Volvo CoPilot Safe Assist
Group 3
Design Description

January 11, 2018

Contents

1 Introduction

1.1 Background

Volvo Construction Equipment is a company that develops, manufactures and markets equipment for construction and related industries. One of the problems that they have encountered in their construction sites is the high risk of accidents due to the close proximity of the worker with the machines at work. As an assignment in the Software Engineering 2 course at Mälardalens University, Björn Brattberg, acting as a representative of Volvo CE, has contracted the Optimus Octavian group to create an application which reduces the number of accidents at construction sites. This application, called Safe Assist, runs both on the workers' handheld devices and the machines' CoPilot. It is Android-based and issues alarms to both machine driver (through machine's CoPilot) and laborer whenever there is close proximity between the former and the latter. More importantly, however, it gives the driver of the said machine a warning that there are unprotected people close by.

1.2 Definitions

- Co-Pilot - a display used in Volvo's construction vehicles. It is an Android-based tablet with applications that have the ability to provide the driver of the vehicle with real-time intelligence from the vehicle. Custom applications that gather information such as the speed of the vehicle, the direction of the movement, the load size, fuel consumption are run in it. Currently, the Co-Pilot offers applications like Dig Assist, Load Assist and Pave Assist, to name some.
- Handheld device - any portable device which supports Android and has access to the Play Store, in order to download and use applications.
- Safe Assist - The Android application developed by the group. It consists of two working versions using the same code. One version works on the Co-Pilot platform and the other one works on normal handheld devices, to receive information such as the vehicles location.
- Worker - any person designated to do any kind of work inside the construction area. These people are considered vulnerable to an accident with a machine.
- Operator - a kind of worker which is currently working with a machine or close to it. This operator could be performing different types of work, such as driving, repairing, or any other type of work next to a machine for a long time.
- Laborer - every worker who is not an operator. Therefore, when the situation of proximity between the laborer and a machine happens, it will be considered as a critical situation.
- Construction site - area which comprehends a square formed by 2 provided points A (A-longitude, A-latitude) and B (B-longitude, B-latitude), automatically being the 2 other points: C (A-longitude, B-latitude) and D (B-longitude, A-latitude). Inside this area the machines and laborers will be notified when in close proximity.
- Unit - every element stored in the database which has one-point coordinates. Elements such as workers and machines are included in this definition, but not construction sites.

2 System Description

2.1 High-level Description

The Safe Assist system is a unique Android application that runs in both machine's CoPilot and in the workers' handheld devices (Android smartphones), adapting the application's functionalities to its current device type. The application makes use of the combined positioning information provided by workers' handheld devices, and the positioning information provided by the construction machines. When a laborer is approaching a machine, the laborer's handheld device, as well as the machine's CoPilot, issues alarms. This occurs if both machine and laborer are in a construction area. A worker can avoid the notifications when he is close to a machine if first he links to that machine through the app. This way, he becomes an operator.

In our design, we have 3 actors (Worker, Manager and Administrator). The worker provides his location while he is inside a construction area for the system to determine if he is too close to a construction machine. In this case, the worker receives an issue alarm. He also receives a notification when he enters to a construction area. Moreover, he can become an operator.

The administrator is able to add, delete or edit users to the database. He is able to edit, delete or set a construction area. The manager has the same features as the worker in addition to the administrator features.

2.2 System Overview

As mentioned above, the Safe Assist provides the laborer/s and the machine with alerts whenever close proximity between the two is calculated. To do such, it must be provided with the geographical position of the two units (worker and machine) within the construction site.

The Safe Assist, running in the laborer's handheld device, receives his geographical position by getting continuous updates from the handheld device's GPS (which runs in the background, together with the application). On each location input from the worker, the Safe Assist updates his coordinates on the database and then an then refreshes the alarm status through an update from the server. If it is necessary, the GUI is changed accordingly. The times in an interval are set by a timer which changes depending on the alarm status.

The Safe Assist, running in the machine's CoPilot, does the same process. The difference is it gathers the information needed by taking input from the CoPilot. The CoPilot, being itself an existing hardware, not to be developed in this project, communicates with the Safe Assist through a framework, which the Safe Assist adapts to. In order to know the geographical position of the vehicle, the GPS system of the CoPilot runs in the background, together with the Safe Assist and updates of the position of the vehicle are provided to the Safe Assist.

There is a server which through a PHP interface is accessed by the application. It is on this server where it is stored the database. This PHP interface allows to fetch and push data, such as geographical information of a specific unit, or the alarm status of a specific unit. Moreover, the server hosts a web tool for manual editing in the database.

2.3 Use Case Diagram

The use case diagram of the Safe Assist application is displayed in Figure 1.

2.3.1 Actors

- Worker: The worker's device with the Safe Assist application installed on it. Can login into the Safe Assist System, receive alarm and associate with CoPilot.
- Manager: Supervisor of the Safe Assist in the construction site. Can do everything a worker does. Also, sets the construction site area, can edit and delete it. Can also add, edit and delete users.
- Administrator: Administrator without worker functionalities. Can add, edit and delete users and set, edit and delete construction sites.

2.3.2 Use Case Description

- Use Case 1

  ->**Name:** Login

  ->**Initiator:** Worker, Manager

  ->**Goal:** Enter into the system to use the services offered by the application

  ->**Main Scenario:**

  1. The worker or manager enters his personal identification data to login.
  2. The system checks if the given information of the Worker or manager is valid.
  3. System allows the user to get into the system.

  ->**Extensions:**

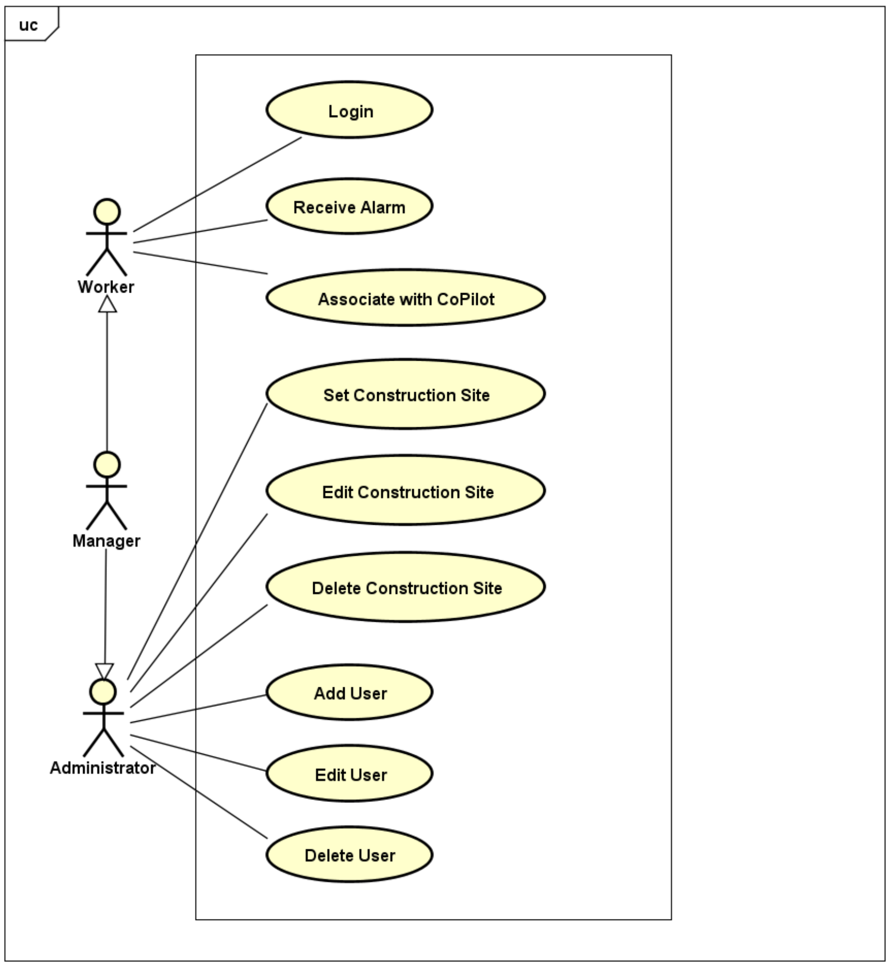


Figure 1: Use case diagram of Safe Assist

1. Given info not valid
   a) System throws an ERROR message
   b) Resume at 1

- Use Case 2

  ->**Name:** Receive Alarm

  ->**Initiator:** Worker, Manager

  ->**Goal:** Receive alarms and notifications when the device gets inside the Construction Site

  ->**Requirement:** The user has logged in and connected

  ->**Main Scenario:**

  1. The user receives a notification just after he gets into the Construction Site

  ->**Extensions:**

  1 The users connection is lost.
    a) The system displays an error message and tries to reconnect
    b) Resume at 1

- Use Case 3

  ->**Name:** Associate with CoPilot

  ->**Initiator:** Worker, manager

  ->**Goal:** For the worker or manager to associate himself with a vehicle when using it

  ->**Requirement**: Worker/manager is inlogged in the handheld device

  ->**Main Scenario:**

  1. Worker/manager turns on the vehicle and the Co-Pilot Safe Assist
  2. Worker/manager associates himself into the Co-Pilot Safe Assist through logging in the CoPilot with the same handheld device credentials

  ->**Extensions:**

  2 The association between the user and vehicle failed
    a) System throws an ERROR message
    b) Resume at 1

- Use Case 4

  ->**Name:** Set Construction Site

  ->**Initiator:** Administrator

  ->**Goal:** For the administrator to set a new construction site

  ->**Requirement:** The administrator has logged in

  ->**Main Scenario:**

  1. The administrator chooses to set a new construction site.
  2. Up-left corner point coordinates and down-right corner point coordinates of the desired rectangle site are inserted and set by the administrator to the system (the other two point are generated automatically.

