Cardiograph

Mini-Capstone

Submitted to
Professor Dr. Lynch William & Mr. Tyler Tyrese
ELEC 390: Engineering Team Design Project

Due date: April 14th, 2016

STUDENT NAME: STUDENT ID

Team C

Jafar Abbas (26346650)

Rajithan Sivapatharajah (26592031)

Jordan Flinker (27023286)

Samih Makhams (26583067)

Michael Bcharah (29693076)

ELEC/COEN 390 - Final Submission
Team C
Jafar Abbas (26346650)
Rajithan Sivapatharajah (26592031)
Jordan Flinker (27023286)
Samih Makhams (26583067)
Michael Bcharah (29693076)

A. Mission Statement

The zephyr heart rate sensor measures the heart rate of the user by connecting to an application running on their android smartphones. Therefore, the sensor is strapped on the chest of the user, while the app will continuously monitor the heart rate of an individual. Our team using Android studio and combined with our knowledge of Java will develop an application which monitors the heart beats, detect problems using R-R intervals and algorithms, and alerts the user if the heart rate reaches a dangerous level. According to Heart and Stroke Foundation, "An estimated 1.6 million Canadians are living with heart disease or the effects of a stroke". Hence, this application will allow monitoring systems to be moved from hospitals to smartphones by using our application. Furthermore, the goal of our group is to create an accurate system which would be continuously measuring the heart rate as long as the phone is on and the sensor is connected. In addition, an automated message will be sent to a preferred contact set by the user in case of any emergency notifying of potential danger. Based on a study from Catalyst on Canada's mobile landscape, 68% of the Canadian population uses a smartphone. Therefore, this product will be successful due to the increase in heart diseases and also the ownership of smartphones. The projected cost of the application will be in the range of $5$10 and the strap with the sensor will range in-between 6070. Hence the product will be affordable for the user since our purpose is to aid people in need of such a technology. Our goal is to assure that majority of the population will be able to afford this product. As a result, the target market is to people who have a heart condition and cardiovascular malfunctions, in addition to elderly people. As long as the phone and the strap are in good condition without any malfunctions,

the product will be functioning properly. In addition, one constraint is that the phone should always be within a certain range in order to keep the Bluetooth connection active.

B. Product Backlog

Listed in order of most important to least important

Story Title: Display live heart-rate and ability to store data	**Priority**: essential for sp1	**Size**: 2 Weeks
Summary		
The customer should be able to visualize their current heart-rate as well as being able to connect the bluetooth to the sensor through the application.		
Conversation		
The app should display (in an easy-to-read format) the current heart-rate via a bluetooth connection, which would be able to auto detect the sensor and allow the user to connect to it through the application itself. The app should also have the ability to store sample readings taken.		
Confirmation		
The application detected the sensor and connects through bluetooth. Live heart rate is displayed.		
Notes	**Status**: Completed	
Format could be plain text or graphs (allow user to click on portion of graph to pop out the values)		

Story Title: Implement the settings along with options to add contacts	**Priority**: Secondary for sp1	**Size**: 2 Weeks
Summary		
The customer would be able to create their own personal profile, set notifications and manage his emergency contacts.		
Conversation		
The application should be able implemented with a settings option which would include the ability to create a personal profile with your information. In addition, create and manage the emergency contacts, by having the ability to enter their names and phone numbers.		
Confirmation		

Settings are implemented and a profile option is active. Emergency contacts can be added in the settings.
Notes | **Status**: Completed
Should be a maximum and minimum heart rates based on their given condition.

Total hours for sprint 1= 90 hours

Story Title: Store data of previous results and be able to form a graph	**Priority**: urgent for sp2	**Size**: 2 Week
Summary		
After storing the data, the user would be able to turn this data into the form of a graph and be better able to view and compare and retrieve previous measurements at different points.		
Conversation		
This is a key feature, in the process of building the application seeing that it will be a better way to visualize the different heart rates at different points in the day. Furthermore, with the usage of graphs and other analytics the user would be able to determine certain information such as averages along with high points and low points. In addition, it makes for an easy way for the doctor to interpret and verify results from different times of the day dating back to as far as a month long. Furthermore, we would implement a feature to compare and retrieve previously stored data.		
Confirmation		
Recording data and graphing is an option in the application. The graph can show analytics and averages. Data can be viewed based on record created.		
Notes | **Status**: Completed		
The application should be able to graph different periods of time, ranging from as short as a day to as long as a month and distinguish high and low points of the heart rate from that graph.		

Story Title: Display heart rate sensors battery percentage	**Priority**: Secondary for sp2	**Size**: 2 Week
Summary		
In the top corner of the applications window there will be a battery percentage for the heart rate sensor, which will alert the user in case the battery is almost dead.		
Conversation		
This feature is secondary in terms of importance, however, it is still a key feature that the sensor should have since it alerts the user when it is almost out of battery. By having this feature it is another way of reducing possible reasons as to why the user may be experiencing issues with the heart rate sensor by knowing when it no longer has any battery.		
Confirmation		
Sensor battery displayed. A notification will be sent if it goes under 20%		
Notes	**Status**: Complete	
Battery should contain percentage, and be visible while not being too big since it is not the main focus of the application.		

Story Title: Researching how to implement the tables and graphs needed for sprint 2	**Priority:**	**Size**: 2 Weeks
Summary		
To meet our expectations for sprint 2 and how we would like to display the live readings from the heart rate we had to refer to the internet for certain aspects we weren't sure about.		
Conversation		
Seeing that we did learn the majority of what was required for this app from the tutorials, there was certain items for visualization that were needed that we weren't sure how to implement. With that being said, we turned to the internet to research to determine it would be needed to be done.		
Confirmation		
By doing the research we are able to increase the depth of our code and program more that we initially weren't sure about.		
Notes	**Status**: Complete	

Total hours for Sprint 2 = 100 hours

Story Title: Alert the user and emergency contacts in case of danger	**Priority**: important for sp3	**Size**: 2 Weeks
Summary		
As a vulnerable person (or family member/friend) I want to be alerted when the user's heart-rate reaches critical points so I can check up on them or get immediate medical assistance.		
Conversation		
This is the most important feature of our application: when a user's heart-rate approaches dangerous levels, we want to avoid undetected heart-attacks, strokes, fainting, and abnormally-low pulse so the user and their emergency contacts should be immediately alerted via automated SMS . the SMS will alert the contact that the user is not stable and he should be contacted to ensure his safety		
Confirmation		
Potential danger is activated. If heart rate reaches danger automated msg will be sent.		
Notes	Status: Completed	
False positives will be addressed in another story. A loud ringing of the phone to accompany danger levels would also be a good idea.		

Story Title: Beat-to-beat interval display	**Priority**: sp3 essential	**Size**: 2 Weeks
Summary		
As another method to measure the heart rate and see if there is any discrepancies in the normal heart rate, by viewing it as a beat-to-beat as another precautionary measure.		
Conversation		
This will be done in order to verify if the heart is not functioning properly. By measuring the intervals,we would be able to see if there is anything abnormal. If this is the case it would then notify the user along with the emergency contacts.		
Confirmation		
distinguish between a false positive and a true positive. The option to view graphs by beat to beat interval is part of the app now. Algorithms can detect abnormalities in the readings and alert the user		
Notes	**Status**: Completed	

Story Title: Research required for the algorithm to create beat-to-beat interval display	**Priority:**	**Size**: 2 Weeks
Summary		
In order to be able to confidently display our beat-to-beat interval display we have decided to spend some time to research what is required to complete the task.		
Conversation		
Seeing that we would like to implement this feature but don't feel fully confident that we are able to with the knowledge we have obtained to date, we plan on spending some time to research how to better approach this task so that it can be successfully completed and have a beat-to-beat interval display as another option for the user.		
Confirmation		
Notes	**Status**: Completed	

Story Title: Edge case testing	**Priority:**	**Size**: 2 Weeks
Summary		
Testing the app at both the maximum and minimum limits that will be determined by the user		
Conversation		
After determining arbitrary maximum and minimum limits, we would test to make sure that the application alerts the user when it reaches those points.		
Confirmation		
Irregularities are filtered Only stable measurements are taken in the graphing.		
Notes	**Status**: Complete	

Total hours for Sprint 3 = 85 hours

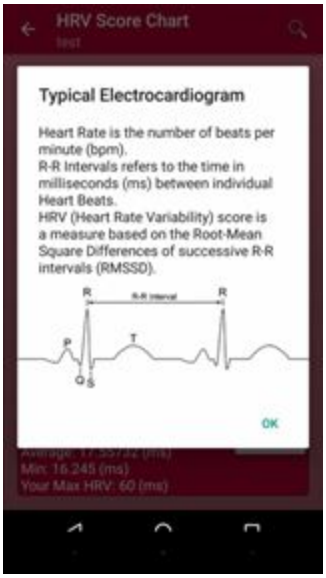
C. Design document

Introduction

A design document is a written description of a software product, that a software designer writes in order to give a software development team overall direction to the architecture of the software project. The software will have three major components; the Bluetooth connection class, the display of the heart rate class and finally the setting class (input of profile of the user and emergency contact). Every single class will contain subclasses that will describe specialized functionalities, such as electro-diagram, troubleshooting connection issues/errors, and recording/storing heart rate data into a database.

Use cases:

UC 1) Defining the medical terms



Prerequisites:

First time opening the app OR

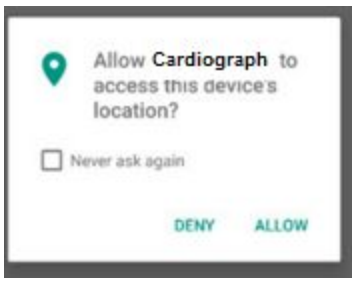
Clicking “Information”

A typical electrocardiogram is shown to the user along with an explanation of the terms “Heart Rate”, “R-R Intervals” and “Heart Rate Variability”. The units are also mentioned, along with the algorithm used to measure heart-rate variability score.

UC 2) Giving the app permission for Bluetooth/sending SMS

Prerequisites:

First time opening the app



The app will ask the user for permission for being able to access the phone's "coarse location" and access the ability to send text messages.

Course location is needed for detecting Bluetooth nearby devices and sending text messages is needed for alerting contacts in case of danger.

UC 3) Troubleshooting prerequisites for recording

Prerequisites:

Permissions granted AND

Bluetooth is not enabled OR

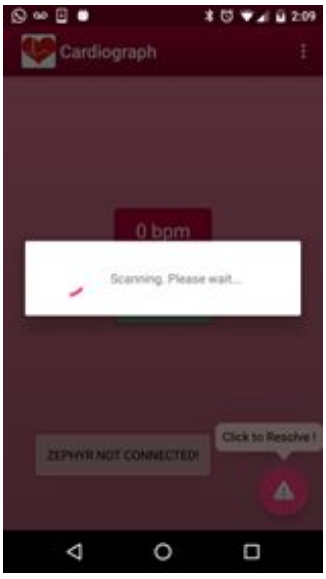
Zephyr is not connected OR

No emergency contacts saved OR

Important setting is missing

A message appears at the bottom of the screen with the appropriate error message and a tooltip saying "Click to Resolve!"

The button to click will automatically redirect the user to the appropriate settings/contacts page or try to enable Bluetooth or connect to the zephyr.



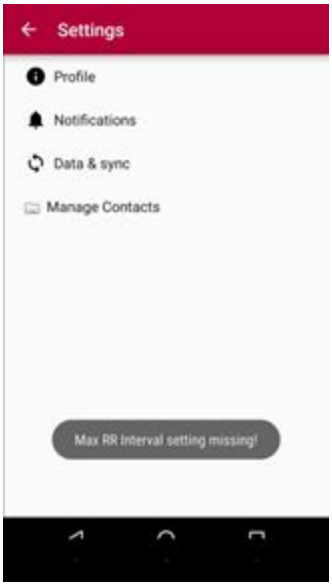
UC 4) Inputting settings

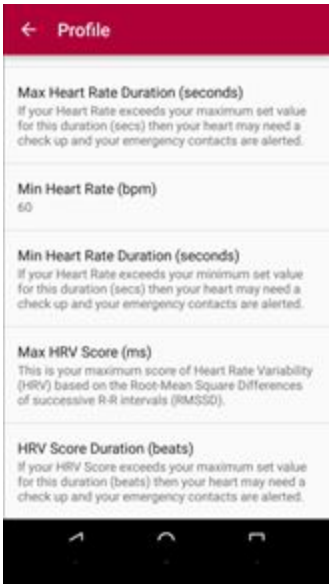
The settings page allows access to the user's profile, notifications preferences, emergency contacts and the zephyr device's information.

