Project Name: Dr. Waker

Team Member Names: Magic 3 (Hsuan Chen, Yunjing Li, Yapeng Li)

Project Description:

This application is a smart alarm clock designed for those who find it hard to wake up or get clear in the morning. Many people find themselves turning off their alarms in the morning with just their subconscious mind without being fully awake. This issue can cause many problems including being late to meetings or classes. As a result, we have decided to tackle this problem and provide the solution in the form of an Android smart alarm clock application.

This alarm clock is equipped with the conceptual idea of only allowing the user to turn off the alarm after the user has completed a task. One task is to solve a mathematical problem correctly while the other task involves taking a selfie where the application can clearly see that the user has opened both eyes.

In this application, we will implement multiple features. For example, we will use the camera to take a wakeup-selfie. We will also implement speaker and GPS. Speaker will be used to sound the alarm when the alarm goes off. GPS will be used to detect the user's location in deciding whether or not the application should sound the alarm.

According to the fact that our project is local-store based application, we will implement the local SQLite database to store the information of our objects. Also, we will implement a remote MySQL database to store our math problems. It will interact with our client Android application using sockets.

We call our application as Dr. Waker. Our application is a professional alarm app which will meet your multiple requests. Download now!

List of Android Features:

- Camera
    - The camera will be used to take various kinds of pictures, including the baseline picture, a wakeup-selfie, and wakeup-picture of objects.
- GPS
    - GPS will be used to detect the user's location in deciding whether or not the application should sound the alarm.
- Speaker
    - Speaker will be used to sound the alarm when the alarm goes off.

User Story:

At a Sunday midnight, Danny has just finished his assignment for his course Java Smartphone Development. However, he has another course on Monday mornings at 8:30 am. So he sets an alarm at 7:30 am since he really wants to have breakfast before he goes to school. He was so tired that he falls asleep as soon as he reached the bed. When it comes to the morning, the alarm rings. Danny does not have our application, so he turned off his alarm without actually completely waking up. After 40 minutes have passed, Danny opens his eyes and checks the time. What makes him so shocked is that it's already 8:10 am and that he has to go to school right way. He is forced to go immediately and skipped his valuable breakfast. After that, he realizes that he really needs an alarm that can keep ringing until he gets up and can make him awake in the morning. So he downloads the application we designed and sets up an alarm at 7:30 am for the next day. Next morning, the alarm goes off with a face detector show up. What Danny needs to do is that he has to smile and open both his eyes for the selfie in order to turn off the alarm. He quickly takes a selfie to turn off the the alarm. However, after that, what really makes him awake is that he has to solve a math question so that he can unlock the phone completely. After doing these, Danny doesn't want to sleep anymore. He gets up and prepares his breakfast. With this alarm, Danny begins a day with a wonderful morning. That night, Danny wants to try "Take a picture" method of waking up the following day. He sets an

alarm at 9:00 am and takes a baseline picture of his microwave in the kitchen. The next morning, his alarm goes off asking him to take a similar picture of his microwave. As a result, Danny has to get out of his bed and walks to kitchen in order to turn off his alarm. After taking a picture of his microwave, his alarm finally turns off and he begins another day with another wonderful morning.

Brown is an officer who has to deal a lot of meetings every day. To get up early and prepare breakfast for his daughter, he has to set a lot of alarms to remind himself in the morning. He sets an alarm at 8:00 am as a weekly alarm. However, one day, Brown has a meeting at 8:00 am. Since this is outside of his normal routine, he forgets he has the weekly alarm and does not deactivate the alarm. When time goes to 8:00 am, the alarm goes off and the ringing makes Brown really embarrassed. After that, he makes up his mind to download our application. Setting the location of the weekly alarm at 8:00 am to be his home, the alarm will not go off unless Brown is near the proximity of his home. Then another day arrives with Brown having another 8:00 am meeting out of his routine. However, this time, even with Brown forgetting to deactivate his alarm, his meeting went on smoothly without the interruption of his 8:00 am alarm. The reason being that at 8:00 am, the application sees that Brown is not at his home, so it does not sound the alarm. After downloading the application, he never worries about the weekly alarm anymore. The weekly alarm will be automatically turned off if he has left home. With our application, Brown will never face this embarrassing situation again.