  ->**Extensions:**

  2 The information about the site is invalid.
    a) System throws an ERROR message
    b) Resume at 1

- Use Case 5

  ->**Name:** Edit Construction Site

  ->**Initiator:** Administrator

  ->**Goal:** For the administrator to edit an existing construction site

  ->**Requirement:** The administrator has logged in.

  ->**Main Scenario:**

  1. The administrator chooses to edit an existing construction site
  2. The administrator picks the existing construction Site to edit from the list displayed
  3. The new values of the site's coordinates are inserted by the administrator.

  ->**Extensions:**

  3 The information about the site is invalid.
  a) System throws an ERROR message
  b) Resume at 2

- Use Case 6

  ->**Name:** Delete Construction Site

  ->**Initiator:** Administrator

  ->**Goal:** For the administrator to delete an existing construction site

  ->**Requirement:** The administrator has logged in

  ->**Main Scenario:**

  1. The administrator chooses to delete an existing construction site.
  2. The administrator chooses the site from the list and deletes it.

  ->**Extensions:**

  2 The existing site could not be deleted.
  a) System throws an ERROR message
  b) Resume at 1

- Use Case 7

  ->**Name:** Add User

  ->**Initiator:** Administrator

  ->**Goal:** For the administrator to add a new user into the system.

  ->**Requirement:** The administrator has logged in

  ->**Main Scenario:**

  1. The administrator chooses to add a new user.
  2. Information about the new user is inserted and set by the administrator to the system.

  ->**Extensions:**

  2 The information about the user is invalid.
  a) System throws an ERROR message
  b) Resume at 1

- Use Case 8

  ->**Name:** Edit User

  ->**Initiator:** Administrator

  ->**Goal:** For administrator to edit an existing user.

  ->**Requirement:** The administrator has logged in

  ->**Main Scenario:**

1. The administrator chooses to edit an existing user.
2. The new values of the site are inserted.

->**Extensions:**

2 The information about the user is invalid.
  a) System throws an ERROR message
  b) Resume at 1

- Use Case 9

  ->**Name:** Delete User

  ->**Initiator:** Administrator

  ->**Goal:** For the administrator to delete an existing user

  ->**Requirement:** The administrator has logged in

  ->**Main Scenario:**

  1. The administrator chooses to delete an existing user.
  2. The administrator chooses the user and deletes it.

  ->**Extensions:**

  2 The existing user could not be deleted.
    a) System throws an ERROR message
    b) Resume at 1

3 Software Architecture

3.1 Overview and Rationale

The system's overall software architecture is a hybrid of Model-View-Controller architecture (MVC) and Client-Server architecture. The Android Application is implemented with MVC. The view has all the GUI functionality. The model has all the logic and the inner functionalities of the application, such as obtaining the local information needed (GPS coordinates) and conducting the communication with the server. The controller connects these two components in order to make them work together.

The server is the external component which stores the database with the important data of each Android Application execution (unique unit). The application accesses the server through a PHP interface. The said PHP interface allows to perform different tasks when communicating with the server, such as login, saving the GPS coordinates or requesting the alarm status. The server has also a web tool aimed for the administrator and manager. Here the information about the workers and construction sites can be created, edited or deleted. The overall Client-Server architecture and the detailed Model-View-Controller architecture can be seen in Figure 2 and 3.

One of the main reasons to use MVC is that the application will be used on devices with differing screen sizes. Having a separate component for the GUI allows for an effortless adaptation to the various conditions. A centralized data storage and computation aiding server were chosen due to the need for shared data and to unload the battery-powered mobile devices with heavy workloads. The Client-Server architecture is needed since there will be many client executions which need a server to communicate.

3.2 Hardware/Software Mapping

The Safe Assist application runs in two devices: the workers' handheld devices and the machines' CoPilot. The application is unique and identifies the hardware it is being run in, to choose the functionalities which will use. The Android application is implemented in Java for the logic and in XML for the GUI. The database, the web tool and the PHP interface run in a server. The database is implemented in MySQL through PHPMyAdmin and the server interface is programmed in PHP. The web tool is implemented with Bootstrap framework using HTML, CSS, PHP and JavaScript. Communication through the clients and the server is done through Internet connection.

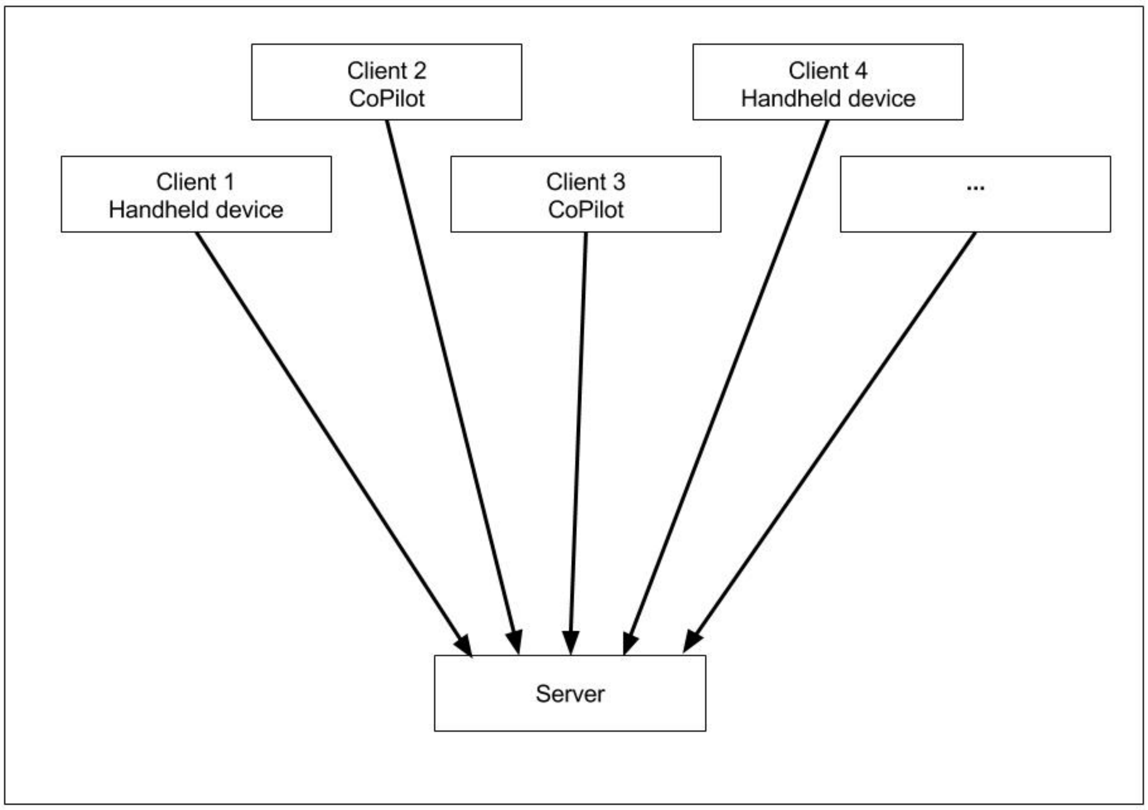


Figure 2: General architecture: Client-Server

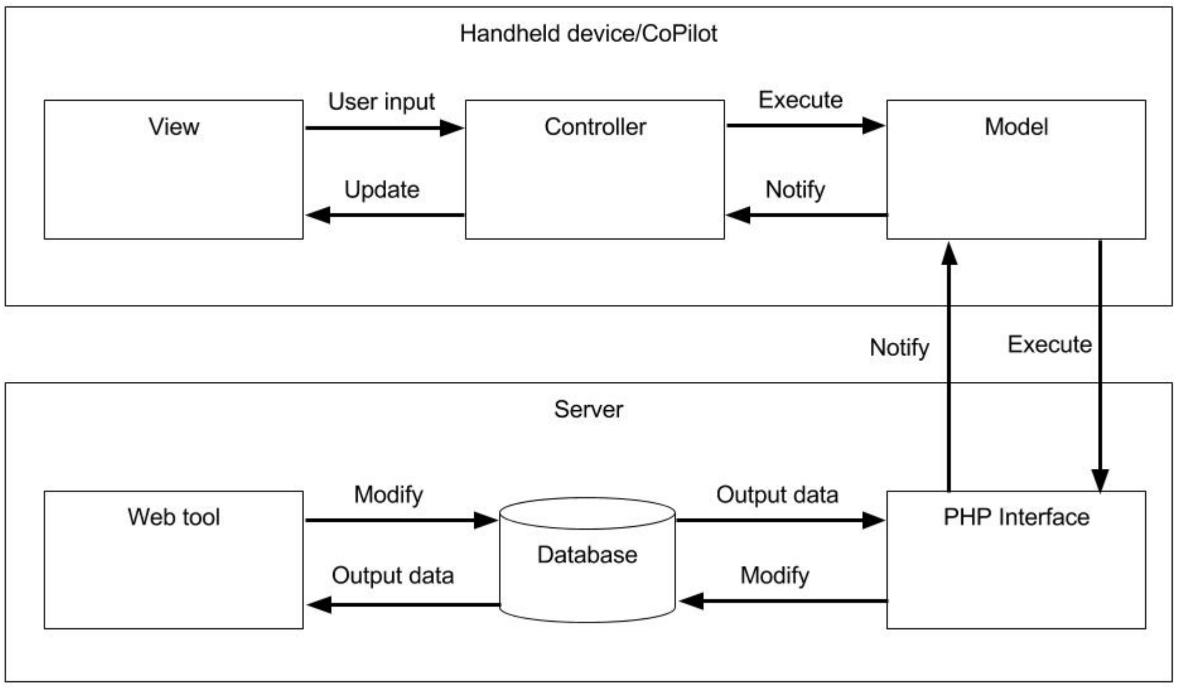


Figure 3: Detailed architecture: MVC and Server

3.3 Persistent Data

Persistent data in the Safe Assist application is comprised of data in the database which provides the manager and administrator rights to specific people within the company.

