Documentation

This document contains instructions for installing and running the GatorMind project.

Dependencies Required

- Android Studio
- Java 8
- Tone analyzer plugin (provided with code)
- AWARE-micro (already set up in the server)
- MySQL database (already set up in the server)
- MySQL client (to check whether the data is flowing in as intended
- Activity Analysis plugin
- Noise Analysis plugin
- Alarm map plugin
- Maven
- Eclipse
- Git on laptop
- Datatable Plugin (https://grafana.com/grafana/plugins/briangann-datatable-panel/installation)
- Button Plugin (https://grafana.com/grafana/plugins/speakyourcode-button-panel)

Application Instructions

1) Install Android Studio (https://developer.android.com/studio)
2) Download the application from GitHub (https://github.com/SarahBrown97/GatorMind) and open the application in Android Studio after unzipping the folder. This project uses Zoom SDK for Android which is used for peer support implementation.
3) Once you download the repository from GitHub, extract it, then follow the links provided below to import the two modules named commonlib and mobilertc into the project application.
4) To import the 2 modules, first we need to download the zoom-sdk for android from: (https://github.com/zoom/zoom-sdk, android/archive/master.zip)
5) Then, unzip the file and follow the steps from: (https://marketplace.zoom.us/docs/sdk/native-sdks/android/getting started/integration). All the dependencies and configurations are already present, all we need to do is import the above-mentioned AAR packages (commonlib and mobilertc) and then build the project and run.
6) Application Installation: Once all the dependencies are included in the Android application, build the application in Android studio. To run the application, you can use the simulator in Android studio or connect an Android device and install the application on the device.

Server Set-Up

1) Follow the instructions on https://github.com/denzilferreira/aware micro to deploy the server.
2) Add the plugin details to the aware-config.json in the following format (set the defaultValue of the setting which specifies the status of the plugin to true if you want the client to start the plugin by default):

```
{
    "package_name" : "com.aware.plugin.activity_analysis",
    "plugin" : "plugin_activity_analysis",
    "settings" : [ {
        "setting" : "status_plugin_activity_analysis",
        "title" : "Active",
        "defaultValue" : "true",
        "summary" : "Analyses activity data"
    }, {
        "setting" : "frequency_plugin_activity_analysis",
        "title" : "Activity analysis interval (in seconds)",
        "defaultValue" : "16",
        "summary" : "16 seconds"
    } ]
}
```

3) API Installation: Open the Digital-Health-Interventions-Main folder in Eclipse. Run the project as a maven build with the goal ‘package’. Find the .jar file in the /target folder of the project. In order to run the API, run `java -jar DHIntervention-1.0-SNAPSHOT.jar`.

Grafana Dashboard Instructions (by Group 4)

For Ubuntu:

sudo apt-get install -y apt-transport-https

sudo apt-get install -y software-properties-common wget wget -q -O -

https://packages.grafana.com/gpg.key | sudo apt-key

add -

sudo apt-get update

sudo apt-get install grafana

Running the Grafana Server:

sudo service grafana-server start

sudo service grafana-server status

Accessing the Grafana Server:

1. Connect to vpn
2. Open the Grafana server url:10.254.0.1:3000
3. Login with username: admin and password: password to access the Grafana server with admin privileges

Users created to login to grafana server and given viewer access:

1) Username : group1, Password: group1

2) Username : group2, Password: group2

3) Username : group3, Password: group3

4) Username : group4, Password: group4

5) Username : group5, Password: group5

6) Username : group6, Password: group6

Group 1 Specific Documentation

Frequency of Updates to Aware Micro Instance:

1. Calls and Messages: Every 60 mins. (User must grant Accessibility permissions)
2. Device Usage: Every instance when the device's screen is on/off.
3. Application usage and notification: As soon as a new application is used.
4. Tone Analyser: Every 60 mins, the tone of the hour is updated and sent. (User must grant Accessibility permissions).
5. Screen time: Every instance of screen lock and unlock is sent as soon as these events occur.

