0&&y8<rows)
```

The Dijkstra is responsible for printing out the results above the graph. The first method is System.gc(); The System.gc() method runs the garbage collector. I'm calling this for the Java Virtual Machine to reuse unused objects to make the memory available for quick reuse. This is to make sure the application runs as quick as possible for a user to find the emergency escape. I then run the get Graph method, which is below, and I will explain in the next paragraph. I make the Dijkstra thread to execute asynchronously with the graph, this means that both the get graph and the text are executed at the same time and one doesn't have to finish before the other one.

The get Graph is pressed when a user presses the Dijkstra button. This is the main feature and algorithm of my final year project. First, we let g be a new graph, I iterate through each row, and J iterates through each column. If grid I and j is not 3, which is when the user draws a fire, we add a vertex, which takes in a counter as id, j and I as the x and y values. If the grid I j value is 4 or 5 or 6, which means, its located, its on the located queue or the path is found, I let grid I j equal to 0 to reset. I then increment the counter id, as that would count as a second route finding.

The nodes have 8 values of x and y around them, each of these must be iterated through, I subtract or add 1 as the values for them had to be in the for loop, so the values resemble the table I have below, except the one that is directly by the node and not Diagonal. The table below is how I implement the x1-x8 y1-y8 values and this is to make sure the algorithm works correctly. This is also to find the end node of the algorithm i.e. the destination and to instruct the algorithm which way to go

X1 j-1 i -1 Weight 1.4	X4 j-1 i+0 Weight 1	X6 j-1 i+1 Weight 1.4
X2 j+0 i-1 Weight 1	Node (j 0 I 0)	X7 j+0 i+1 Weight 1
X3 j+1 i-1 Weight 1.4	X5 j-1 i+0 Weight 1	X8 j+1 i+1 Weight 1.4

I then have if statements to see if x1>0 and x1 is greater than cols and y1 is greater than 0 and less than rows and if nothing is drawn there in red, we add edge grid with a source j and i, we add the destination from that edge x1 and y1 and we set the weight to be 1.4. The if statement is there to make sure that the node we add to the edge grid is not the node we are using as the end node. The weights on the diagonal is greater than on the straight lines. This algorithm tends to avoid the routes with the larger weights unless it is necessary to find the optimal escape route.

I then check in all the 8 if statements if the current x value is greater than 0, meaning it is not found, and it is less than cols, meaning that it is on the screen and if the y value is greater than 0, meaning it is not found and that it is less than rows, meaning its on the screen, I then add an edge grid, with the source, destination and a weight of either 1 or 1.4 depending on the position of the node that is been discovered currently. This goes on for each x and y value, and then I let the graph to be g, meaning the graph is updated and I invalidate, this is an android method used because the system handles resizing, hiding, showing, but it sometimes has issues if the underlying buffer for drawn pixels or if the data has been changed.

```
                if(grid[y8][x8]!=3)
                    g.addEdgeGrid(g.getV(j,i),g.getV(x8,y8), weight: 1.4);
            }
        }
        graph = g;
       invalidate();
    }


//The AsyncTask executes everything in doInBackground() inside of another thread, which does not have access to the GUI where your views are.
    public class async extends AsyncTask<Void,Void,Void>{
        protected Void doInBackground(Void... params) {
            // the part below needs to be graphed
            Vertex start=graph.getV(start_x, start_y);
            Vertex destination = graph.getV(stop_x, stop_y);
            //Initialization
            // d[start]=0 (other vertex's d_value is infinity by default), S={0} , Q = vertex
            LinkedList<Vertex> set = new LinkedList<>();
            PriorityQueue<Vertex> queue = new PriorityQueue<>();
            queue.add(start);
            start.d_value=0;

            //cycle until queue is empty or destination has been inserted into s
            while(!queue.isEmpty()) {
                if (isCancelled()) break;
                Vertex extracted = queue.poll();
                extracted.discovered = true;// when it is in set and now d_value is defined
                set.add(extracted);
                if(grid[(int) extracted.y][(int) extracted.x]!=1&&grid[(int) extracted.y][(int) extracted.x]!=2)
                    grid[(int) extracted.y][(int) extracted.x] = 4;
                publishProgress();
                try {
                    Thread.sleep(animationtime);
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
```

