



VAASAN AMMATTIKORKEAKOULU  
UNIVERSITY OF APPLIED SCIENCES

Lakachew Tadesse, Lakew

# WEB AND MOBILE APPLICATION FOR A SERVICE COMPANY

Information and Technology  
2016

## ACKNOWLEDGEMENTS

Firstly, I would like to express my sincere gratitude to my thesis supervisor Timo Kankaanpää, for his continuous support, motivation, and in lighting my path. His guidance helped me throughout the project and added to my software projects knowledge. I would also like to thank Päivi Auranen for her support during my paper work.

My sincere thanks also goes to Dr. Ghodrat Moghadampour, for having me acquired all the necessary base knowledge in regards to writing and implementing a software project.

Last but not least, I would like to thank my fellow class mate and roommate Yonas Azmera for his sincere support, and assistance during my study and thesis work. His support has made my thesis work fun, short, and entertaining.

## ABSTRACT

Author	Lakachew Lakew
Title	Web and Mobile application for a service company
Year	2016
Languages	English
Page	72
Supervisor Name	Timo Kankaanpää

---

As companies grow managing the work becomes harder and harder, thus the advantages of technology come into hand for increasing communication and facilitating work.

This thesis documents the design and development of a web and mobile application for a service company. The web and mobile applications are built using Laravel and Android Studio respectively. These two applications are designed to communicate with one another so that similar data can be used across these two platforms.

This thesis was developed in such a way that the work data are stored in a centralized location (Cloud Server), so that all privileged personnel can access such data to update and view the work progress using a web browser and a mobile application. The purpose of the web application is to register, assign, display, collect and send data from and to the mobile application. The mobile application displays assigned works, sends and receives data to and from the web application.

The web application is built using PHPStorm IDE with Laravel Framework and the Android application is built using Android\_Studio IDE on Android Os. The communication between the two applications is handled by using a JSON Web Token (Restful API). These used technologies along with the implementation are documented in this thesis.

# CONTENTS

1	INTRODUCTION .....	7
1.1	Development Constraints .....	7
2	SOFTWARE ANALYSIS .....	8
2.1	Requirements specification .....	8
2.2	Functional specification .....	10
2.2.1	Use Cases .....	10
2.2.2	Sequence Diagrams .....	12
2.3	Mock-ups .....	32
2.3.1	Web Application Mockups .....	32
2.3.2	Mobile Application Mock-ups .....	38
2.4	Data Requirement .....	42
2.5	Non-Functional requirements .....	44
3	APPLICATION DESIGN .....	45
3.1	Three-tier organization.....	47
3.2	Framework Applications.....	47
3.3	Architectural Design .....	47
4	USED TECHNOLOGIES .....	50
4.1	Web app .....	50
4.1.1	Laravel Framework .....	50
4.1.2	Composer .....	51
4.1.3	Vagrant.....	51
4.1.4	Laravel Homestead.....	51
4.1.5	Virtual Box.....	52
4.1.6	Alexpechkarev/google-maps.....	53
4.1.7	Cornford/googlemapper .....	53
4.1.8	Tymondesigns/jwt-auth.....	53
4.1.9	Postman.....	54
4.2	Mobile app .....	55
4.2.1	Android Studio .....	55
4.2.2	Java JWT (JJWT).....	56

4.2.3	Material Design.....	56
5	IMPLEMENTATION .....	58
5.1	Building the Web application .....	58
5.1.1	Alexpechkarev/google-maps.....	58
5.1.2	Cornford/googlemapper .....	59
5.1.3	Tymondesigns/jwt-auth.....	60
5.2	Building the Mobile application .....	62
5.2.1	Java JSON Web Token (JWT) .....	62
5.2.2	Material Design.....	63
6	TESTING AND ANALYSIS .....	65
6.1	Testing.....	65
6.2	Analysis.....	67
7	LEARNING OUTCOME.....	68
7.1	Future development .....	68
8	SUMMARY .....	70
9	REFERENCES .....	71

## LIST OF FIGURES

<b>Figure 1- Mobile Application Use Case diagram</b> _____	11
<b>Figure 2 - Web Application Use Case Diagram</b> _____	12
<b>Figure 3 - Sequence Diagram for Adding Employee</b> _____	13
<b>Figure 4 - Sequence Diagram for Adding Customer</b> _____	15
<b>Figure 5 - Sequence Diagram for Adding Work</b> _____	17
<b>Figure 6 - Sequence Diagram for Assigning Work to employee</b> _____	19
<b>Figure 7 - Sequence Diagram for viewing all customers</b> _____	21
<b>Figure 8 - Sequence Diagram for viewing Employees as required</b> _____	23
<b>Figure 9 - Sequence Diagram for viewing Works as required</b> _____	25
<b>Figure 10 - LoginActivity Sequence diagram</b> _____	27
<b>Figure 11 – UserWorkActivity sequence diagram (starting activity, refreshing lists, selecting work)</b> _____	29
<b>Figure 12 – WorkActivity (sequence diagram for: checking, starting, and stopping Work)</b> _____	31
<b>Figure 13 – Welcome page mock-up for computer browser</b> _____	33
<b>Figure 14 - Welcome page mock-up for mobile browser</b> _____	33
<b>Figure 15 – Web app Home page with computer browsers</b> _____	34
<b>Figure 16 – Web app Home page with mobile browsers</b> _____	35
<b>Figure 17 – Selected Work Mock-up</b> _____	36
<b>Figure 18 – Add New Customer ‘Registration Form page’</b> _____	37
<b>Figure 19 - Assign Work to Employee Page Mock-up 1</b> _____	37
<b>Figure 20 – Assigning work to employee page Mock-up 2</b> _____	38
<b>Figure 21 - Login Activity</b> _____	39
<b>Figure 22 – Assigned user work lists Activity</b> _____	40
<b>Figure 23 – selected user work Activity</b> _____	41
<b>Figure 24 - Database Architecture V2</b> _____	43
<b>Figure 25 – Deployment diagram, which shows the hardware and software connections for the hole application</b> _____	46
<b>Figure 26 – Three-tier organization</b> _____	47
<b>Figure 27 – Web application Architectural Design</b> _____	48
<b>Figure 28 – Mobile Application Architectural Design</b> _____	49
<b>Figure 29 – Material design principles</b> _____	57

## **LIST OF ABBREVIATIONS**

IDE	Integrated Development Environment
SDK	Software Development Kit
VM	virtual machine
PHP	PHP: Hypertext Pre-processor
API	Application Programming Interface
JWT	JSON Web Token
URL	Uniform Resource Locator
JJWT	Java JSON Web Token
JSON	JavaScript Object Notation
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
WN	Work Number
MVC	Model, View, and Control

# **1 INTRODUCTION**

This Web and Mobile Application for a Service Company thesis documents the development of mobile and web application for a company called Jiko Ltd. Jiko is a cleaning company providing services in Vaasa and Seinäjoki, Finland. The company was established in 2002 with one employee but now the company has more than 80 employees.

The company Jiko Ltd is seeking a way to get the employees' work status immediately rather than waiting for the employees to submit their working hours, which is done in the middle or end of every month.

This thesis addresses the obstacles by using Web and Mobile applications. The web application can be accessed by the office workers (e.g. manager, accountant, developer) to add individual data (customer, work and employee) and assign work to the employees. The mobile application is used by the field workers to get their tasks and inform their working hours.

I am a Vaasa University of Applied Sciences student, part time working at UpCode Ltd and Jiko Ltd. Due to the frequent communication with Jiko Ltd manager/owner on building an application for solving the mentioned work obstacles, this thesis was developed.

## **1.1 Development Constraints**

Constraints faced while developing the application are as follows:

- Only one developer.
- Implementation of the latest tools.
- Necessity of real time test.



## 2 SOFTWARE ANALYSIS

Nowadays companies are capable of having a web application that can only be managed/controlled by the company personnel. A good example of such a web application is GitHub, a group of people located in different continents, who can use the same repository to work on without being hindered by their location. Likewise, Jiko Ltd will also deploy a web application which can only be accessed by the company employees.

Jiko Ltd requires to have two applications, one running on a mobile phone and the other on the web. The mobile operating system chosen is Android and PHP for the Web application. In the next section, the requirements specification will be described.

### 2.1 Requirements specification

The project requirements have three levels of importance, where #1 indicates “must have”, then followed by importance number #2 indicating “should have” or what the customer requested, and finally importance number #3 “could have” or nice to include in the application but not necessary.

The project has been requested by Mr. Jyrki Kallislahti, the owner/manager of Jiko Oy. The specification requirements for both applications are mentioned in the following tables 1 – 2.

**Table 1:** Android application Requirements specification

<b>Ref.</b>	<b>Claim and description</b>	<b>Importance</b>
A1	Email as a user ID	1
A2	Login	1
A3	Button for registering work start time	1
A4	Button for registering work end time	1
A5	Display employee works	2
A6	Provide address	2

A7	Provide work description	2
A8	Send GPS coordinate	3
A9	Work list refreshing button	3
A10	Lost password recovery	3

**Table 2:** Web application requirements specification

<b>Ref.</b>	<b>Claim and description</b>	<b>Importance</b>
W1	Provide welcome page for public user	1
W2	Login	1
W3	Login is allowed for only privileged personnel	1
W4	Register new employees	1
W5	Register Work	1
W6	Assign work for an Employee	1
W7	Register Customer	2
W8	Work address is provided in both Map and text description.	2
W9	View work data for a range of days.	2
W10	View works as a list	2
W11	View finished and unfinished works	3
W12	View employee's data in a periodic range Period-1: first half of the month (1 <sup>st</sup> - 15 <sup>th</sup> ) and Period-2: second half of the month (16 <sup>th</sup> - 28 <sup>th</sup> , 30 <sup>th</sup> or 31 <sup>st</sup> )	3
W13	View employee contact information.	3
W14	View Assigned and not assigned employees	3
W15	Customer is allowed to sign in and view, there request.	3

## **2.2 Functional specification**

### **2.2.1 Use Cases**

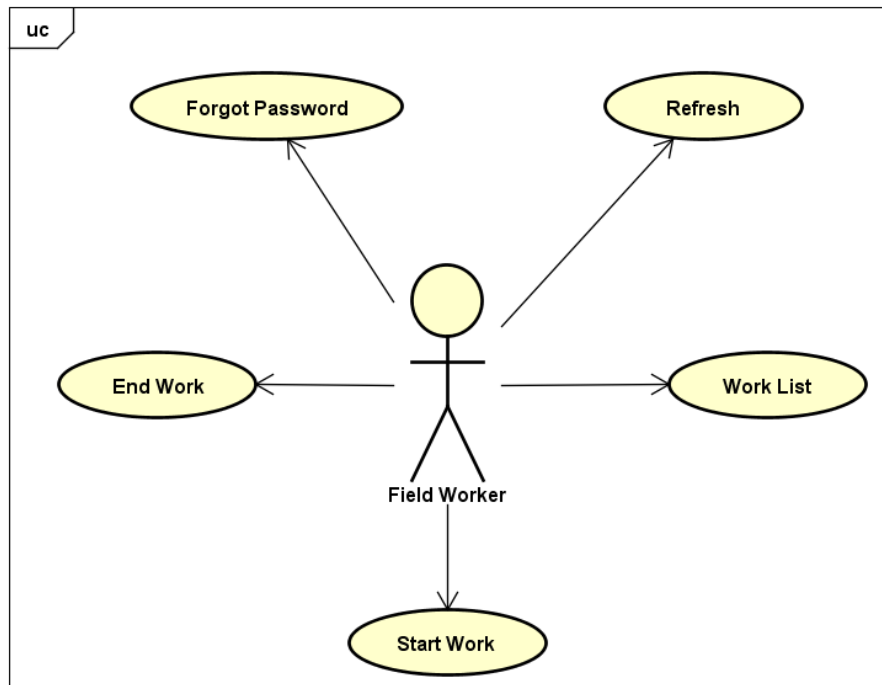
Since both applications provide different functionalities, each project has a different Use Case diagrams (figure representation of the user interaction with the application), which are shown in the following sub section:

#### **2.2.1.1 Mobile Use Case**

In this section, the mobile application Use Case diagram will be discussed. As can be seen in Figure 1, login is omitted since all use cases require authentication.

The 'Work list' Use Case, as the name implies, lists the assigned works for the employee. When the 'work list' page (Activity) is accessed the work lists are retrieved from the server, but if the employee wants to check for new work without leaving the work lists activity, the refresh button can be used to update the work lists. When the Activity is refreshed, the code behind the interface sends a request to the backend to collect all the assigned works for the employee.

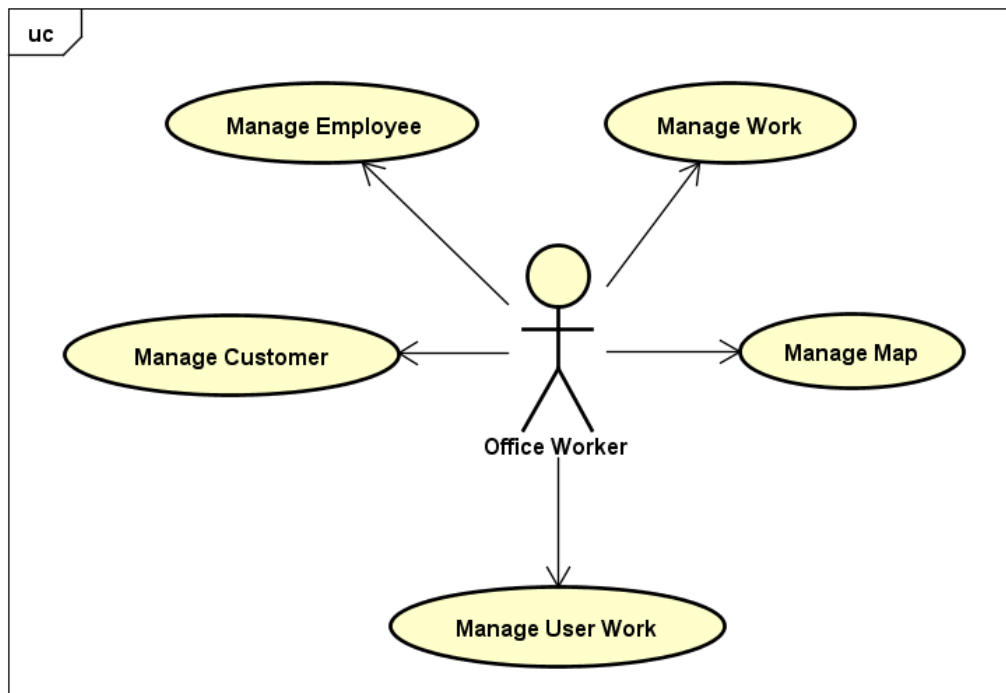
When the required work is selected among the list, Work Activity is generated for providing the 'work start', 'work end', and 'view description'. Each Use Case provide a service for the employee as the name implies, i.e. for starting/setting the start time, end time and/or view some description as needed.



**Figure 1-** Mobile Application Use Case diagram

#### 2.2.1.2 Web Use Case

In the web application the usage of the application is more complex compared to the mobile use since there is a lot more functionalities that are going to be handled by the web application. But for simplicity reasons, this Use Case diagram is provided with the general representation of functionalities, as shown in Figure 2. The detailed functionalities of the application will be described later on.