Things to setup on Server:

Add the following lines of code in aware-config.json

```
{
  "package_name":"com.aware.plugin.tone_analyser",
  "plugin":"plugin_tone_analyser",
  "settings":[
    {
      "setting":"status_plugin_tone_analyser",
      "title":"Active",
      "defaultValue":"true",
      "summary":"Logs tone of hour."
    }
  ]
}
```

Plugin Documentation:

1. Tone Analyser

This plugin analyses the tone of the user for a particular hour using IBM watson tone analyser. Refer https://github.com/KashidVivek/aware-tone-analyser for more details.

Field	Type	Description
_id	Integer	Primary key autoincrement
timestamp	Real	unix timestamp in milliseconds of sample
device_id	Text	AWARE device ID
Tone	Text	Tone analyzed for the previous hour

Group 2 Specific Documentation

Activity Analysis:

- This plugin is dependent on the Aware Google Activity Recognition plugin for data. It uses the data generated by the plugin to perform the analysis. Please refer to Google Activity Recognition for details.

Field	Type	Description
_id	INTEGER	primary key auto-incremented
timestamp	REAL	unix timestamp in milliseconds of sample
device_id	TEXT	AWARE device ID
moderate_activity_time	INTEGER	Time for which the user is walking in the specified interval
high_activity_time	INTEGER	Time for which the user is running or cycling in the specified interval
is_user_active	INTEGER	1 is the user is active in the specified interval

Note: The time fields are in minutes.

Settings:

- Name of the table created: plugin_activity_analysis
- This plugin sends data every **day.** This can be changed in the com.aware.plugin.activity_analysis.Plugin file. You just need to set a new interval time for the Scheduler.

How to activate this plugin?

- Use the line: Aware.startPlugin(getApplicationContext(), "com.aware.plugin.activity_analysis");

Analysis perimeter:

- We define if the user is active by checking if the moderateActivityTime > 23 or highActivityTime > 12.

Noise Analysis:

- This plugin is dependent on the Aware Ambient noise plugin for data. It uses the data generated by the plugin to perform the analysis. Please refer to Ambient Noise Plugin for details.

Field	Type	Description
_id	INTEGER	primary key auto-incremented
timestamp	REAL	unix timestamp in milliseconds of sample
device_id	TEXT	AWARE device ID
low_noise_time	INTEGER	Time for which the ambient noise is below 40
moderate_noise_time	INTEGER	Time for which the ambient noise is between 40 and 75
is_harmful_inday	INTEGER	1 if noise level is harmful or else 0

Note: The time fields are in minutes.

Settings:

- Name of the table created: plugin_noise_analysis
- This plugin sends data every **day**. This can be changed in the com.aware.plugin.noise_analysis.Plugin file. You just need to set a new interval time for the Scheduler.
- By default It sample the data every 5 minutes.

How to activate this plugin?

- Use the line: Aware.startPlugin(getApplicationContext(), "com.aware.plugin.noise_analysis");

Analysis perimeter:

- Low noise (< 40 decibels), normal noise (40~75 decibels), high noise (>75 decibels) and harmful in a day (stay in over 75 decibels environment for over two hours)

Alarm map:

Field	Type	Description
_id	INTEGER	primary key auto-incremented
timestamp	REAL	unix timestamp in milliseconds of sample
device_id	TEXT	AWARE device ID
type_of_place	TEXT	Type of the place the user is visiting
no_of_visits	INTEGER	Number of times the user visits the place

Settings:

- Name of the table created: alarm_map.
- This plugin sends data every day. This can be changed in the com.aware.PLUGIN_ALARM_MAP Plugin file. You just need to set a new interval time for the Scheduler.

How to activate this plugin?

- Use the line: Aware.startPlugin(getApplicationContext(),"com.aware.plugin.alarm_map")

How to update place dedication?