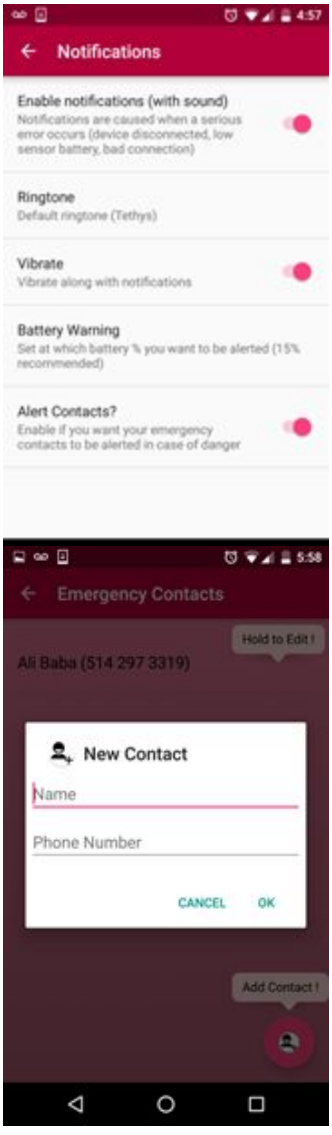
In the profile, the user can set the detection parameters such as the max/min heart-rate and max HRV score as well as the according sensitivity before alerting contacts.

In the notifications, the user can set the minimum sensor battery alert and disable/enable notifications and alerting contacts.

In the Data & Sync, the user can manually enter his zephyr's mac address or name.







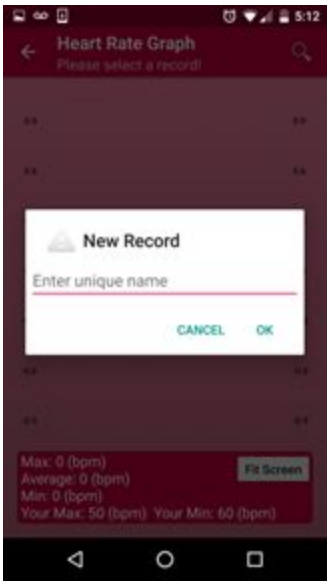
UC 5) Adding contacts

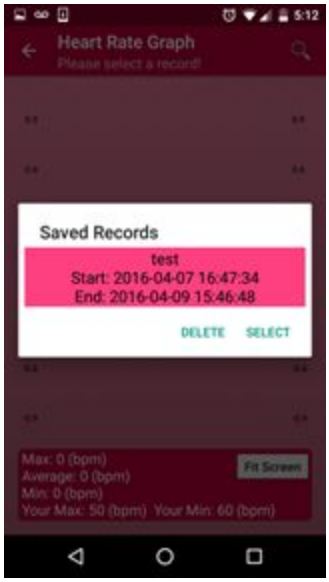
In the "Manage contacts" page, the user can add his emergency contacts by clicking the button in the bottom right where it says "Add Contact!" and filling out the form.

The user can also choose to edit or delete a contact by pressing and holding one of his contacts.

UC 6) Selecting or creating a record

All recorded data is attached to a “record”.
To record new data, the user has to create a new record or select an old one. Records should have unique names.
To view old data, the user has to select an old record based on the name or their start/end timestamps.





UC 7) Recording heart data

Prerequisites:

All profile settings set AND
Bluetooth enabled AND
Zephyr connected AND
Record selected AND
If alerting contacts enabled, 1 contact should exist

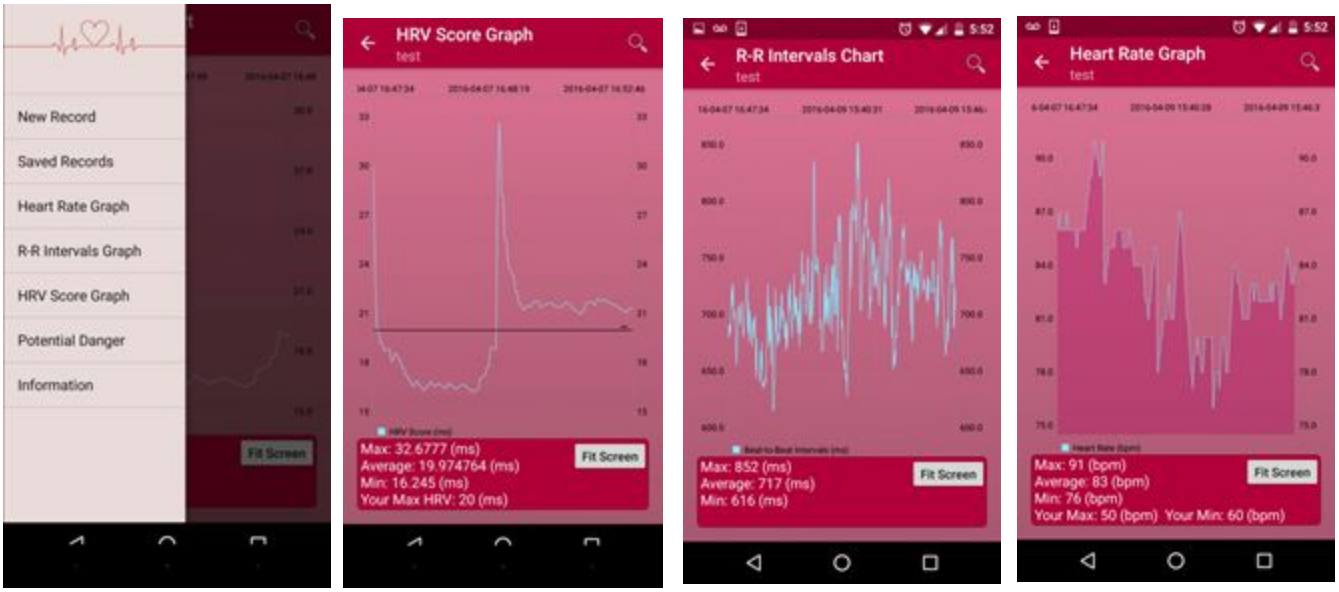
After all the prerequisites are met, the user will finally see the “Start” and “Stop” buttons.
After clicking start, the live heart rate will show and will be replaced by a blue “Record” button.
The “Record” button, when triggered, shows a green circle and starts saving the sensor’s data into the database and tells the app the start updating the live charts if they are showing.

UC 8) Viewing charts

Prerequisite:

Record selected

When the user clicks on “View Data”, he will be redirected to the page where you can visualize your live heart rate and live HRV score, you can also view charts associated with old records.
At the bottom of each chart, the max/min and average value associated with it, as well as a reminder of your set values.



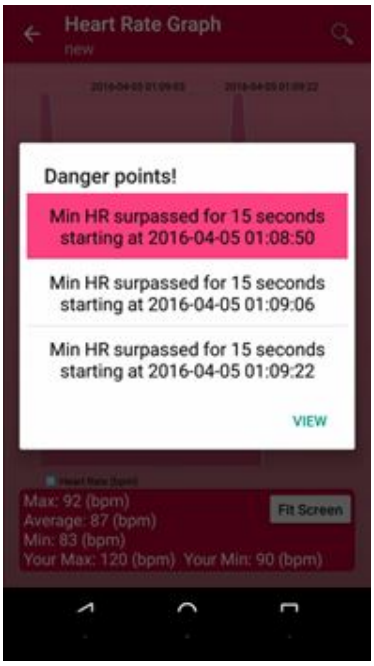
UC 9) Evaluating old records for danger zones

Prerequisites:

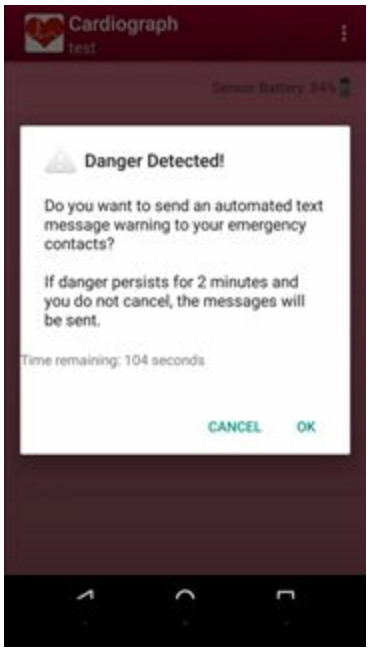
Record selected

Max/min/duration settings set

While going through old charts, the user can click “Potential Danger” to see view where on the graph his max/min heart rate or HRV score was surpassed for the durations he set.
After selecting one of the danger points and clicking “View”, the chart will automatically move to that area.



UC 10) Alerting contacts of danger



Prerequisites:

Recording live data

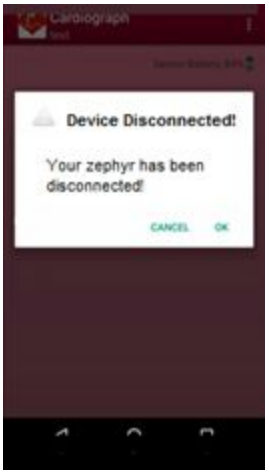
Alerting contacts enabled

Max/min/duration settings set

While the user is using the app and zephyr to record his heart's data and a potential danger is detected based on his settings, the phone will vibrate/ring to notify the user.

The user will be prompted with an interface asking to confirm/deny sending a SMS text message to his contacts. The alert also comes with a 2 minute countdown, after which the messages will be automatically sent if ignored.

UC 11) Alert when sensor is disconnected



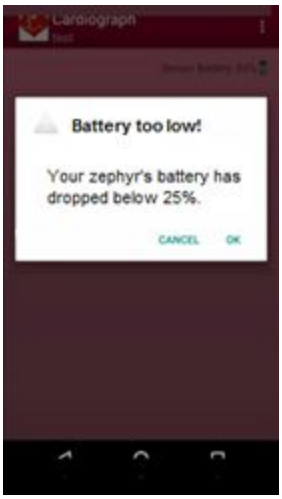
Prerequisites:

Recording live data, Device disconnected

If the user is recording live data and the sensor strap comes loose or somehow the circuit is no longer complete a dialog will be displayed asking the user to acknowledge.

The user will be notified with a vibration/ring if those settings are enabled. Notifications also show on the phone's main screen.

UC 12) Alert when sensor Battery too low



Prerequisites:

Notifications enabled, battery minimum level set

Recording live data, Sensor battery below his level

If the user is recording live data and the sensor battery drops below his minimum set percentage a dialog will be displayed asking the user to acknowledge.

The user will be notified with a vibration/ring if those settings are enabled. Notifications also show on the phone's main screen.

Conclusion

In conclusion, this application has a good software design that is coherent and follows a logical path. The Bluetooth connection and the display of the heart rate was successfully displayed after many test cases were done.

Algorithms used

To detect heart problems, the app's algorithms rely on the settings set by the user which are presumably recommendations given by the user's doctor.

These settings are:

1) Maximum Heart Rate, Maximum Heart Rate Duration
2) Minimum Heart Rate, Minimum Heart Rate Duration
3) Maximum HRV (heart rate variability) Score, Max HRV Duration

Building on these settings, we can detect problems using the data provided by the zephyr sensor.

Problem detection:

1) Maximum Heart Rate, Maximum Heart Rate Duration

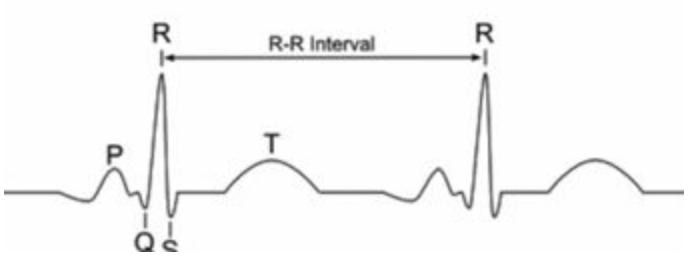
This is straightforward to compute, whenever a heart rate value is above the maximum, we increment the max duration timer. If this timer reaches the maximum duration, then the contacts should be alerted.
Therefore, a problem is detected when the current heart rate surpasses the maximum heart rate for the maximum duration.