```
            //for each vertex into dhe adj list of extracted -> relax.
            for (int i = 0; i < extracted.edges.size(); i++) {
                //edge examined
                Edge edge = extracted.edges.get(i);
                //get neighbour vertex and relax
                Vertex neighbour = edge.destination;
                if (neighbour.discovered == false) {
                    //Relaxation
                    if(grid[(int) neighbour.y][(int) neighbour.x]!=1&&grid[(int) neighbour.y][(int) neighbour.x]!=2)
                        grid[(int) neighbour.y][(int) neighbour.x] = 5;
                    publishProgress();
                    try {
                        Thread.sleep(animationtime);
                    } catch (InterruptedException e) {
                        e.printStackTrace();
                    }
                    if (neighbour.d_value > extracted.d_value + edge.weight) {
                        neighbour.d_value = extracted.d_value + edge.weight;
                        neighbour.parent = extracted;
                        //insert neighbours in queue so we can choose dhe min one
                        queue.remove(neighbour);
                        queue.add(neighbour);
                    }
                }//if discovered

            }
        }
        Vertex current = destination;
        while (current != null) {
            if (isCancelled()) break;
            if (grid[(int) current.y][(int) current.x] != 1 && grid[(int) current.y][(int) current.x] != 2)
                grid[(int) current.y][(int) current.x] = 6;
            publishProgress();
            try {
                Thread.sleep(animationtime);
            } catch (InterruptedException e) {
                e.printStackTrace();
```

```
                }
                current = current.parent;
            }
            return null;
        }
        protected void onProgressUpdate(Void... values) { invalidate(
    }


}
```

In the above screenshots, I have the AsyncTask that executes everything in the background while the graph is drawn. First, I initialise the start and destination Vertex. I have a linked list called set and a priority queue called queue. Set basically stores all the nodes visited that are not part of the solution- transparent in the final demo, blue on the poster. Queue is the ones that are bound to be discovered and are grey in colour. I add then the start to be on the queue and set the d_value to be 0, as this is the start and I will use this to draw out the yellow path from the start. I then cycle until queue is empty or the destination has been reached and inserted into the set. I have an extracted vertex value to be equal to the poll value of the queue. I then set the extracted discovered to be true, and then add it to the set.

If the grid value of the extracted is not the start i.e. !=1 and if the grid extracted is not the end !=2 then the grid must be of value 4 = Transparent i.e. discovered I publish the results. I then call a Java Async method called publishProgress() which pushes the information onto the UI. If the extracted is the destination I break from the animation and the route is drawn.

I then go through the size of the extracted edges and I check all edges and get the neighbour of each edge that was discovered. If the neighbour is not discovered and if it is not the start or the end, it is the future nodes to discovered which is 5, the grey colour on the poster app demo. I publish the progress. The d_value for start is 0, but for any other node it is infinity. If the neighbours.d_value is greater than the extracted.d_value +weight I let the d_value to be the weight and the parent to be extracted. I add the neighbour to the queue to be discovered. This step is to add the future nodes to be explored, excluding the ones marked by the user that indicate there is a fire there.

If the queue is empty, i.e. no more nodes to discover I set the current vertex to be the destination. If the destination is not null, and not the start or the end node, I let the grid of current x and y to be 6, which is the yellow specifying the quickest path. I specify that the current node is the current parent so that there are no more nodes to be discovered. This method returns null, it just updates the UI with the publishProgress(); java method, and I try to sleep the thread for the animation time, just to make sure there are no errors.

Building Activity Class Explained

```
public class BuildingPlanActivity extends FragmentActivity {
    BuildingPlanView grid;
    View view;
    Bitmap bitmap;
    File file;
    // Creating Separate Directory for saving Generated Images
    String DIRECTORY = Environment.getExternalStorageDirectory().getPath() + "/BuildingPlans/";
    String pic_name = new SimpleDateFormat( pattern: "yyyyMMdd_HHmmss", Locale.getDefault()).format(new Date());
    String StoredPath = DIRECTORY + pic_name + ".png";

    // On Create and button handlers
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_buildingplan);
        // Toolbar toolbar = (Toolbar) findViewById(R.id.toolbarBuildingPlan);
        Button start = (Button) findViewById(R.id.start);
        Button stop = (Button) findViewById(R.id.stop);
        Button dijkstra = (Button) findViewById(R.id.dijkstra);
        final Button save = (Button) findViewById(R.id.save);
        final TextView results = (TextView) findViewById(R.id.results);
        grid = (BuildingPlanView) findViewById(R.id.grid);
        view = grid;

