Institiúid Teicneolaíochta Cheatharlach

INSTITUTE *of* TECHNOLOGY CARLOW

At the Heart of South Leinster

LifeBuoy Monitor Design Document

BSc (Hons) in Software Development

Name: Garry Byrne

Student ID: C00120055

Year: 4th Year

Supervisor: Dr. Oisin Cawley

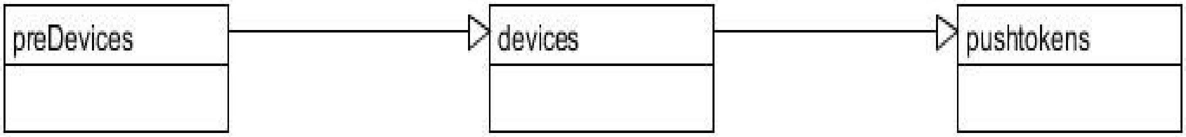
Due date: 18-04-2018

Table Of Contents

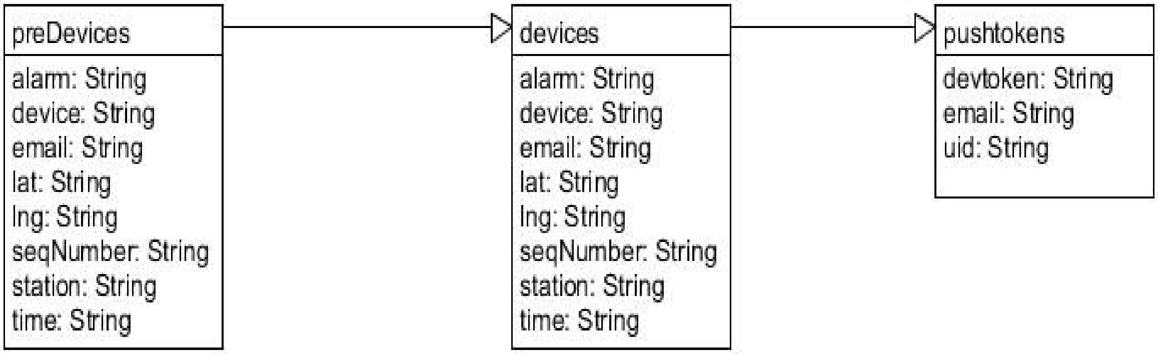
Abstract

The purpose of this document is to show the system architecture and the system design of the app.

Database Structure(Basic)

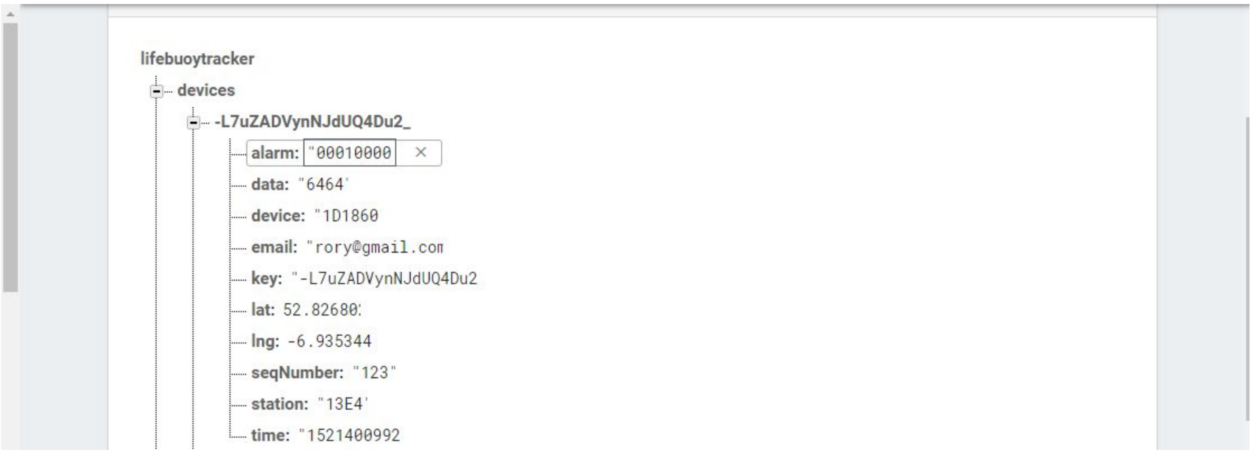


Database Structure(Detailed)



Devices Table

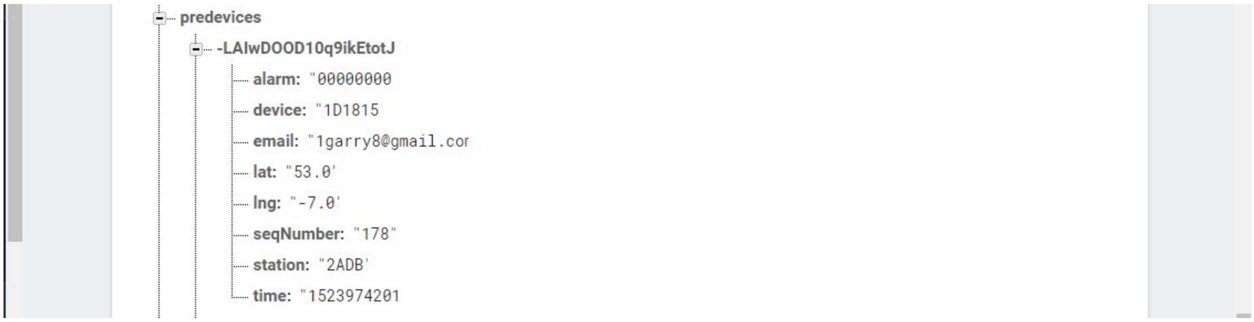
In the below image you can see the layout of the Devices table. This table is for each unique device among all users. This table is the table that gets updated either when an alarm occurs or when a location is updated.



```
lifebuoytracker
  devices
    -L7uZADVynNJdUQ4Du2_
      alarm: "00010000
      data: "6464'
      device: "1D1860
      email: "rory@gmail.com
      key: "-L7uZADVynNJdUQ4Du2
      lat: 52.82680
      lng: -6.935344
      seqNumber: "123"
      station: "13E4'
      time: "1521400992
```