**Figure 2** - Web Application Use Case Diagram

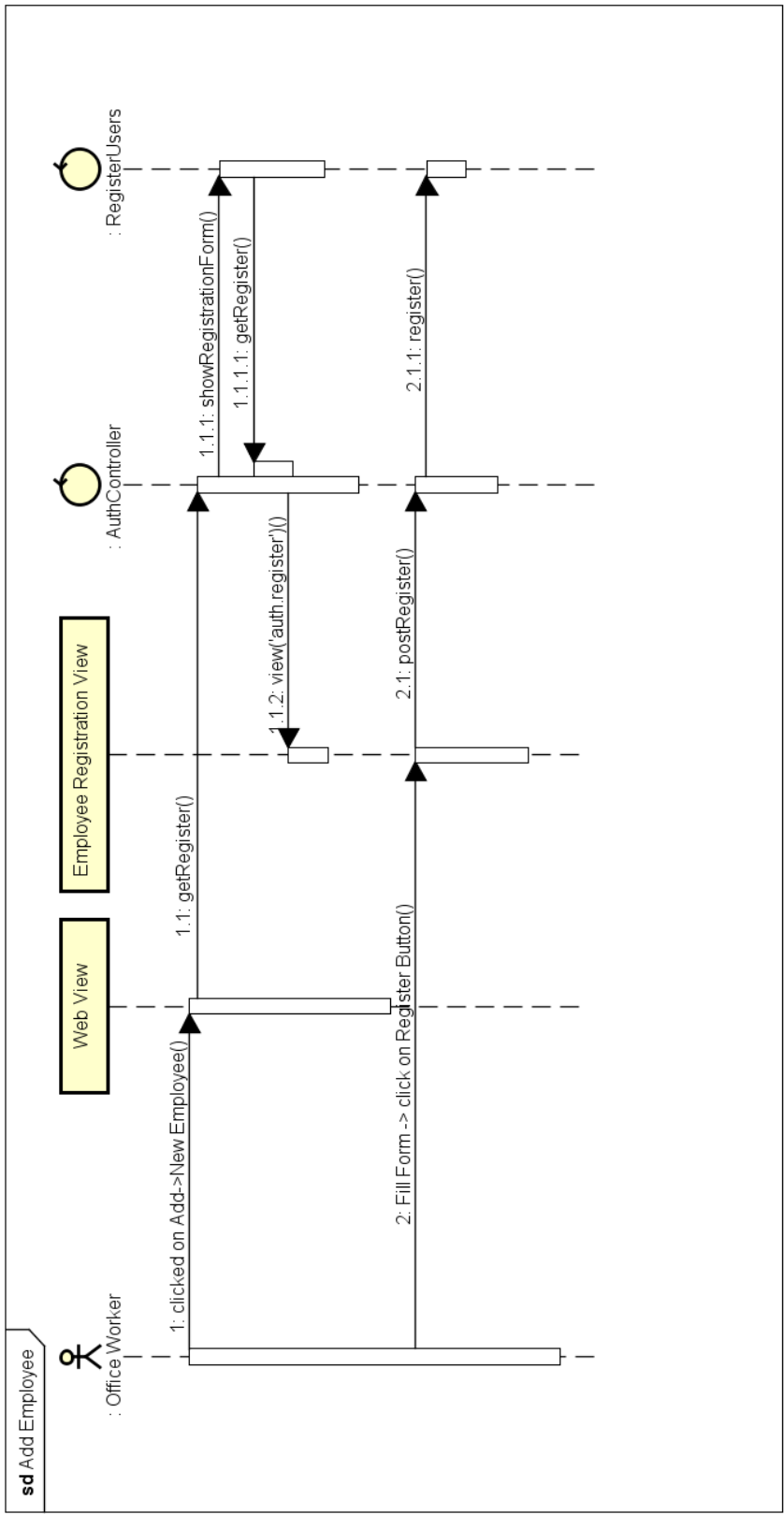
As can be seen from Figure 2, the Office Worker (user) can manage all the models i.e. add, view and assign different functionalities. This interaction with the application requires login in before accessing the functionalities, but the login and logout Use Cases have been removed for visual clarity.

### 2.2.2 Sequence Diagrams

Sequence diagrams are one of the best ways for visualizing/knowing how this project is working under the interface. It also helps us easily find the best way for managing the work flow and/or finding new ways for handling data.

#### 2.2.2.1 Web Sequence Diagrams

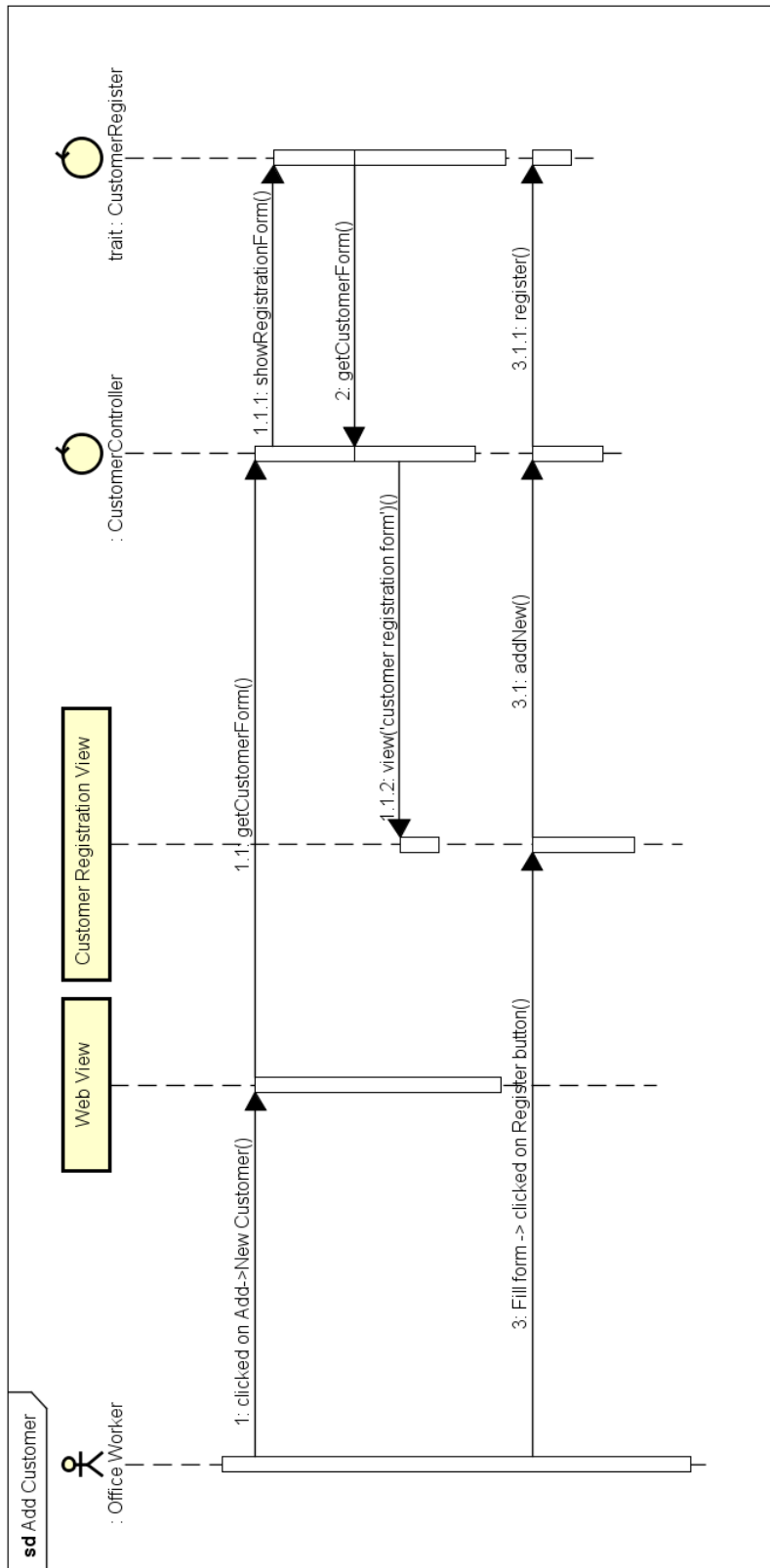
In this section the most usable Sequence Diagrams of the web application shown and each diagram is followed by a small explanation.



**Figure 3 - Sequence Diagram for Adding Employee**

As can be seen from Figure 3, for adding an employee, the Office Worker will first need to fill up a form. To display the form, the employee must first use the 'Add new Employee' from the Add tab, which is located in the top navigation bar. By selecting the 'Add New Employee' the web app route redirects to the getRegister method, which is located in the AuthController Class, after confirming if the user token is authenticated the AuthController class access the showRegistrationForm method from the RegisterUsers Class, which displays the View for the Employee Registration form.

After filling the Registration form the submit button posts the data to the postRegister method, which is found in the AuthController. After validating the form data, it is redirected to the register method in the RegisterUsers Class, which will store the data into the database.

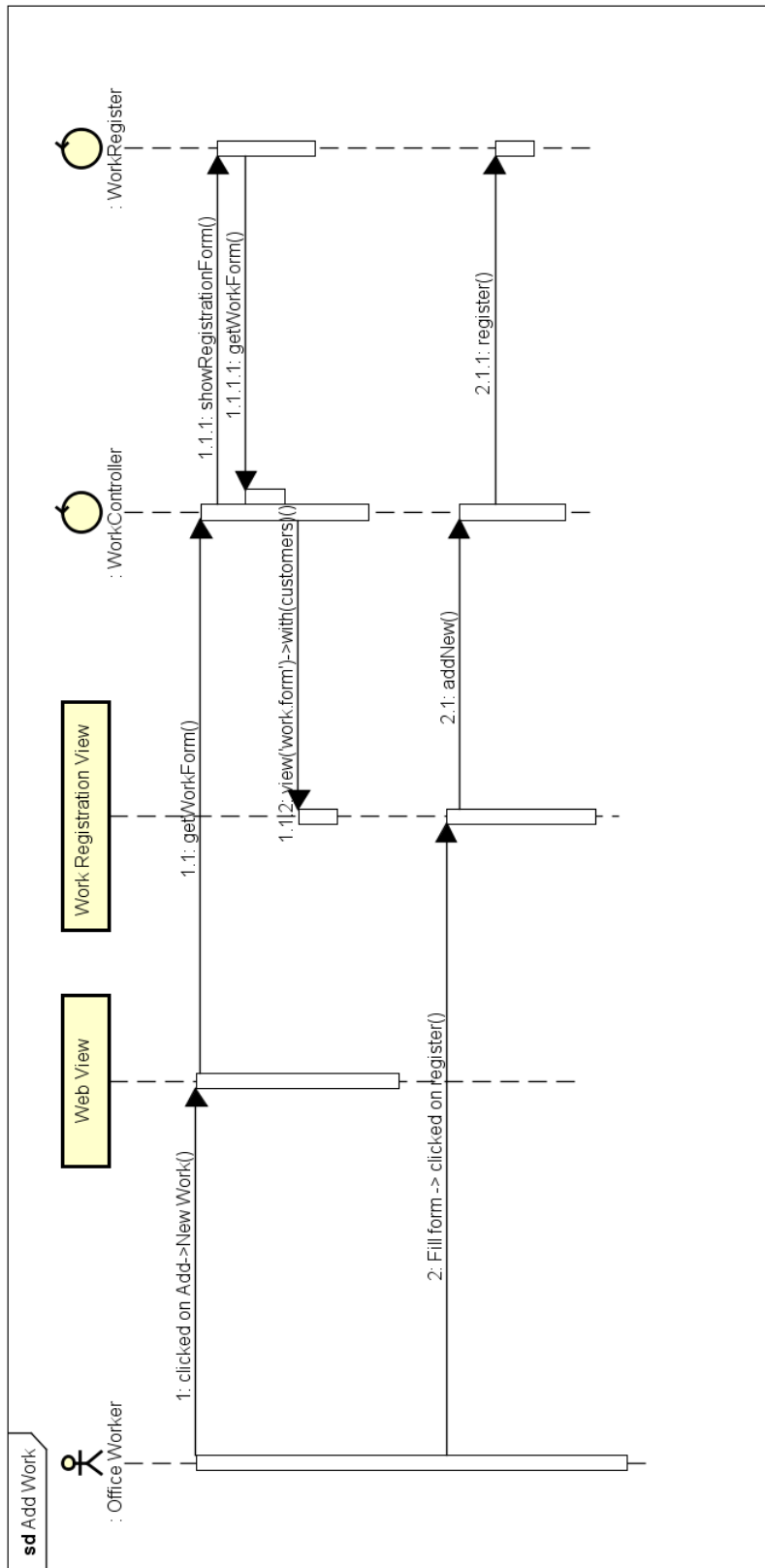


**Figure 4** - Sequence Diagram for Adding Customer



As can be seen in Figure 4, for adding a customer, the employee will first need to fill up the customer form. To display the form, the employee must first use the 'Add new Customer' from the 'Add' or 'Customer' Tab, which is located in the top navigation bar. By selecting the 'Add New Customer' the web app route redirects to the `getCustomerForm` method, which is located in the `CustomerController` Class, after confirming if the user token is authenticated the `CustomerController` class access the `showRegistrationForm` method from the `CustomerRegister` Class, which displays the View for the Customer Registration form.

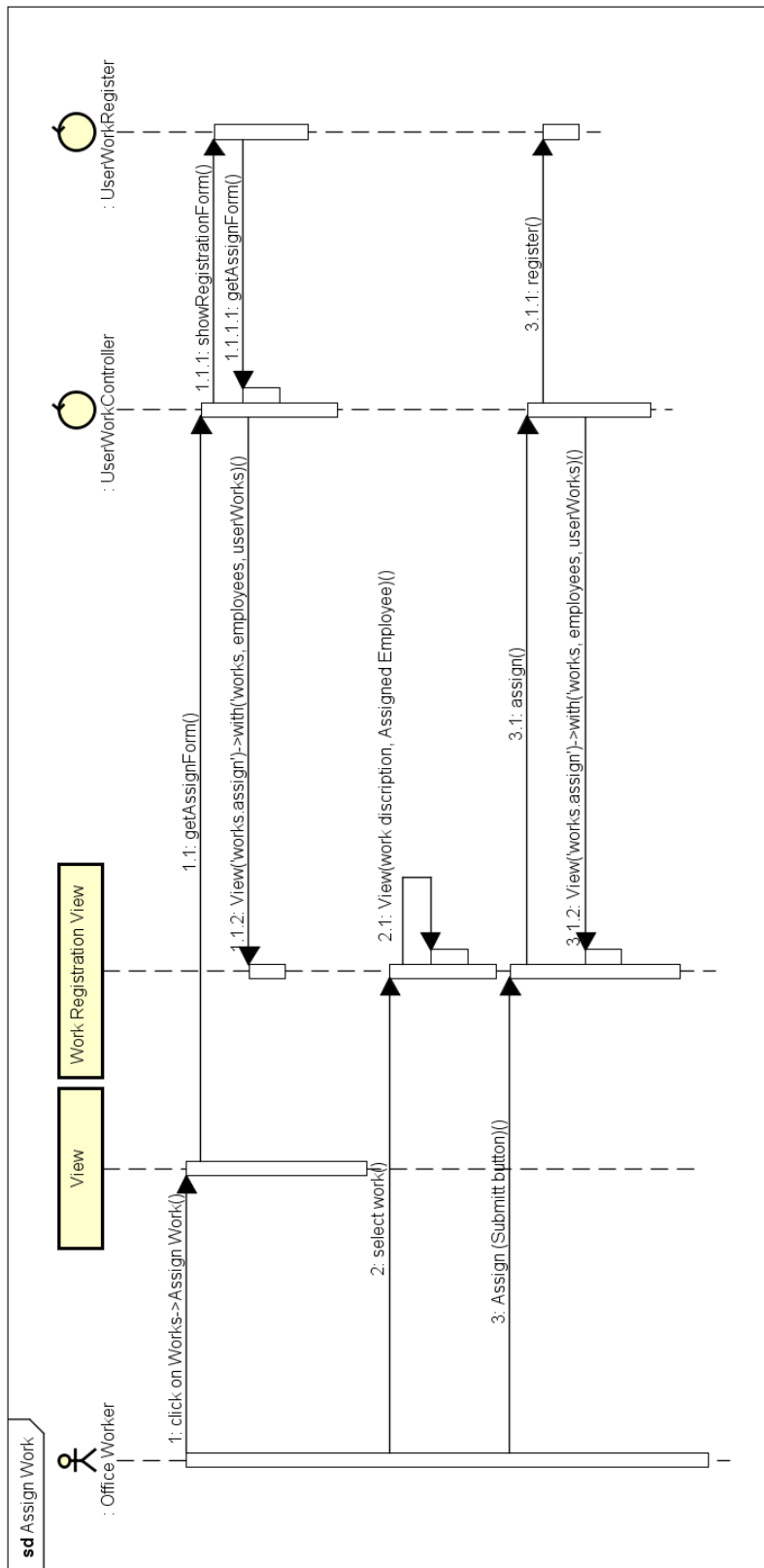
When the Customer form is submitted, the Data is sent to the `addNew` method, which is found in the `CustomerController`. After validating the form data, it is redirected to the `register` method in the `CustomerRegister` Class, which will store the data into the database.