3.4 Access Control

This product (Safe Assist application) is owned by Volvo, and used by Volvo CE Equipment workers. ID and password authentication method is used in the application to grant its users access to it.

3.5 Synchronization and timing

The location update is synchronized so it is updated in the application at the same rate as in the database. This way, it avoids extra location updates which would not be of any use and would drain the battery. The update rate depends on the alarm status of the unit. The closer to a critical situation, the greater the number of location updates per second.

3.6 Start-Up and Shut-Down

The start-up process includes the process of identifying the device where the Safe Assist application has been booted up on to use handheld or CoPilot functionalities. It also includes the login of the user which is done through the server PHP interface.

3.7 Error Handling

In the situation where Internet connection and/or GPS signal is lost, a notification is displayed in the user interface and the application tries to reestablish the connection.

4 Detailed Software Design

4.1 System Components

- Client
  - Component Interface Description

    This component communicates with the actual user executing it and with the server. The communication with the server is done through the server PHP interface from the GeoLocation and alarm classes. The communication with the user is done through the GUI contained in the View.
  - Component Design

    As seen in the class diagram of the Client (depicted in Figure 4), there are ten classes: LoginActivity, MainActivity, Unit, UnitType, Constants, ServerComService, GeoLocation, HandheldLocation, CoPilotLocation and Alarm. The view classes are the layouts stored in the application, which are not included in the Class Diagram. The class LoginActivity has the functionality of the first screen: the login. The class MainActivity is the controller between the main layout and the model of the application. MainActivity stores the class Unit. This way it is easy to check the unit information, which is loaded when the application is booted. The enumerator UnitType is stored in Unit, and it states the unit type: handheld device or CoPilot. The class Constants is used as a dictionary and has constants to make all the code more understandable. A reference to ServerComService is stored in the MainActivity since this class creates it. It has the main functionality of the application, guiding the server communication. This class and the references it has comprise the model of the system. ServerComService has a reference to GeoLocation and Alarm which have the functionality of the main two tasks which can be done while accessing the server. GeoLocation is an abstract class which can be extended by HandheldLocation or CoPilot location, depending on the type of unit which booted the system. This way the two distinct implementations of getting the GPS coordinates locally and pushing it to the server can be separated. Alarm has the functionality to fetch the alarm status. It is not divided since the logic is mainly the same for the two unit types.

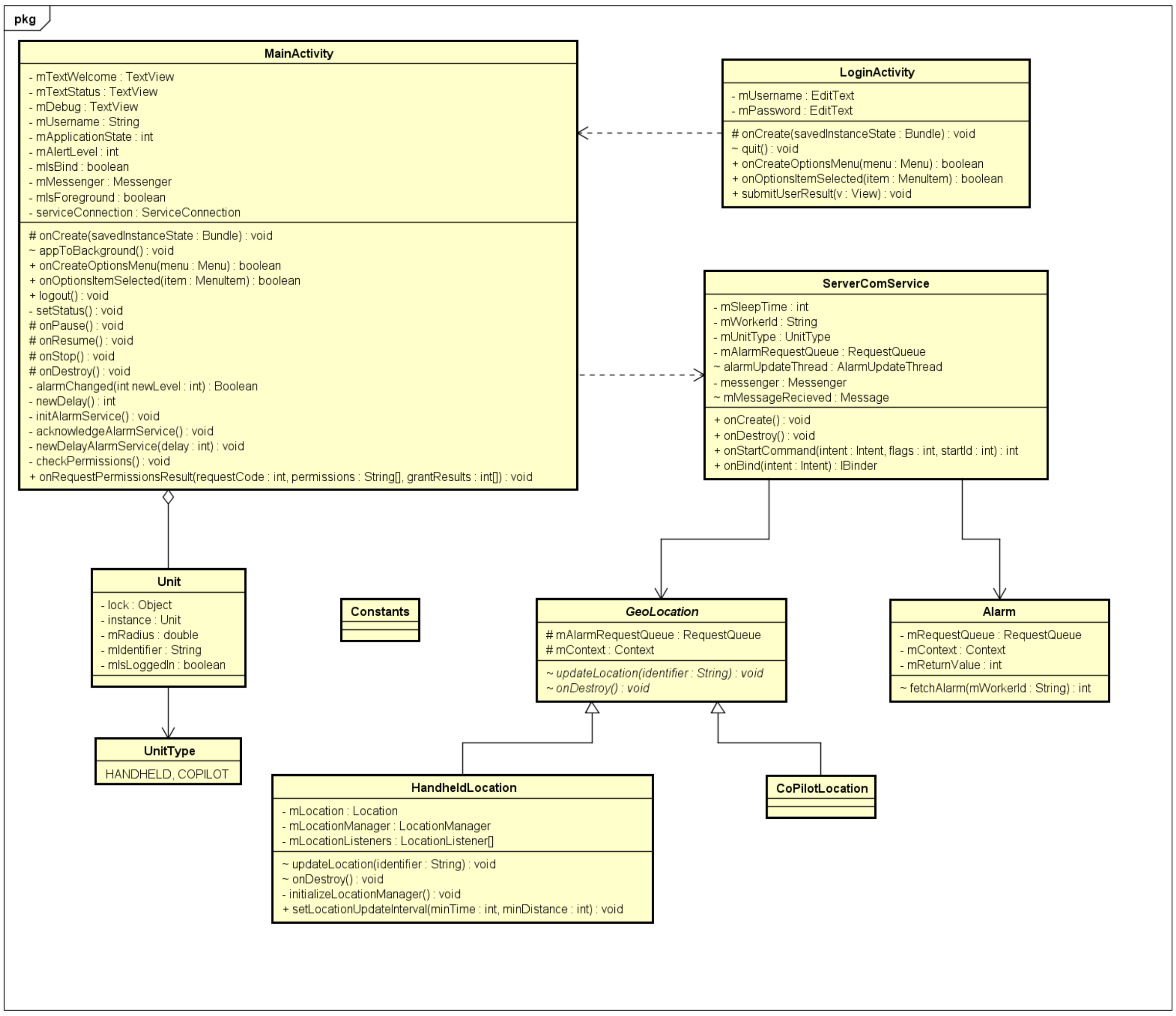

pkg
MainActivity
- mTextWelcome : TextView
- mTextStatus : TextView
- mDebug : TextView
- mUsername : String
- mApplicationState : int
- mAlertLevel : int
- mIsBind : boolean
- mMessenger : Messenger
- mIsForeground : boolean
- serviceConnection : ServiceConnection
# onCreate(savedInstanceState : Bundle) : void
~ appToBackground() : void
+ onCreateOptionsMenu(menu : Menu) : boolean
+ onOptionsItemSelected(item : MenuItem) : boolean
+ logout() : void
- setStatus() : void
# onPause() : void
# onResume() : void
# onStop() : void
# onDestroy() : void
- alarmChanged(int newLevel : int) : Boolean
- newDelay() : int
- initAlarmService() : void
- acknowledgeAlarmService() : void
- newDelayAlarmService(delay : int) : void
- checkPermissions() : void
+ onRequestPermissionsResult(requestCode : int, permissions : String[], grantResults : int[]) : void
LoginActivity
- mUsername : EditText
- mPassword : EditText
# onCreate(savedInstanceState : Bundle) : void
~ quit() : void
+ onCreateOptionsMenu(menu : Menu) : boolean
+ onOptionsItemSelected(item : MenuItem) : boolean
+ submitUserResult(v : View) : void
ServerComService
- mSleepTime : int
- mWorkerId : String
- mUnitType : UnitType
- mAlarmRequestQueue : RequestQueue
~ alarmUpdateThread : AlarmUpdateThread
- messenger : Messenger
~ mMessageRecieved : Message
+ onCreate() : void
+ onDestroy() : void
+ onStartCommand(intent : Intent, flags : int, startId : int) : int
+ onBind(intent : Intent) : IBinder
Unit
- lock : Object
- instance : Unit
- mRadius : double
- mIdentifier : String
- mIsLoggedIn : boolean
Constants
GeoLocation
# mAlarmRequestQueue : RequestQueue
# mContext : Context
~ updateLocation(identifier : String) : void
~ onDestroy() : void
Alarm
- mRequestQueue : RequestQueue
- mContext : Context
- mReturnValue : int
~ fetchAlarm(mWorkerId : String) : int
UnitType
HANDHELD, COPILOT
HandheldLocation
- mLocation : Location
- mLocationManager : LocationManager
- mLocationListeners : LocationListener[]
~ updateLocation(identifier : String) : void
~ onDestroy() : void
- initializeLocationManager() : void
+ setLocationUpdateInterval(minTime : int, minDistance : int) : void
CoPilotLocation

Figure 4: Class Diagram of the Client

  - Workflows and Algorithms
    Depicted in Figure 4.

- Server PHP Interface

  - Component Interface Description As a user starts the application, the client (Safe Assist application) sets up communication with the server to check login information at first. Once the login is confirmed the server uses the location-data being sent to it, to see if the client is inside a working area and update location-data. All the data being sent (locations and areas) is then used to send appropriate alerts, e.g. entering-notification or the level of proximity-alert. The web interface is solely used by the administrator and manager to manage worker accounts and construction sites. The communication with the database can also be done manually through the web interface.
  - Component Design
    Class diagram of the Server component is displayed in Figure 5. There are five classes as you can see in class-diagram of the server (Figure 5). The class AppInterface is the main class in which it is selected the action to be executed. The class Login handles all login authentications. Alarm provides the utility and functionality of sending alarms and notifications. Location has the functionality to update coordinates of a unit. Connection is used by all the functionalities to communicate with database.