- The file alram_map/Plugin.java count 1 each visit based on the location type (currently only ‘bar and liquor store’)
- Supprrted types that can be aded from PLACE API can be found at https://developers.google.com/places/web-service/supported_types

Sync and location update frequency.

- The plugin locates the user every 5 min matching the phone location update frequency. The server gets a location visit count every 24 hours.

Future work for alarm map plugin:

- Improve the accuracy of the code which finds out the type of the place the user is currently present in.
- Improve the accuracy of the code that finds out how many times the user has visited the place.
- Add support for more place types.

Future work for all plugins:

- Need to make code changes to the plugin to sync data using the Sync Adapters rather than the current approach which uses Http post method to send the data to the server.
- Need to set up default values for plugins to sync data if the use has not set up any value in the server.

Group 3 Specific Documentation

1. Once you download the repository from GitHub, extract it, then follow the links provided below to import the two modules named commonlib and mobilertc into the project application.
2. To import the 2 modules, first we need to download the zoom-sdk for android from: (https://github.com/zoom/zoom-sdk android/archive/master.zip)
3. Then, unzip the file and follow the steps from:
    a. (https://marketplace.zoom.us/docs/sdk/native-sdks/android/getting started/integration)
4. All the dependencies and configurations are already present, all we need to do is import the above-mentioned AAR packages (commonlib and mobilertc) and connect to the digital ocean cloud via VPN (167.71.59.111) as the server (providing REST API endpoints) is deployed there.
5. Once everything is set, build the project, and run the application using android studio emulator.

For development and testing purposes of the Zoom-SDK, an SDK Key & Secret from the Zoom marketplace for developers is required. Currently, we have already pushed the code changes with the SDK key & secret using one of our team member's Zoom developer account. In order for future developers to create their own SDK key & secret, please follow the link – https://marketplace.zoom.us/docs/sdk/native-sdks/credentials and paste your SDKkey and secret in the ZoomActivity.java file in the initializeSdk function.

Group 5 Specific Documentation

1. The code is present at the git repository: https://github.com/ruthvik123/Digital-Health-interventions
2. Have ‘eclipse’ IDE installed in your machine
3. Create a jar file using the IDE. Right click on the project in the package explorer tab and run as maven build with the goal as ‘package’.
4. The jar file can be found in the /target folder of the project
5. Run the following command : java -jar DHIntervention-1.0-SNAPSHOT.jar

QoL Questionnaires and Self Reports - We have implemented a functionality using which healthcare professionals can request for QoL Questionnaires and self reports for a particular patient. The questions are created intelligently by analysing the patient data.

End Point: http://10.254.0.1:8081/qolList/{userID}

Description: Provides a list of questionnaires for a particular user.

Sample Response:


```
{
 "questions": [
 {
 "id": 1,
 "status": "Completed"
 },
 {
 "id": 2,
 "status": "Completed"
 },
 {
 "id": 3,
 "status": "Completed"
 },
 {
 "id": 4,
 "status": "Completed"
 }
 ]
}
```

End Point: http://10.254.0.1:8081/qol/{userID}/{questionnaireID}

Description: Returns the requested questionnaire for a particular user.

Sample Response:


```
{
 "questions": [
 {
 "id": 1,
 "question": "How connected do you think you are with your friends and family?",
 "type": "Rating",
 "answer": "sample Answer 1"
 },
 {
 "id": 2,
 "question": "Do you feel lonely",
 "type": "Yes/No",
 "answer": "sample Answer 2"
 },
 {
 "id": 3,
 "question": "Are you in touch with your friends?",  "type":
"Yes/No",
 "answer": "sample Answer 3"
 },
 {
 "id": 4,
 "question": "Do you think you are physically healthy?",  "type":
"Yes/No",
 "answer": "sample Answer 4"
 },
 {
 "id": 5,
 "question": "Would you say you live a sedentary lifestyle?",
 "type": "Yes/No",
 "answer": "sample Answer 5"
 },
 {
 "id": 6,
 "question": "Placeholder2",
 "type": "Rating",
 "answer": "sample Answer 6"
 }
```

```
]
}
```

End Point: http://10.254.0.1:8081/questionnaire

Description: Handles request from healthcare professional for a questionnaire for a particular patient.