**Figure 5** - Sequence Diagram for Adding Work

As can be seen in Figure 5, for adding Work, the Office Worker will first need to fill up the Work form. To display the form, the employee must first use the 'Add new Work' from the 'Add' or 'Work' Tab, which is located in the top navigation bar. By selecting the 'Add New Work' the web app route redirects to the getWork-Form method, which is located in the WorkController Class, after confirming if the user token is authenticated the WorkController class access the showRegistration-Form method from the WorkRegister Class, which displays the View for the Work Registration form.

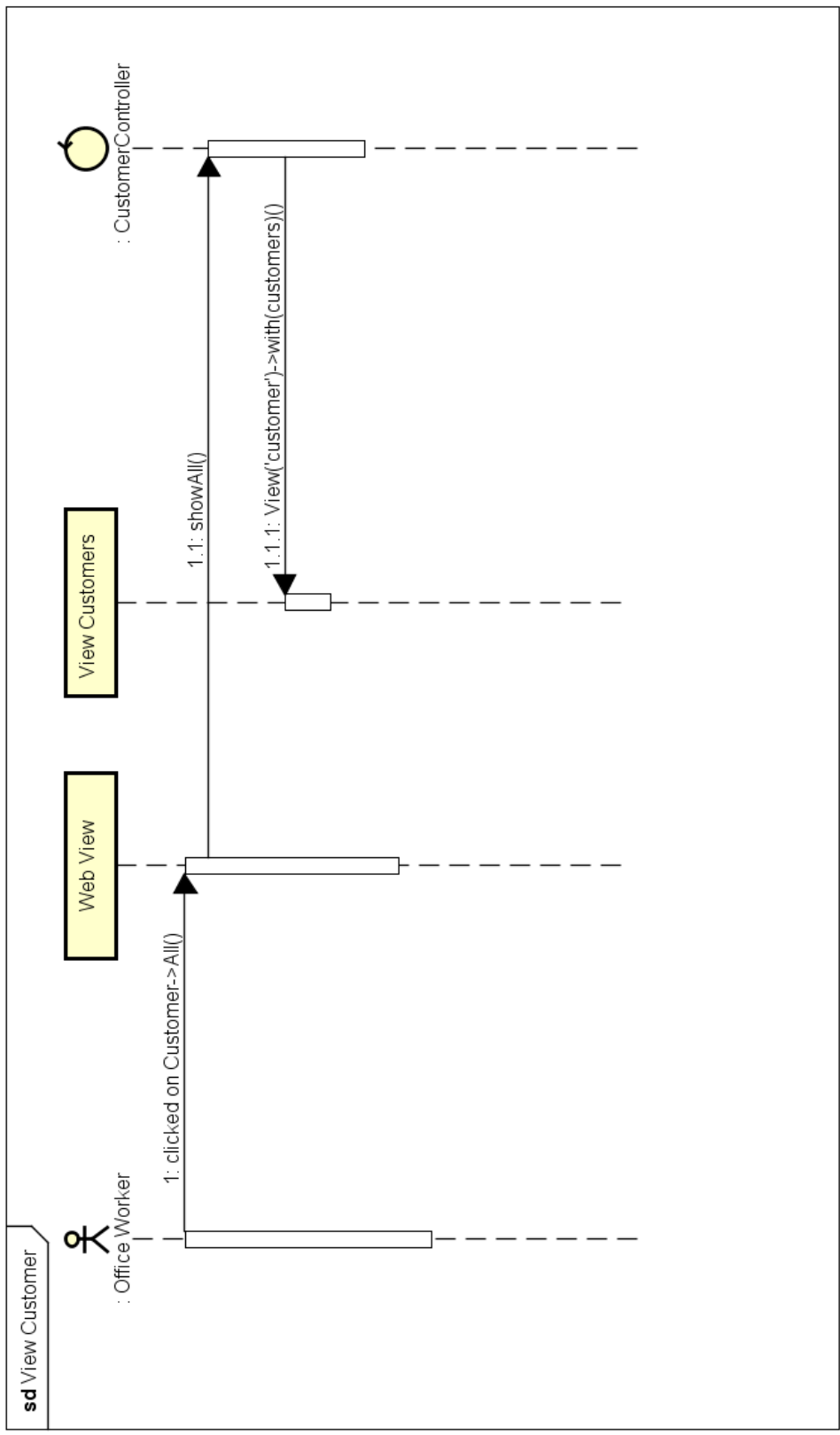
When the Work form is submitted, the data is sent to the addNew method, which is found in the WorkController. After validating the form data, it is redirected to the register method in the WorkRegister Class, which will store the data into the database.



**Figure 6** - Sequence Diagram for Assigning Work to employee

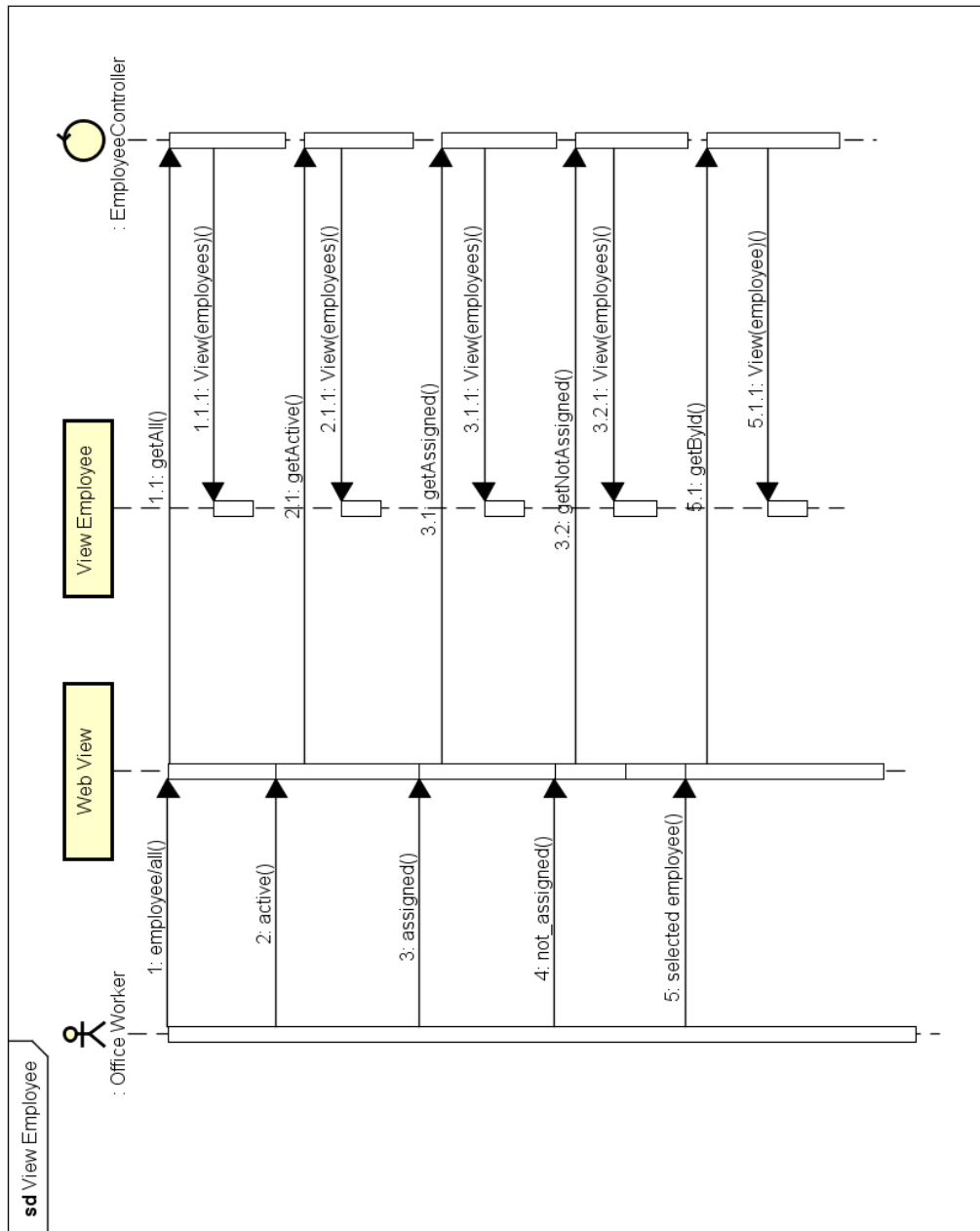
As can be seen from Figure 6, works can be assigned using the 'Assign Work' under the Works Tab. When the 'Assign Work' is clicked it redirects route to access the 'getAssignForm' method, which is located under the 'UserWorkController', this method checks the authenticity of the user and redirects to the 'showRegistration-Form' method, which is found in 'UserWorkRegister' method for displaying the 'Work Assign View'.

The 'Work Assign View' displays two drop down lists, one for the listing the works and the other for listing the employees. Choosing the work will display, the work information and employees list assigned for that work. This information helps the user to confirm if the right work is chosen and also to check the assigned employees. Knowing this information, the user can assign an employee. Assigning the employee will first need to pass through 'UserWorkController' for validation, and if the validation is passed it will redirect to the 'UserWorkRegister' for storing the assigned information into the database.



**Figure 7** - Sequence Diagram for viewing all customers

As can be seen in Figure 7, in order to display all customers, the user is required to select the 'All' under the Customer Tab. Clicking on 'All' redirects the route to 'showAll' method, which is found under the Customer Controller. This method returns all the customers data to the 'Customer View' page, which is displayed on Table 3.



**Figure 8** - Sequence Diagram for viewing Employees as required

Employees can be viewed in different forms; among the many ways the most common ones are shown in the Figure 8. The different forms are discussed in the following Table 3.

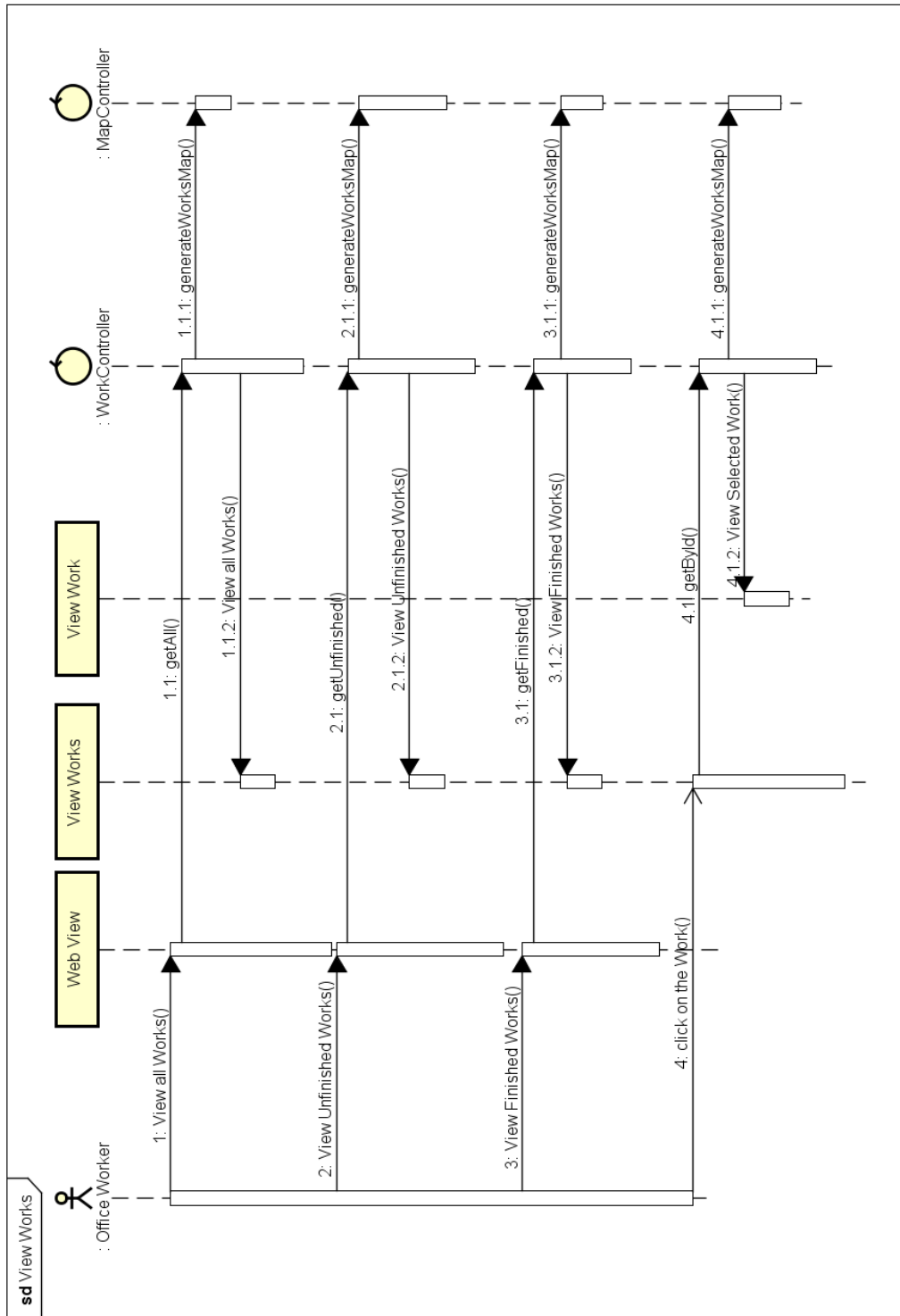
**Table 3:** Available Employee view types with description

No.	Function	Description
-----	----------	-------------



1	All	Displays all Employees
2	Active	Displays all Active Employees
3	Assigned	Displays only Assigned Employees. Assigned Employees are those employees which have a task to do and the task is not completed
4	Not Assigned	Displays only those Employees which are not assigned, i.e. employees who have completed their work and are currently not working.
5	Selected Employee	All the above mentioned functionalities provide the user with a list of employees. The lists display the employees full name and phone number only if the user wishes to display full information of the employee the user can click on each employee name to display full information

Since all the methodology described in Table 3 follow the same sequence only one of the methodology sequences will be discussed. When the user selects 'All' under the Employee Tab, the route is redirected to access the 'getAll' method, which is located in the 'EmployeeController'. This method collects all the Employee data from the database and forwards it to the 'View Employee' page, which displays the retrieved data in a table providing the employee name along with the phone number. If the user wishes to get full information of the employee he/she can click on the name for displaying full information of that employee.



**Figure 9** - Sequence Diagram for viewing Works as required

Works can be viewed in different forms; among the many ways the most common ones are shown in Figure 9. The functionality of different forms is described in following Table 4.