- Database

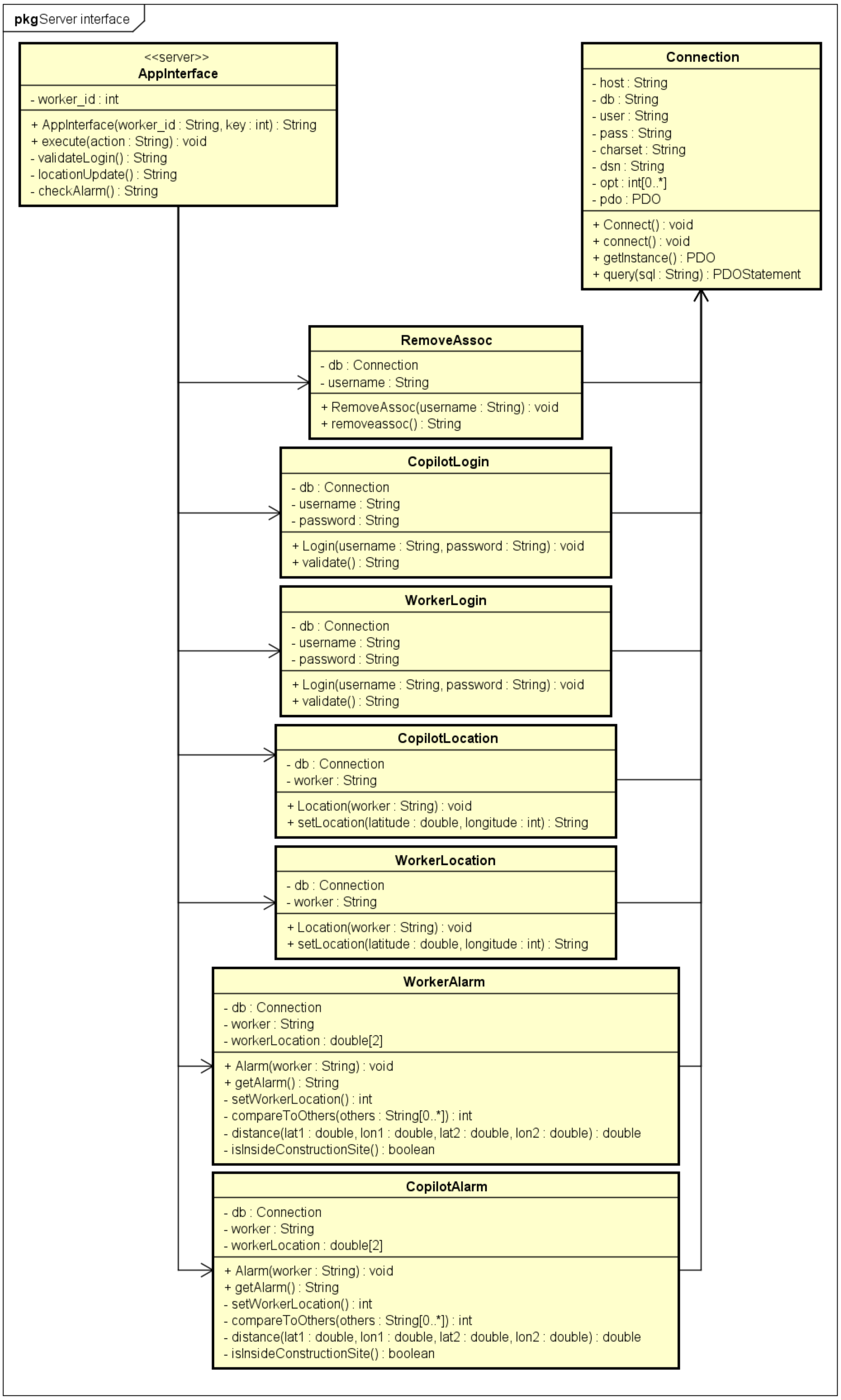

pkg Server interface
<<server>>
AppInterface
- worker_id : int
+ AppInterface(worker_id : String, key : int) : String
+ execute(action : String) : void
- validateLogin() : String
- locationUpdate() : String
- checkAlarm() : String
Connection
- host : String
- db : String
- user : String
- pass : String
- charset : String
- dsn : String
- opt : int[0..*]
- pdo : PDO
+ Connect() : void
+ connect() : void
+ getInstance() : PDO
+ query(sql : String) : PDOStatement
RemoveAssoc
- db : Connection
- username : String
+ RemoveAssoc(username : String) : void
+ removeassoc() : String
CopilotLogin
- db : Connection
- username : String
- password : String
+ Login(username : String, password : String) : void
+ validate() : String
WorkerLogin
- db : Connection
- username : String
- password : String
+ Login(username : String, password : String) : void
+ validate() : String
CopilotLocation
- db : Connection
- worker : String
+ Location(worker : String) : void
+ setLocation(latitude : double, longitude : int) : String
WorkerLocation
- db : Connection
- worker : String
+ Location(worker : String) : void
+ setLocation(latitude : double, longitude : int) : String
WorkerAlarm
- db : Connection
- worker : String
- workerLocation : double[2]
+ Alarm(worker : String) : void
+ getAlarm() : String
- setWorkerLocation() : int
- compareToOthers(others : String[0..*]) : int
- distance(lat1 : double, lon1 : double, lat2 : double, lon2 : double) : double
- isInsideConstructionSite() : boolean
CopilotAlarm
- db : Connection
- worker : String
- workerLocation : double[2]
+ Alarm(worker : String) : void
+ getAlarm() : String
- setWorkerLocation() : int
- compareToOthers(others : String[0..*]) : int
- distance(lat1 : double, lon1 : double, lat2 : double, lon2 : double) : double
- isInsideConstructionSite() : boolean

Figure 5: Class Diagram of the Server PHP Interface

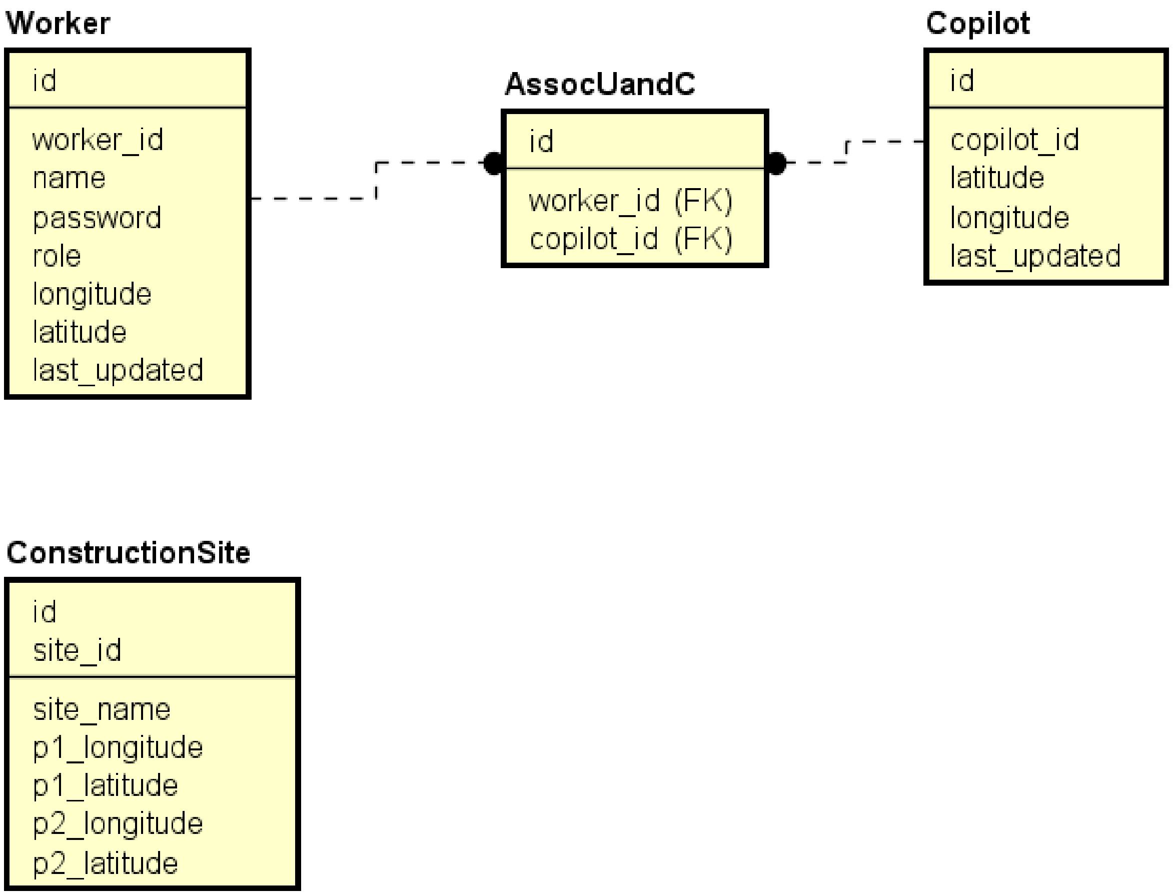


Figure 6: ER Diagram

- Component Interface Description

  The database, so far, is comprised of four tables: Worker, AssocUandC (Associate user and copilot), CoPilot, ConstructionSite. Worker stores the information of every worker. CoPilot stores the information of every machine with CoPilot. ConstructionSite stores all the registered construction sites, and AssocUandC stores the link between CoPilots and workers, which now are operators. All tables have their unique identifiers. Worker, Copilot and ConstructionSite have GPS-location-data in the forms of longitude and latitude. The relation between Worker and AssocUandC, and CoPilot and AssocUand C is 1-1.