2) Minimum Heart Rate, Minimum Heart Rate Duration

The same logic for 1) is reused in this case, except it's based on a minimum value for heart rate.

3) Maximum HRV (heart rate variability) Score, Max HRV Duration

This one is more interesting because the formula for HRV is based on a standard formula used by heart experts to measure the variability of time successive heart beats (defined by R-R intervals).
This formula is known as "RMSSD" which stands for Root Mean Square Successive Differences is applied to the individual heart beat timestamps that are fed by the zephyr sensor.



$$\sqrt{\frac{1}{N-1}\left(\sum_{i=1}^{N-1}\left((R-R)_{i+1}-(R-R)_i\right)^2\right)}$$

Typical Electrocardiogram showing R-R Intervals RMSSD formula

To start, let's consider the information provided by the zephyr exactly every 1 second:

Byte/Bit	7	6	5	4	3	2	1	0	Field
0	STX								STX
1	0x26								Msg ID
2	55								DLC
3	Firmware ID								Payload
5	Firmware Version								
7	Hardware ID								
9	Hardware Version								
11	Battery Charge Indicator								
12	Heart Rate								
13	Heart Beat Number								
14	Heart Beat Timestamp #1 (Newest)								
16	Heart Beat Timestamp #2								
18	Heart Beat Timestamp #3								
20	Heart Beat Timestamp #4								
22	Heart Beat Timestamp #5								
24	Heart Beat Timestamp #6								
26	Heart Beat Timestamp #7								
28	Heart Beat Timestamp #8								
30	Heart Beat Timestamp #9								
32	Heart Beat Timestamp #10								
34	Heart Beat Timestamp #11								
36	Heart Beat Timestamp #12								
38	Heart Beat Timestamp #13								
40	Heart Beat Timestamp #14								
42	Heart Beat Timestamp #15 (Oldest)								
44	Reserved								
46	Reserved								
48	Reserved								
50	Distance								
52	Instantaneous speed								
54	Strides								
55	Reserved								
56	Reserved								
58	CRC								CRC
59	ETX								ETX

We can see that the zephyr provides the heart rate in beats per minute, the number of heart beats counted and the timestamps of the last 15 heart beats.

Note: The timestamps are counted using an unsigned integer as an internal counter which rolls over after 65535 ms. The heart beat number is also an unsigned integer which rolls over after 255. This rolling over is accounted for before saving the data into the database.

Computing the HRV Score:

Using the timestamps between individual heart beats, we can calculate the R-R intervals. With every pair of R-R Intervals we can calculate the new RMSSD or HRV Score.

We consider the initial value of the RMSSD to be half the user's maximum setting. This is a good starting point so we don't get an unrealistic value if the first R-R intervals are abnormally high or low.

The RMSSD formula involves adding up all the previous R-R intervals. This will take longer and longer to calculate as we record for longer. Instead, we derived the formula for computing the new RMSSD using the old value which has the old sum encoded in it.

Note: Through testing, we observed the R-R intervals to sometimes be too long. This is a flaw of the zephyr sometimes skipping a beat, or whatever it may be. These successive R-R intervals that are longer than 400 milliseconds are filtered out.

Static Model

The application is created using many classes and function in order to get the desired result from the heart rate monitor. The coding will help connect the device to the phone and it will start measuring the heart rate. In addition, we will be coding where it will store the data and display it in a graph while when someone has a risk of getting a heart attack, it will contact the emergency right away.

BluetoothConnection class

protected ZephyrProtocol _protocol;
protected NewConnectedListener _NConnListener;
protected BluetoothAdapter mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
protected ArrayList<BluetoothDevice> mDeviceList = new ArrayList<BluetoothDevice>();
protected BluetoothDevice mZephyr;
private Context mMainPageContext;
private MainActivity mMainActivity;
private ProgressDialog scanningDialog;
protected boolean recordMeasurement = false;
static protected boolean updateLiveHeartRateChart = false;
static protected boolean updateLiveECGChart = false;
private final int HEART_RATE = 0x100;
private final int BATTERY_PERCENT = 0x102;
private final int HEART_BEAT_TIMESTAMPS = 0x103;
private final int HEART_BEAT_NUMBER = 0x104;

The BluetoothConnection class is a class created to make sure the device connects properly to the cellular phone.

public BluetoothConnection (Context ctx)

This function scans for Bluetooth devices.

private void pairDevice(BluetoothDevice device)

This function will pair a device that has its Bluetooth enabled.

private void unpairDevice(BluetoothDevice device)

This function will disconnect a device that was previously connected.

public void connectWithZephyr()

This function will connect the Zephyr with the phone.

public void connectListener()

This function will connect the listener from acting on received messages.

public void disconnectListener()

This function will disconnect the listener from acting on received messages.

private void showSaveDeviceDialog (final BluetoothDevice device)

This function will save the dialogs between the Zephyr and the phone.

public void enableBluetooth()

This function enables Bluetooth on the app.

public void setBluetoothListener()

This function ensures that the Zephyr stays connected to the app.

DB Contract

```
protected static final class ContactsEntry implements BaseColumns {
protected static final String TABLE_NAME = "Contacts";
protected static final String NAME_COLUMN = "Name";
protected static final String NAME_COLUMN_TYPE = "TEXT";
protected static final String PHONE_COLUMN = "Phone";
protected static final String PHONE_COLUMN_TYPE = "TEXT";
protected static final String PRIORITY_COLUMN = "Priority";
protected static final String PRIORITY_COLUMN_TYPE = "TEXT";
protected static final String ACTION_COLUMN = "Action";
protected static final String ACTION_COLUMN_TYPE = "TEXT";
```

This class implements the layout of the contacts and the heart rate display window. It contains three subclasses.

public static final class ContactsEntry implements BaseColumns

This subclass defines the layout of the contact list. It displays the name, phone number, and the priority of this contact (whether it should be the first one to call in case of emergency).

public static final class InstantaneousHeartRateEntry implements BaseColumns

This subclass implements the layout of the instantaneous heart rate as it's recorded second by second in the database.

public static final class AverageHeartRateEntry implements BaseColumns

This subclass implements the layout of the average heart rate as it's recorded on an average time.

DataGraph class

```
private DB_Helper myDBHelper;
static protected List<heartRate> mHeartRates = new ArrayList<>();
static protected List<rrInterval> mRRIntervals = new ArrayList<>();
static protected LineChart mChart;
/** Heart Rate chart stuff **/
static private ArrayList<Entry> mHR_Entries;
static private ArrayList<String> mHR_labels;
static private LineDataSet mHR_DataSet;
private boolean limit_lines_add_once = false;
private LimitLine Max_LimitLine;
private LimitLine Min_LimitLine;
/** RR interval chart stuff **/
static private ArrayList<Entry> mRR_Entries;
static private ArrayList<String> mRR_labels;
```

static private LineDataSet mRR_DataSet; private Button delete_alL_btn; private Button reset_scale_btn; /** Drawer **/ private ListView mDrawerList; private DrawerLayout mDrawerLayout; private ArrayAdapter<String> mAdapter; private ActionBarDrawerToggle mDrawerToggle;
This class graphs the heart rate data that is stored in the database. private void listAllHeartRates() This function will get the data for all instantaneous heart rates. public boolean onOptionsItemSelected(MenuItem item) This function let's the user click on the home button or scroll up and down through the heart rate data.

SettingsActivity class
This class implements the settings button. private static boolean isXLargeTablet(Context context) This function binds a preference's summary to its value. More specifically, when the preference's value is changed, its summary (line of text below the preference title) is updated to reflect the value. The summary is also immediately updated upon calling this method. The exact display format is dependent on the type of preference. private static void bindPreferenceSummaryToValue(Preference preference)

This function sets the listener to watch for value change. It triggers the listener immediately with the preference's current value.

private void setupActionBar()

This function shows the Up button in the action bar.

public boolean onMenuItemSelected(int featureId, MenuItem item)

This function let's the user click on the home button or scroll up and down through the heart rate data.

public void onBuildHeaders(List<Header> target)

This function stops fragment injection in malicious applications.

protected boolean isValidFragment(String fragmentName)

This fragment shows the general preferences only. It is used when the activity is showing a two-pane settings UI.

public static class ProfilePreferenceFragment extends PreferenceFragment (subsclass)

This subclass binds the summaries of EditText/List/Dialog/Ringtone preferences to their values. When their values change, their summaries are updated to reflect the new value.

ContactsPage.java

```
private ToolTipView mAddContactToolTipView;
private ToolTipView mHoldToEditToolTipView;
private List<String> mContacts;
private List<contact> true_Contacts;
private ArrayAdapter<String> mContactsAdapter;
```

This class records the contact information of the person/institution that should be reached in case of emergency. It has name and number field and more than one contact can be added to the emergency contact list. The private data members are name, phone, priority, action. Here are the most important functions and the functionality of each.

We will elaborate on two functionality on this class and we chose private void showAddContactDialog and private void showEditContactDialog. The

showAddContactDialog adds the contact to the emergency list while the showEditContactDialog edits the number if the person/institution has changed the number.

DB_Helper.java

```
private static final String CONTACTS_TABLE_CREATE = "CREATE TABLE " +
DB_Contract.ContactsEntry.TABLE_NAME +
" (" + DB_Contract.ContactsEntry.NAME_COLUMN + " " +
DB_Contract.ContactsEntry.NAME_COLUMN_TYPE + "," +
DB_Contract.ContactsEntry.PHONE_COLUMN + " " +
DB_Contract.ContactsEntry.PHONE_COLUMN_TYPE + "," +
DB_Contract.ContactsEntry.PRIORITY_COLUMN + " " +
DB_Contract.ContactsEntry.PRIORITY_COLUMN_TYPE + "," +
DB_Contract.ContactsEntry.ACTION_COLUMN + " " +
DB_Contract.ContactsEntry.ACTION_COLUMN_TYPE
+ ");";
private static final String INSTANTANEOUS_HEART_RATE_TABLE_CREATE =
"CREATE TABLE " + DB_Contract.InstantaneousHeartRateEntry.TABLE_NAME +
" (" + DB_Contract.InstantaneousHeartRateEntry.DATE_COLUMN + " " +
DB_Contract.InstantaneousHeartRateEntry.DATE_COLUMN_TYPE + "," +
DB_Contract.InstantaneousHeartRateEntry.HEART_RATE_COLUMN + " " +
DB_Contract.InstantaneousHeartRateEntry.HEART_RATE_COLUMN_TYPE + "," +
DB_Contract.InstantaneousHeartRateEntry.NOTE_COLUMN + " " +
DB_Contract.InstantaneousHeartRateEntry.NOTE_COLUMN_TYPE
+ ");";
private static final String AVERAGE_HEART_RATE_TABLE_CREATE = "CREATE
TABLE " + DB_Contract.AverageHeartRateEntry.TABLE_NAME +
" (" + DB_Contract.AverageHeartRateEntry.DATE_COLUMN + " " +
DB_Contract.AverageHeartRateEntry.DATE_COLUMN_TYPE + "," +
```

```
DB_Contract.AverageHeartRateEntry.HEART_RATE_COLUMN + " " +
DB_Contract.AverageHeartRateEntry.HEART_RATE_COLUMN_TYPE + "," +
DB_Contract.AverageHeartRateEntry.NOTE_COLUMN + " " +
DB_Contract.AverageHeartRateEntry.NOTE_COLUMN_TYPE
+ ");";
private static final String RR_INTERVALS_TABLE_CREATE = "CREATE TABLE " +
DB_Contract.RRIntervals.TABLE_NAME +
" (" + DB_Contract.RRIntervals.DATE_COLUMN + " " +
DB_Contract.RRIntervals.DATE_COLUMN_TYPE + "," +
DB_Contract.RRIntervals.RR_COLUMN + " " +
DB_Contract.RRIntervals.RR_COLUMN_TYPE + "," +
DB_Contract.RRIntervals.NOTE_COLUMN + " " +
DB_Contract.RRIntervals.NOTE_COLUMN_TYPE
+ ");";
private static final String CONTACTS_TABLE_DROP = "DROP TABLE IF EXISTS " +
DB_Contract.ContactsEntry.TABLE_NAME + ";";
private static final String INSTANTANEOUS_HEART_RATE_TABLE_DROP = "DROP
TABLE IF EXISTS " + DB_Contract.InstantaneousHeartRateEntry.TABLE_NAME + ";";
private static final String AVERAGE_HEART_RATE_TABLE_DROP = "DROP TABLE IF
EXISTS " + DB_Contract.AverageHeartRateEntry.TABLE_NAME + ";";
private static final String RR_INTERVALS_TABLE_DROP = "DROP TABLE IF EXISTS "
+ DB_Contract.RRIntervals.TABLE_NAME + ";";
```

Public void onCreate(SQLiteDatabase sqLiteDatabase)

The application will constantly will get updated heart rate from the heart rate monitor. It will record down all the data into a table.

public void insertContact(String name, String phone, String priority, String action)

The application will insert the contact/user to the phone by recording down the name, the telephone number and the priority and the action it needs to be taken.

public Cursor getAllContacts()

It will take down all the contact into the list.

public void deleteAllInstantaneousHeartRates()

This will clear the entire list of the data.

public Cursor getAllInstantaneousHeartRates()

This will constantly record the heart rate of a person every second.

public void removeContactByPhone(String phone)

This will delete a contact that was added into the application.