**Table 4** – Available work view types with description

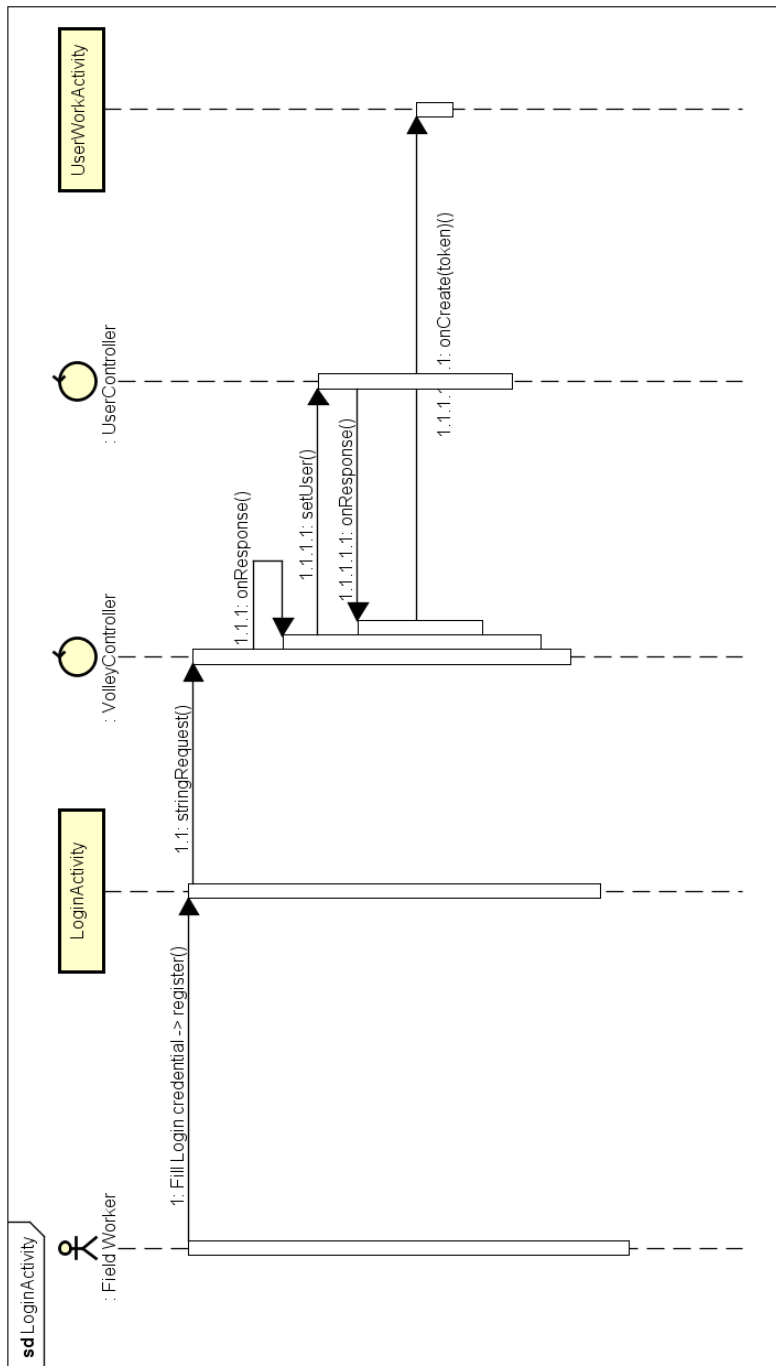
No.	Function	Description
1	View All Works	Displays all Works
2	Unfinished Works	Displays all unfinished Works
3	Finished Works	Displays only Finished Works
4	Selected Work	Displays the selected Work in details

Since all the display methodology follow the same sequence only one of the methodology sequences will be discussed. When the user selects 'Finished Works' under the Work Tab, the route is redirected to access the 'getFinished' method, which is located in the 'WorkController'. This method collects all the finished works from the database and access the 'MapController' for generating the map using the collected Works data. After generating the Map, the data is forwarded to the View Work page, which displays the retrieved data using a marked map and the work information using a table.

When displaying works as a list (i.e. All, Unfinished, and Finished Works) the application not only displays the data but also provides a means of updating the status and a link for displaying full information of the selected work. As can be seen in the above Figure 9 when a work is selected from the 'View Works' page, it is routed to a method called 'getById' method under the WorkController Class, which generates a Map and redirect the data to a 'View Work' page.

### 2.2.2.2 Mobile Sequence Diagrams

In this section, the most usable sequence diagrams for the mobile application are shown followed by a small description explaining the diagram.

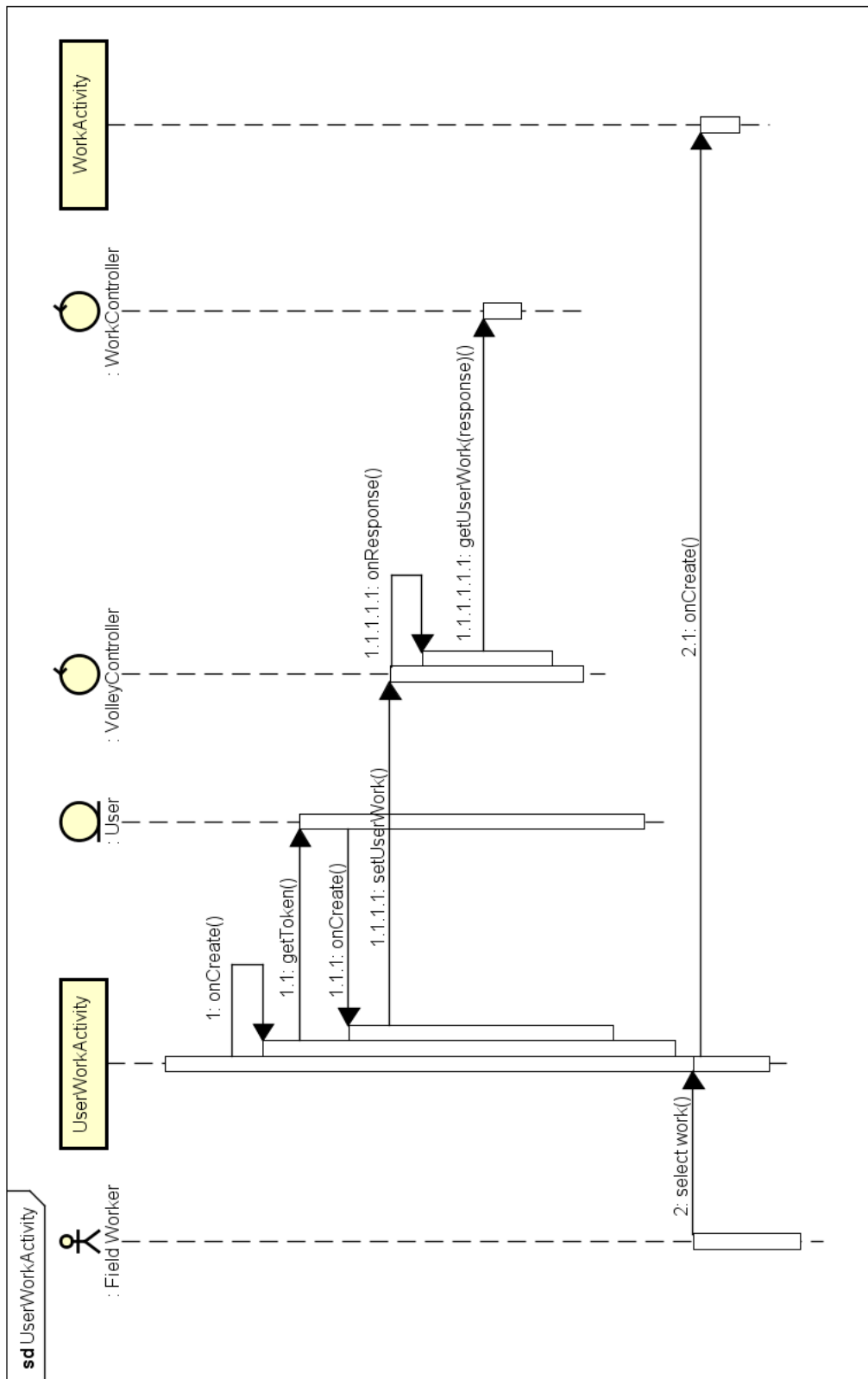


**Figure 10** - LoginActivity Sequence diagram

When opening up the mobile application the user is requested to login, using his/her email address and password, which are provided by the administrator. When the user submits this required information, the application uses a 'VolleyController' Class for connecting to the server and authenticating the user credentials. If the user is authenticated, a valid token will be generated from the server side and responded back to the application, at this time the Login Activity will be destroyed and UserWorkActivity will be generated. On the other hand, if the user is not authorized an error message pops up informing the user to retry.

When the server responds for authorized user, it sends user credential along with the generated token. The token is used for further interaction with the server as long as the user did not sign out or close the application. The application uses the responded credential to set the user information. This credentials are stored in a local database (SQLite) till the user signs out. The credentials are stored by using the 'setUser' method of the 'UserController' Class, as can be seen in Figure 10.

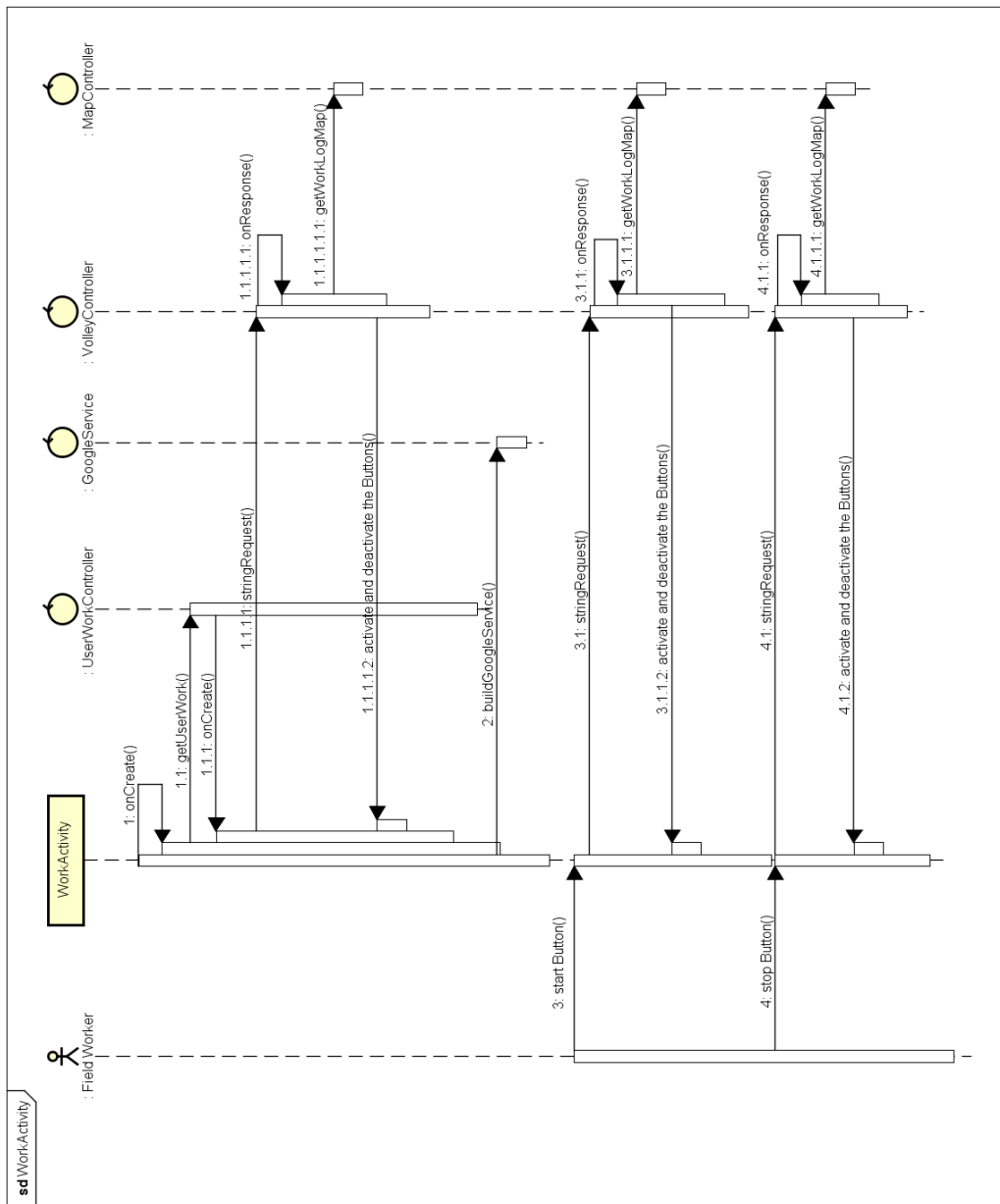
At this point the user has logged into the application and 'LoginActivity' will be terminated but meanwhile 'UserWorkActivity' will be generated. This 'UserWork-Activity' sequence diagram is shown in Figure 11.



**Figure 11** – UserWorkActivity sequence diagram (starting activity, refreshing lists, selecting work)

Generating the 'UserWorkActivity' page goes through multiple sequences before displaying the view content. As discussed in the previous 'LoginActivity' Sequence diagram, the user credentials have been stored in the 'User' Model (SQLite), this User object is used for providing the token using 'getToken' method of the User Class. This token is what the server side application uses to authenticate the user and also know who is accessing the server so that the server responds for that particular user. So, using the 'VolleyController' the mobile application will connect to the server and get the available works, which will be listed in the 'UserWorkActivity'.

The refresh button, which is found in the UserWorkActivity, regenerates the page i.e. generating 'UserWorkActivity' and refreshing the page goes through the same process. If the user is assigned to a work, there will be lists of work available for the user, which can now be selected to check out or start work.



**Figure 12** – WorkActivity (sequence diagram for: checking, starting, and stopping Work)

As can be seen in Figure 12, the ‘WorkActivity’ is the most complicated activity with multiple message exchanges with the server. When the ‘WorkActivity’ is generated it connects to the UserWorkController (in the background) for retrieving the user work information and status. This is accomplished by using the ‘getUserWork’ method.



Using the stored user credentials and selected work id (from the UserWorkActivity), the application connects to the server using the VolleyController for retrieving the Map information. This Map holds the information whether the work has previously been activated or not. If not, the page is created without the starting time (start button is active), otherwise the page will have additional information of the started time (start button is not active but stop button is active).

## **2.3 Mock-ups**

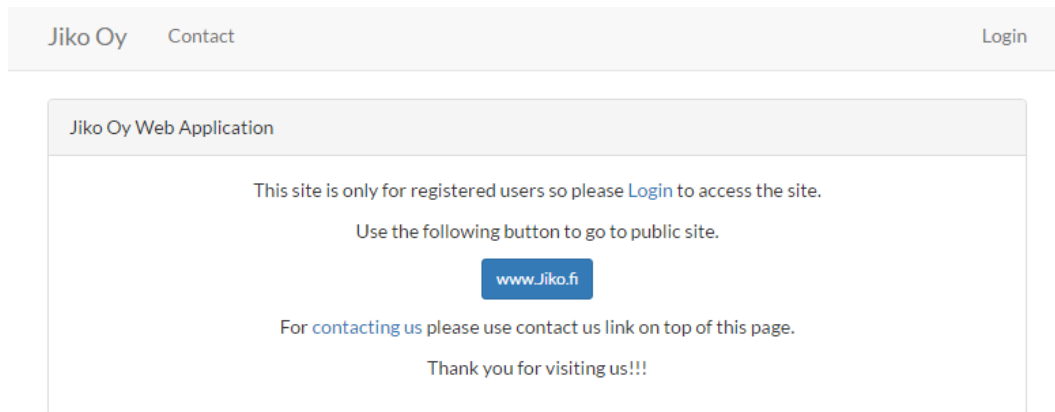
### **2.3.1 Web Application Mockups**

Jiko Web application is a private site used for the employees only and for such reason Jiko web application has only two public page. This public pages are the Welcome and the Contact page. The remaining pages are all for the Jiko Ltd registered employees only.

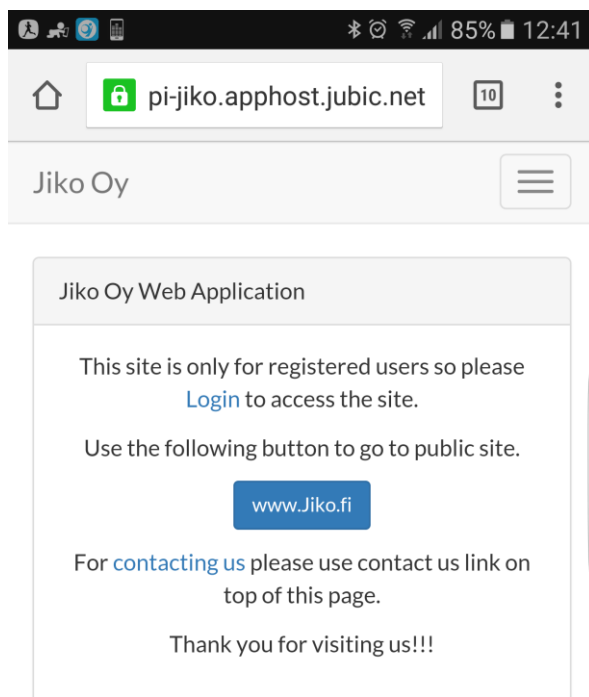
This web application is built to be mobile responsive, i.e. the web application looks elegant when accessed with a mobile application browser also. For showing these looks both the computer browser and mobile browser mock-ups are documented side by side.

In order to avoiding redundancy, one common Mock-up is used with a small description.