Predevices table

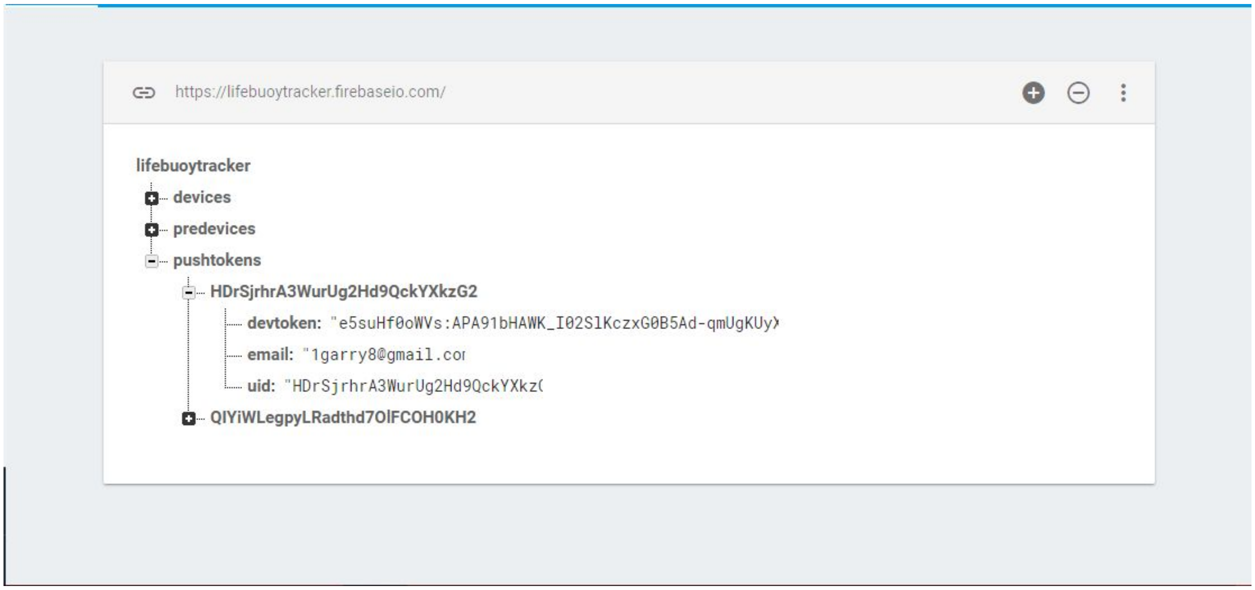
In the below image you can see the layout of the Predevices table. This table is for when a alarm message comes in or when a new device is added I use this table to store incoming messages, i thn use a cloud function to sort the entries and to make sure no duplicates enter the devices table.



```
predevices
  -LAIwDOOD10q9ikEtotJ
    alarm: "00000000
    device: "1D1815
    email: "1garry8@gmail.cor
    lat: "53.0'
    lng: "-7.0'
    seqNumber: "178"
    station: "2ADB'
    time: "1523974201
```

Push Token Table

In the below image you can see the layout of the Push Token table. This table is for when a user logs in their FCM token , email and uid get stored here. I use this table so that i can send push notifications.



https://lifebuoytracker.firebaseio.com/

```
lifebuoytracker
  + devices
  + predevices
  - pushtokens
      - HDrSjrhrA3WurUg2Hd9QckYXkzG2
          devtoken: "e5suHf0oWVs:APA91bHAWK_I02SlKczxG0B5Ad-qmUgKUy
          email: "1garry8@gmail.co
          uid: "HDrSjrhrA3WurUg2Hd9QckYXkz
      + QIYiWLegpyLRadthd7OlFCOH0KH2
```

Detailed Use Case

Detailed Use Case 1: Login

Actors: User, Database

Brief Description: This use case begins when a user wishes to log into the app. The user with be prompted to enter a username and password. If the user enters their details and they get the login details correct they will be allowed to progress to the next page relevant to that user.

If the user enter their details and they get the login details incorrect they will not be allowed to progress to the next page instead they will be asked to re-enter their details. This use case ends when the user has either successfully logged in to the web client or has given up trying to log in. If the user has logged in successfully, details of the user's login (time, date) will be stored in the relevant database.

Main Success Scenario:

1. This case starts when the user wants to Login to their account.
2. The user is shown the Login page.
3. The user fills in the form.
4. The app validates that both fields are filled in.
5. The data is sent to the server to be validated.
6. The server responds that their login credentials are correct.
7. The user is brought to the relevant page.
8. The user is now successfully logged in.

Alternatives:

4a1. The form had empty fields.
4a2. The form is returned to the user with the empty fields highlighted.
4a3. The user then fills in the empty fields.
4a4. The system returns to 4 to validate the form.

5a1. The server is not responding.
5a2. The user is returned a message informing them that the server is not responding.
5a3. The user is returned to 2.

Detailed Use Case 2: Logout

Actors: User, Database

Brief Description: This use case begins when the user wants to logout of the app. The user clicks the logout button. This use case ends when the user has successfully logged out and the login screen is displayed.

Main Success Scenario:

1. This case starts when the user wants to Logout of their account.
2. The user selects Logout.
3. The user is brought to the relevant page.
4. The user is now successfully logged out.

Detailed Use Case 3: Register

Actors: User, Database

Brief Description: This use case begins when the user wishes to Login but has yet to create an account. The user will get to the Login page where there will be an option to create an account if the user does not currently have one. The user must enter the required details. This use case ends when the user has successfully entered the required details and created an account. These user details will be stored in the relevant database.

Main Success Scenario:

1. This case starts when the user wants to Login but does not have a account.
2. The user is shown the Login page.
3. The user clicks the register button.
4. The user is shown the Register page.
5. The user fills in the form.
6. The app validates that both fields are filled in.
7. The data is sent to the server to be stored.
8. The user is brought to the relevant page(Login page).
9. The user is now successfully registered

Alternatives:

6a1. The form had empty fields.
6a2. The form is returned to the user with the empty fields highlighted.
6a3. The user then fills in the empty fields.
6a4. The system returns to 6 to validate the form.

7a1. The server is not responding.
7a2. The user is returned a message informing them that the server is not responding.
7a3. The user is returned to 2.

Detailed Use Case 4: View Devices

Actors: User, Database
Brief Description: This use case begins when the user wishes to view a device that have occurred. The user will be able to the click the ID of the desired Lifebuoy monitor device.The user will then be presented the details of the device, and will also be able to see the status of the device..

Main Success Scenario:

1. This case starts when the user wishes to view the status of a device.
2. The user selects “Device” from within the app.
3. The relevant page is displayed for the user by the app.
4. The user enters the ID of the desired lifebuoy or selects a ID from a supplied list.
5. The app validates the form (checks no empty fields).
6. The user is is then shown the details of the selected device(alarm,id, lat and long).
7. The user returns to the home screen.

