Timelog System on Android OS
Abstract
Usage of smart phones has become more common nowadays. Timelog system is a user friendly and web based application which systemizes the business of a company with the focus on time tracking. Following the Timelog system from small screen of smart phones is not user-friendly because the Timelog application and its features have not been designed to be accessed from these devices.

The solution is creating a smart phone client for the Timelog system which can connect to the database of the system. An appropriate approach is to design a Web Service as an intermediate layer to communicate between the client side and the database.

The result is a smart phone client that can be used by users of Timelog system to manage their business when they use their mobiles instead of personal computers. The Web Service can also be used for other smart phone platforms. As a result it can be considered as a generic protocol for the mobile-user of Timelog system.

Keywords:
Timelog, Android, Web Service, SOAP, REST, Hibernate, HTTP Client, Jersey, JSON
Acknowledgement

I would like to express my sincere gratitude to my supervisor Dr. Rüdiger Lincke for his support and guidance throughout this thesis.

I would also want to appreciate my family for their support all along my studies.
# Contents

1 Introduction .................................................................................................................. 1  
  1.1 Problem .................................................................................................................... 1  
  1.2 Motivation ............................................................................................................... 2  
  1.3 Goals and Criteria .................................................................................................... 2  
  1.4 Overview .................................................................................................................. 3  

2 Background .................................................................................................................... 4  
  2.1 Definitions ............................................................................................................... 4  
    2.1.1 Project .................................................................................................................. 4  
    2.1.2 Category .............................................................................................................. 4  
    2.1.3 Deliverable ......................................................................................................... 4  
    2.1.4 Activity ............................................................................................................... 4  
    2.1.5 Task .................................................................................................................... 4  
  2.2 Web Service ............................................................................................................. 5  
    2.2.1 SOAP .................................................................................................................. 5  
    2.2.2 REST .................................................................................................................. 6  
    2.2.3 Compare SOAP, REST ........................................................................................ 6  
  2.3 Hibernate .................................................................................................................. 7  
  2.4 HTTP Client ............................................................................................................ 9  
  2.5 Smart Phone Operating System ............................................................................... 10  
    2.5.1 Android OS ........................................................................................................ 10  
    2.5.2 Android Version ................................................................................................ 11  
    2.5.3 Android Hardware Buttons .............................................................................. 13  

3 Requirements ............................................................................................................... 15  
  3.1 Use-Cases ............................................................................................................... 15  
    3.1.1 Login .................................................................................................................. 16  
    3.1.2 Add Task ............................................................................................................ 16  
    3.1.3 View Task ......................................................................................................... 16  
    3.1.4 Statistics .......................................................................................................... 17  
  3.2 Functional and Non-Functional ............................................................................... 17  

4 Architecture and Design .............................................................................................. 19
4.1 Overall Architecture ................................................................. 20
4.2 Web Service Architecture .......................................................... 20
4.3 Android Design ................................................................. 21
  4.3.1 List View and Grid View ..................................................... 21
  4.3.2 Progress Dialog .............................................................. 22
  4.3.3 Action Bar ................................................................. 22
  4.3.4 Search Interface ............................................................ 24
5 Implementation ............................................................................. 26
  5.1 Overall View ......................................................................... 26
  5.2 Login ..................................................................................... 27
  5.3 Home ................................................................................... 28
  5.4 Select Project ...................................................................... 29
  5.5 Select Deliverable ................................................................. 30
  5.6 Select Activity ....................................................................... 31
  5.7 Add Task .............................................................................. 32
  5.8 View Task ............................................................................ 33
  5.9 Statistics .............................................................................. 34
6 Case Study ..................................................................................... 37
  6.1 Login, Home Page ................................................................. 37
  6.2 Select Project ...................................................................... 37
  6.3 Add Task .............................................................................. 38
  6.4 View Task ............................................................................ 39
  6.5 Statistic ............................................................................... 39
  6.6 Sorting ............................................................................... 40
7 Conclusion and future work ............................................................. 42
  7.1 Conclusion .......................................................................... 42
  7.2 Future Work ....................................................................... 42
References ......................................................................................... 45
List of Figures