### 2.3.1.1 Welcome page Mock-up



**Figure 13** – Welcome page mock-up for computer browser



**Figure 14** - Welcome page mock-up for mobile browser

The above Figure 13 and 14 are mockups for the welcome page of the web application. This page is a public page used for redirecting to the Jiko Ltd public site ([www.jiko.fi](http://www.jiko.fi)), contact page, and a login page for the visitor. The contact page is also accessible for the public visitors in case they need the phone numbers or email addresses of the company personnel.

### 2.3.1.2 Home page Mock-up

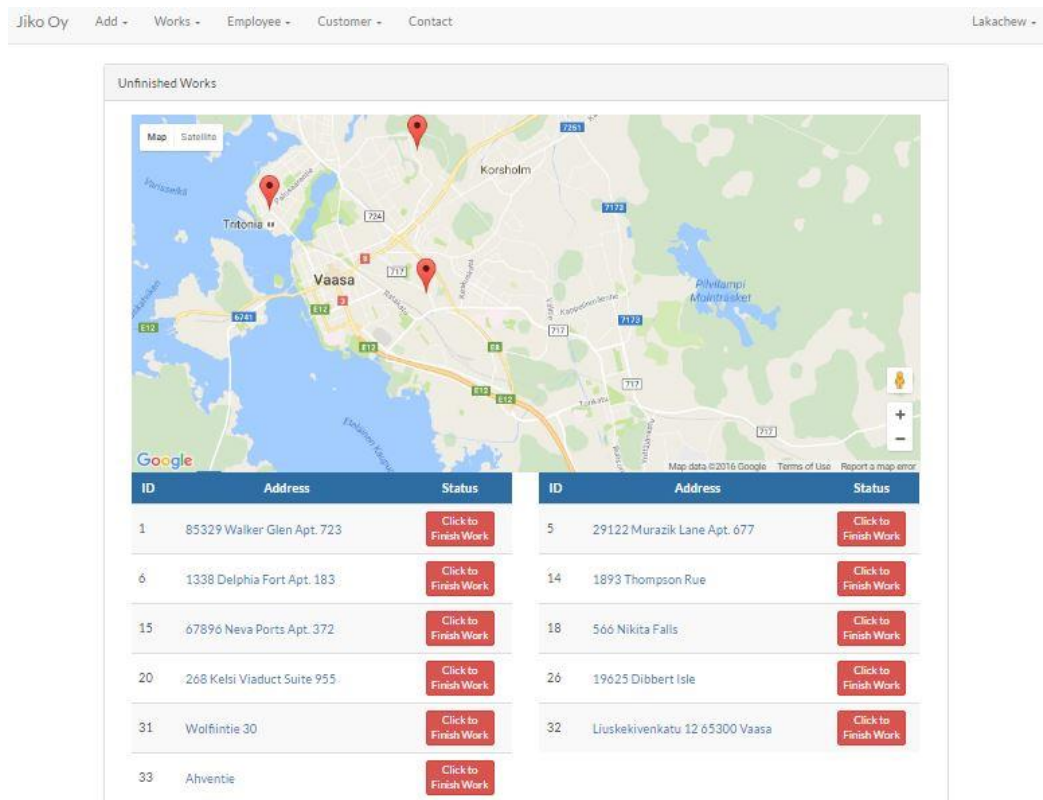
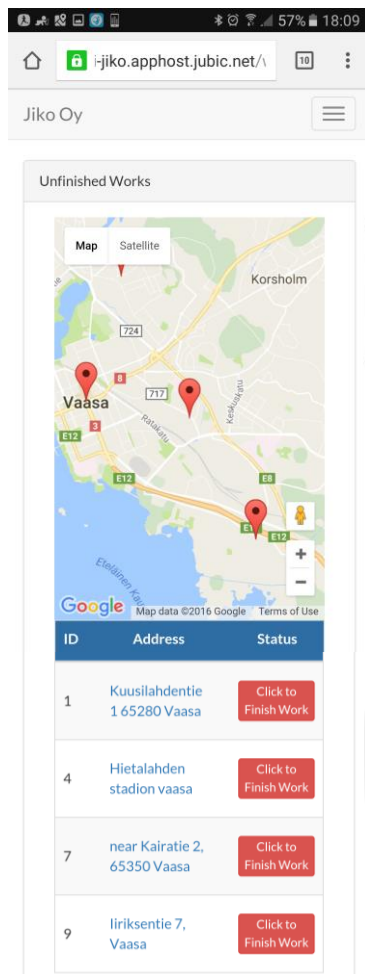


Figure 15 – Web app Home page with computer browsers



**Figure 16** – Web app Home page with mobile browsers

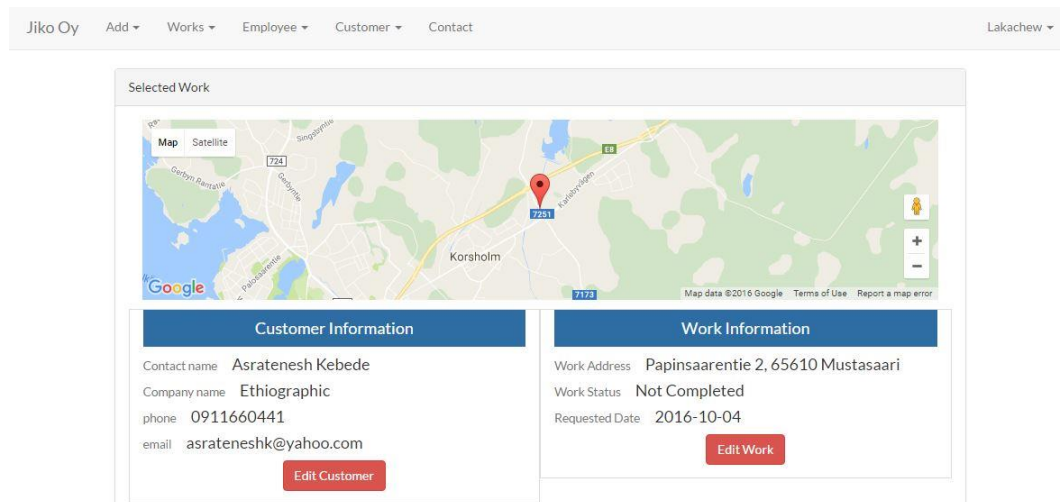
When the user logs in, the first page that appears is the unfinished works page, also used as the home page, since the office's common task is to check the unfinished works.

As can be seen in Figures 15 and 16 the page displays the Map followed by list of works. The list contains Work id, address, and status. Due to the available space with computer browser the lists are shown in two columns but the mobile browser displays only one. The Map provides each work location with a marker for two dimensional representations.

The list is not only for displaying the customer name address and status but also linked and buttoned to provide further-information and update-status respectively.

Customer and Address columns are equipped with a link to provide further information on the selected work (Figure 17). However, the status column contains a button for updating the status of the work to be completed.

### 2.3.1.3 Selected Work Mock-up



**Figure 17** – Selected Work Mock-up

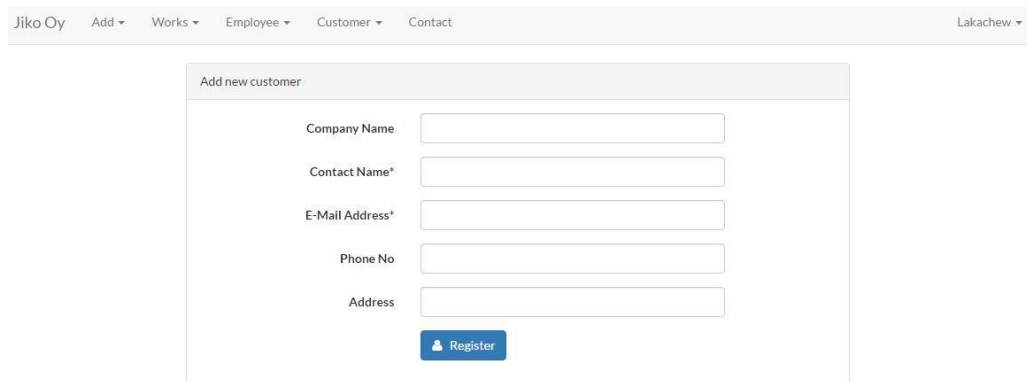
In the previous Mock-up (Figure 15 - 16), the list holds the Work address, but the addresses provide the user a link to check detailed information of the work. Clicking on this address link the user is redirected to the selected work information page, which is shown in Figure 17 above. This page provides information of the work and the customer. The map assists in finding the work location by providing only the selected work marked, which increases user comfort and visual assistance. For the coming future this page will also provide the user with the hours being counted for the selected work task.

### 2.3.1.4 Register page Mock-up

There are three different registration pages: the customer, work, and employee register. Since all use the same design and mechanism, it is enough to show only one registration Mock-up and represent the other registration pages (Figure 18).

As can be seen in Figure 18, the registration page provides the user with a user-friendly form. Not all of the fields are required to be filled and those, fields that are required to be filled are indicated with a '\*' sign. The registration pages also provide

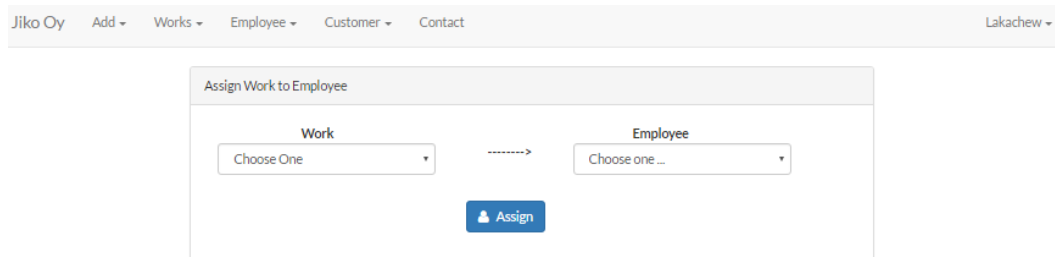
security by restricting the user from entering more than the required characters, and validating the data entered on the server side before storing the data.



**Figure 18** – Add New Customer ‘Registration Form page’

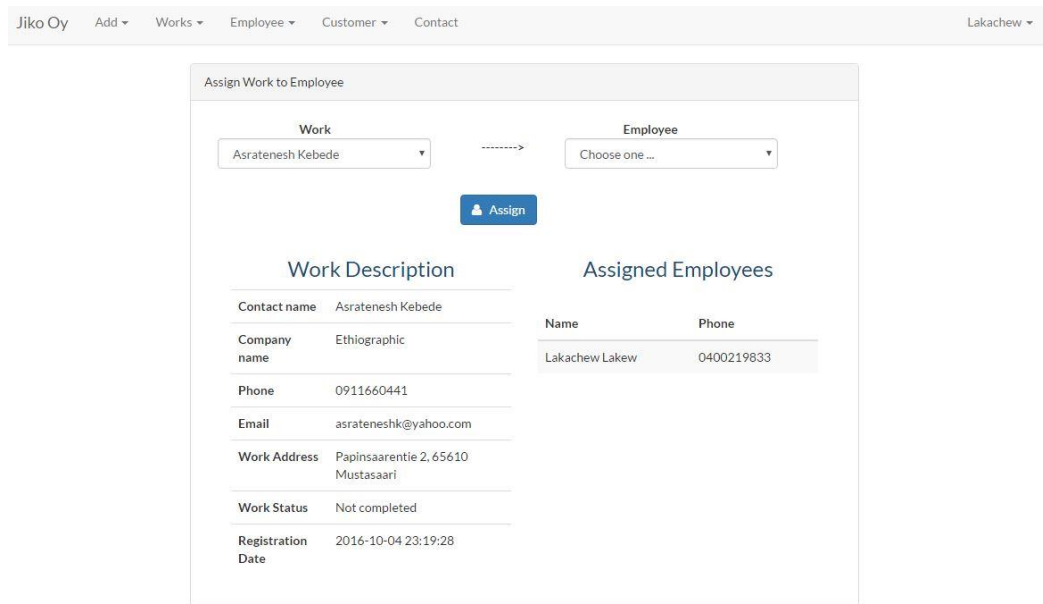
### 2.3.1.5 Assign Work to Employee Page Mock-up

Assigning work to employees is one of the most vital parts of this web application. Every field worker is using such assigned task to perform his/her duties and also the recorded data are stored and accessed using assigned identification number. When first opening up this page, there are two drops down lists hanging side by side, where one is for choosing the work and the other for the employee to assign the work to. The Mock-up is shown in Figure 19.



**Figure 19** - Assign Work to Employee Page Mock-up 1

By selecting the work from the drop down list the user will be able to see the work description and the assigned employees as shown in Figure 20. Such information is useful for checking the details of the work selected, are enough employees assigned for the work, and who is assigned for the work.



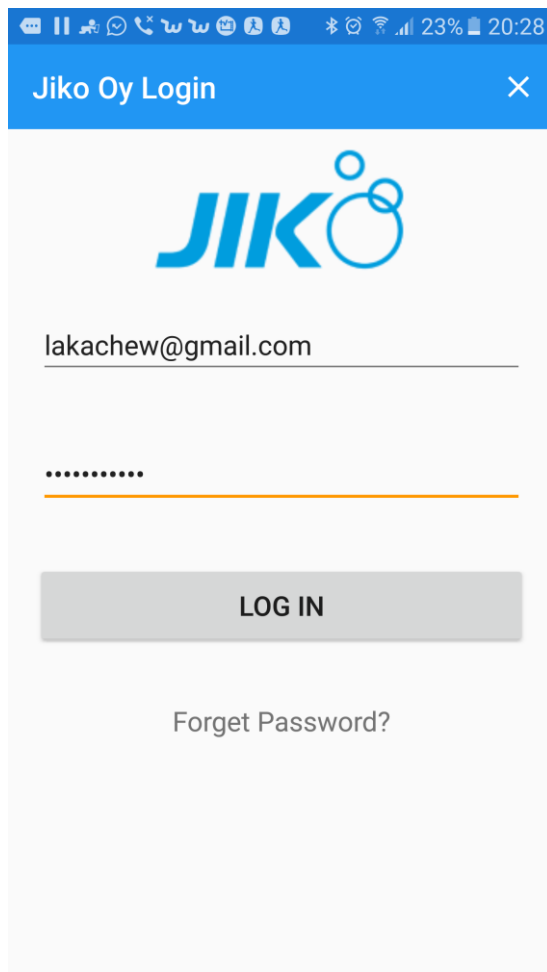
**Figure 20** – Assigning work to employee page Mock-up 2

## 2.3.2 Mobile Application Mock-ups

Unlike the Web application, there are only three Mobile Application Mock-ups, even though the background tasks are more complicated. In this section, all the Mobile Mock-ups are shown along with their description.

### 2.3.2.1 Login Activity Mock-up

This page is what the user sees first when opening the application. The user can login or request for a new password if forgotten, as can be seen in Figure 21.

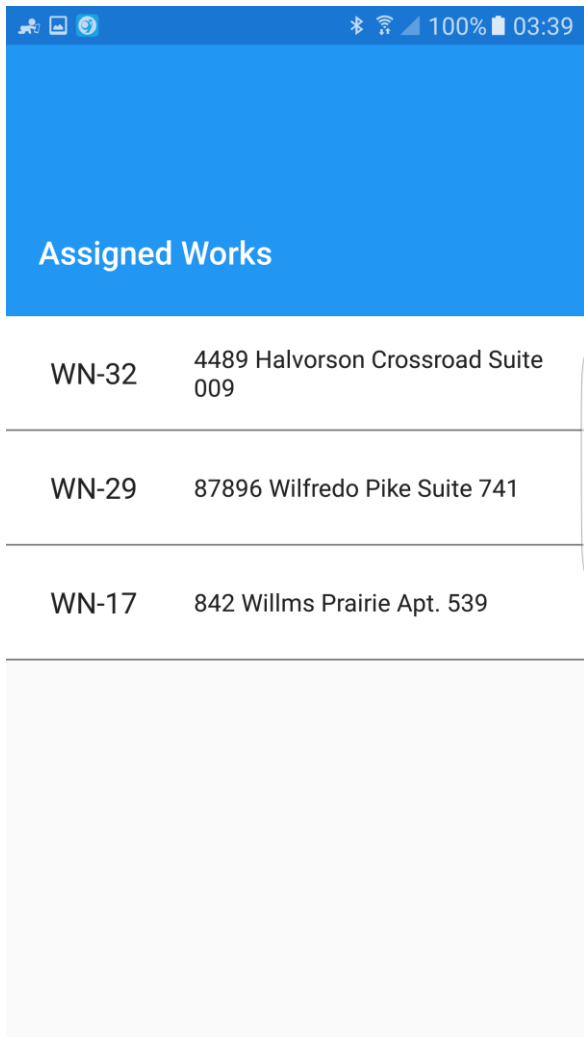


**Figure 21** - Login Activity

### 2.3.2.2 User Works Activity

When the user is Authenticated, the user credentials like the name and user distinguishing key are stored in the User object model, so in every connection made to the server this credentials are used in order to provide the users the assigned tasks and record their activities.