Alternatives:

5a1. The server is not responding.
5a2. The user is returned a message informing them that the server is not responding.
5a3. The user is returned to 3.

Detailed Use Case 5: Search Devices

Actors: User, Database
Brief Description: This use case begins when the user wishes to search for a device. The user will be able enter the full/partial ID of the device .The user will then be presented the details of the device/devices matching the ID. The user will be click on the wanted device, they will then be able to see the details of the device.

Main Success Scenario:

1. This case starts when the user wishes to view the status of a device.
2. The user selects “Device” from within the app.
3. The relevant page is displayed for the user by the app.
4. The user enters the ID of the desired lifebuoy or selects a ID from a supplied list.
5. The app validates the form (checks no empty fields).
6. The user is is then shown the details of the selected device(alarm,id, lat and long).
7. The user returns to the home screen.

Alternatives:

5a1. The form had empty fields.
5a2. The form is returned to the user with the empty fields highlighted.
5a3. The user then fills in the empty fields
5a4. The system returns to 5 to validate the form

Detailed Use Case 6: Reset Alarms

Actors: User, Database
Brief Description: This use case begins when the user wishes to reset a alarm. The user will search for the desired device, once the device is found the user will click the device and be able to view all relevant device information. On this page the user will be able to click the reset alarm button. This will then reset the alarm and google maps marker. The use case ends when the user has successfully reset the alarm.

Main Success Scenario:

1. This case starts when the user wishes to reset a device.
2. The user selects "Device" from within the app.
3. The relevant page is displayed for the user by the app.
4. The user enters the ID of the desired lifebuoy or selects a ID from a supplied list.
5. The app validates the form (checks no empty fields).
6. The user then clicks on the relevant device .
7. The user is then shown the information of the selected device.
8. The user will then select the reset button.
9. The relevant field is updated in the table.
10. The user returns to the home screen.

Alternatives:

5a1. The form had empty fields.
5a2. The form is returned to the user with the empty fields highlighted.
5a3. The user then fills in the empty fields
5a4. The system returns to 5 to validate the form

9a1. The server is not responding.
9a2. The user is returned a message informing them that the server is not responding.
8a3. The user is returned to 2.

Detailed Use Case 7: Update Location

Actors: User, Database
Brief Description: This use case begins when the user wishes to update the location of a device. The user will search for the desired device, once the device is found the user will click the device and be able to view all relevant device information. On this page the user will be able to click the update location. This will update the longitude and latitude of the device and then reposition the relevant google maps marker. The use case ends when the user has successfully updated the alarm.

Main Success Scenario:

1. This case starts when the user wishes to update the position of a device.
2. The user selects “Device” from within the app.
3. The relevant page is displayed for the user by the app.
4. The user enters the ID of the desired lifebuoy or selects a ID from a supplied list.
5. The app validates the form (checks no empty fields).
6. The user then clicks on the relevant device .
7. The user is then shown the information of the selected device.
8. The user will then select the update location button.
9. The relevant field is updated in the table.
10. The user returns to the home screen.

Alternatives:

5a1. The form had empty fields.
5a2. The form is returned to the user with the empty fields highlighted.
5a3. The user then fills in the empty fields
5a4. The system returns to 5 to validate the form

9a1. The server is not responding.
9a2. The user is returned a message informing them that the server is not responding.
8a3. The user is returned to 2.

Detailed Use Case 8: Display Map

Actors: User, Database
Brief Description: This use case begins when the user wishes to view all devices(represented on a map).The user logs in, on the redirected page the user will be shown a google map with all their markers displayed through use of markers. The user will be able to click on a marker, this will display some basic information about the device to the user. The use case ends when the user has viewed the map.

Main Success Scenario:

1. This case starts when the user wants to view all their devices on a map.
2. The user logs in.
3. The relevant page is displayed for the user by the app.
4. The user can then see all their markers displayed on the map.
5. The user can click on any of their markers to see more information.
6. When the user is finished they can return to the homescreen.

Alternatives:

3a1. The server is not responding.
3a2. The user is returned a message informing them that the server is not responding.
3a3. The user is returned to 2.

Detailed Use Case 9: Push Notifications

Actors: User, Database, Arduino
Brief Description: This use case begins when an event has occurred with the arduino. This event will be if the accelerometer/altimeter senses any movement. If an event has occurred, the arduino will send a message to both the user through push notification and the database. This use case ends when the user has received the push notification and the details of the alert have been saved to the database.

Main Success Scenario:

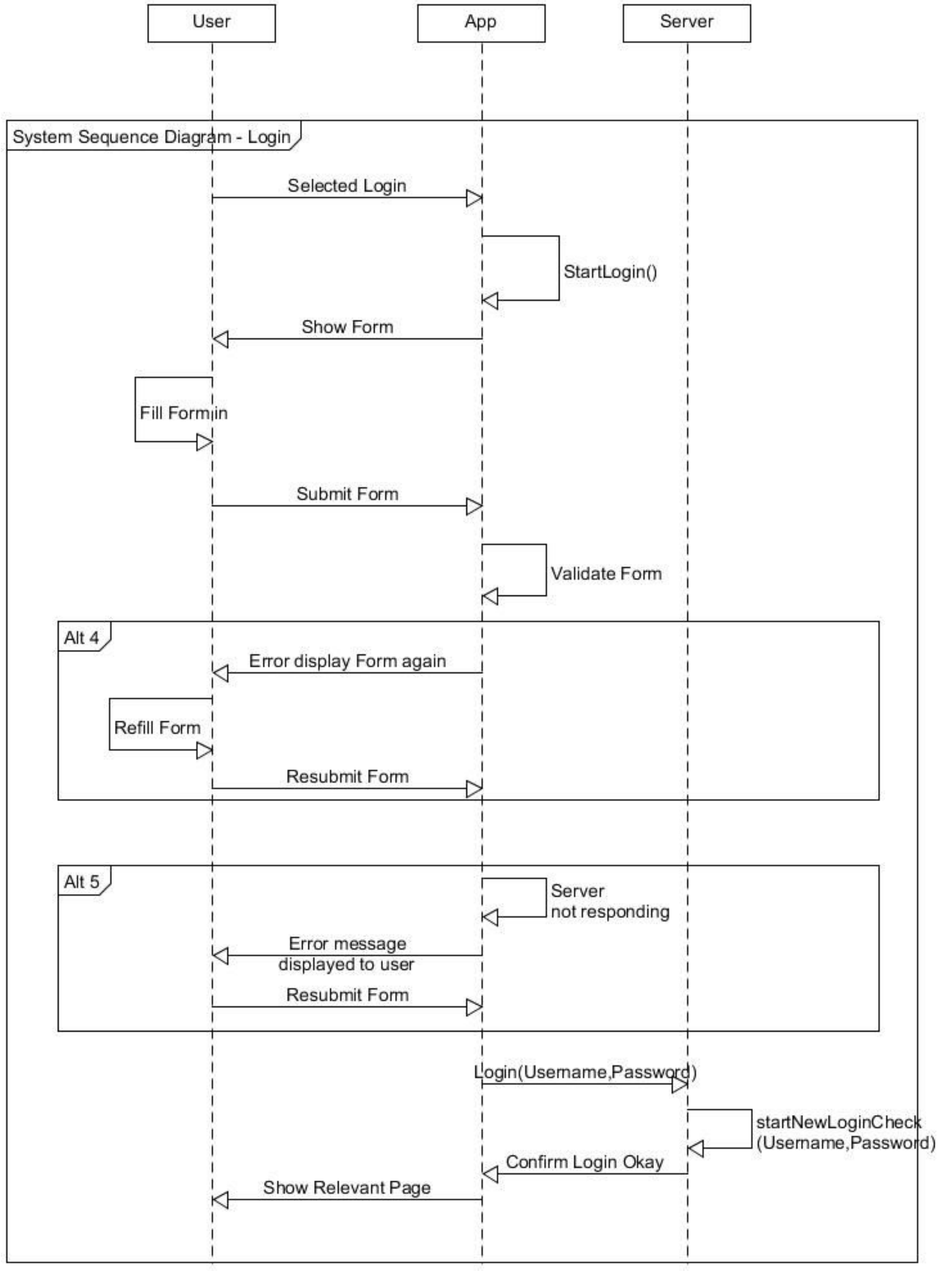
1. This case starts when the sensor senses movement.
2. The arduino sends a message to the Sigfox backend..
3. The Sigfox backend sends the received data to the database .
4. The cloud function sends a push notification to the relevant device.

Alternatives:

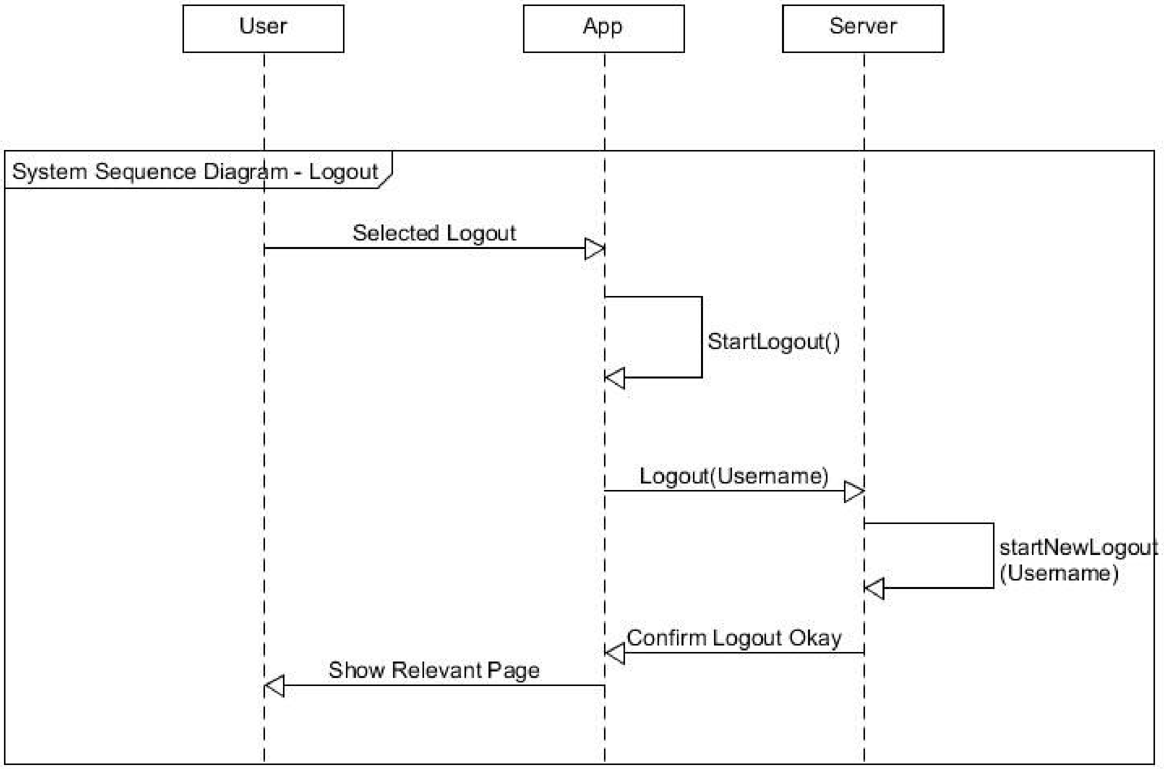
2a1. The Sigfox network is not responding.
2a2 The arduino will continue to try and send the message.