Figure 1.1: Timelog web application on Android OS .......................................................... 1
Figure 2.1: JSON String format example ........................................................................... 7
Figure 2.2: Eclipse, database and hibernate setup ............................................................. 8
Figure 2.3: HttpClient, Web Service and database connection ........................................... 9
Figure 2.4: Smart phone operating system popularity ....................................................... 10
Figure 2.5: Android API level in the manifest file ............................................................ 12
Figure 2.6: Android version popularity ............................................................................. 12
Figure 2.7: Android hardware buttons .............................................................................. 14
Figure 4.1: Overall architecture of the system ................................................................. 19
Figure 4.2: Hibernate object-oriented models for the Timelog .......................................... 20
Figure 4.3: ListView in comparison to Grid View ............................................................. 22
Figure 4.4: ProgressDialog in Android .......................................................................... 23
Figure 4.5: Action Bar in Android .................................................................................. 23
Figure 4.6: Overflow menu on Action Bar in Android ....................................................... 24
Figure 4.7: Search Widget on an Action Bar in Android ................................................... 24
Figure 4.8: Search Dialog in Android ............................................................................. 25
Figure 5.1: Overall view of the Android client ............................................................... 26
Figure 5.2: Login Screen ................................................................................................. 27
Figure 5.3: Home Screen ............................................................................................... 28
Figure 5.4: Select Project Screen .................................................................................... 29
Figure 5.5: Select Deliverable Screen ............................................................................. 30
Figure 5.6: Select Activity Screen .................................................................................. 31
Figure 5.7: Add Task Screen .......................................................................................... 32
Figure 5.8: View Task Screen ....................................................................................... 34
Figure 5.9: Statistic Screen ............................................................................................ 35
Figure 5.10: Horizontal Statistic Screen ......................................................................... 36
Figure 6.1: Login Page Progress ..................................................................................... 37
Figure 6.2: Home Page Progress ................................................................................. 38
Figure 6.3: Select Project Progress ............................................................................... 38
Figure 6.4: Add Task Progress ..................................................................................... 39
Figure 6.5: View Task Progress ................................................................. 39
Figure 6.6: Statistic Page Progress .............................................................. 40
Figure 7.1: Operating system popularity for the most well-known smart phone ............ 43
Figure 7.2: Floating context menu and the Contextual Action Bar ............................ 43
Figure 7.3: Fragment in Android .................................................................... 44
List of Tables

Table 2.1: Relation between SQL and HTTP commands ................................................. 6
Table 3.1: Use-Case for Login page .............................................................................. 16
Table 3.2: Use-Case for Add Task .................................................................................. 16
Table 3.3: Use-Case for View Task ............................................................................... 16
Table 3.4: Use-Case for Statistic .................................................................................. 17
Table 3.5: Login Requirements ..................................................................................... 17
Table 3.6: Add Task Requirements ............................................................................... 17
Table 3.7: View Task Requirements ............................................................................. 18
Table 3.8: Statistic Requirements .................................................................................. 18
Table 6.1: Compare sorting in the Server-side and Client-side .................................... 40
1 Introduction

One of the important keys for running a successful organization is planning and scheduling of projects and resources. Analysis of the time to complete the current projects will enable both managers and employees to follow every single detail.

A Timelog application is a web based application with a user friendly interface that systemizes the scheduling approach [1]. It can be used by companies to manage their business. The focus of this application is time tracking for different projects of the company. It also enables both managers and employees to estimate and analyze the time for different projects by viewing some statistics for the projects.

It allows a company to have different project managements. This system helps both managers and employees to save the time in order to continue different projects. With this application it becomes easy for project managers to create new tasks in addition to allocate the tasks to the employees.

1.1 Problem

Access to current implementation of the Timelog application from a smart phone over the internet is not user-friendly since the web application is not optimized to be accessed from the small screen of a smart phone as shown in Figure 1.1. Timelog web application functionalities have been designed to be accessed from the devices with a bigger screen such as desktops and laptops which are suitable to show a relatively large amount of data.

On a large computer screen multiple functionalities can be provided but small screen of a smart phone always need to be considered to show a limited and special functionalities of the Timelog system. Therefore to represent every functionality different pages (activities) need to be designed in the smart phone.