Page Flow Diagrams:

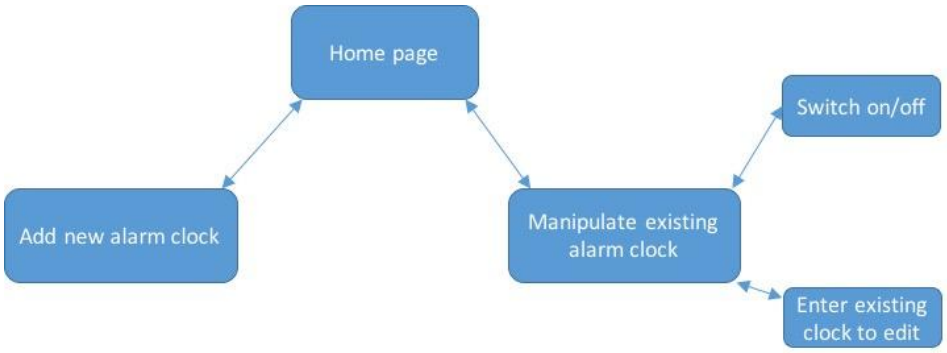

Figure 1 – Flow of Activities.

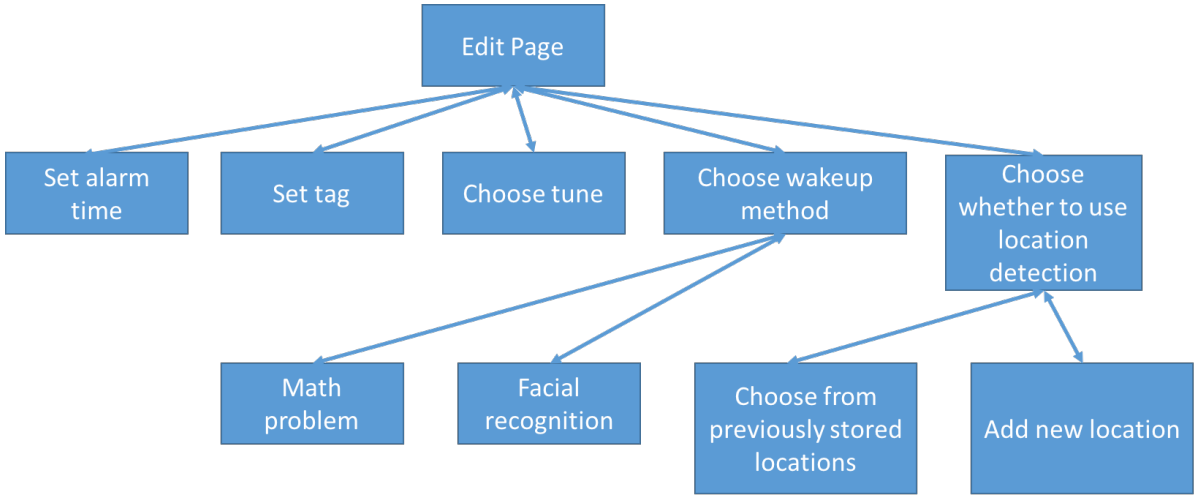

Figure 2 – Flow of Alarm's Edit Page.