- Component Design

  Database design is described in Figure 6.

4.2 State Machine Diagram

State machine diagram of the Safe Assist is displayed in Figure 7. When Safe Assist application is initialized, it identifies the unit and it enters the Login state depending on which device it is activated upon. In case it is activated in the CoPilot, after logging in, it goes into Set Operator state, which associates the operator, who will run the vehicle, with the vehicle. Afterwards, if successful, it enters the state of Outside Construction Site. On a regular interval, it checks if it is inside the construction area and if that is true, it goes into the state Inside Construction Area, and on entering the state it issues a notification to the unit. This state is checked by the constant event of sending Coordinates. After the appropriate calculations, there are two possible states: Risk level 1 - which issues alert messages - and Risk level 2 - which issues alarm. In the case of No GPS or Internet Connection, the Safe Assist passes on the No Connection state which issues warnings and establishes GPS and Internet Connection.

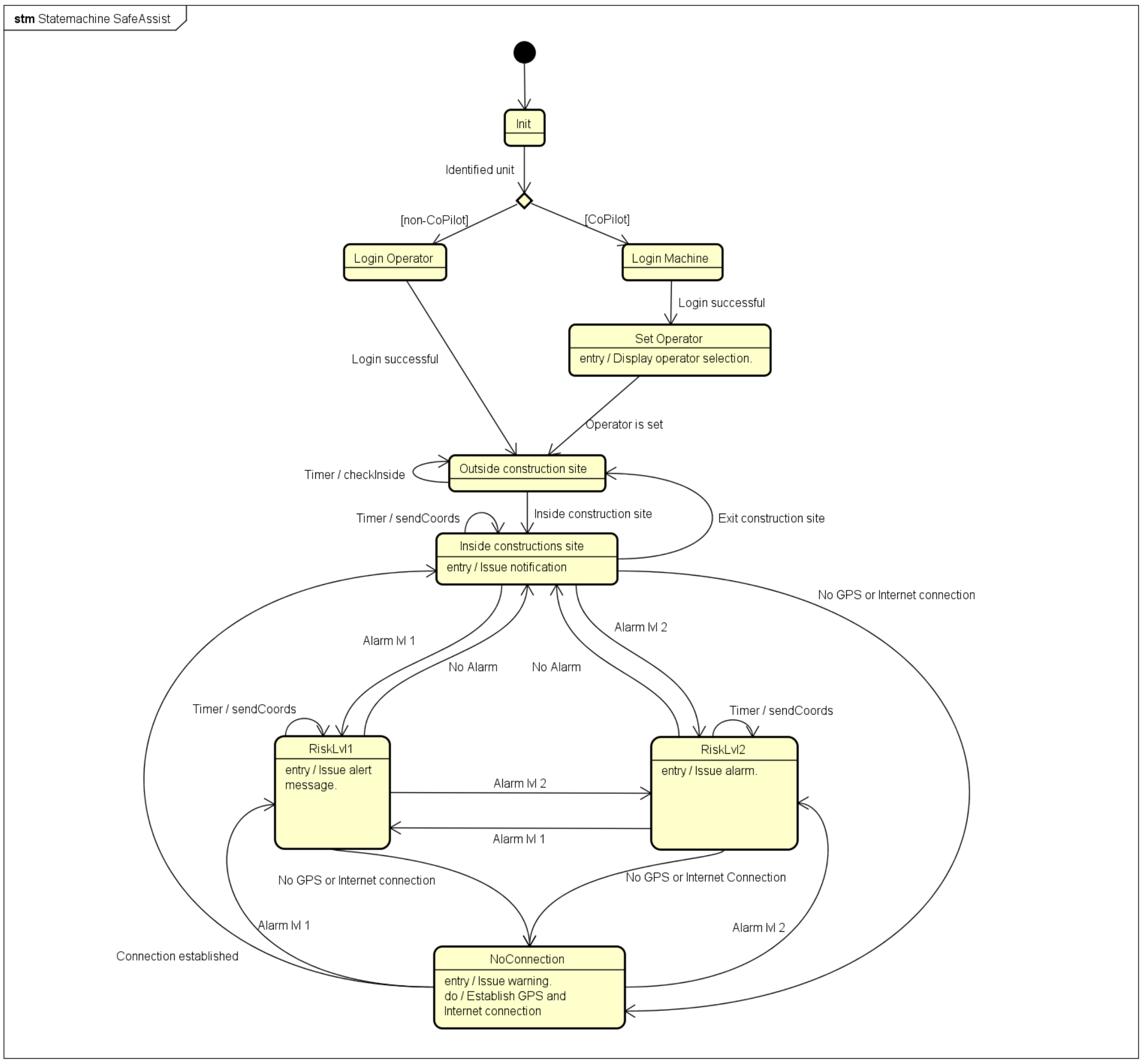


Figure 7: State Diagram of Safe Assist

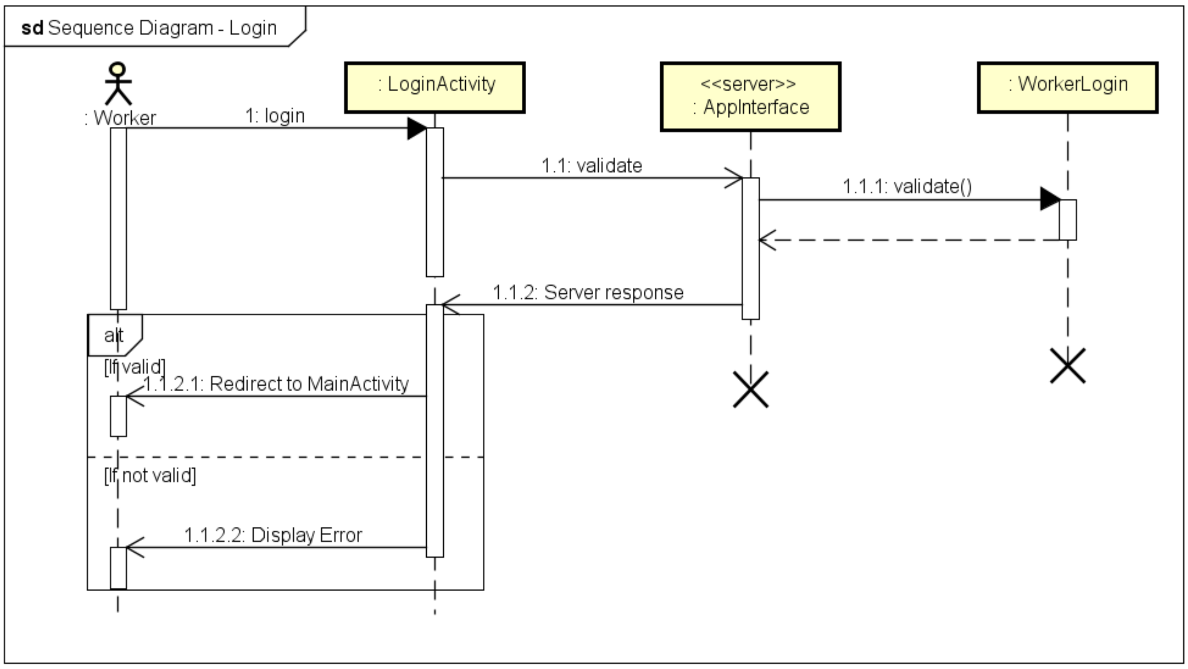


Figure 8: Sequence Diagram of Login

4.3 Dynamic Behavior

Dynamic behavior of the system is displayed in Figure 8 and 9.

5 Graphical User Interface

5.1 Overview of the user interface

There is an Android application for the user and web application for the administrator. The Android application provides all of the users the possibility to login with their unique ID and password. By doing so, the application is able to get the users' exact location (longitude and latitude) and save them in the database, to be subsequently compared to other users' location.

After the user logs in, he can choose to exit the application - meaning it will run on background - or keep it open on the handheld device/CoPilot. The construction area location will be saved in the database. By being able to get the user's location first, we check if the user is already inside a working area. If he isn't, it displays "You are outside of all construction sites" (Figure 11). In this occasion, we notify him with an alert in the application with the message "Caution! You are inside a construction site." (Figure 12). Afterwards, it is checked and compared the distance between the worker and the machine operator. If a laborer and a machine are a bit close, an alert is displayed: "Alert! Look out for vehicles!" (Figure 13). If they are too close, we notify them with a pop up alert with the message "Warning! Vehicles too close!" (Figure 14).

Regarding the administrator's and manager's interface, he has a web application to manage the users. Basically, the web application provides the administrator with these functionalities: log in, log out, list all users, add users, edit user data and delete them. For the second development phase, a map of the construction area and the workers on it can be displayed.

In the next subsection, from Figure 15 to 19, the screenshots of how the web page of the administrator look like are displayed.

5.2 Screen Images

In the following images, it is shown how the Android and web application look like are displayed, together with a brief explanation.