Figure 1.1: Timelog web application on Android OS.
1.2 Motivation
Usage of smart phones has become more common nowadays. Creating a smart phone client for the Timelog system can be beneficial for different users of the system. In many cases managers and employees do not have access to their personal computers for instance while travelling or being in a meeting. Their smart phones are a way to access to the internet. Therefore a smart phone client can helps them to manage their tasks.

According to the latest statistics iPhone and Android are the most common smart phone devices [2]. In spite of iPhone having a good progress in the recent months, Android devices are still the most common smart phone devices [2]. As a result designing an Android client for the Timelog system can be more preferable in compare with others.

Access the data of the Timelog database from the Android client directly is not a proper solution since it is not a secure connection. Design a Web Service as an intermediate layer can be an appropriate approach to guarantee more security because Web Service can include a security mechanism to transfer data between the Android client and the database.

An optimized graphical user interface (GUI) for the Android client is an important part of implementation and designing since the Android client will be designed to improve on the not optimized view of Timelog web application when users access to it from smart phone devices. Following the latest guidelines of Android UI to develop a proper interface should be considered in the development of this application.

1.3 Goals and Criteria
The thesis includes a main goal and some sub-goals in order to solve the problem.

The main goal is to create an Android client for the Timelog system. This client can be considered as a substitution for a part of web application when users do not have access to their personal computers. The main features in this client are adding a new task to the system, view the tasks and review statistics in an optimized way for a smart phone.

To support the main goal, a Web Service needs to be designed as an intermediate layer between the Android client and the database. The Web Service provides more security for the system. It can play another important role later on when another smart phone client will also be developed for the Timelog system. The same Web Service can be used to connect to the database by other smart phones. It saves times and codes for every smart phone client. As a result this Web Service can be considered as a generic protocol for the mobile-user of Timelog system.

The final goal is to design a user interface for the Android client which can be compatible with latest version of Android devices. Android 4.x version has been released recently. For instance Action Bar Compatibility version is a new feature that has been published with this new version [3].

We decide to support three important mechanisms of Timelog system in the Android App:

- Add a new Task.
- View Tasks.
• Display Statistics.
The goal is reached if these functionalities support different version of Android devices and we can have a generic Web Service that guarantee more security for the system.

Now we have a native Android App for Timelog. Users of Timelog system can use this App whenever they do not have access to their personal computers. This App has a user-friendly interface and can be accessed from different version of Android devices.

1.4 Overview
Background knowledge that is required for this project will be discussed in Section 2. This section starts with describing the Web Service. Two of the most important and useful Web Services (SOAP and REST) are described and compared. Hibernate technology and HTTP client will also be presented. Smart phones will be introduced and Android operating system and versions will be discussed at the end of this section.

Requirements of the system will be presented in Section 3. It includes presenting of Use-Cases for different features of the system in Section 3.1. Functional and non-functional requirements of the system will be discussed in Section 3.2 based on the Use-Cases.

In Section 4, overall architecture and design of the system will be presented. It includes the Web Service architecture and Android design. In the Android design section some of the controls such as List View, Grid View, Progress Dialog and Action Bar will be presented.

Implementation of the system will be presented in Section 5. It includes an overall view of the Android client at first. Implementations of every feature including a brief description and Web Service specification will be presented also.

In Section 6, improvement in the design of each feature will be presented. It represents different attempts that have been done to make the user interface more user-friendly.

In Section 7 we will discuss if the goal has been met. Some suggestion for the future work will also be provided at the end.
2 Background
In this section background knowledge to implement an Android client will be discussed. This Android client is able to communicate with an external database. To make this connection work, a Web Service between client and database is a proper solution.

There will be an introduction of what a Web Service is and how it works. SOAP and REST which are two of the most well known Web Service approaches will be compared. Finally the smart phones and in particular Android operating system will be introduced. A short history about the Android versions and specially improvement in the latest versions will also be represented.

2.1 Definitions
We have some entities in the Timelog system such as Project, Category, Deliverable, Activity and Task. A brief description is provided for them in this section.

2.1.1 Project
When a company receives a project, they need to enter it in the system. This entity is known as Project in the system. It includes a name, key name, description, Category (Section 2.1.2) and Deliverables (Section 2.1.3).

2.1.2 Category
Each Project in the system should have a Category. This Category includes the Activities (Section 2.1.4) that need to be done. Every Category can contain one or several Activities. It includes a name and description in the system.