System Sequence Diagrams

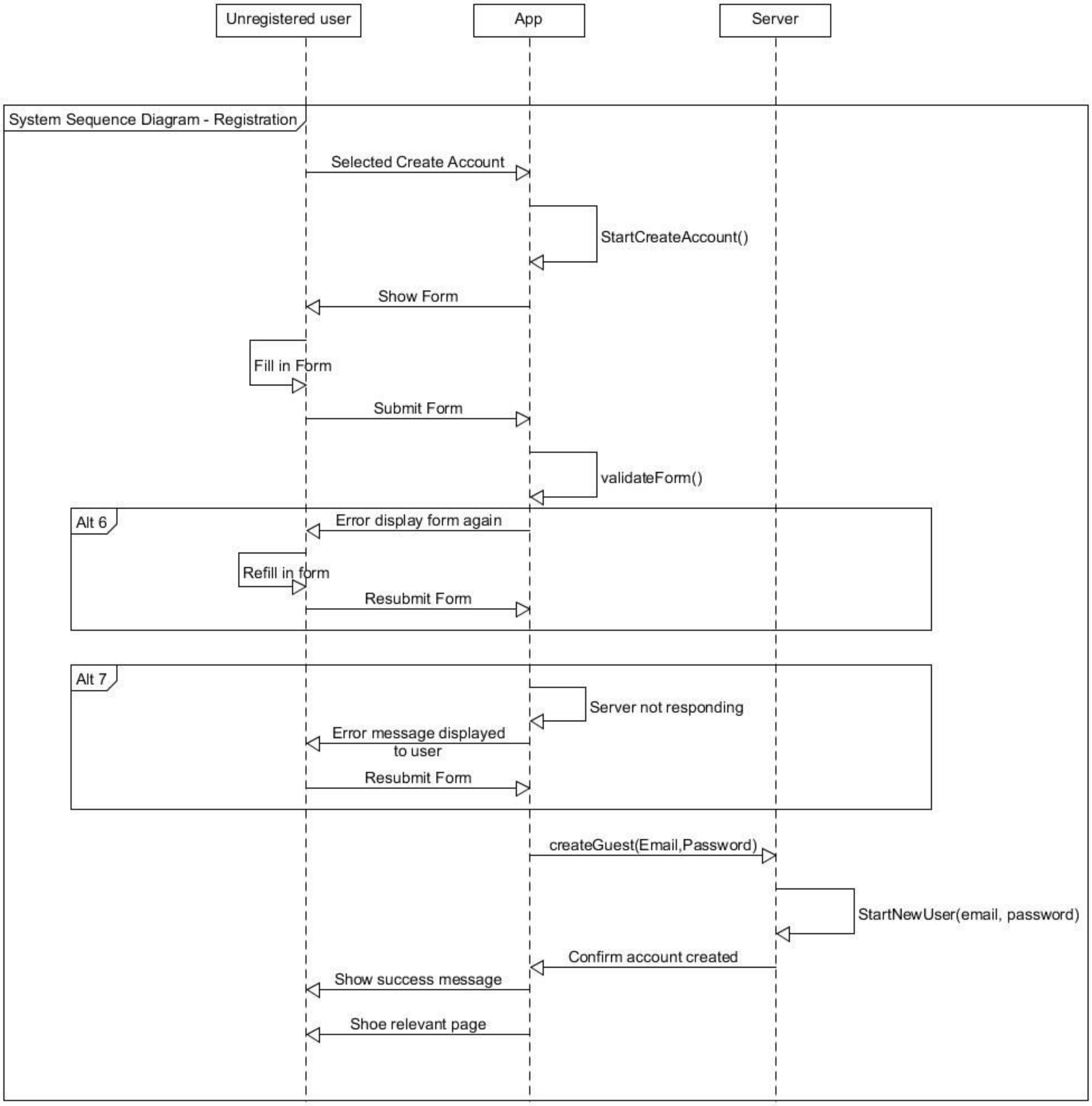
Login



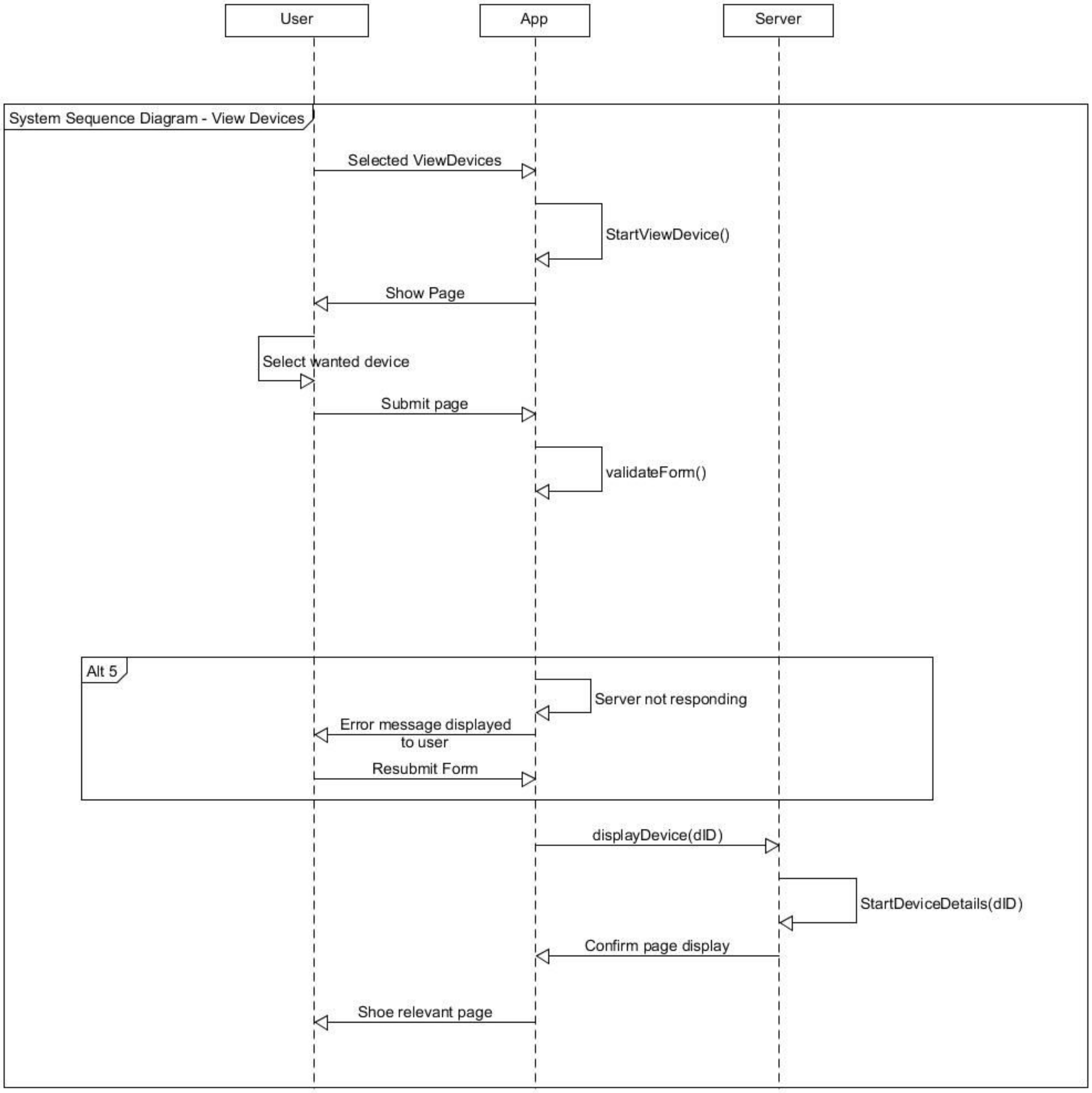
Logout


User
App
Server
System Sequence Diagram - Logout
Selected Logout
StartLogout()
Logout(Usemame)
startNewLogout
(Usemame)
Confirm Logout Okay
Show Relevant Page