Sample Request:

```
{
   "patientId": "1",
   "status": "Pending"
}
```

The functionality also keeps track of the completion status of each of the questionnaires sent so that they can be rendered as pending/completed in the user interface developed by Project 3.

End Point: http://10.254.0.1:8081/qol

Description: This endpoint stores user responses to the questionnaires and updates the completion status accordingly.

Sample Request:

```
{
 "userID": "admin",
 "questionnaireID": 1,
 "userResponse":[
 {
 "id": 1,
 "answer": "abc"
 },
 {
 "id": 2,
       "answer": "abc2"
 }
 ]
 }
```

Sample Response:

success

Worklist - This functionality provides healthcare professionals with a list of pending tasks. This can be visualized using the dashboard developed by Project 4. Updates from QoL and self-report as well as any new determinations by the intervention algorithm are added to the worklist for the healthcare provider to view.

Triaging - This functionality triages the patients based on the criticality so that the healthcare professionals can provide his/her services to patients who need it the most. In order to do this we are using K Means clustering. In order to calculate it we are normalizing the data by assigned a score to each biomarker. A score of 100 in the best and 0 is the worst. For example if a person has more that 150 mins of high physical activity a week as recommended by the AHA(American Heart Association) he gets a score of 100 and if he has low physical activity his score will be closer to zero. We will calculate the clusters based on this score. Once the clusters are formed we choose the cluster based on its proximity to the centre([0,0,0,0,0,0,0] : worst scores in each aspect).

Biomarker Scores(all over a week)		
Biomarker	Value	Score
Physical Activity(in mins)	>150 mins	100
	0 < x < 150 mins	(x/150)*100
Ambient Noise(in mins)	>= 60	0
	0 < x < 60	100 - (x / 60 * 100)
Alarm Map(instances of bar or liquor store visits)	>= 5	0
	0 < x < 5	100 - (x * 20)
Device Usage(in milliseconds)	>= 70*60*60*1000	0
	7*60*60*1000 <= x <= 14*60*60*1000	75
	0 <= x < 7*60*60*1000	100 - (x/(7*60*60*1000)*100/4)
	14*60*60*1000 < x < 70*60*60*1000	75 - ((x - 14*60*60*1000)/((70-14)*60*60 *1 000)*75)

Calls(in mins)	>= 35*60	0
	3.5*60 < x < 35*60	100 - ((x - 3.5*60)/((35-3.5)*60)*100)
	0 <= x <= 3.5*60	x / (3.5*60) * 100
Tone(instances of Anger, Sadness, Fear)	>= 42	0
	0 < x < 42	100 - (x / 42 * 100)

If no data is present the default score is 50.

Peer support - This functionality has been developed using clustering algorithms to create peer groups based on behavioural patterns and similarity in bio markers. Each patient can visualize or see the group that they belong to using the interface developed by Project 3. Patients are grouped by K Means clustering similar to Triaging. In case the cluster is large we divide them into clusters of no more than 10 peers.

End Point: http://10.254.0.1:8081/peer/{userID}

Description: This endpoint returns the Peer Group and Group members of a particular user.

Sample Response:


```
{
 "peerGroupId": "1",
 "peerIds": [
 "1",
 "10",
 "11",
 "12",
 "13",
 "14",
 "15",
 "16",
 "17",
 "18"
 ]
}
```


End Point: http://10.254.0.1:8081/peerNotes
Description: This endpoint stores responses from users/members of particular group after group meet-ups.

Sample Request:

```
{
 "peerGroupId": 1,
 "peerId":"23",
 "notes": "Attended peer support meeting and felt good after that" }
```

Automatic intervention - We have developed this service to periodically check for changes in the user data. Healthcare professionals can see any significant warnings due to low biomarker scores in the form of entries in work lists in the dashboard.