        // Method to create Directory, if the Directory doesn't exists
        file = new File(DIRECTORY);
        if (!file.exists()) {
            file.mkdir();
        }
```

This class is responsible for the buttons that are pressed at the bottom of the screen and the links with the GUI. I also have a method to save the image to a bitmap, and possibly re-use the image as the application would develop. I create in the On Create method the buttons, the results tab, the grid and I let the view to be the grid.

I also create a directory of the format as seen in the String DIRECTORY prior to the On Create method. I check if the directory to save the image exists and if it does not I create it.

```
start.setOnClickListener((v) → {
        grid.stopClicked = 0;
        grid.startClicked = 1;
});
stop.setOnClickListener((v) → {
        grid.startClicked = 0;
        grid.stopClicked = 1;
});
dijkstra.setOnClickListener((v) → {
        String text = grid.Dijkstra();
        results.setText(text);
});
save.setOnClickListener((v) → {
        isStoragePermissionGranted();
        view.setDrawingCacheEnabled(true);
        save(view, StoredPath);
        Toast.makeText(getApplicationContext(), text: "Successfully Saved", Toast.LENGTH_SHORT).show();
});

}

// To save the image to the phone
ublic boolean isStoragePermissionGranted() {
  if (Build.VERSION.SDK_INT >= 23) {
      if (getApplicationContext().checkSelfPermission(Manifest.permission.WRITE_EXTERNAL_STORAGE) == PackageManager.PERMISSION_GRANTED) {
          return true;
      } else {
          ActivityCompat.requestPermissions( activity: this, new String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE},  requestCode: 1);
          return false;
      }
  } else {
      return true;
  }
```

In the above I have a start button, stop button, Dijksktra button and save button on click listeners. Each of these buttons have rolls. The start sets the start node, stop sets the end node, the Dijkstra button runs all of the methods in the view and also sets the results to be the text obtained from the Graph class. The save first checks if the storage permission is granted, then saves the view to the stored path and a toast message is then printed to confirm a save occurred.

I have a boolean method to check if the storage permission is given. This is a feature the user must implement from versions 23 or above. If the permision is given return true else false, and that will then save or not save the bitmap.

```
@Override
protected void onStop() {
    if (grid.dijkstrathread.getStatus() == AsyncTask.Status.RUNNING)
        grid.dijkstrathread.cancel( mayInterruptIfRunning: true);
    super.onStop();
}

//Saving permision
@Override
public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    if(grantResults.length > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED){

        grid.setDrawingCacheEnabled(true);
        save(view, StoredPath);
        Toast.makeText(getApplicationContext(), text: "Successfully Saved", Toast.LENGTH_SHORT).show();
        // Calling the same class
        recreate();
    }
    else
    {
        Toast.makeText( context: this,  text: "The app was not allowed to write to your storage- grant it this permission", Toast.LENGTH_LONG).show();
    }
}
```

The On Stop method above checks if the Background task is running and cancels it. There is a method called onRequestPermisionResults which saves the picture.

```
//Save
public void save(View v, String StoredPath) {
    Log.v( tag: "log_tag",  msg: "Width: " + v.getWidth());
    Log.v( tag: "log_tag",  msg: "Height: " + v.getHeight());

    if (bitmap == null) {
        bitmap = Bitmap.createBitmap(grid.getWidth(), grid.getHeight(), Bitmap.Config.RGB_565);
    }
    Canvas canvas = new Canvas(bitmap);
    try {
        // Output the file
        FileOutputStream mFileOutStream = new FileOutputStream(StoredPath);
        v.draw(canvas);

        // Convert the output file to Image such as .png
        bitmap.compress(Bitmap.CompressFormat.PNG,  quality: 90, mFileOutStream);
        mFileOutStream.flush();
        mFileOutStream.close();