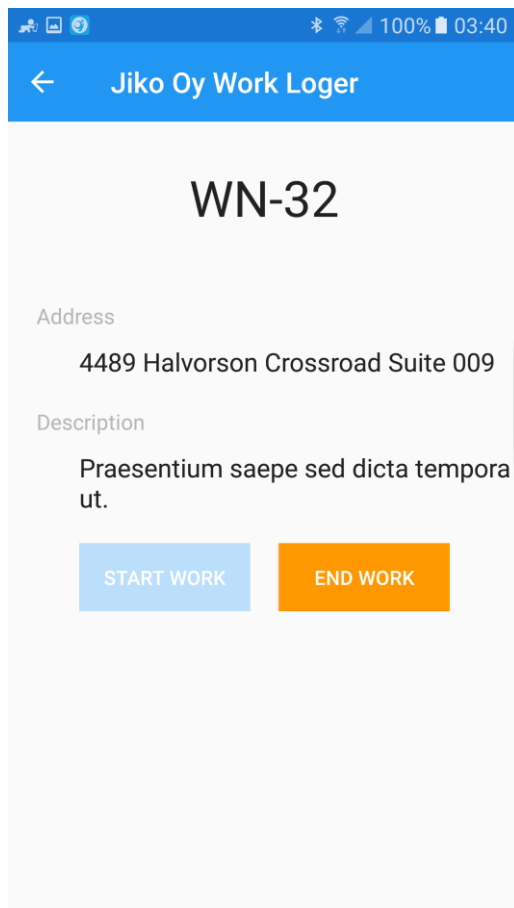


**Figure 22** – Assigned user work lists Activity

As can be seen in Figure 22, the page provides the user with a list of works which are assigned to the specific user only. In order to distinguish the various works list, each one is provided with a unique Work Number (WN) and Address. Whenever the work is completed the Office employees will update the status of the work to ‘Finished’, which in turn will remove the work from the mobile user lists, till then the mobile users will see all the assigned tasks. The refresh button, which is located at the top right corner, helps the user to update the work lists so that the finished works will be removed and new assigned tasks will be added. Each work list is clickable and takes the user to the selected works page; such a page is shown in Work Activity sub section below.

### 2.3.2.3 Work Activity

The work activity is a page where the user interacts with the selected task (assigned work). This page interaction will be recorded on the server side, which will be visible for the office employees.



**Figure 23** – selected user work Activity

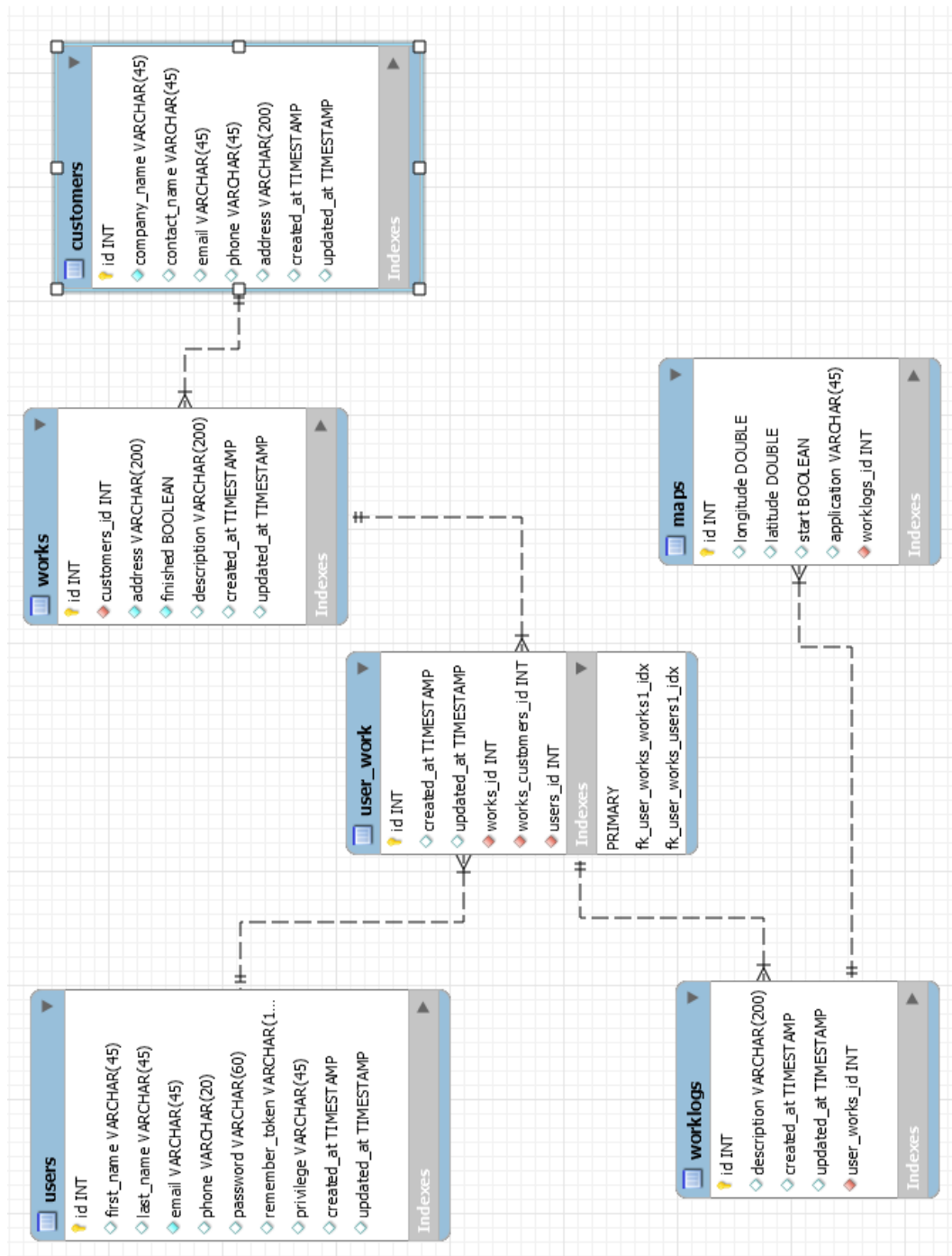
As can be seen in Figure 23, the user is provided with more information about the task. Figure 23 is a good example for showing an activated work (START WORK is not active in other work it has already been activated), i.e. if the work is active or not is registered on the server side and the user can activate the work and log out of the application and get the same activated data when returned to the Work Activity page. This is accomplished for the reason that every record made to the work is stored in the server and the application checks or gets the data from the server for building the page. In turn also, no information is lost if by any means the mobile

shuts down. This mechanism enables the office employees to track the work activity remotely in real time.

## **2.4 Data Requirement**

Since similar types of data are being used among the two clients, the data requirements for both applications can be document together in this section.

Models/Database tables used for the documented project are shown in the following database diagram (Figure 24).



**Figure 24 - Database Architecture V2**

The database architecture design was shown so that the relationship among the Models can be easily be visualized.

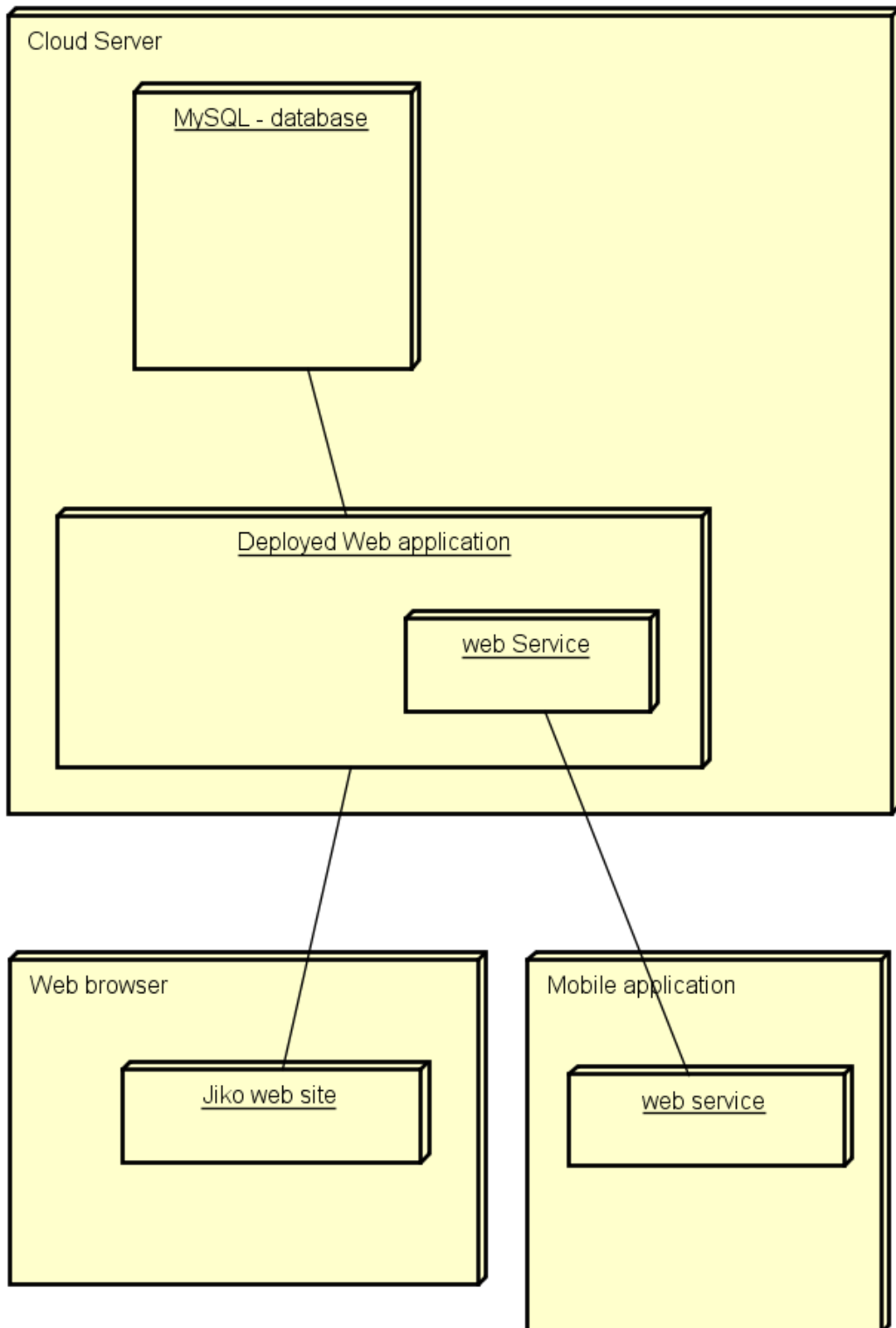
## 2.5 Non-Functional requirements

The non-functional requirements for Jiko ltd are listed below:

- Security
  - authorized personnel should only be allowed to access the applications.
  - Level 1 Secured communication between client and server.
  - The user can recover a lost password.
- Performance
  - The server should respond in less than 2 second for all requests.
  - The application should provide different functionalities to handle different requirements.
- Capacity
  - The server is made to handle a maximum capacity of 1000 employees to be safe even though there are about 80 employees at this moment.
  - Server storage should be a scalable, for the initial stage 10GB is used.
- Availability
  - Available at all times
  - Only in Finland
  - Internet required
  - Backup
  - System recovery
  - Up after reboot.

### **3 APPLICATION DESIGN**

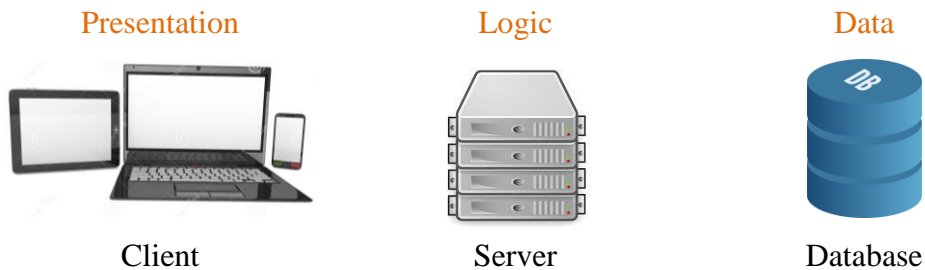
The application is designed in such a way that the web application is hosted in a cloud server. The connection between the mobile app, the web app, and the database is handled on the server. The Deployment Diagram or the application hardware and software connection is shown in Figure 25.



**Figure 25** – Deployment diagram, which shows the hardware and software connections for the whole application

### 3.1 Three-tier organization

In three-tier organization applications, three physical tiers are used. This tiers are shown in Figure 26.



**Figure 26** – Three-tier organization

### 3.2 Framework Applications

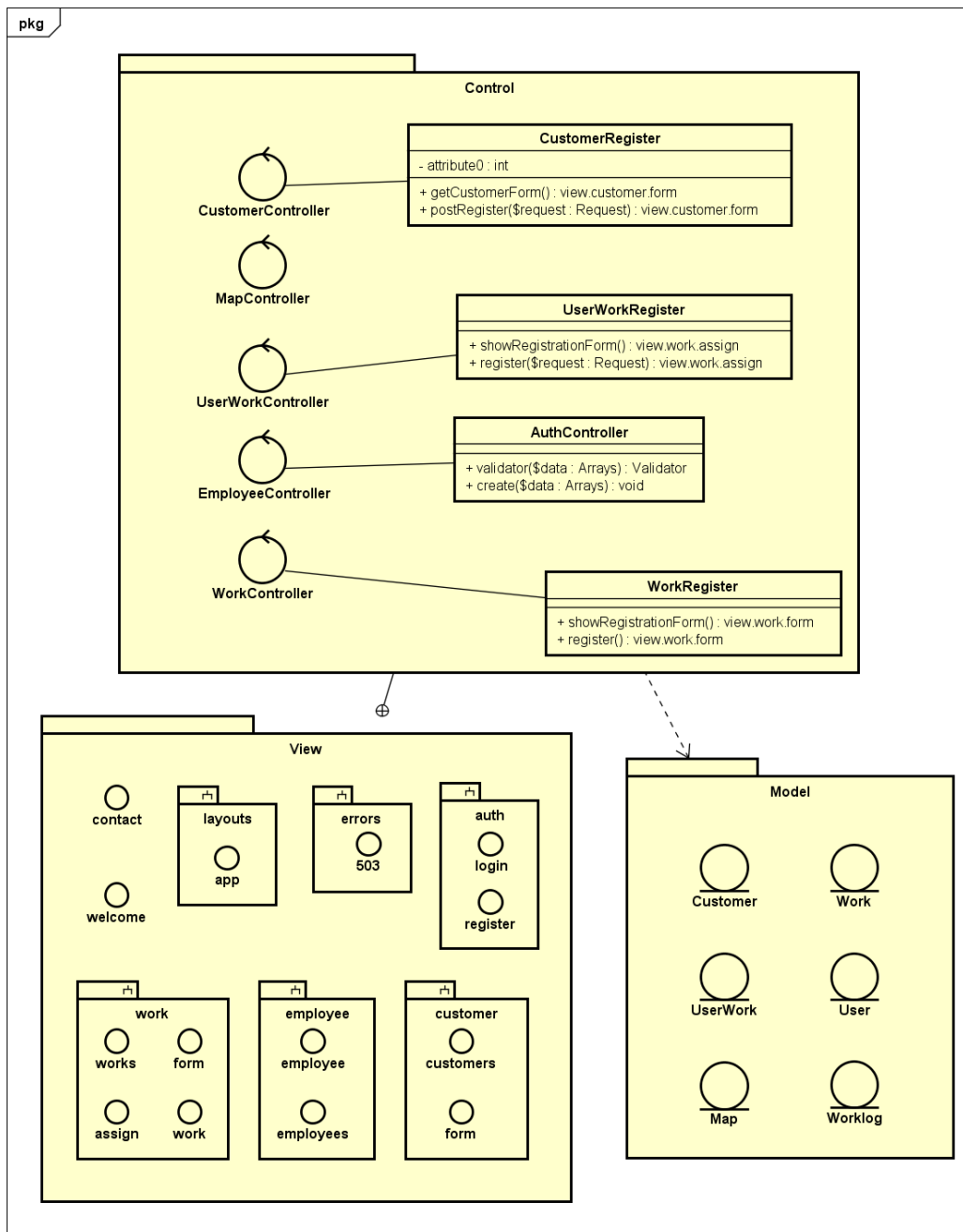
Frameworks are built to assist on the construction of the web application based on a single programming language. Generally, application frameworks support and guide the developer to use short, maintainable, and convenient methods for building an application. Such methodology encourages other developers to easily understand the project and collaborate on the development since the application will be built on a general guide line. This guideline is not a onetime developed architecture but rather a serious of development (and is still in progress), which makes usage of Framework popular among developers. Generally, frameworks are built to support mapped URL, integration of tools (API), object-relational mapping (ORM), Bootstrap, and many more tools and features are being added from time to time.