MainActivity.java

```
private String BLUETOOTH_NOT_ENABLED = "BLUETOOTH NOT ENABLED!";
private String NO_DEVICES_FOUND = "NO DEVICES FOUND!";
private String ZEPHYR_NOT_CONNECTED = "ZEPHYR NOT CONNECTED!";
public ArrayList<String> PROBLEMS_DETECTED;
/** UI **/
private ToolTipView mErrorWarningToolTipView;
protected boolean batteryDialogShown = false;
protected AlertDialog batteryLowDialog;
protected boolean badConnectionDialogShown = false;
protected AlertDialog badConnectionDialog;
protected boolean deviceDisconnectedDialogShown = false;
protected AlertDialog deviceDisconnectedDialog;
private FloatingActionButton warning_fab;
private TextView warning_tv;
protected TextView live_pulse_tv;
protected TextView sensor_battery_tv;
private Button measure_btn;
private Button stop_measure_btn;
private Button start_recording_btn;
private Button view_data_btn;
/** Backend **/
protected DB_Helper myDBHelper;
private BluetoothConnection mBluetoothConnection;
private SharedPreferences.OnSharedPreferenceChangeListener prefListener;
private SharedPreferences prefs;
```

public void onStart()

The application will start recording and measuring one's heart rate.

public void onDestroy()

This will disconnect the connection between the device and the phone.

public void onStop()

This will stop measuring the heart rate from the heart rate monitor.

public boolean onCreateOptionsMenu(Menu menu)

This goes to the setting menu.

NewConnectedListener class

```
private Handler _aNewHandler;
private int GP_MSG_ID = 0x20;
private int GP_HANDLER_ID = 0x20;
private int HR_SPD_DIST_PACKET =0x26;
private final int HEART_RATE = 0x100;
private final int BATTERY_PERCENT = 0x102;
private final int HEART_BEAT_TIMESTAMPS = 0x103;
private final int HEART_BEAT_NUMBER = 0x104;
private HRSpeedDistPacketInfo HRSpeedDistPacket = new HRSpeedDistPacketInfo();
```

public void Connected(ConnectedEvent<BTClient> eventArgs)

The Zephyr will connect to the device and will ensure there is a data connection where the information can be sent to the phone and vice versa.

Testing Document

Introduction

The application is called Cardiograph, and its purpose is to monitor the heart rate of a person with a heart condition (or prone to having a heart condition). Testing of the app was done on individuals with differing levels of heart conditions. The heart monitor was strapped on their

chest and they were asked to run at a slow pace, and then to accelerate to their top speed. The heart beats were monitored and compared to medical research results. The tests were performed over a period of 3 days.

Methodology

One of the ways to help design our application is to use the monitor on actual person. In order to verify the acceptance test, we measured the heart rate of a person through the device and compare it to the heart measured manually using the traditional index and middle fingers methods. In addition, we compared the heart rate result using another heart rate monitor from our Samsung Galaxy S5 to prevent from any error. As a result, the actual heart rate monitor produced the same result as the Samsung Galaxy S5 heart rate and the old-fashioned traditional finger measurement. The test we designed were a physical test. We made sure that the device connects to the heart rate monitor through a Bluetooth connection. Once the connection is made, we checked if the heart rate is displaying on the application. The tests performed were mainly to verify the responsiveness of the Zephyr heart rate monitor. This was a physical test.

All the code that we wrote specifically addresses the physical test that we defined above.

Test cases

1. Bluetooth connection test

Test id	Tc-3-2	Associated Backlog Item (or method for unit test)	PBI # 3.2
Description	Verify successful connection of modules Zephyr and Nexus 5		
Acceptance criteria	Communication between the two devices is established until disconnected by user		
Designed by	Our team	Run by	Instructor
Product Owner Check			

Initial conditions	Application has been started Heart rate monitor has been initialized
Inputs	Heart Rate Beats
Output	Displayed Beats on application

Test Procedure

Step	Action	Results	Pass/Fail/Comments
1	Run the app	App runs smoothly	Pass
2	Connect the Zephyr	Bluetooth connection established	Pass
3	Test heart rate monitor	Heart beats displayed successfully	Pass

2. Heart rate display on screen

Test id	Tc-3-2	Associated Backlog Item (or method for unit test)	PBI # 3.2
Description	Verify if heart rate display and be able to store data		
Acceptance criteria	Communication between the two devices and starts to record data and stores		
Designed by	Our team	Run by	Instructor
Product Owner Check			

Initial conditions	Application has been started to measure heart rate live It is storing the data		
Inputs	Heart Rate Beats		
Output	Displayed Beats on application and being recorded		
Test Procedure			
Step	Action	Results	Pass/Fail/Comments
1	Run the app	App runs smoothly	Pass
2	Display heart rate	Shows the heart rate	Pass
3	Records Data	Stores the Data	Pass

3. Setting menu display on screen

Test id	Tc-3-2	Associated Backlog Item (or method for unit test)	PBI # 3.2
Description	Verify if setting menu is displaying the appropriate options		
Acceptance criteria	Communication between the two devices is established until disconnected by user		
Designed by	Our team	Run by	Instructor
Product Owner Check			
Initial conditions	Application shows the setting option It is properly displaying the options to choose/modify through setting		

Inputs	Application opens up		
Output	Displayed Correctly the Setting		
Test Procedure			
Step	Action	Results	Pass/Fail/Comments
1	Run the app	App runs smoothly	Pass
2	Check Setting Button	Gives various options to modify	Pass

4. Graphing data

Test id	Tc-3-2	Associated Backlog Item (or method for unit test)	PBI # 3.2
Description	Verify if record can be displayed in the form of graph		
Acceptance criteria	A graph formed with different measurement shown at different times		
Designed by	Our team	Run by	Instructor
Product Owner Check			
Initial conditions	Application shows the graph and the user can see the different measurement taken at times and high and low points can distinguished		

Inputs	Application opens up		
Output	Displayed Correctly the graph		
Test Procedure			
Step	Action	Results	Pass/Fail/ Comments
1	Run the app	App runs smoothly	Pass
2	Select view data	Gives record to choose from	Pass

5. Disconnection alert

Test id	Tc-3-2	Associated Backlog Item (or method for unit test)	PBI # 3.2
Description	Verify if device alerts the user in case of disconnection		
Acceptance criteria	A notification alerts the user when sensor disconnects in notification panel plus alert sound should be produced		
Designed by	Our team	Run by	Instructor
Product Owner Check			
Initial conditions	Application notifies the user		
Inputs	Sensor disconnects from the user		
Output	Displayed alert message on the phone plus notification		

Test Procedure			
Step	Action	Results	Pass/Fail/ Comments
1	Run the sensor	App runs smoothly	Pass
2	Disconnect sensor	Gives notification in the panel plus alerts the user	Pass

6. Potential danger alert

Test id	Tc-3-2	Associated Backlog Item (or method for unit test)	PBI # 3.2
Description	alerts the user in case of emergency and sends automated message to an emergency contact		
Acceptance criteria	if app alerts the user of dangerous heart rate in addition to generating an automated message to emergency contact		
Designed by	Our team	Run by	Instructor
Product Owner Check			
Initial conditions	application measures heart rate and if its above max or below min a notification should be sent plus automated message would be generated		
Inputs	max and min heart rates		
Output	Automated message sent to emergency contact		

Test Procedure			
Step	Action	Results	Pass/Fail/ Comments
1	Set max and min heart rates	Lower max is set for test purposes	Pass
2	Alert the user and option to send automated message	If not cancelled by user a message will be automatically sent	Pass

Results

3 bugs detected. 100% Pass.

There were three testing activities. The first test was running the app. It involved turning on the app and clicking on the different options in the settings. The second test was connecting the Zephyr through Bluetooth. We simply activated the Bluetooth option in the app, and the device was located and connection was established. The third test was testing the heart rate monitor. The Zephyr was strapped on the chest of a healthy person. He was asked to run at a fast pace for 10 seconds, and his heart beats were monitored all along. The fourth test was graphing data which involved graphing a record providing detailed measurements, another test involved testing if the application alerts the user in case the sensor disconnects by sending a notification which will displayed in the notification panel in addition to an alert so the user would verify the sensor and reconnect. Finally, we tested the potential danger setting where the application alerts and notifies the user in case his heart rate raises above the Max heart rate or below the Min heart rate, if the user does not respond within 20 seconds, the application will send an automated message to an emergency contact already saved in the setting.In conclusion. this app was successful. We encountered no issues during testing and all the code run smoothly with barely any bugs.

Edges cases

1. Separating Zephyr from phone

The zephyr was brought to a distance of approximately 160 m from the phone until a perceived disconnection occurred.

2. Strap slides from user

The strap (with the zephyr attached to it) was deliberately pulled from user. A message appeared on the screen indicating that the device has been disconnected.

3. Battery completely drains out

The Zephyr was worn on until the built-in battery completely emptied. Once the battery percentage (as displayed in the Cardiograph) reached zero, a message appeared warning the user that the Zephyr has been disconnected.

Libraries

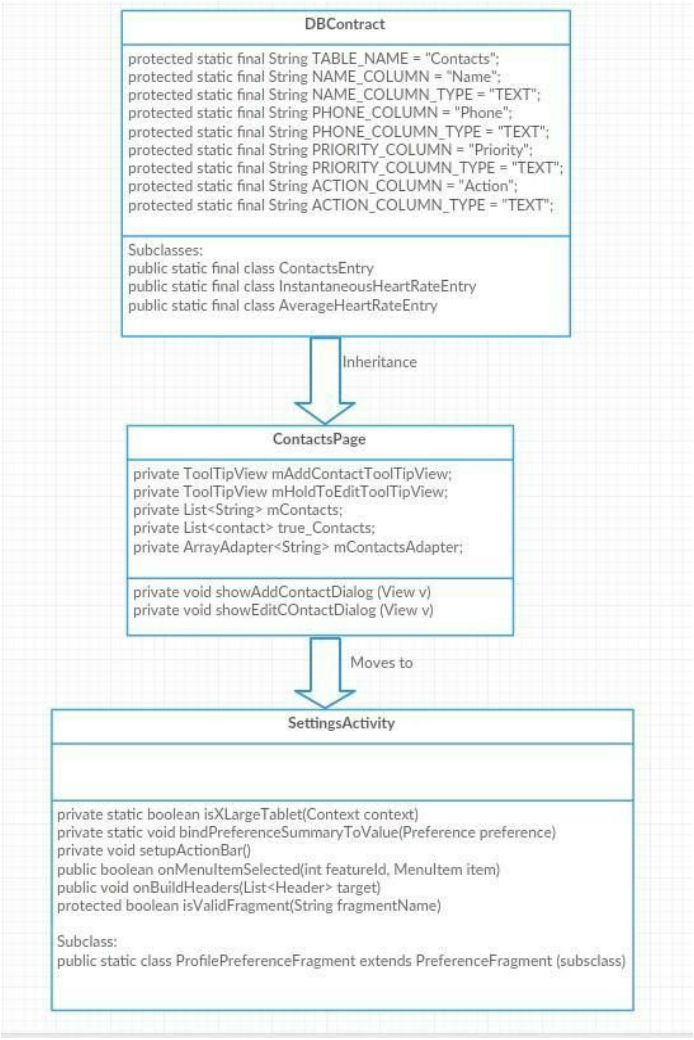
Zephyr library: This library provides all the functions to establish a Bluetooth connection between the device and the phone. It also enables the basic display of the heart rate on the screen. In addition, the functions that permit to identify the Zephyr are all from this library.