2.1.3 Deliverable
When a Project entered in the system, it should be break down to smaller part to assign to employees. They are known as Deliverable in Timelog system. A Deliverable can be parent for some other Deliverables or a child in the system. It includes a name, key name description and Project which it belongs to. Every Project can contain one or several Deliverables in the system.

2.1.4 Activity
It represents what a user performs in his Task (Section 2.1.5) such as meeting, development or consulting. Every Activity has a specific cost in the system that will be used to calculate statistics. It includes a name, key name, cost and the Category which it belongs to.

2.1.5 Task
A user starts his work with entering a Task in the system. He needs to specify the Project, Deliverable, Activity, start and end time to enter the Task. A short description can also be provided by the user.
2.2 Web Service
Web Service is a protocol to communicate between two applications via Internet. It can connect applications together over the web and transfer information between them.

World Wide Web Consortium (W3C) is the main international standards organization for the World Wide Web. It explains the Web Service as a software protocol that can support communication between programs over a network [4].

Two of the most significant and arguable Web Services are Simple Object Access Protocol (SOAP) and REpresentational State Transfer (REST) Web Service. There are lots of arguments about these two kinds of Web Service implementation in the Web [4]. In the next sections these Web Services will be introduced and compared.

Web Services do not provide a Graphical User Interface (GUI). They exchange data over a network instead. Developers can fetch the information and represent them in a nice GUI such as a web page or a smart phone client. The most important advantages of a Web Service between a smart phone client and database are:

**Usability:** If we create a Web Service between an Android client and database, other kind of smart phone such as iPhone and Windows phone can use the same Web Service. We do not have to create another kind of mechanism to communicate between client and database. We can consider the Web Service as a generic protocol for the mobile-user of the system.

**Flexibility:** If we want to change the database configuration, we do not need to make any further changes in our client applications as long as we use the same Web Service interface. So it makes it easier to maintain our system. We just need to adapt our Web Service to the new database configuration.

**Security:** It is not a good solution to connect a client application via Web directly to the database. We need to have some kind of authentication mechanism. It can make the communication between clients and server more secure. Web Services give us the ability to implement an authentication mechanism.

We pay for these benefits by increasing a level of complexity since we add a Web Service between Android client and database [5]. It would probably be faster if we have a direct access from the client to the database but we lose the advantages of using Web Service. Mostly it is recommended to use Web Service since just a little bit faster is not really important.

2.2.1 SOAP
Simple Object Access Protocol (SOAP) is a communication protocol specification to transfer massages and information in the structure of a Web Service. It uses Extensible Markup Language (XML) to transfer messages.

This protocol exchanges massages over Hypertext Transfer Protocol (HTTP) specification [6]. One of the best ways to exchange information via Internet is over HTTP. The main reason is all of the Internet browsers and servers support HTTP protocol [7].
Basically SOAP is not depended on a programming language. It is a protocol to exchange information between applications. These applications can be developed on different operating systems [6].

### 2.2.2 REST
REpresentational State Transfer (REST) is a software architectural style that helps developers to design Web Services [8]. REST is not a standard. It cannot be sold by companies such as Microsoft, Sun or IBM [9].

It is just a style of architecture which we need to understand and then design our Web Services based on it. The idea of REST has been introduced by Roy Thomas Fielding in his doctoral thesis called “Architectural Styles and the Design of Network-based Software Architectures” in 2000 at the University of California, Irvine [8]. It did not have much attention at first but after some years it became one of the significant Web Service approach.

It uses Hypertext Transfer Protocol (HTTP) specification. The HTTP methods can be used as a relationship between Structured Query Language (SQL) and HTTP commands as shown in Table 2.1 [10].

<table>
<thead>
<tr>
<th>Action</th>
<th>SQL</th>
<th>HTTP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create a new data</td>
<td>Insert</td>
<td>POST</td>
</tr>
<tr>
<td>Retrieve the data</td>
<td>Select</td>
<td>GET</td>
</tr>
<tr>
<td>Modify the data</td>
<td>Update</td>
<td>PUT</td>
</tr>
<tr>
<td>Remove the data</td>
<td>Delete</td>
<td>DELETE</td>
</tr>
</tbody>
</table>

Table 2.1: Relation between SQL and HTTP commands.