The detailed part of the technologies used for each application is documented in the coming Used Technologies Chapter.

### 3.3 Architectural Design

In order to make the application easily maintainable and scalable, building a proper architecture is required. Best practice for building such architecture is considering the separation of tasks. The most widely used architectural design is the Model, View, and Control (MVC) Architecture. In this section the architectural design for both mobile and web applications is documented.



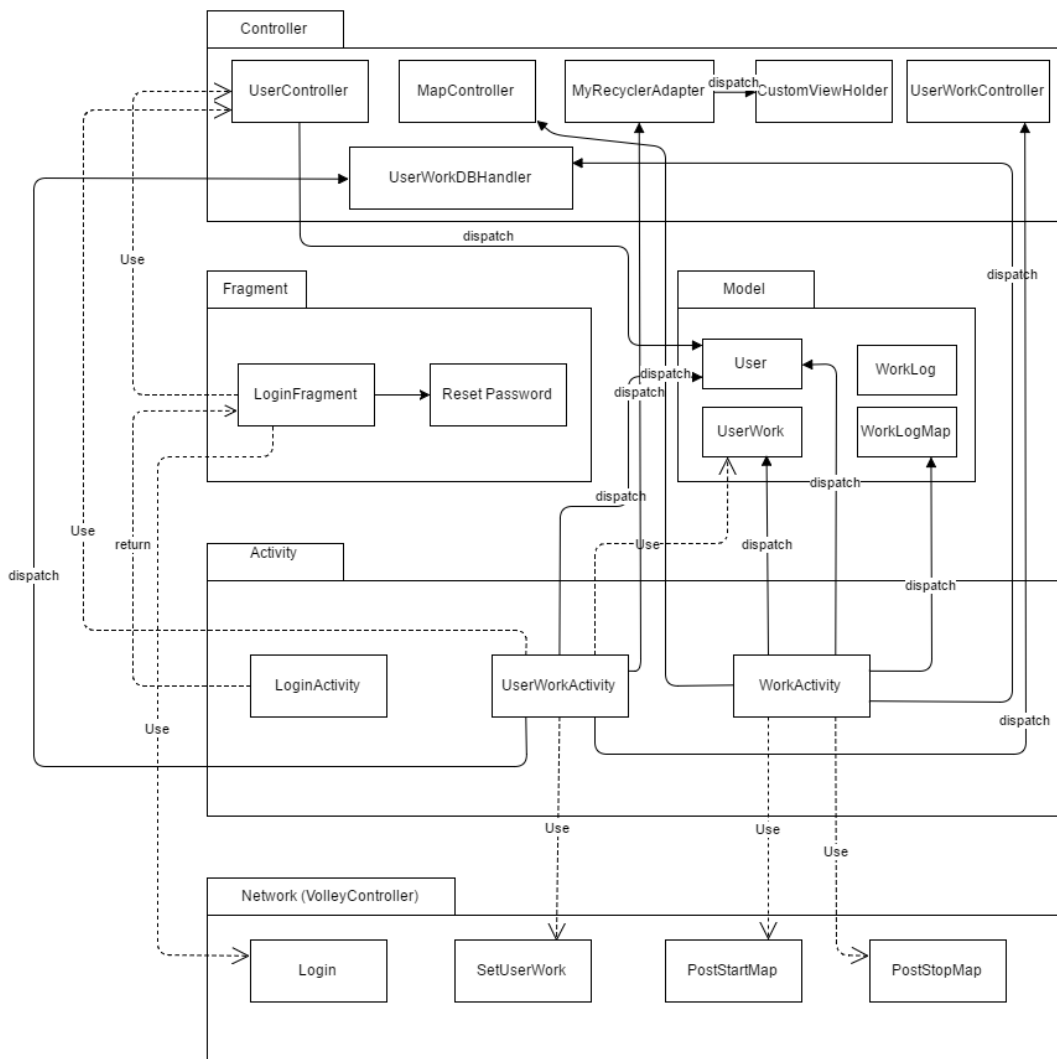


**Figure 27** – Web application Architectural Design

As can be seen in the above Figure 27, there is a distinct separation of tasks. The most distinct packages are Model, View and Control. Then each of the packages is further divided into sub sections, model and class so that each performs a specific task. i.e. the ‘WorkController’ task is controlling everything related to the Work Model and Work View, but again the ‘WorkRegister’ is separated since its task

involves extra functions, which is changing the model data content. The same goes for the remaining Controller class's.

Furthermore, the Mobile Architectural design is developed in such a way that there is a distinct separation of tasks, as the web application. i.e. MVC architectural design, which is shown in Figure 28.



**Figure 28** – Mobile Application Architectural Design

## 4 USED TECHNOLOGIES

The technologies and tools used for building the applications are documented in this chapter.

### 4.1 Web app

The web application is built using Laravel Framework and will be hosted in a cloud server. The Framework uses Composer, Vagrant and Virtual Box for installing and deploying Laravel projects on a localhost or PHP hosting server. For the Jiko Ltd web application also other relevant technologies are used, and they are listed in the following sub-sections.

#### 4.1.1 Laravel Framework

Laravel Framework is a free, open-source PHP web framework, created by Taylor Otwell. The framework is intended for the development of web applications supporting the MVC architectural pattern. The framework also provides an expressive and elegant syntax for coding and managing the project. Laravel attempts to take the redundant hustle out of development by easing common tasks used in the majority of web projects, such as authentication, routing, sessions, queueing and caching.

The Laravel framework has few system requirements including:

- PHP  $\geq$  5.5.9
- OpenSSL PHP Extension
- PDO PHP Extension
- Mbstring PHP Extension
- Tokenizer PHP Extension

However all of its requirements are bundled when using the Laravel Homestead virtual machine. For such reason it is recommended to use Laravel Homestead instead of downloading the Laravel project alone. /2/

Laravel is a well-documented framework with a lot of community involving in the development and supporting side. Laravel have integrated powerful tools needed for large, robust applications. It has an excellent architectural design for control container, expressive migration system and an integrated unit testing support, which are all needed to build any kind of web application.

This framework supports the addition of other tools by using a dependency manager known as Composer.

#### **4.1.2 Composer**

Composer is a tool for dependency management for PHP. It allows the developer to import and install libraries to the project. Composer is not only limited on importing it but also updates, downgrades, locks and removes packages during development or afterwards. /1/

#### **4.1.3 Vagrant**

Vagrant is a tool for building complete development environments detached from the local operating system. With an easy-to-use workflow and focus on automation, Vagrant lowers the development environment setup time by just tweaking few configurations, which in turn increases development or production parity, and makes all vagrant project work on similar work environment or machine. /3/

Laravel can be installed in Vagrant development environment and all advantage of Vagrant is inherited. For such reason Laravel developing team have created a small tool known as Homestead for easing the integration of this two technologies and also for simplifying the transfer of project along with the development environment.

#### **4.1.4 Laravel Homestead**

Laravel Homestead is developed and distributed by Vagrant. Vagrant box provides you a wonderful development environment without requiring you to install PHP, HHVM, a web server, and any other server software on a local machine. No more miss-configuring the development environment and requiring specific operating system. Vagrant boxes are completely disposable if not required or clean booting

the server. If something goes wrong with the virtual machine (VM) while developing, it can be switched to a new VM by destroying the running VM and deploying a new one, in minutes while all local files are still intact.

Homestead can be used in Windows, Mac, or Linux system and it includes the following:

- Ubuntu 16.04
- Git
- PHP 7.0
- HHVM
- Nginx
- MySQL
- MariaDB
- Sqlite3
- Postgres
- Composer
- Node (With PM2, Bower, Grunt, and Gulp)
- Redis
- Memcached
- Beanstalkd.

All of this can be used to develop not only Laravel Framework but different types of web applications, which rank homestead among the best virtual machine hosting local server configuration. /4/

Laravel Homestead requires a virtual machine. On this documentation Virtual Box is used. This tool is documented in the following section.

#### **4.1.5 Virtual Box**

Virtual Box is a cross-platform virtualization application. In other words, it can run on Windows, Mac, Linux or Solaris operating systems. Using Virtual Box extends

the capabilities of your existing computer so that it can run multiple operating systems (inside multiple virtual machines) at the same time. /5/

#### **4.1.6 Alexpechkarev/google-maps**

Alexpechkarev/google-maps (Collection of Google Maps API Web Services for Laravel) is an API build for Laravel projects for requesting Google maps API in a convenient way Using Google Maps APIs web services. This Laravel API can be found using Packagist, by the name “alexpechkarev/google-maps”. As when used on this project alexpechkarev/google-maps version is 1.0.5.

Packagist is the storage site where all public PHP packages installable with Composer are found, or in other word Packagist is the main composer repository. Any of composer handled dependencies can be searched and downloaded from the packagist site (<https://packagist.org>).

Google Maps APIs web services, is officially realized by Google and is a collection of HTTP interfaces to Google services providing geographic data for maps using applications.

#### **4.1.7 Cornford/googlemapper**

Cornford/googlemapper (an easy way to integrate Google Maps with Laravel) is an API built for Laravel projects so that it can access Google Maps for getting google map along with of two dimensional global view, also being able to assign markers, view level, direction, and all available google utilities.

This API is chosen because the map is being built and assigned markers programmatically and finally rendered on the view while the user can interact with the map.

#### **4.1.8 Tymondesigns/jwt-auth**

Tymondesigns/jwt-auth (JSON Web Token Authentication for Laravel and Lumen) is a simple tool, which can be added to Laravel projects and providing a simple

means of authentication. This tool can be added to the project by using Composer, then addressing the service provider, configuration the aliases and payload to point to the jwt-auth tool and then publishing and generateing the JWT configuration and key respectively.

JWTs are an encoded representation of JSON object. JSON might have zero, one or many name/value pairs, and such information is being converted into URL-safe (base64 encoded) while maintaining its unreadability (encrypted), and unmodifiable (signed).

JWTs can have different usage and implementation depending on the communication needed. For this thesis project, the usage is to authenticate the mobile user to the server and allow the authorized personnel to upload and download information to and from the server.

A JWT sample token would look something like this:

“dBjftKJFTrdTrfg98Gu.mB92KK29skkkHDJFO.wInLjUj8j97HGvfD”

This is a Base64 encoded string. When it is broken apart there are three distinct separate strings. This separator is the dot in between long strings. Each of the sectioned strings have their own representation and usage.

The first section is the header that describes the token, followed by the payload, which contains the data and finally the signature hash that is for checking the validity of the token. /8/

#### **4.1.9 Postman**

Postman is a software built for sending and receiving data to and from a server. This application works for all kind of servers as long as it has a URL local/remote address. The application provides a lot of ways for the parameters, headers, authorization and many more features to add on http/https request. /10/

Postman is not only used for the development of the web application but also for the mobile application. Postman can be downloaded as a Chrome application for windows OS, that is Postman uses chrome's privilege for accessing the Internet.

## **4.2 Mobile app**

The Mobile Application is built using Android Studio, the official IDE (Integrated Development Environment) for Android application development. Android Studio has reached the current stage with a series of update or changes to the IDE.

Building an Android application requires different set of development tools, most of the tools required for this application are included in Android SDK (Software Development Kit). The added tools are discussed in this section.

### **4.2.1 Android Studio**

Android Studio is the Official IDE for developing an Android application. The application is owned, developed, and maintained by Google. Google has announced in 2013 that Android Studio is freely available for anyone to develop Android mobile applications for both personal and business use. At this time of project development Android studio is at Version 2.1.2. /7/

The Android studio development is based on IntelliJ IDEA, in other words, on top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance the productivity of Android applications. The features provided are:

- A flexible Gradle-based build system,
- A fast and feature-rich emulator,
- A unified environment where all Android applications can be developed,
- Instant Run to build changes without building a new APK (Android Application Package),
- Code templates and GitHub integration,
- Extensive testing tools and frameworks,



- Lint tools to catch performance, usability, version compatibility and other problems,
- C++ and NDK support
- Built-in support for Google Cloud Platform, making it easy to integrate Google Cloud messaging and App Engine.

#### **4.2.2 Java JWT (JJWT)**

JJWT (Java JSON Web Token) is a library used for communication between applications. JJWT is particularly designed for Java and Android applications.

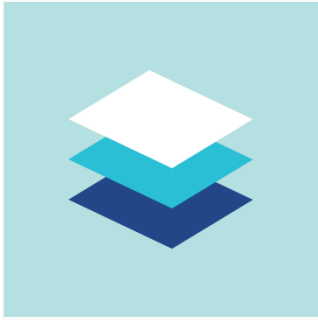
The aim of JJWT is to provide an easily usable and understandable library for creating and verifying JSON web Tokens (JWTs) on the Java Virtual Machines (JVM). This library was created by Stormpath's CTO, Les Hazlewood and now the library is being maintained by a community. /11/

#### **4.2.3 Material Design**

Material Design is a Google support library for assisting on the design of mobile applications. It allows developers to build elegant user interfaces with multiple type of patterns at their disposal. The support library has based certain principles for building application with user-friendly standards, which have been achieved with a lot of research at the Google Development Centre.

Using Material Design allows for a unified experience across platforms with different size and resolution. Design patterns is built to work in both the older and the latest models of Android operating systems (AOS), but on the older devices the new features will be omitted keeping the desired information still working on older devices. Some features even work on older AOS. The principles of material design are shown in Figure 29.

Material is the metaphor



Bold, graphic, intentional



Motion provides meaning



**Figure 29** – Material design principles

## **5 IMPLEMENTATION**

The above mentioned technologies were all used for building the applications. In this chapter the implementation of the two applications is documented.

### **5.1 Building the Web application**

The web application is built with Laravel framework. As mentioned above (in the used technologies section) Laravel projects add tools by using Composer. The implementation of this tools will be explained in the following sub sections.

#### **5.1.1 Alexpechkarev/google-maps**

Jiko Ltd web application uses Alexpechkarev/google-maps API for converting text addresses to geographical location as in longitude and latitude. Since such web services have a limited request allowed per day for Google services, the application architecture is made in such a way that the retrieved google responses are saved so that there won't be a need to access the google services every time an address is needed.