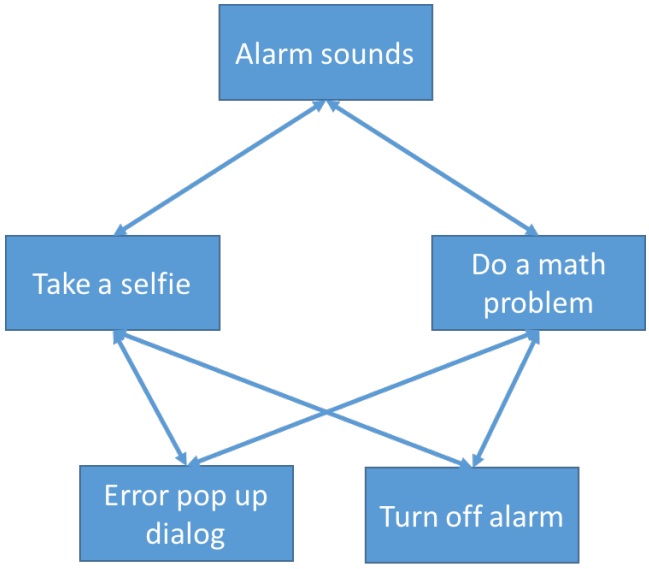

Figure 3 – Flow When Alarm Sounds.

Use Cases:

Table 1 Use Case Details – DISP-ALARM: Display Main Screen with All Alarms

Use Case ID: DISP-ALARM	**Use Case Name:** Display Main Screen with All Alarms
Primary Actor(s):	User
Secondary Actor(s):	N/A
Description:	User opens the application to display all previously added alarms with "Add New" button at the top right.
Preconditions:	N/A
Normal Flow of Events:	1. User opens the application. 2. The system retrieves all information on previously added alarms from local storage. 3. The application displays the alarms according to the information retrieved on the main screen with "Add New" buttons at the top right corner.
Postconditions:	After step 3, screen is filled with all existing alarms. If no alarms were added before, the screen only has the "Add New" button at top right corner.
Frequency of Use:	High
Alternative Flows:	N/A
Exceptions:	N/A
Assumptions:	N/A
Issues:	TBD
Source:	N/A
Includes:	N/A
Associated Requirements:	TBD

Table 2 Use Case Details – ADD-ALARM: Add New Alarm

Use Case ID: ADD-ALARM	**Use Case Name:** Add New Alarm
Primary Actor(s):	User
Secondary Actor(s):	N/A
Description:	User clicks on the "Add New" button in the main screen and adds a new alarm with different options.
Preconditions:	The application is in Main Activity with the "Add New" button at top right corner.
Normal Flow of Events:	1. User clicks on "Add New" button. 2. The application displays all the following options with "Done" button at top left and "Delete" button at top right: a. "Set Time" dropdown menus with all the hours, minutes, and AM/PM. b. "Choose Tune" dropdown menu with all available tunes and "Test" button and "Record" button.

	c. "Choose wake up method" with options: i. Math Calculation ii. Face Recognition iii. Take a picture d. "Repeat" with options: i. Once ii. Every Day iii. Weekly e. "Tag" text field. f. "Location" dropdown menu with saved addresses and an "ON/OFF" switch. 3. User select the desired settings for all options. 4. If the user selected "Take a picture" as the "Wake up method", the back camera will turn on to let user take a picture as the baseline picture for this alarm. 5. User clicks on "Done" button to create a new alarm. a. If settings are all filled out, then the new alarm is created. b. If at least one is left out, the application will show warning dialog to remind user to fill out all setting fields. Repeat steps 3, 4 and 5 until user has all settings filled out.
Postconditions:	After step 4, a new alarm with all the settings is created and stored in a local storage.
Frequency of Use:	High
Alternative Flows:	1. User clicks on "Delete" button to cancel adding a new alarm. 2. The application returns to the main screen of list of existing alarms.
Exceptions:	1. User clicks on "Done" without creating a new alarm.
Assumptions:	N/A
Issues:	TBD
Source:	N/A
Includes:	N/A
Associated Requirements:	TBD