    } catch (Exception e) {
        Log.v( tag: "log_tag", e.toString());
    }
```

This method is run when the button is pressed, as it's the OnClick Handler. It logs to the admin, could change that to user but felt like it's unnecessary information, the image Width and Image height. If the bitmap is null it creates the bitmap with the grid width, height and the colours. It then stores the canvas and loads it back in. The bitmap is then converted to a png format, for the user to be able to view, even if the application fails to function after the path was given out, or to save battery due to the location of the user being check for.

Edge Class Explained – Dijkstra's algorithm

```
public class Edge {
    public Vertex source;
    public Vertex destination;
    public double weight;
    //edge constructor
    public Edge(Vertex source,Vertex destination,double weight){
        this.source=source;
        this.destination=destination;
        this.weight=weight;
    }
}
```

Each edge has a source, a destination and a weight. Weight can be 1 or 1.4. 1 means that the nodes adjacent are not diagonal and a weight value of 1.4 means that the node is on a diagonal. Vertex source and Vertex destination means the angular point of the source and destination, i.e. the first two nodes.

Graph Class Explained – Dijkstra's algorithm

```
package org.wit.emergencyescape.models;

import java.util.*;
public class Graph {

    public ArrayList<Vertex> vertex=new ArrayList<~>();

    public void addVertex(int id,double x,double y){
        Vertex v1 = new Vertex(id,x,y);
        vertex.add(v1);
    }

    public void addEdgeGrid(Vertex source,Vertex destination,double weight){
        //for grid we will initialize its vertex edges
        source.edges.add(new Edge(source,destination,weight));
    }


    public Vertex getV(double x,double y){
        for(int i=0;i<vertex.size();i++)
            if(vertex.get(i).x==x&&vertex.get(i).y==y)
                return vertex.get(i);
        //if not found
        return null;
    }
```

Add Vertex adds a vertex with an id, x and y value. AddEdgeGrid adds the source, destination and weight and it's used in the BuildingPlanView class. The getV method gets the vertexes with a specific I value, when given the x coordinate and the y coordinate. Returns null when there is no vertex found.

```
// This is the thing that prints out on top o the screen
public String Dijkstra(Vertex start,Vertex destination){
    String text="Dijkstra "; //we will keep the processing results here
    //Initialization
    long startTime = System.nanoTime();
    LinkedList<Vertex> set = new LinkedList<~>();
    PriorityQueue<Vertex> queue = new PriorityQueue<~>();
    queue.add(start);
    start.d_value=0;
```

This is what pressing the Dijkstra button prints out. I initialise the start time. I have a linked list called set and a priority queue called queue. Set basically stores all the nodes visited that are not part of the solution- transparent in the final demo, blue on the poster. Queue is the ones that are bound to be discovered and are grey in colour. I add then the start to be on the queue and set the d_value to be 0 , as this is the start and I will use this to draw out the yellow path from the start.

```
//cycle until queue is empty or destination has been inserted into s
while(!queue.isEmpty()){

    Vertex extracted = queue.poll();
    extracted.discovered=true;//when it is in set and now d_value is defined
    set.add(extracted);
    if(extracted==destination){
        break;
    }
    //for each vertex into dhe adj list of extracted -> relax.
    for(int i=0;i<extracted.edges.size();i++){
        //edge examined
        Edge edge = extracted.edges.get(i);
        //get neighbour vertex and relax
        Vertex neighbour = edge.destination;
        //if no neighbour
        if(!neighbour.discovered){
            //Relaxation
            if(neighbour.d_value>extracted.d_value+edge.weight){
                neighbour.d_value=extracted.d_value+edge.weight;
                neighbour.parent=extracted;
                //insert neighbours in queue so we can choose the min one
                queue.remove(neighbour);
                queue.add(neighbour);
            }
        }//if discovered
    }
}
long stopTime = System.nanoTime();
if(destination.parent==null)
    text="There is no route found";
else{
    text+=" Vertex num set: "+set.size();
    //Dijkstra process finished, now we will take our path and print it
    Stack<Vertex> stack = new Stack<~>();
    Vertex current = destination;
```