In order to use such service from Google, the "Google Maps Geocoding API" needs to be activated. Such privilege is possible after having an authenticated credential. Once the user is authenticated (has an account), the user can generate a server key or credentials (type of API key), which are used to access whenever a request is needed from Google. The key is included in every request made to Google so that Google can monitor the traffic flow and location. Google also provides such monitored values in the "Google APIs" web application. /6/

The geographical coordinates can be collected using the following code:

```

// initializing a collection
$collection = collect();

try
{
    // collect the coordinates response from Google Maps
    $response = (array) \GoogleMaps::load('geocoding')
        ->setParam(['address' => $address])
        ->get();

    // get the longitude and latitude from the json response
    $longitude = json_decode($response[0])->results[0]->geometry->location->lat;
    $latitude = json_decode($response[0])->results[0]->geometry->location->lng;

    $collection = $collection->merge([$longitude, $latitude]);
} catch (\ErrorException $e)
{
    //if the coordinates are not found send back an error message
    $collection = collect(['error' => 'Address not found']);
}

```

**Code 1** – Collecting the geographical coordinates from Google Maps

### 5.1.2 Cornford/googlemapper

Cornford/googlemapper API can be downloaded/installed using a composer. There are two ways downloading this tool, one is using the command line and the other is using composer.json as shown in Code 2 and Code 3 respectively.

```
$ composer require cornford/googlemapper:2.10
```

**Code 2** - command line code for downloading and installing the mapper tool

```

"require": {
    | "cornford/googlemapper": "^2.10"
    |
},

```

**Code 3** - Imbedding the mapper tool using the composer.json file

In order to use such service from Google, the “Google Maps JavaScript API” needs to be activated. Such privilege is possible after having an authenticated credential. Once the user is authenticated (has an account), the user can generate a server key or credentials (type of API key), which are used to access whenever a request is needed from Google. The key is included in every request made to Google so that

Google can monitor the traffic flow and location. Google also provides such monitored values in the “Google APIs” web application. /6/

An Implemented snippet code, for generating a Map marked with work location (coordinate) making the centre point at Jiko Ltd office is shown in Code 4.

```
//Generating a Map making Jiko Oy the origin
Mapper::map(63.0925796,21.6516582,
  ['draggable' => false,
   'eventMouseDown' => 'console.log("mouse down");',
   'zoom' => 12,
   'cluster' => false,
   'markers' =>
    [ 'title' => 'Jiko Oy',
      'scale' => 1000,
      'animation' => 'DROP']]);

//going through all requested works list
foreach($works as $work)
{
  $work = Work::find($work['id']);
  $workCustomer = $work->getCustomerName();
  if(isset($work['longitude']))
  {
    // adding location to the generated Map
    Mapper::informationWindow($work['longitude'],$work['latitude'], $workCustomer,
      ['clusters' => ['size' => 10, 'center' => true, 'zoom' => 15],
       'markers' => ['symbol' => 'circle',
                    'scale' => 1000,
                    'animation' => 'DROP']]);
  }
}
```

**Code 4** – Generating Marked Works Map

### 5.1.3 Tymondesigns/jwt-auth

Tymondesigns/jwt-auth API can be downloaded/installed using a composer. composer can be used in two ways. One is using the command line as shown in Code 5 and the other is using the composer.json file as shown in code 6.

```
Lakachew@Lakachew-A56 MINGW64 /
$ composer require tymon/jwt-auth:0.5.9]
```

**Code 5** - Command line code for downloading the Tymondesigns/jwt-auth.

```
"require": {
    | "tymon/jwt-auth": "^0.5.9"
},
```

**Code 6** – Adding Tymondesigns/jwt-auth tool using composer.json file.

Jwt-auth has multiple usages. Each usage is as important as the others and for this reason the Implementation of the jwt-auth is shown in this section.

The jwt-auth api is used firstly for authenticating the mobile application users, and for providing a key for accessing the server if authenticated. The key lasts for a time-to-live (ttl) limit of 1 week. The implemented code for authentication using jwt-auth api is shown on Code 7.

```
try{
    if (! $token = JWTAuth::attempt(['email' => $email, 'password' => $password])) {
        //return response()->json(['error' => 'invalid_credentials'], 401);
        return $this->invalidCredential('invalid_credentials');
    }
} catch (JWTException $e) {
    // Error handling: informing 'something went wrong'
    //return response()->json(['error' => 'could_not_create_token'], 500);
    return $this->respondNotFound('could not create token');
}
```

**Code 7** – Authentication by user name and password

The generated token will be sent back to the mobile application. On the mobile application side this is when the mobile user gets authenticated and allowed to log into the application. And once the application has started all the user requests will be authorized by checking the token. Such authorization methodology is shown on Code 8.

```

try {
    if (! $user = JWTAuth::parseToken()->authenticate() {
        return $this->invalidCredential('user_not_found');
    }
} catch (TokenExpiredException $e) {
    return $this->exceptionThrowResponse("token_expired", $e->getStatusCode());
} catch (TokenInvalidException $e) {
    return $this->exceptionThrowResponse("token_invalid", $e->getStatusCode());
} catch (JWTException $e) {
    return $this->exceptionThrowResponse("token_absent", $e->getStatusCode());
}
}

```

**Code 8** – Authentication by using the token

As can be seen from Code 8, if the authentication fails, the response will contain an error message, which can be used on the mobile app to inform the user.

## 5.2 Building the Mobile application

In this section the implementation of the added tools will be documented. The tools used are shown in section 4.2.

### 5.2.1 Java JSON Web Token (JJWT)

The JJWT is used for handling authenticated user credential with the server. That is the server token response can be manipulated using the JJWT library. This library can be integrated to the application by using the Gradle dependencies as shown in Code 9.

```

dependencies {
    compile 'io.jsonwebtoken:jjwt:0.6.0'
}

```

**Code 9** – Gradle code for adding the JJWT tool

Collecting the user credential from the token using the JJWT api is shown on Code 10.

```

try {
    Claims claims = Jwts.parser().parseClaimsJwt(jwtHeaderAndClaim).getBody();

    int id = Integer.parseInt(claims.getSubject().toString());
    user.setId(id);
    user.setEmail(claims.get("email").toString());
    user.setFirstName(claims.get("first_name").toString());
    user.setLastName(claims.get("last_name").toString());
    user.setPrivilege(claims.get("privilege").toString());
    user.setCreateAt(claims.get("created_at").toString());
    user.setToken(this.token);

    isSet = true;
} catch (Exception e) {
    Log.e("Claims Exception ", e.toString());
}
}

```

**Code 10** – Implementation of JJWT

### 5.2.2 Material Design

In the used technologies chapter, the features of Material Design are discussed. In this section the implementation of the application will be shown along with the code.

Material Design support libraries can be added to the application project by addressing the tools address in the Gradle file, as shown on Code 11.

```

dependencies {

    compile 'com.android.support:recyclerview-v7:24.0.0'
    compile 'com.android.support:cardview-v7:24.0.0'
    compile 'com.android.support:design:24.0.0'
    compile 'com.android.support:support-v4:24.0.0'

}
}

```

**Code 11** – Adding the Material Design support libraries

The Material Design support library has been used for a different part of the application development. For this reason, one part of Material Design usage will be shown.

One of the Material Design supports was used for listing the works for the user or when viewing the UserWorkActivity page. Each work on the list has its own view



class known as Recycle Viewer. This RecyclerView class can be used to generate and connect the view (layout-resource) to the object value like the work id, and address. Code 12 shows the implementation of RecyclerViewer class.

```
@Override
public CustomViewHolder onCreateViewHolder(ViewGroup viewGroup, int viewType)
{
    mView = LayoutInflater.from(viewGroup.getContext()).inflate(R.layout.user_work_layout,
        viewGroup, false);

    return new CustomViewHolder(mView);
}

@Override
public void onBindViewHolder(CustomViewHolder customViewHolder, final int position)
{
    UserWork userWork = userWorkList.get(position);
    /*      Settes the TextView      */
    customViewHolder.titleTextView.setText("WN-" + userWork.getId());
    customViewHolder.contentTextView.setText(userWork.getAddress());
    /*      Settes the ClickListener      */
    View.OnClickListener clickListener = getOnClickListener(position);
    customViewHolder.rowLayout.setOnClickListener(clickListener);
}

@NonNull
public View.OnClickListener getOnClickListener(final int position) {
    return (view) -> {
        UserWork userWork = userWorkList.get(position);
        Toast.makeText(mContext, userWork.getAddress(), Toast.LENGTH_SHORT).show();
        Intent intent = new Intent(mContext, WorkActivity.class);
        intent.putExtra("user_work_list_id", position);
        mContext.startActivity(intent);
    };
}

@Override
public int getItemCount() { return (null != userWorkList ? userWorkList.size() : 0); }
```

### Code 12 – Recycler View class generating a custom made list

The getItemCount method first counts the number of lists available so that such number of lists will be created. When through the Code there will be such a number the onBindViewHolder method will be accessed. On each access the view holder will be initialized with the work information and an OnClickListener will be assigned to each work row. A sample of the list is shown in Figure 22.

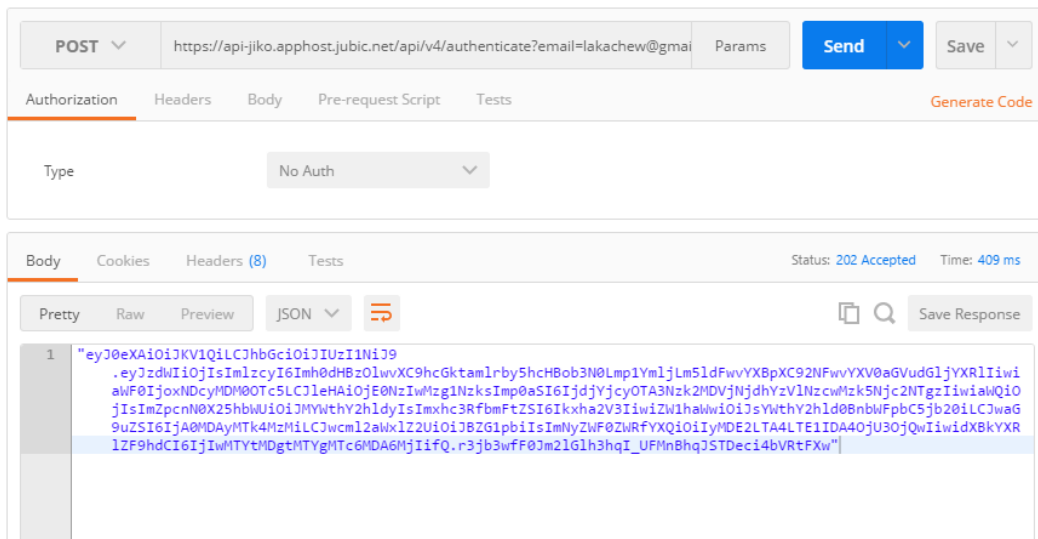
## **6 TESTING AND ANALYSIS**

This section provides an overview the tests made to check the applications features and the analysis of the skills achieved through developing the applications.

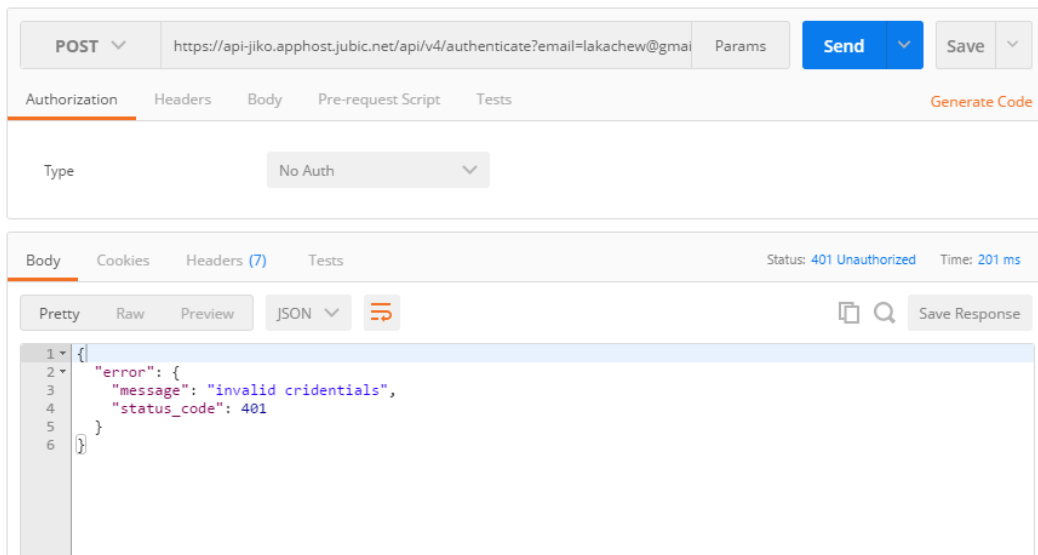
### **6.1 Testing**

The Web application is first deployed on a local host server for testing all the features without going through other configurations concerning the server and mobile application. After building most of the application functionalities the application is ready to be tested on a cloud server.

For testing the web services, Postman was used for both the local host and the cloud server. By using Postman, the web services response can be tested/checked while building the web services. A test sample snippet is shown in Image 1 and 2 (below) for both authorized and un-authorized user responses respectively.



**Image 1** – Postman response snapshot for authenticated user



**Image 2** – Postman response snapshot for un-authorized user

Such responses checked in Postman are results that needed to be handled in the mobile application so Postman usage is not just only for the testing the web application response but also for testing the mobile web services.

The mobile application was tested with a built in android emulator with API Level 17 for common check-up and an API Level 21 device (Samsung S6) was used.

After going through all of the above mentioned tests, the project was hosted on Jubic Ltd OpenShift Cloud server and further development and/or tests was made to make the application operation smooth. This test might be the last step for the development stage, but there will be application development and testing as long as the customer requests additional features.

## **6.2 Analysis**

Running the application on Openshift Cloud server was challenging due to the lack of assistant tool for the project framework (Laravel) when this application was being developed (Aug, 2016). But once the application has integrated with the Cloud, Openshift provides different features for handling the server.

## **7 LEARNING OUTCOME**

As the developer of this project, I had a prior knowledge of building web and mobile application developments. The Laravel Web applications were new to me including JSON web Tokens, Android Material Design and Cloud host management. For this reason, this application has given me a time consuming challenge but at the end it was a great achievement with an addition of knowledge with experience of this new and popular technology.

In each new technology implementation, the most challenging part was the initial stage, that is knowing how the technology is working and how it can be integrated to the project. This stage requires an in-depth understanding of both technologies, so that they perform as desired.

Finally, being self-motivated is my key competence for the thesis accomplishment. I knew from the beginning that I was going to learn multiple technologies which would require time and dedication. At the end I am satisfied to see that all the technologies are communicating and performing as desired.

### **7.1 Future development**

One of the features this project accomplishes is the fact that this application is scalable. That is, its architecture is built so that more features can be added. Not only is the addition of features possible but addition of memory (volume) is also possible in case the number of employees and/or company grows.

Finally, there are some features that I do not want to pass without mentioning, they are:

- Each employee can login into the server to check his/her work status and update a few things like profile and work description.
- Customers can also login and check their work progress.
- Letting the customer add work.
- Letting the customer provide feedback of the work accomplished.

- Creating a method and page for calculating the distance from the office (Jiko Ltd) to the working place (customer).

Some features listed might not be as important as the others so most of the updates will be requested by the company (Jiko Ltd).

## 8 SUMMARY

Jiko Oy benefits the following by using this software:

1. Instantly collecting employees work hours when the work is done.
2. Getting a better data on knowing who is working where by using the assigned employee's page.
3. Revising the available work locations using Google map.
4. Easley assigning employees.
5. Easley check available, active, and unassigned employees
6. Easley registering works, customers and employees.
7. Avoiding paper work, which can be lost and misplaced.
8. Having the application at hand since the web application is built to work on a smart phone, tablet and laptop.

Building the application with the documented approach accomplishes a stable, scalable, and upgradable application. A person with a good knowledge of the Laravel project can easily start developing the application in less than a day. The same axes for a developer in starting to develop the mobile application.

Finally, the application is hosted on a cloud server, which makes the server response quick even for companies located in different continents, if the company wishes to branch out in different countries. Still, cloud services are relatively cheap since the company pays for the used space only.

## 9 REFERENCES

### Numbered Website Resources/References

- /1/ Composer official site. <https://getcomposer.org/doc/00-intro.md>
- /2/ Laravel Official site. <https://laravel.com/docs/5.2>
- /3/ Vagrant official site: <https://www.vagrantup.com/about.html>
- /4/ Laravel Homestead <https://laravel.com/docs/5.2/homestead>
- /5/ Virtual Box official site. <https://www.virtualbox.org/manual/ch01.html>
- /6/ Google Maps APIs – Authentication <https://developers.google.com/maps/documentation/geocoding/get-api-key>
- /7/ Android Developer site: <https://developer.android.com/studio/index.html>
- /8/ JSON Web token for Java and Android: <https://stormpath.com/blog/where-to-store-your-jwts-cookies-vs-html5-web-storage>
- /9/ Android API Level: <https://source.android.com/source/build-numbers.html>
- /10/ Postman: <https://www.getpostman>.
- /11/ JSON Web Token for Java and Android: <https://stormpath.com/blog/jjwt-how-it-works-why>