Table 3 Use Case Details – DELETE-ALARM: Delete an Alarm

Use Case ID: DELETE-ALARM	**Use Case Name:** Delete an Alarm
Primary Actor(s):	User
Secondary Actor(s):	N/A

Description:	User deletes a previously added alarm in the main screen.
Preconditions:	The user is in the main screen with an alarm that was previously created.
Normal Flow of Events:	1. User clicks on the the alarm that is to be deleted. 2. System switches the screen to the edit screen with all options and “Done” button at top left and “Delete” button at top right. 3. User clicks on the “Delete” button.
Postconditions:	After step 3, the alarm deleted and removed from local storage.
Frequency of Use:	High
Alternative Flows:	N/A
Exceptions:	1. The alarm cannot be deleted.
Assumptions:	N/A
Issues:	TBD
Source:	N/A
Includes:	N/A
Associated Requirements:	TBD

Table 4 Use Case Details – EDIT-ALARM: Edit an Alarm

Use Case ID: EDIT-ALARM	**Use Case Name:** Edit an Alarm
Primary Actor(s):	System
Secondary Actor(s):	N/A
Description:	User edits a previously added alarm in the main screen.
Preconditions:	The user is in the main screen with an alarm that was previously created.
Normal Flow of Events:	1. User clicks on the the alarm that is to be edited. 2. System switches the screen to the edit screen with all options and “Done” button at top left and “Delete” button at top right. 3. User performs edit and change the options. 4. Once done, user clicks on “Done” button to save all the changes.
Postconditions:	After step 4, the alarm’s changed options are saved to local storage.
Frequency of Use:	High
Alternative Flows:	N/A
Exceptions:	1. The alarm cannot be edited.
Assumptions:	N/A
Issues:	TBD
Source:	N/A
Includes:	N/A

Associated Requirements:	TBD

Table 5 Use Case Details – LOCATION-SOUND: Sound an Alarm with Location

Use Case ID: LOCATION-SOUND	**Use Case Name:** Sound an Alarm: Location
Primary Actor(s):	System
Secondary Actor(s):	N/A
Description:	The application sounds the alarm after checking user location.
Preconditions:	The user added an alarm before with the location service enabled.
Normal Flow of Events:	1. The application checks user location. 2. It sees that the user is in the proximity of the location that was previously saved. 3. Application sounds the alarm and starts the task that the user will have to complete.
Postconditions:	After step 3, the alarm will be ringing.
Frequency of Use:	High
Alternative Flows:	N/A
Exceptions:	1. The user location cannot be detected.
Assumptions:	N/A
Issues:	TBD
Source:	N/A
Includes:	N/A
Associated Requirements:	TBD

Table 6 Use Case Details – LOCATION-MUTE: Mute an Alarm with Location

Use Case ID: LOCATION-MUTE	**Use Case Name:** Mute an Alarm: Location
Primary Actor(s):	Application
Secondary Actor(s):	N/A
Description:	The application doesn't sound the alarm after checking user location.
Preconditions:	The user added an alarm before with the location service enabled.
Normal Flow of Events:	1. The application checks user location. 2. It sees that the user is not in the proximity of the location that was previously saved. 3. Application does not sound the alarm.
Postconditions:	The application will not sound the alarm.
Frequency of Use:	High
Alternative Flows:	N/A
Exceptions:	1. The user location cannot be detected.
Assumptions:	N/A
Issues:	TBD