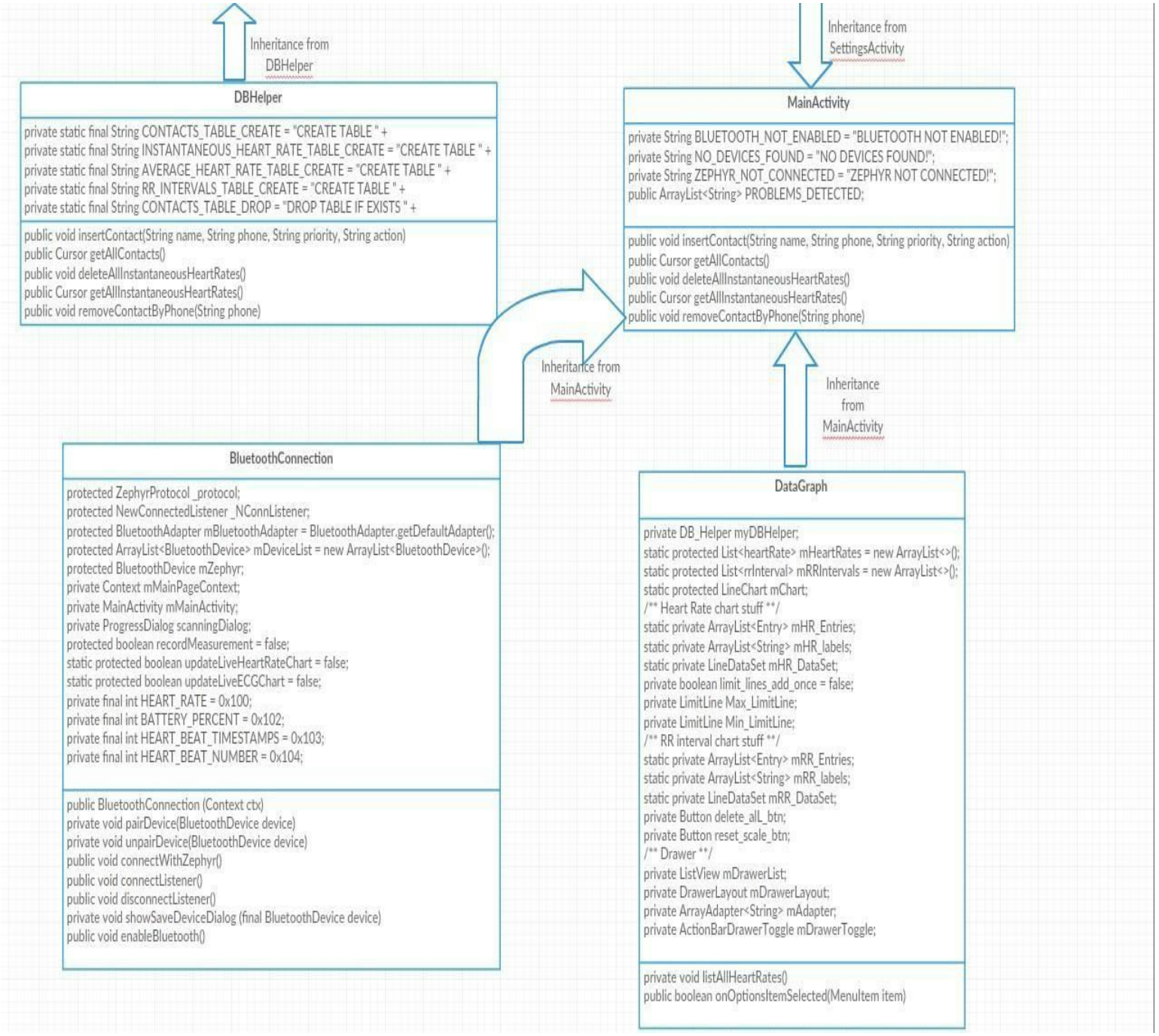
ToolTip library: This library enables the display of important messages and warnings. This includes the phone disconnect warning, battery percentage display, and heart rate notice (if drops below a certain level).

MPAndroid Chart library: A powerful Android chart view / graph view library, supporting line- bar- pie- radar- bubble- and candlestick charts as well as scaling, dragging and animations. Provides the features to display the instantaneous bar chart of every heartbeat (every second or so).