```
            while(current!=null){
                stack.push(current);
                current = current.parent;
            }
            text+=" Nr.Hops:"+(stack.size()-1)+" Time: "+(stopTime-startTime)+" ns";
        }//end else for the existence of the path
        return text;
    }
}
```

I then cycle until queue is empty or the destination has been reached and inserted into the set. If the extracted is discovered I add it to the set. I then iterate through the extracted edges. I get the edge examined and get the neighbour vertex. If no neighbour discovered I, then check the d_value and I let the neighbour.parent value to be equal to the extracted and delete the neighbour. I then have a stop time, and if the destination parent is null I print out that there is no route found. Else I have the set size, meaning how many nodes were looked at, I will then check if the destination is not null and I will print out the number of hops, which is the stack size-1 since the stack starts with 1. I then also get the time in nanoseconds that it took for the route to be found. I then return the text

Vertex Class Explained – Dijkstra's algorithm

```
package org.wit.emergencyescape.models;
import java.util.*;
public class Vertex implements Comparable<Vertex> {
    public int id;
    public Vertex parent=null;
    public double d_value=Double.POSITIVE_INFINITY;
    public LinkedList<Edge> edges=new LinkedList<~>();
    public boolean discovered = false;//found true ie vertex is in SET (Dijkstra)
    //grid coordinates
    public double x,y;

    public Vertex(int id) { this.id=id; }
    public Vertex(int id,double x,double y){
        this.id=id;
        this.x=x;
        this.y=y;
    }