It can transfer Extensible Markup Language (XML), JavaScript Object Notation (JSON) or other kinds of media types [10]. Yahoo, Amazon and eBay use REST approach for their services [11].

### 2.2.3 Compare SOAP and REST
When we started to develop the application, I suppose to investigate about the type of Web Service in our system. SOAP(section 2.2.1) and REST(section 2.2.2) were two of our options. I compare REST and SOAP and see which one is a better case for the Timelog app. The criteria for comparison are based on the functionality of the Timelog app. The architecture of the Timelog app will be discussed in Section 4. Following are the cases that a SOAP approach is more proper than a REST approach.

**Web Service Security (WS-Security):** In cases where security is really important, SOAP can be a good choice. Both the Restful Web Service and SOAP support Secure Socket Layer (SSL) but SOAP can add some enterprise security components to the WS-Security. Soap is a good option for some enterprise scenarios [11].

**Web Service Reliable Messaging:** Soap has a standard messaging system. It receives a message when data is sent successfully. In case that connection fails, it has also a retry
logic which tries to receive information again. The Restful Web Service does not have this kind of logic. Rest waits for the client to deal with connection failure. It expects client to retry again. As a result SOAP is a better choice when having this standard messaging system is important. For instance we want to connect from an Android phone to our bank account. It is recommended to use SOAP as a Web Service because when we want to transfer money, we need to be sure about transferring process. If we use a Restful Web Service it means that we do not have the standard messaging system (successful / retry logic). If transferring is done successfully but response from client to Web Service is failed by any reason, transferring can be done two times and it would be a huge problem [11].

In many cases Restful Web Services are used. Following are the most important reasons that REST is a proper approach.

**Simplicity:** Implementation of a Restful Web Service can be simpler than a SOAP one. Basically SOAP needs a toolkit to make a connection to the client such as Apache Axis. Axis is an implementation of the SOAP. On the other hand SOAP implementation requires specific knowledge in XML parsing. Restful Web Services are easier to implement since they are independent from any toolkit [12]. They use Uniform Resource Locator (URL) to communicate between server and client [13]. URL is a formatted kind of string to find the location of a resource on the Internet. It has been widely spread nowadays. Therefore any developer can easily figure out how to deal with URL and identify a web resource.

**Compare JSON and XML:** Generally JSON is simpler than the XML despite of JSON has same potential like XML. An XML file includes lots of information that we may not use especially when developers need to transfer the information. It needs to declare new tags and attributes to keep information and data inside them. It can be considered as a document markup language [15].

JSON does not need to use tags. It has a lighter structure than XML. As a result JSON is more proper to transfer the data and it can be parsed faster than XML [11]. By the way when developers need to transfer document based data, XML is a better solution [15].

```json
```

Figure 2.1: JSON String format example.

2.3 Hibernate

Almost every application should have a mechanism to store data nowadays. The storage system could be an external database.
Therefore developers need to have a system to connect to the database from their object-oriented models. Developers can easily do it with SQL and Java Database Connectivity (JDBC) [16].

One of the major problems is when they want to change the database for instance from Oracle to MySQL Server. Some relations and commands which have worked on Oracle may not work in MySQL Server. It means that lots of modifications need to be done in the application. As a result it is not a good solution for maintenance of the system to use JDBC library always. Developers need to have an appropriate way to transfer information from the data storage to the object-oriented models and vice versa. Hibernate is a technique which solves this problem by mapping from an object-oriented model to the database.

Hibernate is an open source Object Relational Mapping (ORM) library for the Java programming language which provides a mapping from a object-oriented entity to a Structured Query Language (SQL) data storage[16].

Hibernate has been introduced in 2001. Two years later JBoss Company which is a part of Red Hat Software Company nowadays hired the developers of this technology and started to work on it. The latest version is Hibernate Core 4 which has been released in December 2011. It is one of the most well-known ORM libraries which has been spread considerably nowadays [17].