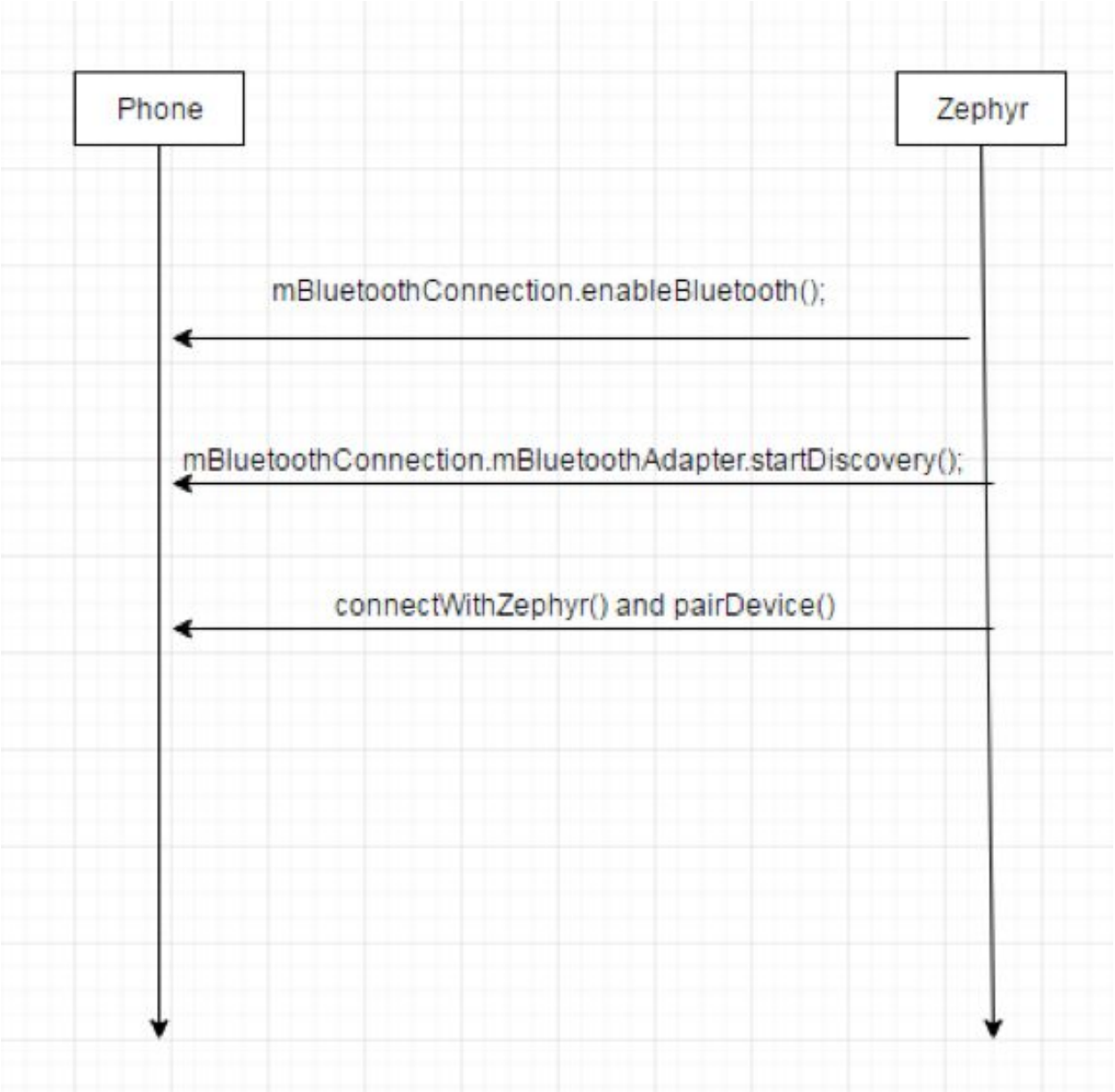
Dynamic Model

Class Diagram

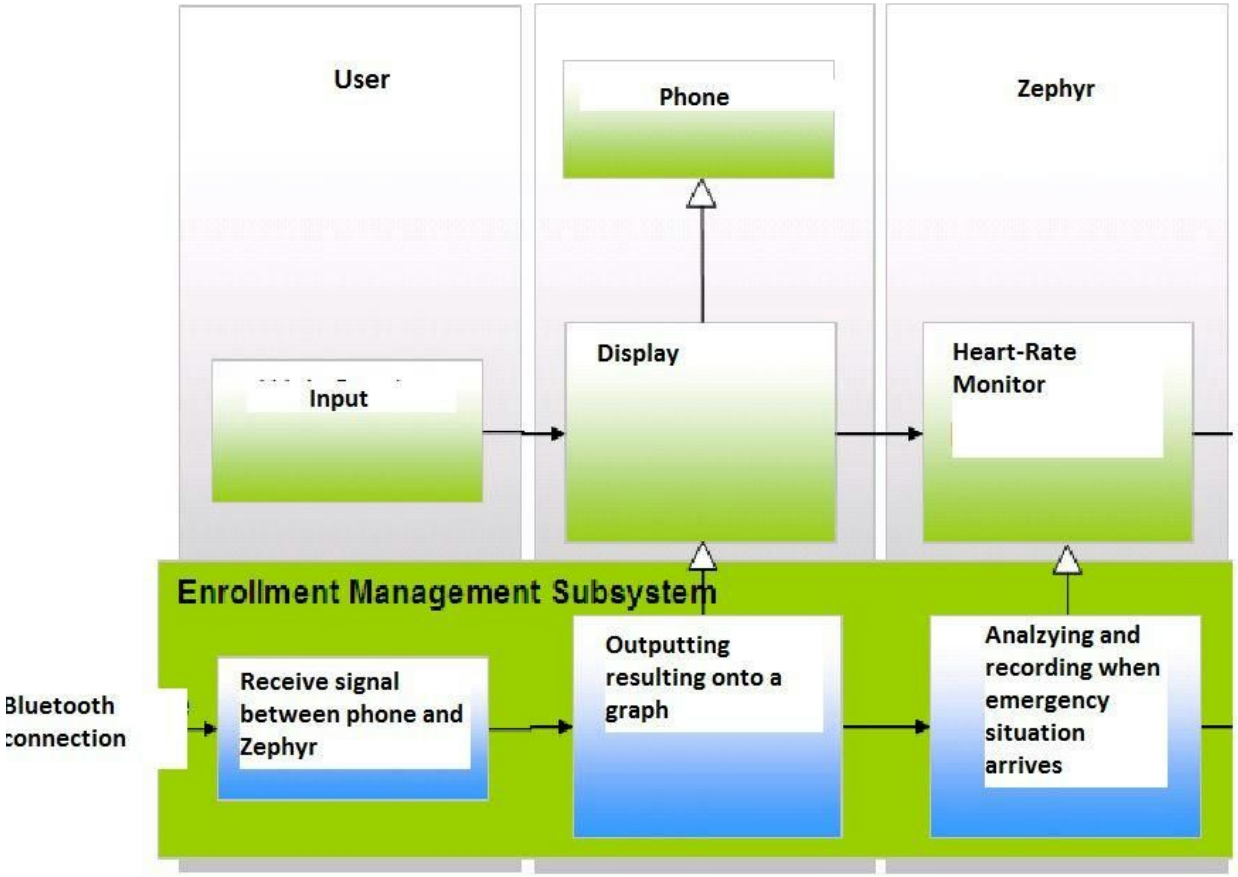



Inheritance from
DBHelper
DBHelper
private static final String CONTACTS_TABLE_CREATE = "CREATE TABLE " +
private static final String INSTANTANEOUS_HEART_RATE_TABLE_CREATE = "CREATE TABLE " +
private static final String AVERAGE_HEART_RATE_TABLE_CREATE = "CREATE TABLE " +
private static final String RR_INTERVALS_TABLE_CREATE = "CREATE TABLE " +
private static final String CONTACTS_TABLE_DROP = "DROP TABLE IF EXISTS " +
public void insertContact(String name, String phone, String priority, String action)
public Cursor getAllContacts()
public void deleteAllInstantaneousHeartRates()
public Cursor getAllInstantaneousHeartRates()
public void removeContactByPhone(String phone)
Inheritance from
SettingsActivity
MainActivity
private String BLUETOOTH_NOT_ENABLED = "BLUETOOTH NOT ENABLED!";
private String NO_DEVICES_FOUND = "NO DEVICES FOUND!";
private String ZEPHYR_NOT_CONNECTED = "ZEPHYR NOT CONNECTED!";
public ArrayList<String> PROBLEMS_DETECTED;
public void insertContact(String name, String phone, String priority, String action)
public Cursor getAllContacts()
public void deleteAllInstantaneousHeartRates()
public Cursor getAllInstantaneousHeartRates()
public void removeContactByPhone(String phone)
Inheritance from
MainActivity
Inheritance
from
MainActivity
BluetoothConnection
protected ZephyrProtocol _protocol;
protected NewConnectedListener _NConnListener;
protected BluetoothAdapter mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
protected ArrayList<BluetoothDevice> mDeviceList = new ArrayList<BluetoothDevice>();
protected BluetoothDevice mZephyr;
private Context mMainPageContext;
private MainActivity mMainActivity;
private ProgressDialog scanningDialog;
protected boolean recordMeasurement = false;
static protected boolean updateLiveHeartRateChart = false;
static protected boolean updateLiveECGChart = false;
private final int HEART_RATE = 0x100;
private final int BATTERY_PERCENT = 0x102;
private final int HEART_BEAT_TIMESTAMPS = 0x103;
private final int HEART_BEAT_NUMBER = 0x104;
public BluetoothConnection (Context ctx)
private void pairDevice(BluetoothDevice device)
private void unpairDevice(BluetoothDevice device)
public void connectWithZephyr()
public void connectListener()
public void disconnectListener()
private void showSaveDeviceDialog (final BluetoothDevice device)
public void enableBluetooth()
DataGraph
private DB_Helper myDBHelper;
static protected List<heartRate> mHeartRates = new ArrayList<>();
static protected List<rrInterval> mRRIntervals = new ArrayList<>();
static protected LineChart mChart;
/** Heart Rate chart stuff **/
static private ArrayList<Entry> mHR_Entries;
static private ArrayList<String> mHR_labels;
static private LineDataSet mHR_DataSet;
private boolean limit_lines_add_once = false;
private LimitLine Max_LimitLine;
private LimitLine Min_LimitLine;
/** RR interval chart stuff **/
static private ArrayList<Entry> mRR_Entries;
static private ArrayList<String> mRR_labels;
static private LineDataSet mRR_DataSet;
private Button delete_alL_btn;
private Button reset_scale_btn;
/** Drawer **/
private ListView mDrawerList;
private DrawerLayout mDrawerLayout;
private ArrayAdapter<String> mAdapter;
private ActionBarDrawerToggle mDrawerToggle;
private void listAllHeartRates()
public boolean onOptionsItemSelected(MenuItem item)

Sequence diagram for establishing a Bluetooth connection between phone and zephyr



Software Architecture



E. Ethical Dimensions of the Heart Rate Sensor

When first building this app we had to consider two frames of reference, the individual, the society and lastly the potential stakeholders who would be interested in this application. More specifically the client and user reference which would be the doctor, the client and the user being either someone with a known heart condition or someone elderly who may also be at risk of having a heart condition.

Ultimately, the goal of this app combined with the usage of the sensor was for a number of reasons. Specifically as a preventative measure, detecting any heart rate abnormalities and avoiding emergencies as well as a method of convenience. For many it can become extremely

inconvenient to be constantly going to the hospital for checkups and stuff. So the goal of this app was to potentially reduce the number of visits needed to the doctor, yet still be able to monitor their heart rate.

By wearing the heart rate monitor (band) it will not limit you from doing any activities you were doing before, but if anything be able to do more while being aware of the current heart rate, if for some reason that you might need to take a break and sit out for a few minutes until it goes down to a certain level again.

In terms of affecting individuals rights, especially in today's technological era, that was something we made sure to look at carefully. Fortunately seeing that there is no connection with the internet in terms of storing data, there is minimal risk of having any results taken in any way. By doing so privacy will be kept seeing that the user can decide who he/she wants to share the results with. Again in terms of safety by the minimalistic design of the heart rate band and the people interviewed there seemed to be no objections to wearing it for long periods of time if needed. Lastly, in terms of safety it could only be more beneficial being aware of your heart rate at all times, especially during some time under exertion where the user can be aware when they're reaching a max heart rate.

One issue that could easily arise as a form of misuse from the application, would be if a user were to use the application without consulting with a doctor beforehand. Due to the fact that we have implemented a max and min heart rates, anything above or below that region would alert the emergency contacts. In the rare occasion that a user were to get the application on their own without previously consulting with a doctor they could set the heart rate regions. Now if a user were to set them on their own without recommendations from a doctor, not only could they be extremely wrong, but in a worst case scenario result in serious health issues. It is for this reason that we recommend to consult with a local physician or a doctor that the user is regularly seeing. Fortunately, we have identified the client to be the doctor and the user to be the patient with a heart condition.

With that being said, along with a warning message that would come with the application, there would be less of a concern for misuse. That being said, it directly leads into the next topic which is liability, dealing with it and ideally reducing it or eliminating it completely. By not emphasizing and making it blatantly clear that the user is recommended to see a medical consultant before using the app, it would not only lead to legal issues for negligence.

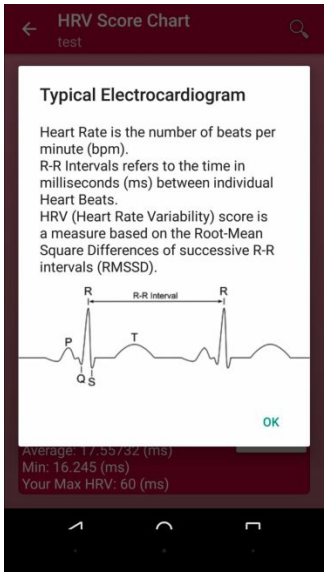
Now that we've listed all these potentially issues that could have risen throughout the whole process of creating the application, to potential misuse and finally how we can best mitigate these issues so that they don't arise. As previously mentioned one major issue that is addressed early on and taken care of is having the doctor recommend what should be the max and minimum heart rates.

For this application, one thing that we had struggled with initially was targeting the proper stakeholder. In fact the confusion between the customer vs. user relationship. Finally, after some time, we have identified the customer to be the doctor/physician who would be "prescribing" this application/heart rate band and the user being the person with a heart condition. Seeing that this app is geared more towards a medical profession, it would be unlikely to have two or more investors with different perspectives seeing that they would all most likely be coming from the same background or area of expertise.

To conclude, this should now have given a better understanding of the thought process that went on while building and creating the application while taking public safety and concerns as a priority.

Appendix

Appendix A – Screenshots and Use-cases



<u>UC 1) Defining the medical terms</u>

Prerequisites:

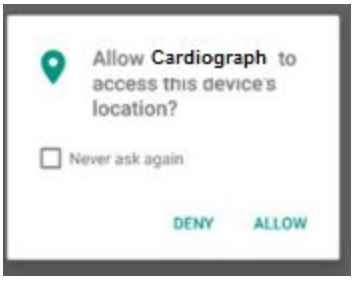
First time opening the app OR
Clicking “Information”

A typical electrocardiogram is shown to the user along with an explanation of the terms “Heart Rate”, “R-R Intervals” and “Heart Rate Variability”. The units are also mentioned, along with the algorithm used to measure heart-rate variability score.

<u>UC 2) Giving the app permission for Bluetooth/sending SMS</u>

Prerequisites:

First time opening the app



The app will ask the user for permission for being able to access the phone’s “coarse location” and access the ability to send text messages.

Course location is needed for detecting Bluetooth nearby devices and sending text messages is needed for alerting contacts in case of danger.

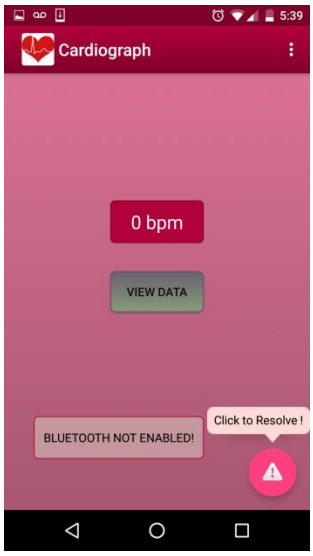
<u>UC 3) Troubleshooting prerequisites for recording</u>

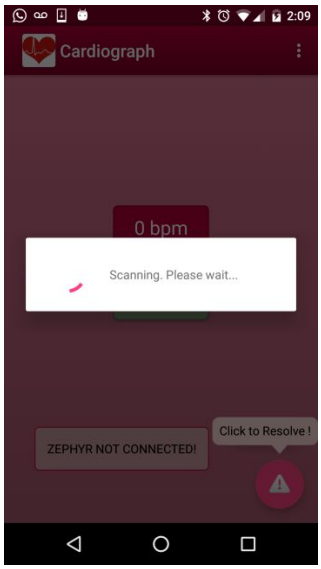
Prerequisites:

Permissions granted AND
Bluetooth is not enabled OR
Zephyr is not connected OR
No emergency contacts saved OR
Important setting is missing

A message appears at the bottom of the screen with the appropriate error message and a tooltip saying “Click to Resolve!”

The button to click will automatically redirect the user to the appropriate settings/contacts page or try to enable Bluetooth or connect to the zephyr.





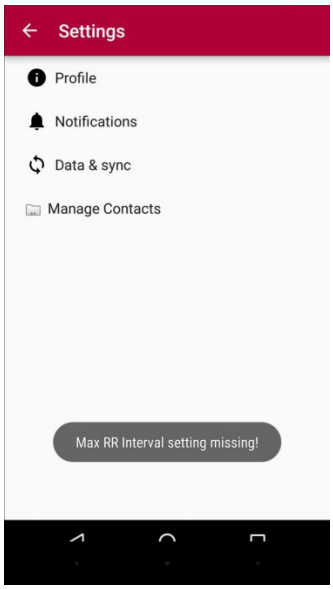
UC 4) Inputting settings

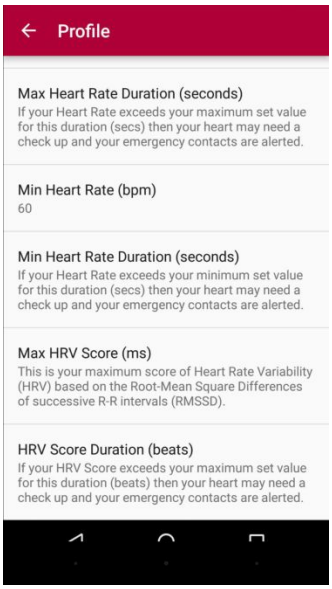
The settings page allows access to the user’s profile, notifications preferences, emergency contacts and the zephyr device’s information.

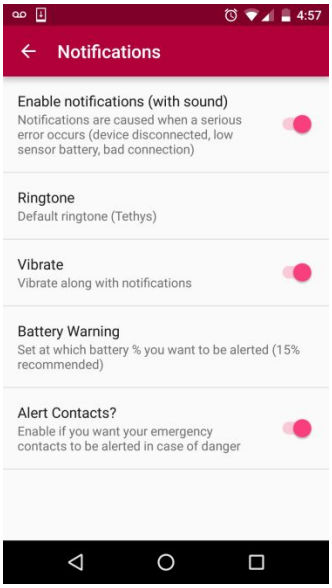
In the profile, the user can set the detection parameters such as the max/min heart-rate and max HRV score as well as the according sensitivity before alerting contacts.

In the notifications, the user can set the minimum sensor battery alert and disable/enable notifications and alerting contacts.