Source:	N/A
Includes:	N/A
Associated Requirements:	TBD

Table 7 Use Case Details – ALARM-MATH: Alarm Rings with MATH

Use Case ID: ALARM-MATH	**Use Case Name:** Alarm Rings with MATH
Primary Actor(s):	System
Secondary Actor(s):	User
Description:	The alarm rings and user has to turn off the alarm by doing a mathematical problem.
Preconditions:	An alarm was added with solving mathematical problem as the wakeup method.
Normal Flow of Events:	1. If the alarm has the location option on, the system checks user's location against the location option. a. If the user's location is within the proximity of the location option, continue to Step 2. b. If the user's locating is not within the proximity of the location option, do not ring the alarm. 2. The alarm goes off at the time it was set up with. 3. The system retrieves a random mathematical problem and its answer from our database. 4. The screen displays a mathematical problem for the user to solve in order to turn off the alarm along with an answer text field and "OK" button. 5. User enters an answer in the answer text field. 6. User clicks on "OK" button to submit answer and to turn off the alarm. 7. The answer submitted evaluated against the answer the system retrieved from our database earlier. a. If the answer is correct, the alarm turns off. b. If the answer is incorrect, a dialog pops up to tell the user to enter another answer. Repeats steps 5, 6 and 7 until the user enters the correct answer.
Postconditions:	1. After step 7a, the alarm will be turned off. a. If the alarm's "Repeat" option was set to "Once", then the alarm will be deactivated.

	b. If the alarm's "Repeat" option was set to either "Every Day" or "Weekly", the alarm will still be activated.
Frequency of Use:	High
Alternative Flows:	N/A
Exceptions:	1. Mathematical problem and its answer cannot be retrieved from our database.
Assumptions:	N/A
Issues:	TBD
Source:	N/A
Includes:	N/A
Associated Requirements:	TBD

Table 8 Use Case Details – ALARM-FACE: Alarm Rings with FACE

Use Case ID: ALARM-FACE	**Use Case Name:** Alarm Rings with FACE
Primary Actor(s):	System
Secondary Actor(s):	User
Description:	The alarm rings and user has to turn off the alarm by taking a selfie with opened eyes.
Preconditions:	An alarm was added with face recognition as the wakeup method.
Normal Flow of Events:	1. If the alarm has the location option on, the system checks user's location against the location option. a. If the user's location is within the proximity of the location option, continue to Step 2. b. If the user's locating is not within the proximity of the location option, do not ring the alarm. 2. The alarm goes off at the time it was set up with. 3. The system turns on the front camera. 4. User takes a selfie. 5. The picture taken will be evaluated. a. If a face is recognized with both eyes opened, the alarm turns off. b. If a face cannot be recognized with both eyes opened, a dialog pops up to tell the user to retake another selfie. Repeats steps 3, 4, and 5 until the user's is detected.
Postconditions:	1. After step 5a, the alarm will be turned off.

	a. If the alarm's "Repeat" option was set to "Once", then the alarm will be deactivated. b. If the alarm's "Repeat" option was set to either "Every Day" or "Weekly", the alarm will still be activated.
Frequency of Use:	High
Alternative Flows:	N/A
Exceptions:	1. Front camera cannot be turned on.
Assumptions:	N/A
Issues:	TBD
Source:	N/A
Includes:	N/A
Associated Requirements:	TBD

Table 9 Use Case Details – ALARM-PIC: Add New Location

Use Case ID: LOCATION-NEW	**Use Case Name:** Add Location
Primary Actor(s):	User
Secondary Actor(s):	System
Description:	The user searches for an location and add it to the application.
Preconditions:	The user was already in Settings and just clicked on "Add New Location" button.
Normal Flow of Events:	1. The user enters a location and click on the "Search" button. 2. The system looks up the address and display it with a marker on the map. 3. The user sees the location and press "Save" button to save the location.
Postconditions:	1. After step 3, the new location is saved on the phone.
Frequency of Use:	High
Alternative Flows:	N/A
Exceptions:	1. Location cannot be found.
Assumptions:	N/A
Issues:	TBD
Source:	N/A
Includes:	N/A
Associated Requirements:	TBD

Wireframes:

Figure 4 – Main Screen and Alarm's Edit Screen.