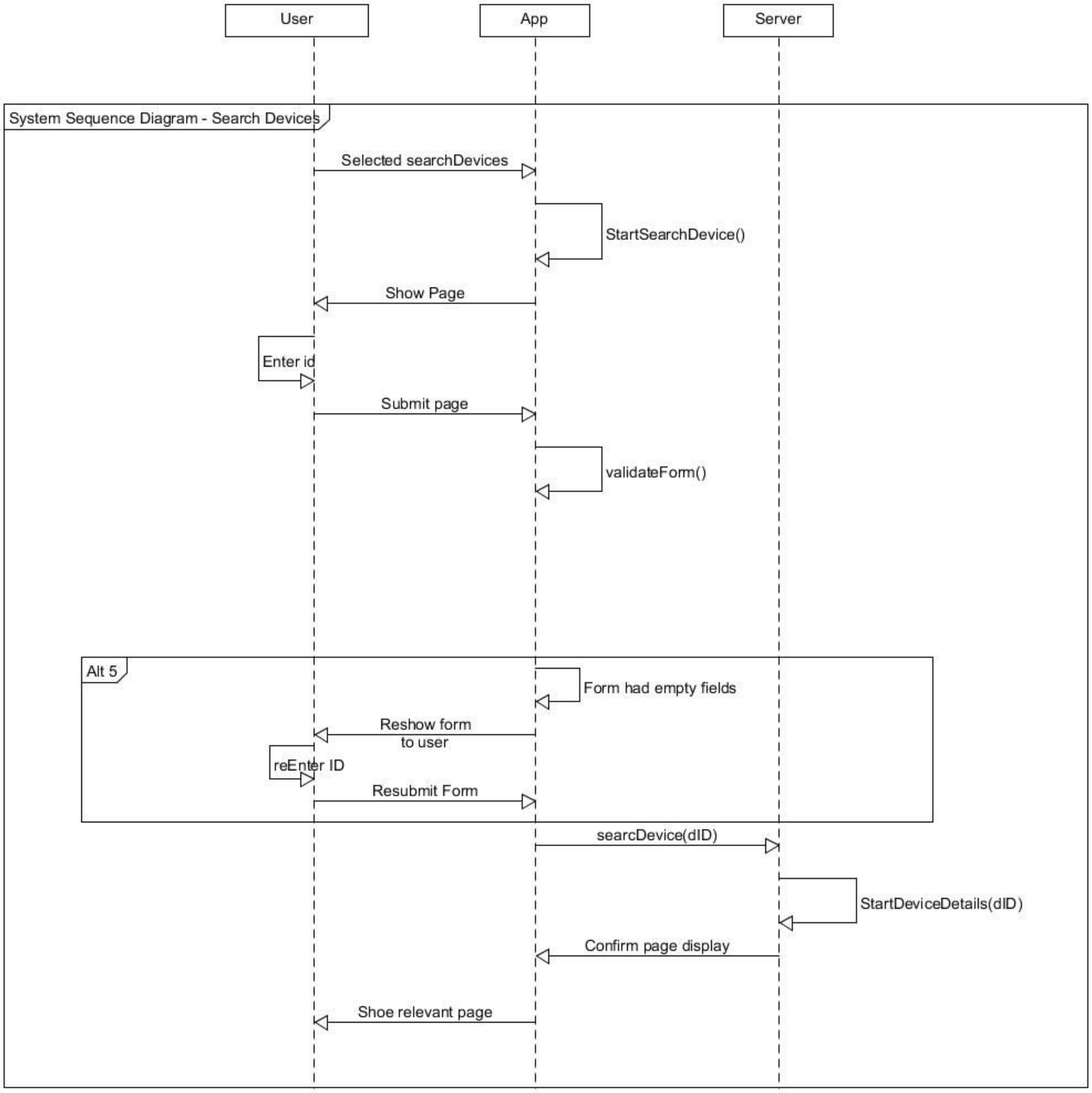
Register


Unregistered user
App
Server
System Sequence Diagram - Registration
Selected Create Account
StartCreateAccount()
Show Form
Fill in Form
Submit Form
validateForm()
Alt 6
Error display form again
Refill in form
Resubmit Form
Alt 7
Server not responding
Error message displayed to user
Resubmit Form
createGuest(Email,Password)
StartNewUser(email, password)
Confirm account created
Show success message
Shoe relevant page