In the Data & Sync, the user can manually enter his zephyr’s mac address or name.



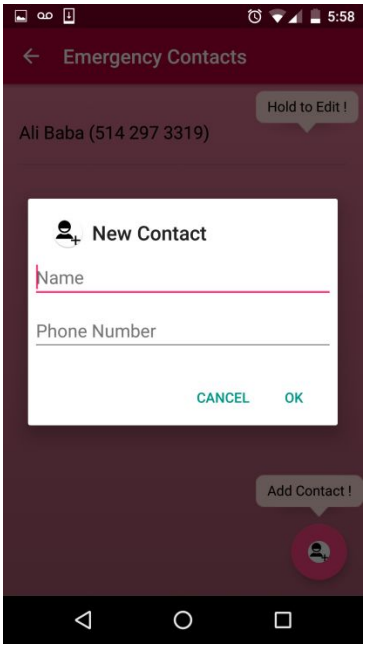




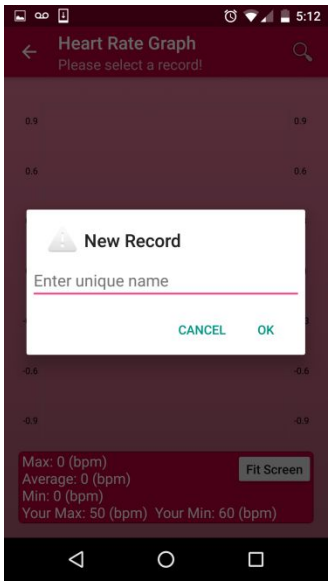
UC 5) Adding contacts

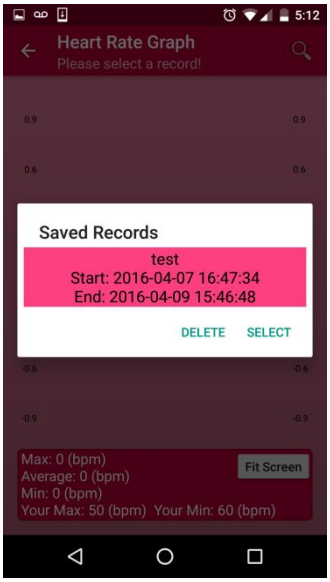
In the “Manage contacts” page, the user can add his emergency contacts by clicking the button in the bottom right where it says “Add Contact!” and filling out the form.

The user can also choose to edit or delete a contact by pressing and holding one of his contacts.



UC 6) Selecting or creating a record





All recorded data is attached to a "record".

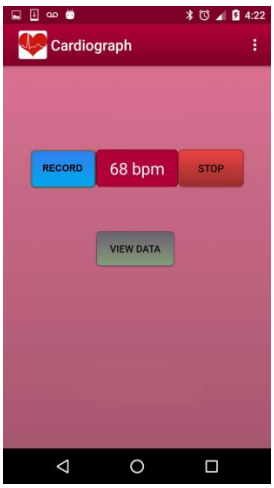
To record new data, the user has to create a new record or select an old one. Records should have unique names.

To view old data, the user has to select an old record based on the name or their start/end timestamps.

UC 7) Recording heart data

Prerequisites:

- All profile settings set AND
- Bluetooth enabled AND
- Zephyr connected AND
- Record selected AND
- If alerting contacts enabled, 1 contact should exist



After all the prerequisites are met, the user will finally see the "Start" and "Stop" buttons.

After clicking start, the live heart rate will show and will be replaced by a blue "Record" button.

The "Record" button, when triggered, shows a green circle and starts saving the sensor's data into the database and tells the app the start updating the live charts if they are showing.

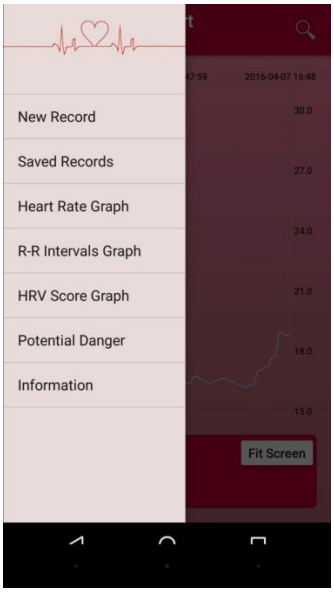
UC 8) Viewing charts

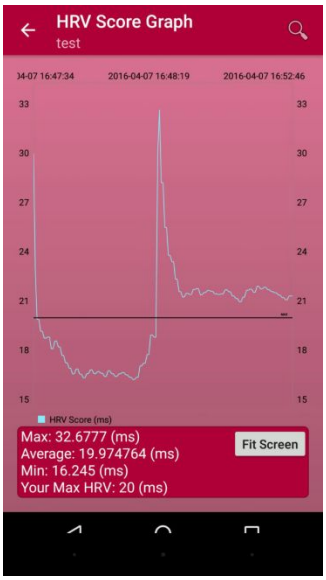
Prerequisite:

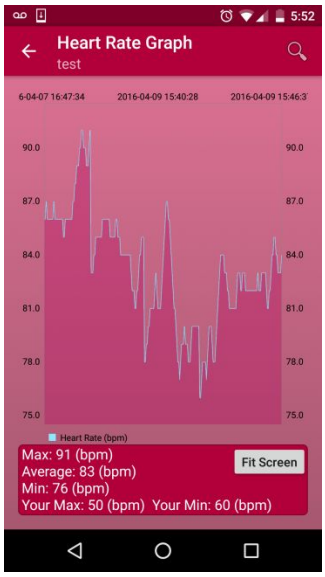
- Record selected

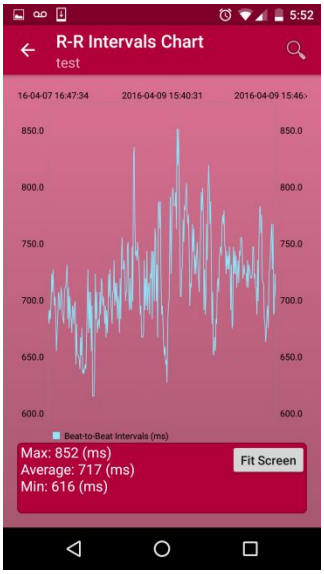
When the user clicks on "View Data", he will be redirected to the page where you can visualize your live heart rate and live HRV score, you can also view charts associated with old records.

At the bottom of each chart, the max/min and average value associated with it, as well as a reminder of your set values.









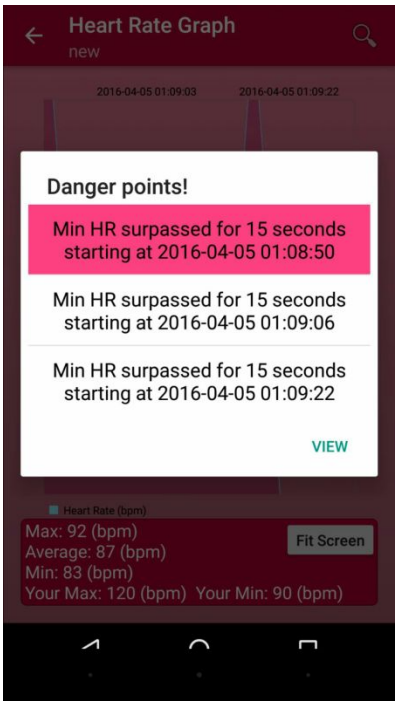
<u>UC 9) Evaluating old records for danger zones</u>

Prerequisites:

Record selected

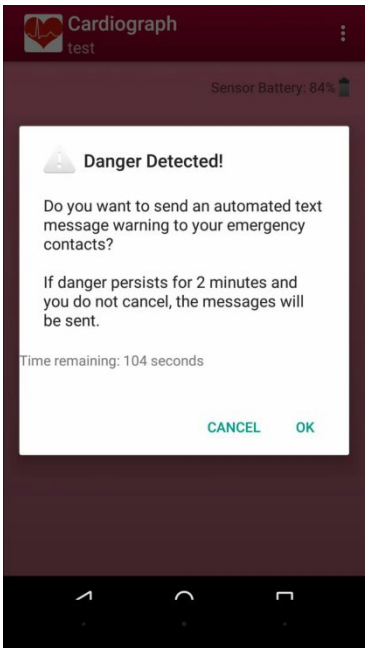
Max/min/duration settings set

While going through old charts, the user can click “Potential Danger” to see view where on the graph his max/min heart rate or HRV score was surpassed for the durations he set.



After selecting one of the danger points and clicking “View”, the chart will automatically move to that area.

UC 10) Alerting contacts of danger



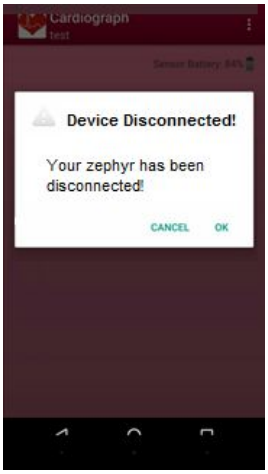
Prerequisites:

Recording live data
Alerting contacts enabled
Max/min/duration settings set

While the user is using the app and zephyr to record his heart's data and a potential danger is detected based on his settings, the phone will vibrate/ring to notify the user.

The user will be prompted with an interface asking to confirm/deny sending a SMS text message to his contacts. The alert also comes with a 2 minute countdown, after which the messages will be automatically sent if ignored.

UC 11) Alert when sensor is disconnected



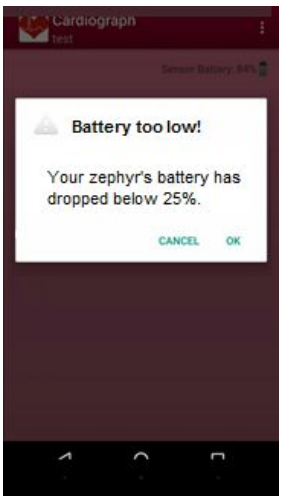
Prerequisites:

Recording live data, Device disconnected

If the user is recording live data and the sensor strap comes loose or somehow the circuit is no longer complete a dialog will be displayed asking the user to acknowledge.

The user will be notified with a vibration/ring if those settings are enabled. Notifications also show on the phone's main screen.

UC 12) Alert when sensor Battery too low



Prerequisites:

Notifications enabled, battery minimum level set
Recording live data, Sensor battery below his level

If the user is recording live data and the sensor battery drops below his minimum set percentage a dialog will be displayed asking the user to acknowledge.

The user will be notified with a vibration/ring if those settings are enabled. Notifications also show on the phone's main screen.

G. Team Blog

Date	Who	Activity and length of time	Purpose	Output
Jan 25th	All	Worked at school in lab room, 1.5 hours	Start to brainstorm initial ideas for first milestone	Starting to get an initial idea of what hardware components we would potentially use
Jan 27th	Jordan,Jafar, Michael	1 hour at school	Continued to brainstorm ideas and start to create a more structured list of potential ideas	Came up with 6 potential ideas and going to meet with rest of group to get down to top 3 ideas.
Jan 29th	Rajithan, Samih, Michael, Jordan, Jaffar	2 hours	Come up with three ideas for first milestone, along with setting along with evaluating each opportunity statement	All three mission statements decided on, evaluated the opportunity statements and ranked them as well
Jan 30th	Rajithan & Samih	1 hour	Write mission statement and wrap up first milestone	Completed first milestone and was submitted shortly after.
Jan 31st	Jordan	1 hour	Wrapped out Milestone	Ready to start working on Milestone 2
Feb 7th	All	1.5 hours	Reviewed commentary from Milestone 1 Eval	Started dividing up the work needed to be done for milestone 2.