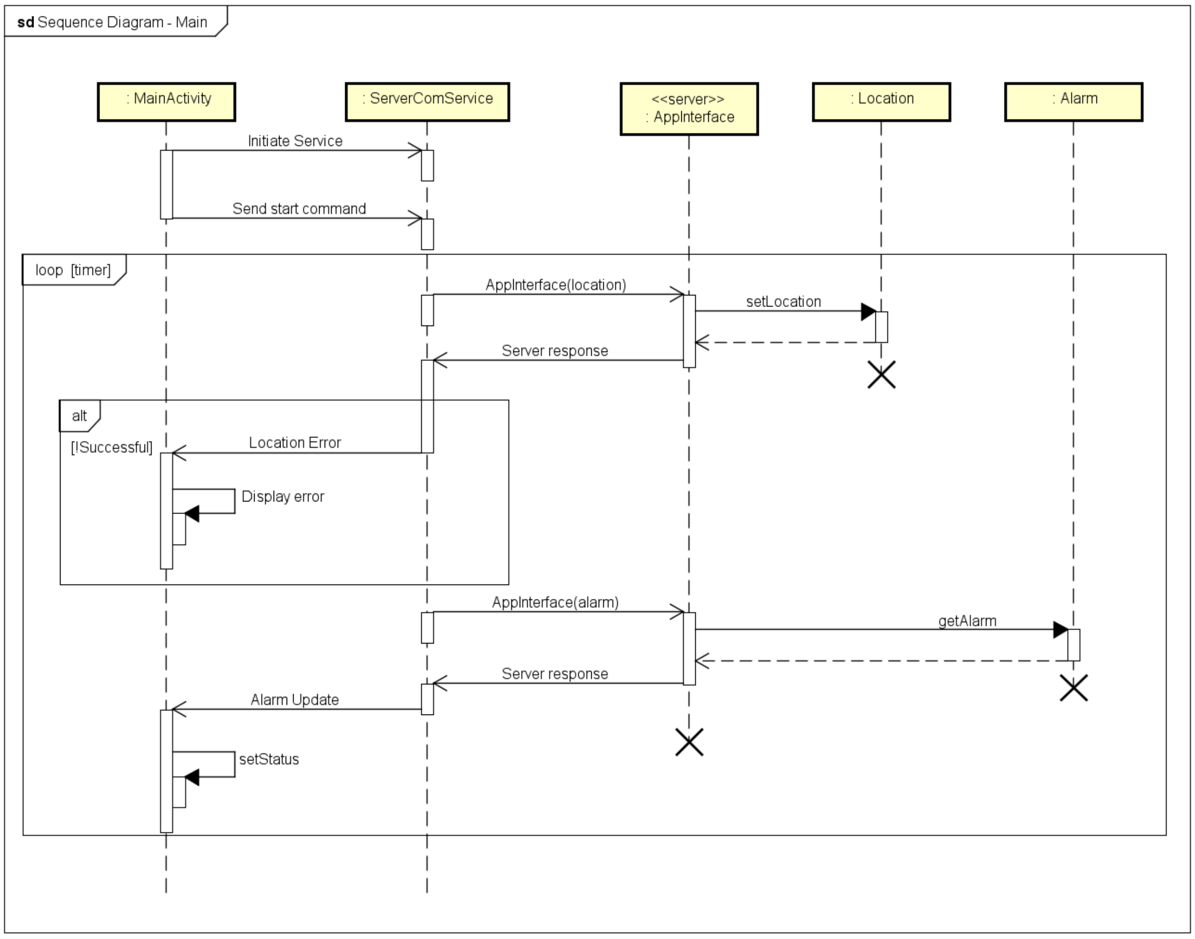


Figure 9: Main Sequence Diagram

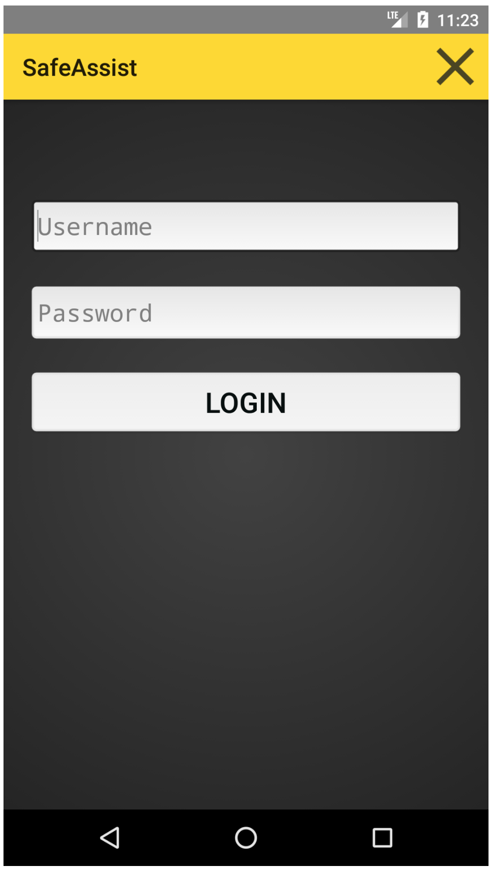
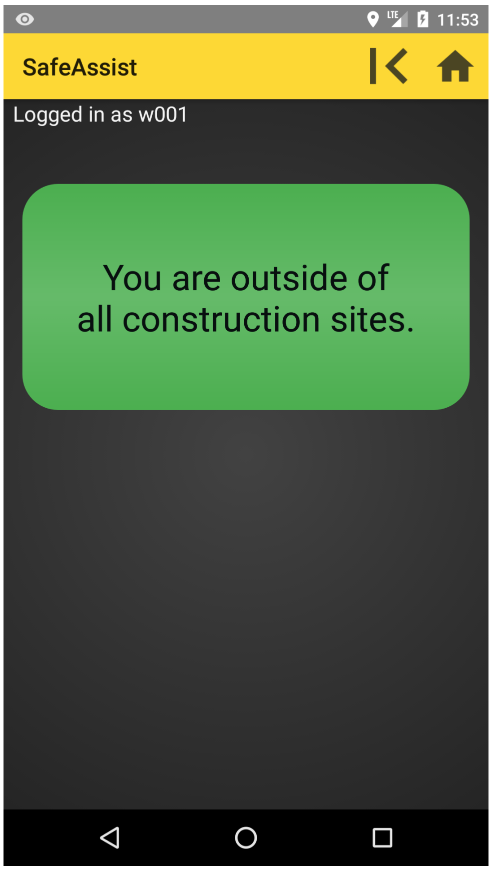


Figure 10: Login GUI
Figure 10 displays the log in view. By providing his/her unique ID and password, the worker, machine operator or the manager are able to login in the application and enter the system in order to get a safer environment.

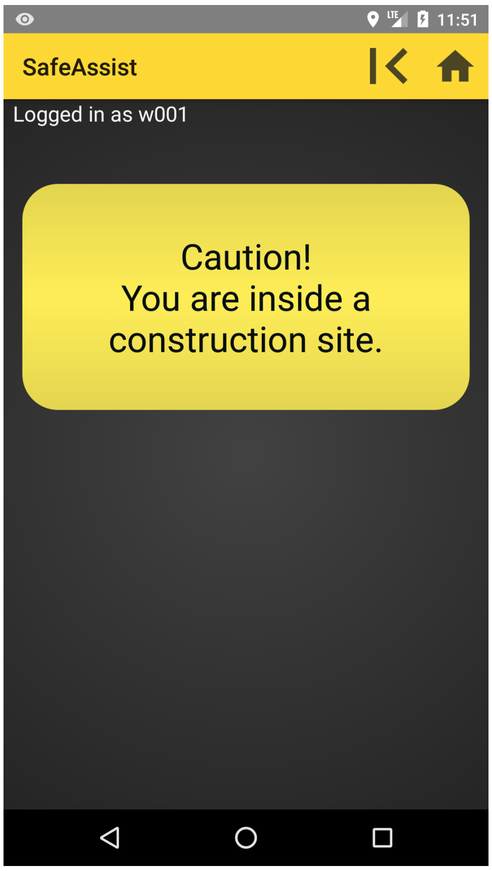


Logged in as w001

You are outside of
all construction sites.

Figure 11: GUI of being outside the working area

Figure 11 shows the display when the user is outside a working area. The pop up is shown in the case when a user exits the working area or hasn't even entered it.

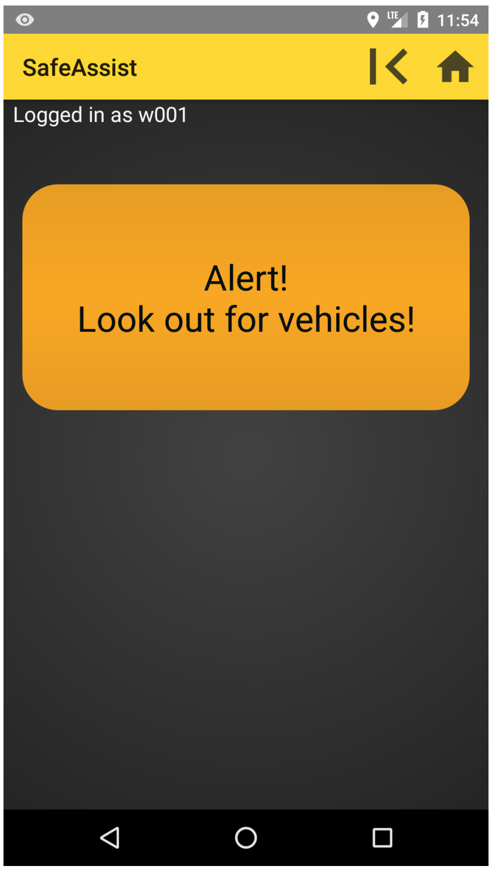


Logged in as w001

Caution!
You are inside a construction site.