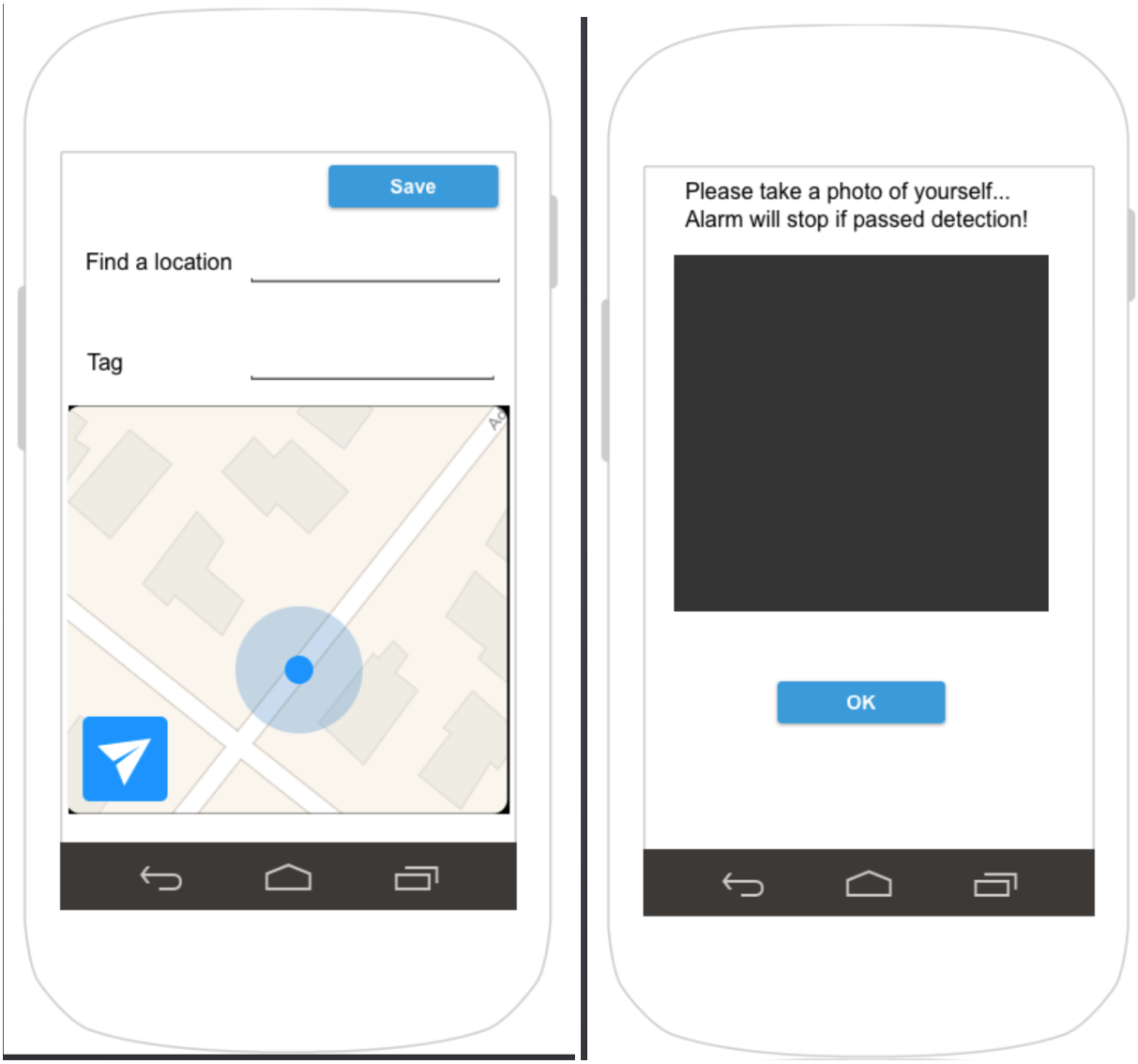

Figure 5 – Setting Location and Selfie Screen.

Figure 7 – Wakeup-Math and Not Passing Assigned Task Dialog.