    @Override
    public int compareTo(Vertex o) { return Double.compare(this.d_value,o.d_value); }
```

The Vertex implements a comparable as I had compared vertex to each other. It has an id, a parent that is set to Null by default. A random d_value that is infinite, depending on the weight added in the programme, but at the start it's infinite. I have a LinkedList of edges and a Boolean whether the vertex is discovered, set to false by default. The Vertex has a double x and y value. I then have a constructor and a compareTo method, that compares the d_values.

How security is dealt with?

Security was important for this project, as each person will be having to specify their own location. I will make sure the variables for latitude and longitude are not kept in any database, and that each user would have to log in, and when logged in they would only be able to see their own private location. No other person's location could be given away, and personal information such as name and surname and other details are given at the register stage are not stored, hence they can't be given away. The login/ register feature is there to provide security to avoid many people entering the system and trying to ddos the system. I will also know some details, like emails of the users using my app so that increases the security. Thanks to the use of firebase authentication I could allow users to log in securely.

Language

I wrote the application in Java, as in the second semester I learned java with Android studio, but also, I had a backup language called Kotlin. Kotlin is a newer version of java, with many things just made easier for the developer to write code in. Java is used, as Kotlin is only a new language, and as much as it simplifies the creation of applications by making some of the methods easier, the language is based on Java. Java was my go-to language due to the fact Java apps are more compact – in comparison to Kotlin, Java apps tend to be lighter, which is good for my final year project as I do not know what the phone of the user is going to be. A Kotlin app that includes complex computing processes in its code can turn out to work slowly on user devices with low technical specs. Java also allows top development speed, and faster build process than Kotlin according to Netguru (netguru, 2018). The build time for my assignment is vital as the quicker the user gets the route, and the quicker the application opens, the more chance the user would have to escape an emergency. According to Netguru Kotlin also has a small developer community, which means limited learning resources and difficulty in finding answers to questions. At looking only on Stackoverflow there are only around 8000 questions tagged with Kotlin against 1.37 million questions about Java.

Plan for the application after the final year project

Firstly, I plan for the application in the future to get the administrator and building owner rights in onto the system, administrators can add building owners, and building owners can add or delete maps. The building owners would have to verify manually, and therefore I did not implement this feature in this project now, but it could be part of future development. The user would not have any access to adding or deleting maps, or owners. The user will only have a map of the building, and when an emergency happens the user will be able to see the path. In building the application.

The application could be connected to the Amazon web services to be deployed in the cloud for many users- but the Arduino boards would have to be more widespread and used by more building administrators and users. There would be a need to install fire alarms in all the buildings and connecting them through Bluetooth, but I see this step being viable in the future as the world becomes connected and IOT – the internet of things begins to be a daily thing.

As my application would develop I could use neural networks to find the best route, if the number of users was enough, to test if the network would be enough. The users would be inputs to the neural network, and the neural network second layer could be different locations and based on the current location of the user the route would be shown. This would be an overkill for my final year project at this stage though as I do not have many users.

I would also take more time in developing the building plans and to match them correctly with specific buildings, as this is a draft application I just created a plan in a graphics programme called GIMP, but more professional image designs would be a nice feature. Push notifications would be added, but after reading up I would need a Wizznote internet offload board to make sure that this functionality works perfectly.

Draft user manual

The user would download the application from the app store. The user then would have to register for safety reasons and for the information to be used by the application regarding location, to keep a database of the user. The user would then log into the application where both a map would be displayed, showing the current location and a building plan. The user tries to get out of a fire. He or she must get the current location by pressing the get location button. Then he or she is going to get the route to the exit. If a fire develops that blocks the path, the user would draw the fire onto the canvas of the application, he or she would then press the button again, and a new route would be calculated and displayed. This image can be saved by the user to the phone device, in case the battery is low and the application that requires location is draining too much battery. The building administrator would press the button for the buzzer to notify all the users that there is an emergency taking place.

Learning curve and challenges

This was the first time where I had to link things that I have learned throughout different subjects and combine them together to get a fully working effect. The things I learned at doing this final year project are vital for my future as an IT graduate. I had researched the things that I planned to for the application and the things that I will do if things do not work out for me. I planned to continue this approach in different areas of my project, and I plan to use my problem-solving skills to the best of my ability. The project was a challenge to show my time keeping skills, my programming skills and my way of working around problems that will arise in the project. Connecting the Arduino with the different modules, pins and wires was also a challenge since it was the first time I was doing it. I had also problems with the Google API keys, about which I had to contact Google. I would invest more money on the Bluetooth module, as the signal is sometimes very weak. I had small problems bringing all the libraries and API's together and that took some time, also to fix all the gradle files where an error occurred due to a dependency being of a wrong version.

Citations

Firebase. 2018. Cloud Storage | Firebase. [ONLINE] Available at: https://firebase.google.com/docs/storage/. [Accessed 01 December 2018].