Figure 12: GUI of entering the working area

Figure 12 displays the application interface when the user is inside a working area but is not close to a machine. Since the user is logged in the application, we have the possibility to get his exact location in the construction area. The construction area is set beforehand by the manager and its longitude and latitude is saved in the database. We get the user (worker, machine operator or manager) location, and compare it to the construction area position. If the user is inside a construction are, we show him a pop up with the message "You are inside a construction site.".


11:54

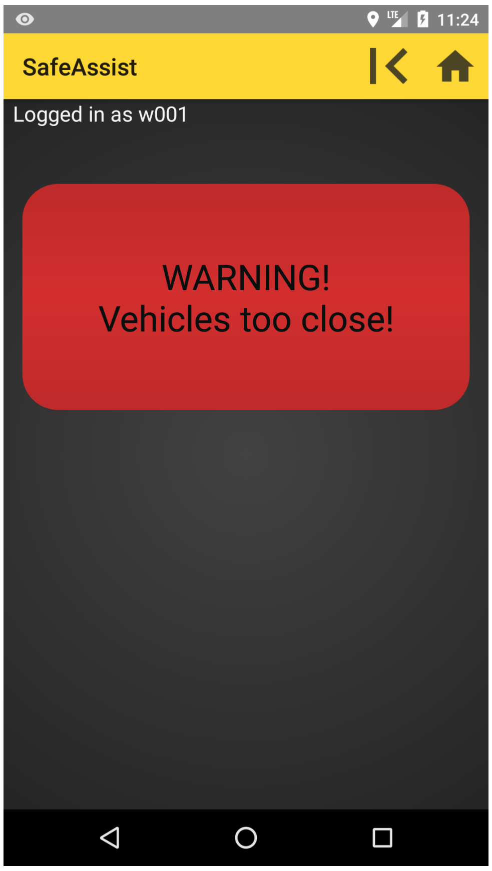
SafeAssist

Logged in as w001

Alert!
Look out for vehicles!

Figure 13: GUI of warning the worker/operator

Figure 13 displays the application interface in case the user is inside a construction site and close to a machine but not critical situation to notify the user that she/he has to be very careful. The pop up which is shown in this case is: "Alert! Look out for vehicles!".

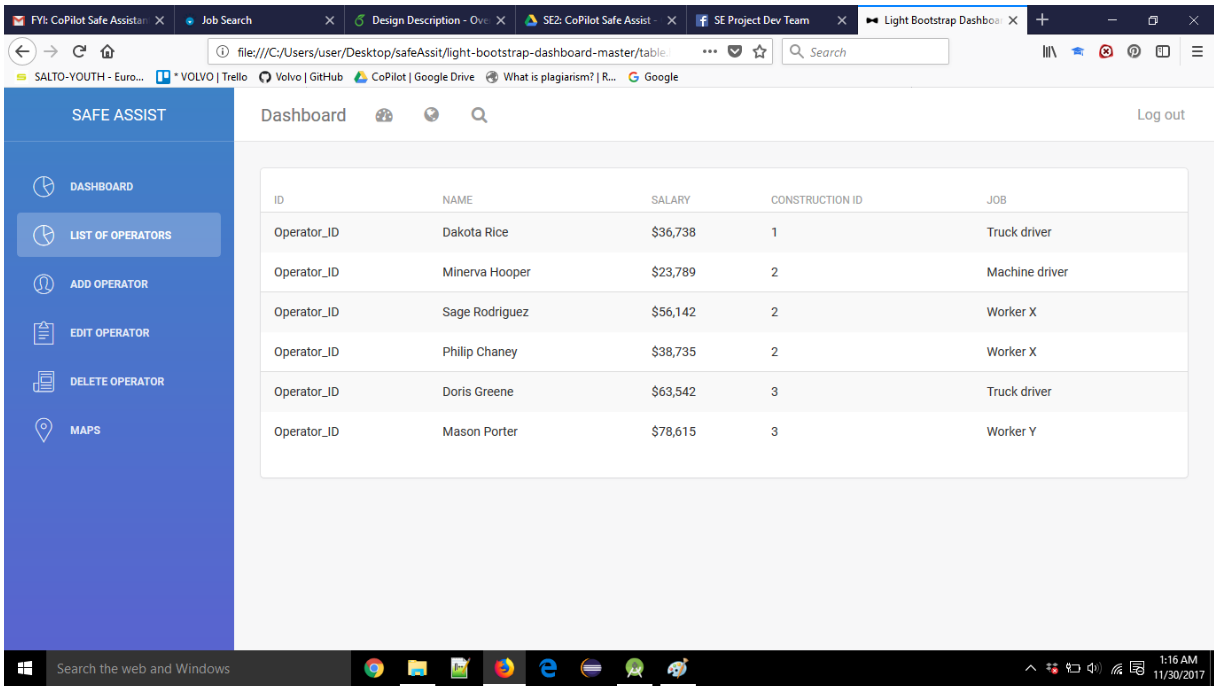


Logged in as w001

WARNING!
Vehicles too close!

Figure 14: GUI of WARNING of being close to a machine

Figure 14 displays the application interface in case of close proximity to a machine. If the user is inside a construction are, we also compare his distance with the other machine operators on the working area. In the case that machine and laborer are too close to each other and might be a risk, we show to both them a pop up with the message "WARNING! Vehicles too close!".



ID	NAME	SALARY	CONSTRUCTION ID	JOB
Operator_ID	Dakota Rice	$36,738	1	Truck driver
Operator_ID	Minerva Hooper	$23,789	2	Machine driver
Operator_ID	Sage Rodriguez	$56,142	2	Worker X
Operator_ID	Philip Chaney	$38,735	2	Worker X
Operator_ID	Doris Greene	$63,542	3	Truck driver
Operator_ID	Mason Porter	$78,615	3	Worker Y