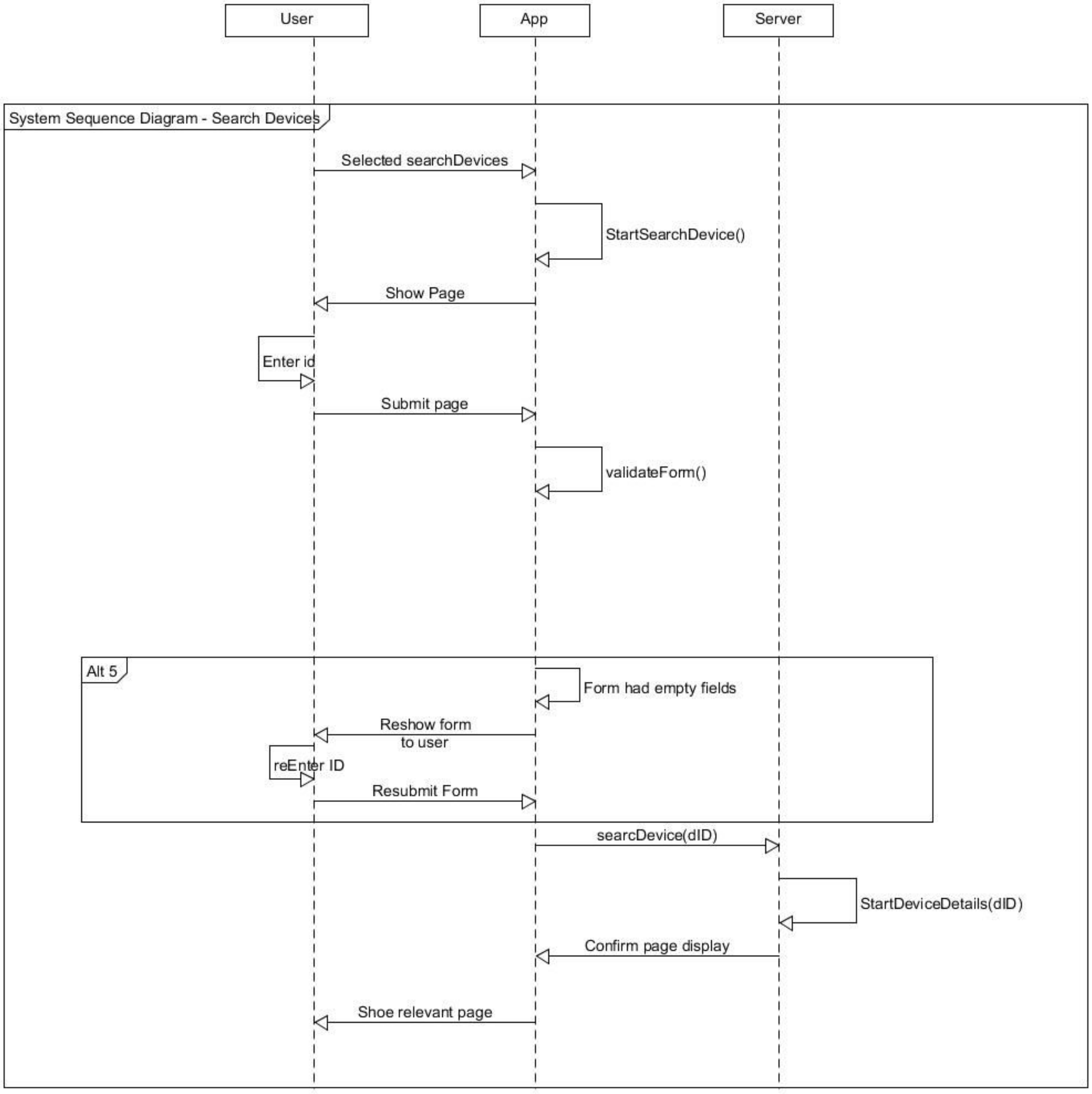
View Devices



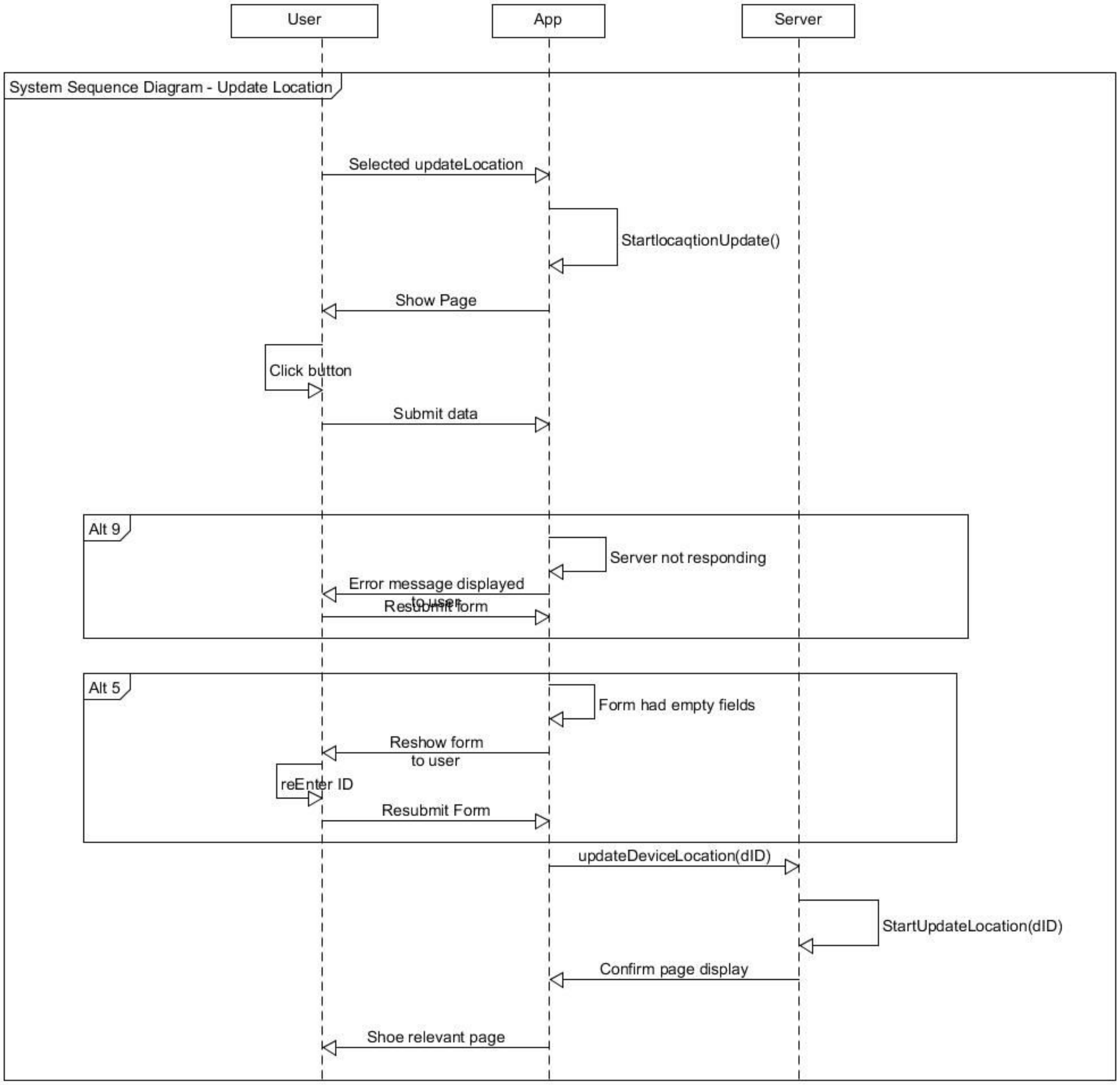
Search Devices



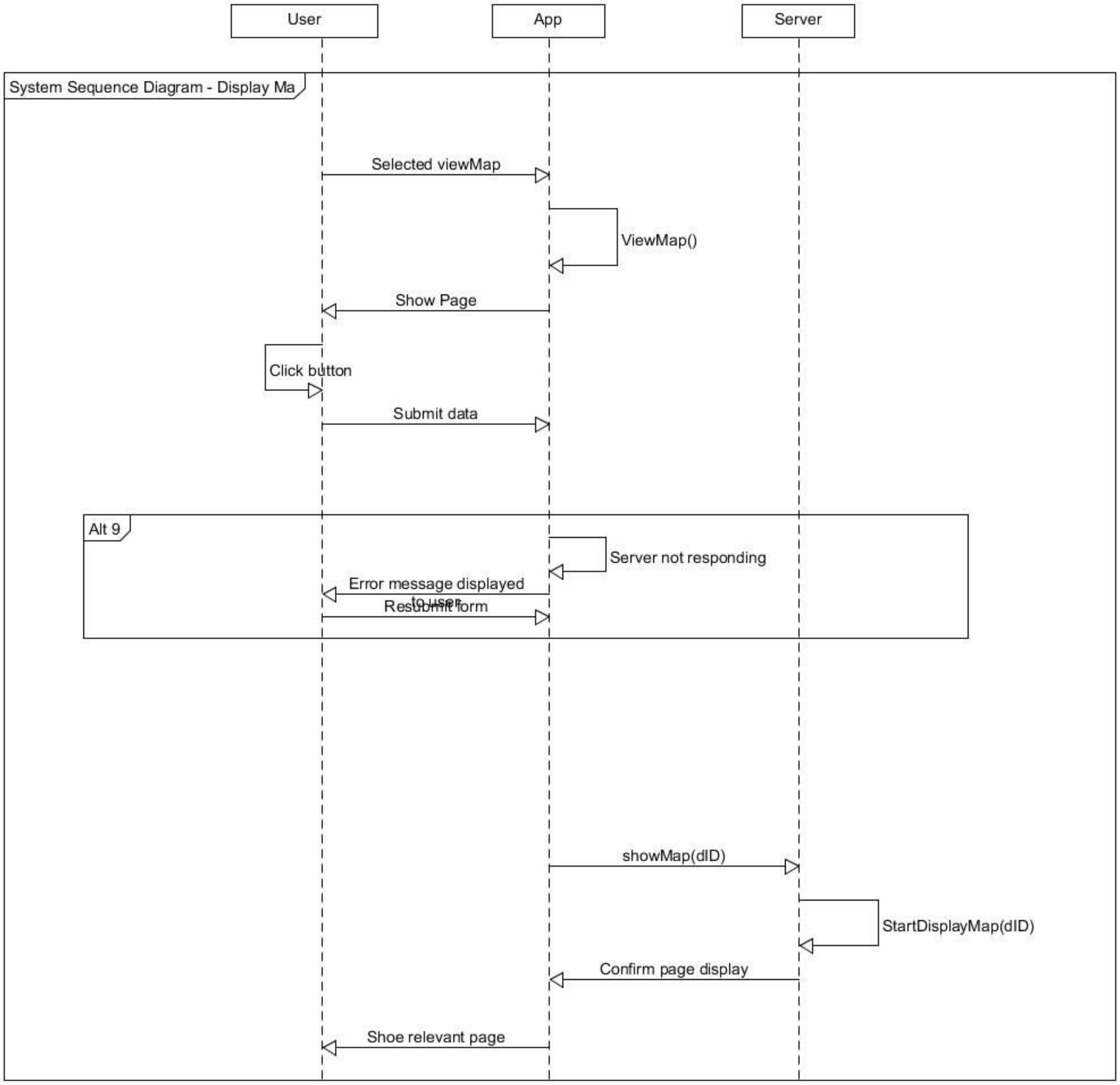
Reset Alarms



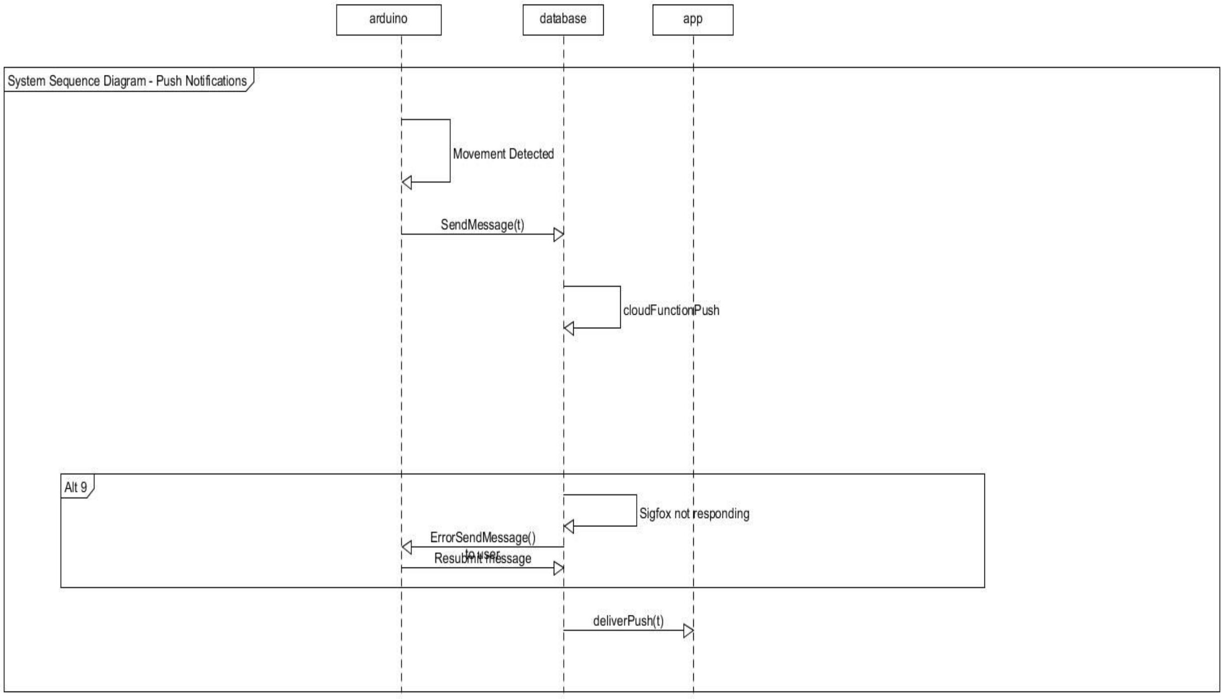
Update Location



Display Map


User
App
Server
System Sequence Diagram - Display Ma
Selected viewMap
ViewMap()
Show Page
Click button
Submit data
Alt 9
Server not responding
Error message displayed to user
Resubmit form
showMap(dID)
StartDisplayMap(dID)
Confirm page display
Shoe relevant page

Push Notifications


arduino
database
app
System Sequence Diagram - Push Notifications
Movement Detected
SendMessage(t)
cloudFunctionPush
Alt 9
Sigfox not responding
ErrorSendMessage()
to user
Resubmit message
deliverPush(t)