Lifewire. 2018. Pros and Cons of Splash Pages in Web Design. [ONLINE] Available at: https://www.lifewire.com/splash-pages-pros-cons-3469116. [Accessed 03 April 2019].

Name year Dummy's Codes: Using Java RxTx library for Serial Communication with Arduino - Part 1. Available at: http://dummyscodes.blogspot.com/2014/08/using-java-rxtx-library-for-serial.html. [Accessed 10 November 2018].

Name 2018 Kotlin vs. Java: Which One You Should Choose for Your Next Android App | Netguru Blog on Kotlin. [ONLINE] Available at: https://www.netguru.co/blog/kotlin-java-which-one-you-should-choose-for-your-next-android-app. [Accessed 01 December 2018].

ProgrammableWeb. 2019. Google Maps API | ProgrammableWeb. [ONLINE] Available at: https://www.programmableweb.com/api/google-maps. [Accessed 06 April 2019].

The Firebase Blog. 2018. The Firebase Blog: What's new at Firebase Summit 2018. Available at: https://firebase.googleblog.com/2018/10/whats-new-at-firebase-summit-2018.html. [Accessed 01 December 2018].

Wikipedia. 2019. Dijkstra's algorithm - Wikipedia. [ONLINE] Available at: https://en.wikipedia.org/wiki/Dijkstra%27s_algorithm. [Accessed 10 April 2019].

Minutes from meetings – Semester 1

Meeting Date	Work Completed	Y/N	Work to Do
18/Oct/18	• Discussed grading scheme of project • Discussed project Idea	Y	• Project proposal start • Technologies that I will use

Meeting Date	Work Completed	Y/N	Work to Do
22/Oct/18	• Discussed the diagrams of the project and what my project is about	Y	• Write up a report about the technologies that I will use

Meeting Date	Work Completed	Y/N	Work to Do
5/Nov/18	• Basic report completed • Done over email due to meetings and me working	Y	• Fix the things highlighted by the supervisor • Make the report more aimed at my project

Meeting Date	Work Completed	Y/N	Work to Do
12/Nov/18	• Report Fixed • Technologies changed and looked up • Fixed Harvard referencing.	Y	• Do minutes for all the reports? • Write about alternatives of the technologies I will use.

Meeting Date	Work Completed	Y/N	Work to Do
30/Nov/18	• Minutes fixed • Alternatives wrote up	Y	• Presentation • Spelling fixes • Citation fixes

Minutes from meetings – Semester 2

Meeting	Work Completed	Y/N	Work to Do
Week 1	• Refactored login screen • Used proto.io to decide on the look of the application • Meeting time arrangements	Y	• Plan out timetable • UML diagrams • Start looking at W3C guidelines • Splash screen

Meeting	Work Completed	Y/N	Work to Do
Week 2	• Planned out timetables • UML Diagrams finalised • Develop a splash screen myself to avoid copyright infringements	Y	• Settings to allow users to see their location • Coordinates placeholder • Google Maps API

Meeting	Work Completed	Y/N	Work to Do
Week 3	• Settings to allow users to see their location • Coordinates placeholder • Google Maps API	Y	• Building plan tab • Drew out a building plan • Classes for coding the Dijkstra's algorithm • Cognito form for the industry day

Meeting	Work Completed	Y/N	Work to Do
Week 4	• Building plan tab • Drew out a building plan • Classes for coding the Dijkstra's algorithm • Cognito form for the industry day	Y	• Fix library imports • Arduino connection • Sensors and modules in Arduino code • Connecting Arduino with application

Meeting	Work Completed	Y/N	Work to Do
Week 5	• Fix library imports • Arduino connection • Sensors and modules in Arduino code • Connecting Arduino with application	Y	• Connectivity checks • GitHub documentation and converting all to Java, as had a few Kotlin classes- as that was my previous module and I had to learn the Java way of implementing something • Buzzer test • Coordinates need to be displayed in the placeholder.

Meeting	Work Completed	Y/N	Work to Do
Week 6	• Connectivity checks • GitHub documentation and converted all to Java, as had a few Kotlin classes- as that was my previous module and I had to learn the Java way of implementing something • Buzzer test • Coordinates displayed in the placeholder.	Y	• XML updates to make application look more appealing • Logo of the application will be developed in GIMP • Add help window that helps the users to use the application • Add info tab, for users to contact me the developer with any inquiries

Meeting	Work Completed	Y/N	Work to Do
Week 7	• XML updates to make application look more appealing • Logo of the application developed in GIMP • Added help window that helps the users to use the application • Added info tab, for users to contact me the developer with any inquiries	Y	• Add a Bluetooth file written in Arduino to connect to the application. • Display ip address of Bluetooth • Download nRF Connect on the phone as it gives a debugging tool on the Bluetooth signals from the Arduino. • Add an option to deactivate an account. • Updated Change Password method • Add Parser method, that parses the building plan and adds and creates rooms. • Connect buzzer and button works when I press to buzz. • Add consistent backgrounds in the app and consistent app bar names

Meeting	Work Completed	Y/N	Work to Do
Week 8	• Added a Bluetooth file written in Arduino to connect to the application. • The ip address now shows up on the application i.e. they are talking. • Needed to download nRF Connect on the phone as it gives a debugging tool on the Bluetooth signals from the Arduino, so I knew what was going on when I failed to connect. • Added an option to deactivate an account. • Updated Change Password method to keep the email of signed in user, instead of typing it out again • Added Parser method, that parses the building plan and adds and creates rooms. • Connected buzzer and button works when I press to buzz. • Added consistent backgrounds in the app and consistent app bar names	Y	• Dijkstra's algorithm to obtain a correct graph to display and look at possibly older versions i.e. legacy items to display the thing I would like to display. • Create a poster • Update buttons for a better user experience

Meeting	Work Completed	Y/N	Work to Do
Week 9	• Coded Dijkstra's algorithm gets a correct graph to display and look at possibly older versions i.e. legacy items to display the thing I would like to display. • Created a poster – draft model • Update buttons for a better user experience	Y	• Trying to figure out escape route still, the graph works but only if I specify the nodes statically and calculate the route myself. • Finish the poster • Start the report write up • Test the application

Meeting	Work Completed	Y/N	Work to Do
Week 10	• Finish the poster • Start the report write up • Test the application	Y	• Add a better UI • Allow users to save the escape route diagram onto the phone • Report • Video • Print Poster

Meeting	Work Completed	Y/N	Work to Do
Week 11	• Printed Poster • Added a better UI with diagrams • Added Save button for users to save the diagram • Report Started	Y	• Report • Further Application Improvements • Video

Meeting	Work Completed	Y/N	Work to Do
Week 12	• Tested the application • Report Started • Video Started	Y	• Report • Further Application Improvements • Video End