Figure 15: List of all the workers - Administrator interface

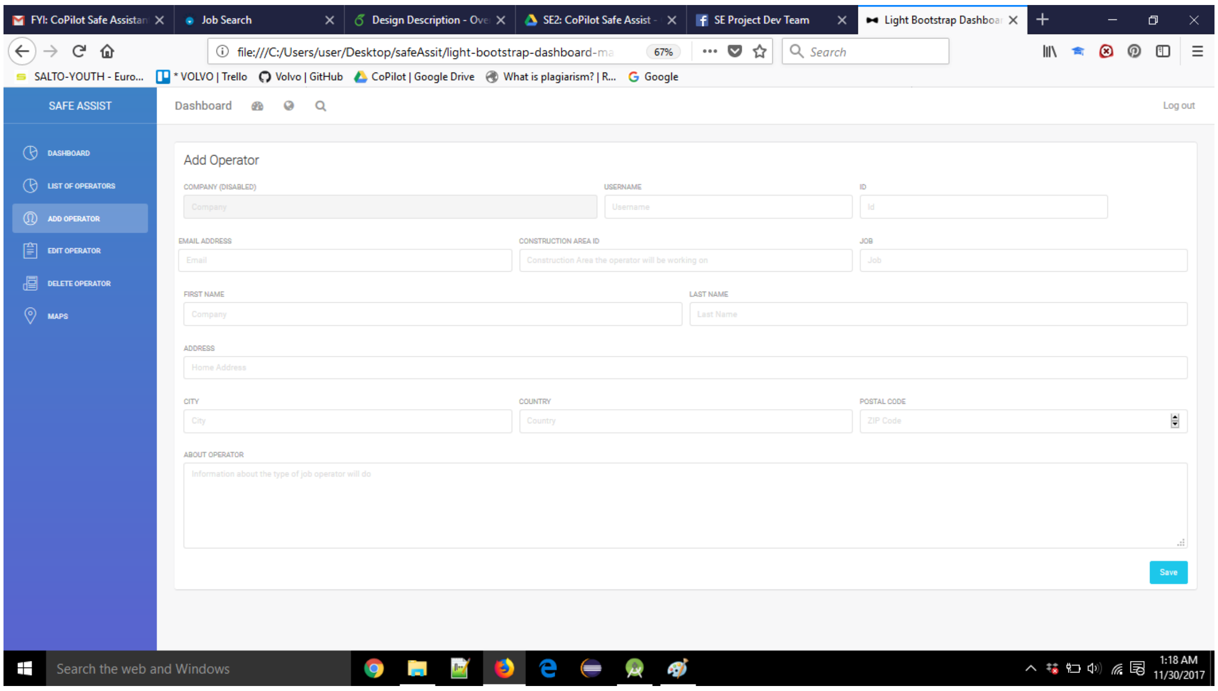


Figure 16: GUI of adding workers - Administrator interface

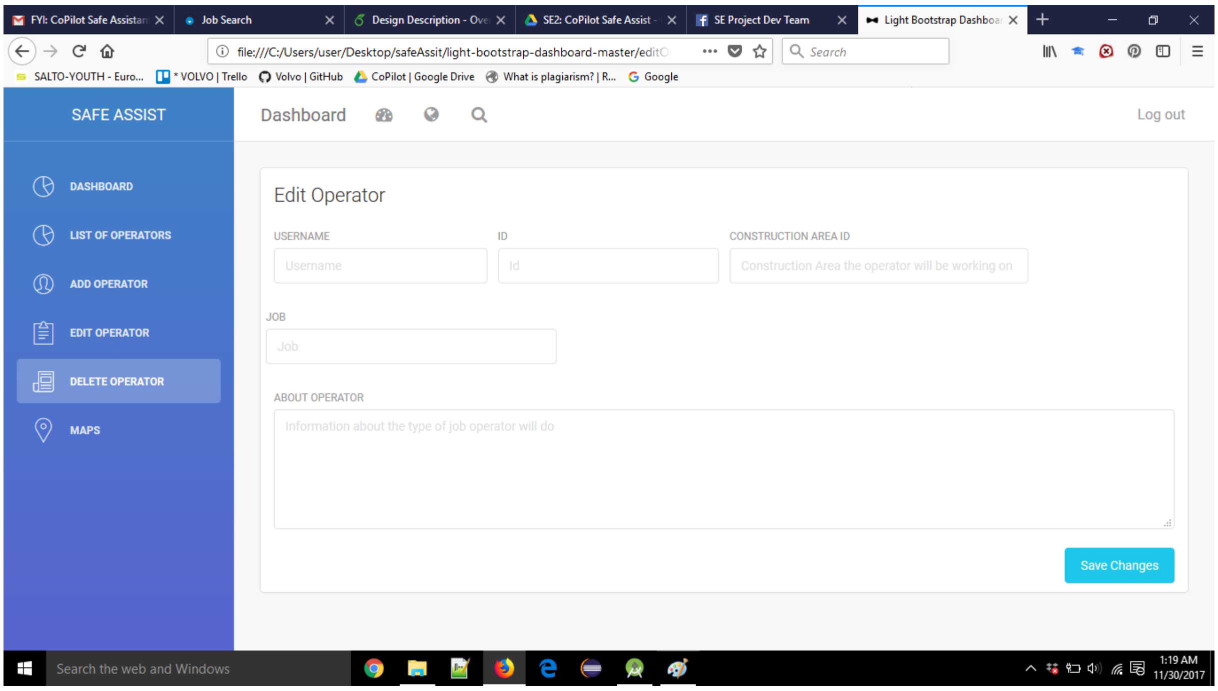


Figure 17: GUI of editing the workers data - Administrator interface

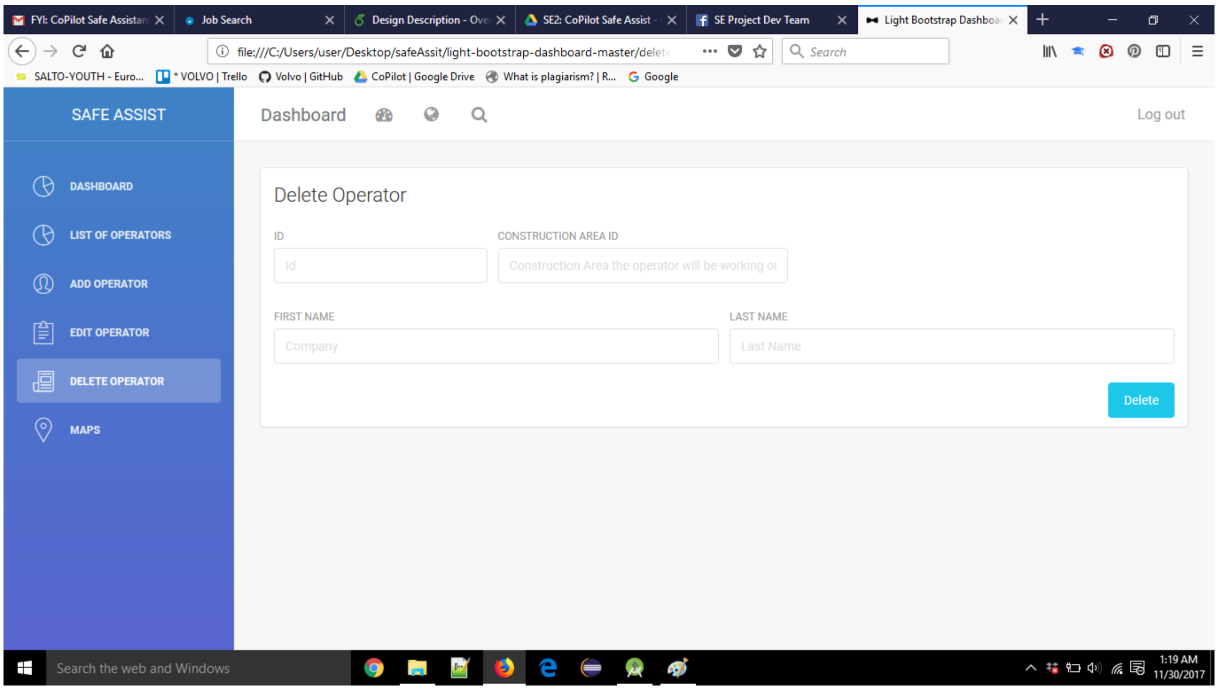


Figure 18: GUI of deleting workers from the working area - Administrator interface

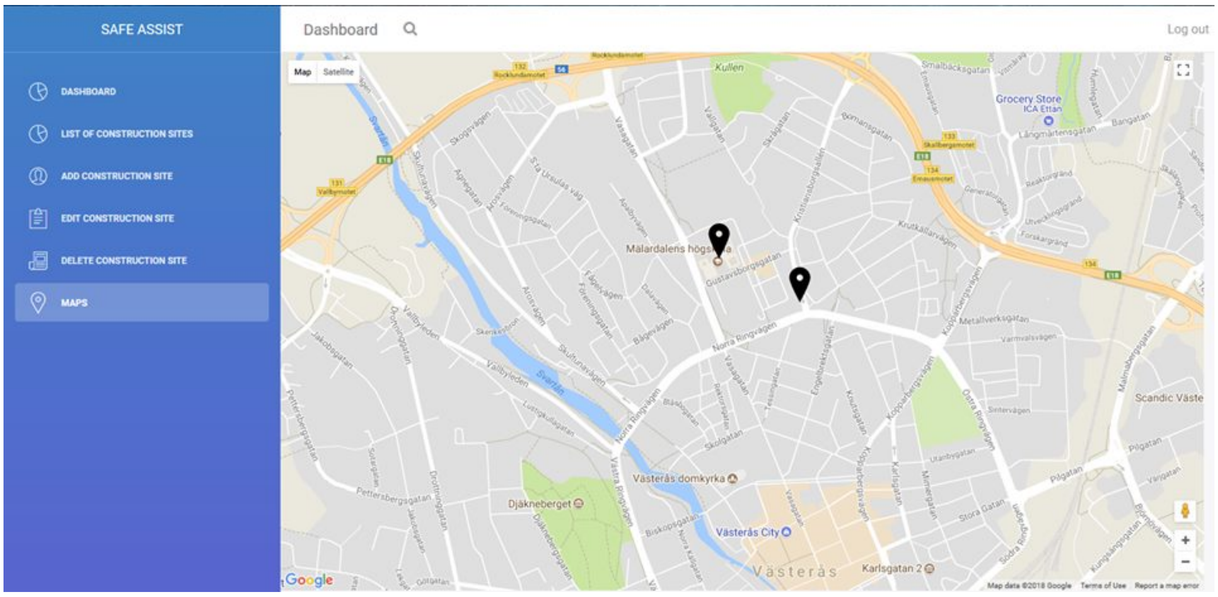
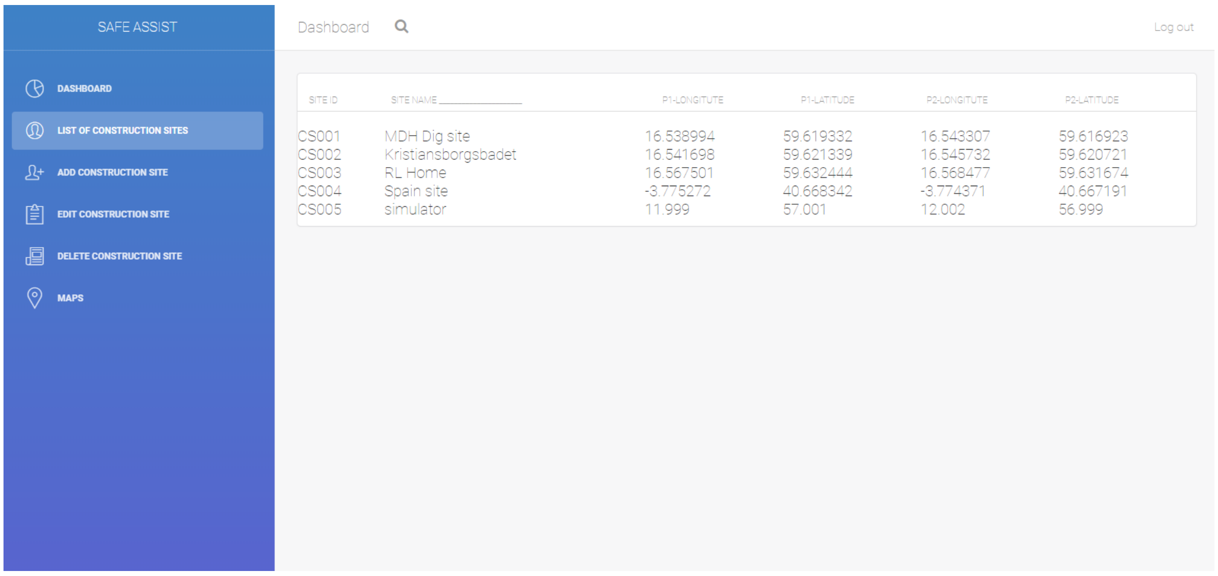


Figure 19: GUI of a map with the location of existing construction sites - Administrator interface



SITE ID	SITE NAME	P1-LONGITUTE	P1-LATITUDE	P2-LONGITUTE	P2-LATITUDE
CS001	MDH Dig site	16.538994	59.619332	16.543307	59.616923
CS002	Kristiansborgsbadet	16.541698	59.621339	16.545732	59.620721
CS003	RL Home	16.567501	59.632444	16.568477	59.631674
CS004	Spain site	-3.775272	40.668342	-3.774371	40.667191
CS005	simulator	11.999	57.001	12.002	56.999

Figure 20: GUI of a list of existing construction sites - Administrator interface