Feb 8th	Michael & Jordan	2.5 hours	Worked on interview script for stakeholders interested in the product	Ready to interview the stakeholders with the interview script that has now been prepared
Feb 10th //////////	Michael & Jordan //////////////	3 hours ///////////////	Interviewed the stakeholders with the script prepared. Managed to interview both in the same day.	Task completed and was put into the folder of stuff that needs to be completed before submitting milestone 2
Feb 11th	Jordan	2 hours	Wrote out ideas for the one page submission along with intent of finishing it after brainstorming.	Finished one page submission took longer than expected. Originally planned for 1.5 hours in total including brainstorming, took 2 hours in total.
Feb 14th	Raj & Samih	2.5 hours	Met and started discussing the simulation plan and what needed to be done	Finished the simulation plan, again took longer than expected. Initially predicted it taking 2 hours, took closer to 2.5 after some research that was needed to be done.
Feb 16th	Jafar	2.5 hours	Met to start discussing what needed to be done for the product backlog.	Started to work on product backlog, ran out of time, agreed to meet again

				tomorrow and finish it.
Feb 17th	Raj & Samih	1 hour	Met to continue product backlog, continued discussing from where everything was left off yesterday.	Product backlog had been finished shortly after and was sent to rest of group members to look over.
Feb 18th	All	2 hours	Everyone met reviewed everything that was done, made sure nothing was missing along with correcting any errors.	Almost done completing everything needed to submit Milestone 2
Feb 19th	All	1 hour	Everyone met together put all materials together that was needed to complete Milestone 2, made sure everything was finished	Submitted Milestone 2
Feb 29th	All	2 hours	Group met to put together everything that was needed for sprint 1 meeting with Tyler	Everything was completed and submitted onto moodle so that it would be able to be reviewed by Tyler before the meeting.
March 1st	All	1 hour	Meeting with Tyler	Allowed us to get a better idea of how to approach sprint 1 and iron out any issues that we may have faced later on

March 4th	All	1.5 hours	Met and discussed who would be doing what, for what is required for sprint 1	In groups of 2-3 everything that is needed to be done has been assigned to which groups felt they were able to handle it best.
March 6th	Jordan and Michael	2 hours	Worked on updated product backlog and sprint backlog based around Tyler's recommendations	Spoke with Jafar, needs more work but an improvement from what was done before.
March 7th	Samih and Raj	3 hours	Worked on the simulation plan . Worked on the coding of the app with Jafar.	We found all the issues to be studied and how it will be resolved in our application
March 8th	Jordan and Michael	2 hours	Continued to work on updating product backlog and sprint backlog to Tyler's recommendations and based on what Jafar had mentioned for us to fix as well.	Finished product backlog, and sprint backlog which are now tailored to the new recommendations.
March 9th	Raj & Samih	1 hour	Revised code to make small adjustments before determining it to be finished for sprint 1	Changes made simply to make it more user friendly when using the application. Notifications when bluetooth

				is connected and disconnected.
March 10th	All	1.5 hours	Met as a group to make sure everything was completed for sprint 1 for the meeting coming up next week and complete anything that was missing.	Completed everything required for the demo and started to discuss when to meet to start working on the required materials for sprint 2
March 11th	All	2 hours	Met together to figure out what was done and what needed to be done in terms of sprint 2 documents so that it can be uploaded.	Finished everything that was missing with regards to sprint 2 and uploaded them to Moodle before having the meeting with Tyler.
March 13th	All	3 hours	Went over what was discussed in the sprint 2 meeting, how to better improve and split up the work for sprint 3	Figured out how to be more efficient with our work along with have a good idea of who is doing what for sprint 3
March 16th	1)Michael & Jordan 2)Samih & Jafar	2 hours 2 hours 4 hours TOTAL	1)Continued to work on the product backlog and sprint backlog. 2)Discussed certain aspects of the code that needed to be fixed or more efficient and how they would go about coding	1)Fixed up based on what was recommended from our previous sprint 2)Started to work out some bugs, still having some issues with the code randomly crashing.

			what's needed for sprint 3	
March 17th	Jordan & Michael	4 hours	Started working on the product and sprint backlog of sprint 3	Started to determine what was missing and what needed to be done still need a few hours to finish everything.

March 19th	Raj & Samih	3.5	Started to work on the class diagram, sequence diagram and flowchart based on previous recommendation s on how to fix it.	Finished most of class diagram and sequence diagram going to meet again to review it and finish the rest.
March 23th	Jafar & Raj	3	Made some adjustments on the code after meeting with Thinesh and app was crashing so spent some time figuring out why.	Managed to debug the code and got rid of the bug that had the app crashing every time we tried to show a feature.

March 24nd	Jordan and Samih	4	Helped and continued to work on the class diagram, and specifically the sequence diagram and the flow chart.	Finished the class diagram, sequence diagram and flowchart.
March 25th	All	2 hour	Met as a group quickly to be sure everyone was on track for what was needed to be	Some clarification was needed for Michael and Raj, aside from that everyone was on track.

			done by the end of the sprint	

March 26th	Michael and Raj	3 hours	Corrected the class diagram and sequence diagram	Ready to meet as a group to go through all the documents and finalize the sprint
March 27th	All	5 hours	Went through all the documents, made sure everything was complete, looked over what the TA's had mentioned we should cover as well, made sure everything was properly formatted and nothing was missing.	Successfully submitted all the required documents and able to meet with Tyler now.
March 31st	Michael & Jordan	2.5 hours	Met to start working to put together the powerpoint for the demonstration	Started to put together the powerpoint for the demonstration need to add in pictures from all the different features of the application
April 2nd	Samih and Raj	2 hours	Added to the static model	Finished the static model and now need to meet again to finish the dynamic model

April 3rd	Raj and Jafar	1.5 hours	Met together discussed what was done on the static model and then worked on the dynamic model	Dynamic model is finished starting to slowly wrap up everything for the final presentation.
April 5th	Jafar, Sami and Raj	2 hours	Ran through the code, making sure there is no sudden errors or bugs coming up before the oral presentation	Thoroughly went through the code to make sure everything was checked and all cases were looked over so that there will be no errors or issues during the oral presentation.
April 6th	Michael and Jafar	3 hours	Making changes to product backlog as suggested by Tyler.	Product backlog is finished now going to finish sprint backlog in next meeting with the group.
April 8th	Jordan and Jafar	2.25 hours	Worked on sprint backlog after Michael and Jafar finished product backlog	Sprint backlog is also now completed after the changes that were recommended.
April 9th	All	4 hours	All came together to start practicing for the demonstration and going over who would be discussing what	Started to work on the oral presentation will need to meet again in order to finalize everything and have it be more smooth during the oral.

April 10th	All	6.5 hours	Met to make sure everything was completed for the demonstration, almost everything completed for the final report along with practicing for the oral demo	Started working towards completing everything left that is needed for the final submission.
April 11th	All	2.5 hours	Ran through the whole code made sure nothing is ambiguous in the code, comments are added before submitting.	Everything is confirmed to be working, comments are added where needed.
April 12th	All	3 hours	Finished final documents made sure everything was in order and nothing was wrong before final submission	Everything completed and finalized. Thank you for everything from Team C!

Team C:

Jordan Flinker (27023286)

Jordan Flinker

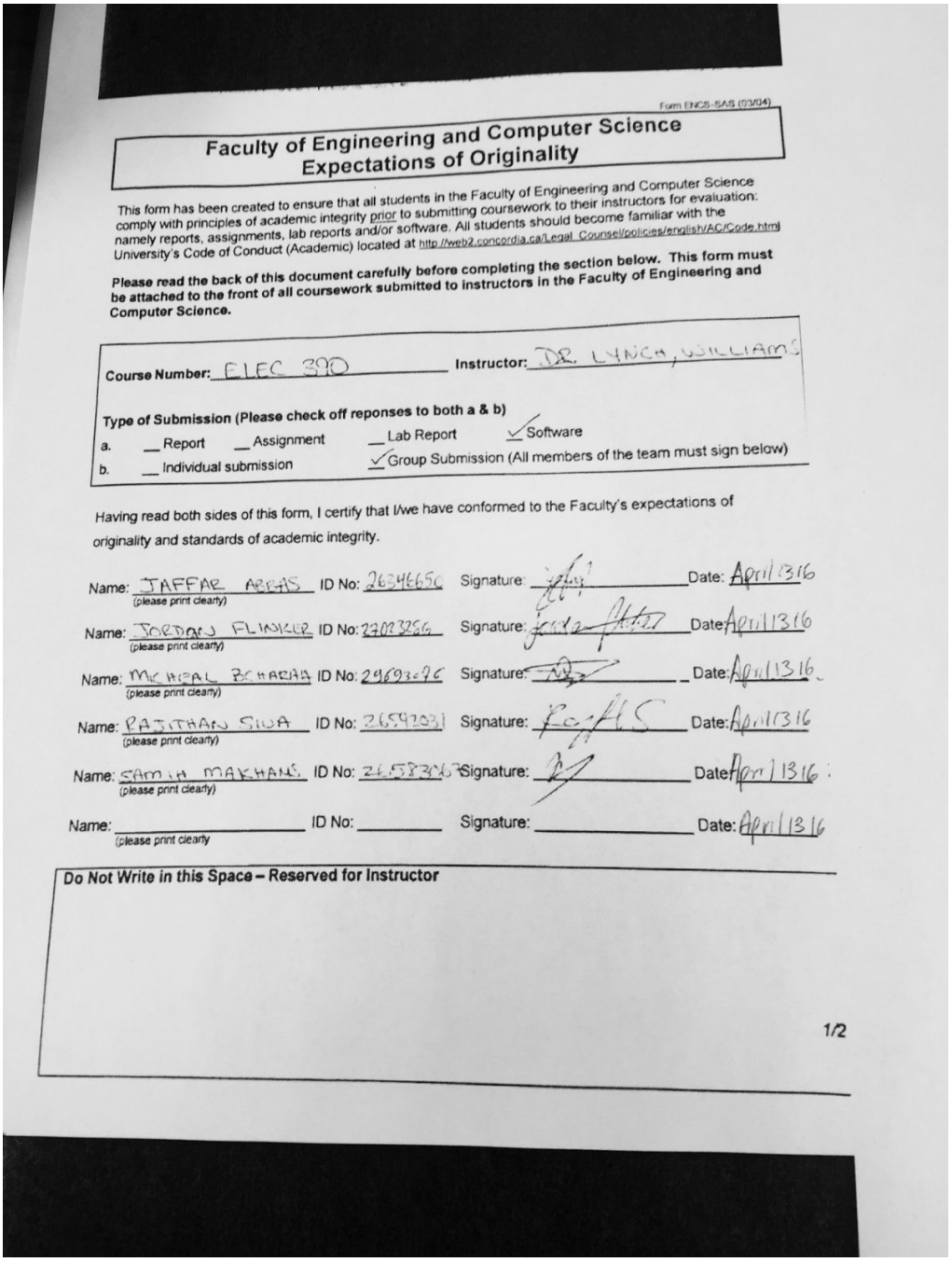
Jafar Abbas 26396650

Michael Bcharah (29693076)

Rajithan Sivapatharajah (26592031)

Samuth MaKhanns 26583067

H. Expectation of originality form



Form ENCS-SAS (03/04)

Faculty of Engineering and Computer Science
Expectations of Originality

This form has been created to ensure that all students in the Faculty of Engineering and Computer Science comply with principles of academic integrity prior to submitting coursework to their instructors for evaluation: namely reports, assignments, lab reports and/or software. All students should become familiar with the University's Code of Conduct (Academic) located at http://web2.concordia.ca/Legal_Counsel/policies/english/AC/Code.html

Please read the back of this document carefully before completing the section below. This form must be attached to the front of all coursework submitted to instructors in the Faculty of Engineering and Computer Science.

Course Number: ELEC 390 **Instructor:** DR. LYNCH, WILLIAMS

Type of Submission (Please check off reponses to both a & b)

a. __ Report __ Assignment __ Lab Report ✓ Software

b. __ Individual submission ✓ Group Submission (All members of the team must sign below)

Having read both sides of this form, I certify that I/we have conformed to the Faculty's expectations of originality and standards of academic integrity.

Name: JAFFAR ABBAS (please print clearly) ID No: 26346656 Signature: _____ Date: April 13 16

Name: JORDAN FLINKER (please print clearly) ID No: 27013256 Signature: _____ Date: April 13 16

Name: MICHAEL BCHARA (please print clearly) ID No: 29693096 Signature: _____ Date: April 13 16

Name: RAJITHAN SIVA (please print clearly) ID No: 26592031 Signature: _____ Date: April 13 16

Name: SAMIH MAKHANI (please print clearly) ID No: 26583067 Signature: _____ Date: April 13 16

Name: _____ (please print clearly) ID No: _____ Signature: _____ Date: April 13 16

Do Not Write in this Space – Reserved for Instructor

1/2