Lightness of this tool made it so beneficial [16]. There is no need to write any additional code when we use Hibernate technology to communicate with database. It is just enough to attach the appropriate library to the entities and their data and finally add the configuration file (hibernate.cfg.xml) to the Project. This configuration file is in the XML format. It includes all the mapping from a client application to the SQL data storage. Developers who use Hibernate just need to modify this configuration file when they decide to change their data storage system [18]. Figure 2.2 is an overall view of how to use Hibernate in Java to connect to the database.

![Diagram](image.png)

Figure 2.2: Eclipse, database and hibernate setup.
2.4 HTTP Client

All source data from database should be transferred to the Android App. We have a Web Service between them. HTTP Client is a library supported in Android that we use to talk to the database using Web Service. Figure 2.3 is a view of the connection between the smartphone Client, Web Service and the database via HTTP client.

HTTP client uses the Hyper Text Transfer Protocol (HTTP) to transfer information from a web server to the client over the Internet. Web browsers are perhaps the most famous kind of these clients. It can also be used to transfer data from a client application to a database via a Web Service [19].

HTTP protocol is one the most well-known protocol which is used on the Internet nowadays. Web Services and network based applications use HTTP protocol to transfer the information mostly. Source code of HTTP protocol is freely available under Apache License [19].

The last version of HTTP client has been released by Apache in December 2007. The common HTTP Project has been finished. Despite of HTTP Project will not be developed no longer a new Project has been started by Apache which is called Apache HTTP Components. It is supposed to show better performance and have more flexibility [20].

An important feature supported by HTTP client is the implementation for SQL commands. Basically GET, POST, PUT and DELETE which read, add, update and remove data from data storage respectively are used by HTTP client. Therefore database transaction can be supported by a client application which uses HTTP protocol to transfer the information.

It also supports Secure Hypertext Transfer Protocol (HTTPS) [19]. HTTPS is a protocol for the World Wide Web which provides a secure transaction over SSL by encryption and decryption of information over the Internet. As a result a secure connection to the server from the client application could be considered over a HTTP protocol.

Figure 2.3: Http client, Web Service and database connection.
2.5 Smart Phone Operating system

Nowadays mobile phones are not just a device to write a message or make a phone call. They can be used as our desktop computers or laptops. We can surf on the Internet, send an email to a friend or make a video call. Smart phones are kind of mobile devices that give us these abilities. It is not possible to change the operating system of the phone therefore it is important for buyers to make the best choice. Functionality is one of the most important attribute for buyers to make a decision.

There is a real competition between companies around functionalities of smart phones. Perhaps the users are the real winner of this competition. Followings are some of the most important properties of smart phones that a programmer may consider to develop a smart phone application.

- Limited battery life.
- Small screen size.
- The computational power is restricted.
- Some extra hardware support such as GPS and movement sensors are provided in compare with ordinary computers.

In the Figure 2.4 which is the latest research from Nielsen Company and released on 18 January 2012, the most well-known smart phones has been represented. Android is the most common smart phone with more than 50 percent of smart phone users.

Figure 2.4: Smart phone operating system popularity [2].

2.5.1 Android OS

Android operating system which belongs to Google and has been powered by the Linux kernel is one the most popular smart phone operating system nowadays [21]. The kernel is a kind of abstraction layer between the hardware and the software which is installed on the device [22].
Android is an Open Source Project (AOSP). Openness of Android is perhaps made it significant and popular. It provides an open framework for developers to develop any kind of application that they wish. Android provides powerful Application Program Interface (API) to handle graphical user interface (GUI) based on the hardware features on it [23].

Fast progress of Android devices is the other significant benefit for Android users. Developers can always improve their application as a result of this rapid process [23].

The programming language for Android applications is Java. XML specification is using to design graphical user interface. XML is also using for configuration in a Manifest file [24]. Manifest will be discussed more in continue.

Android applications have a Software Development Kit (SDK). Basically it is a large Java library (API) that provides support for developing Android applications. It compiles all packages and resource files of an Android project to a file with .apk format [24]. This file can be run on an Android device with compatible version. In the next section the compatibility versioning will be discussed in more details.

Every Android application includes some components to be able to run. These components are the building features for an Android project. Generally Android supports four different kinds of component:

- Activity
- Service
- Content provider
- Broadcast receiver

Every component has its own life cycle which represents creation and destroying of the component. [24].

When an Android operating system wants to start an application, it needs to know about all components in the application. Manifest which is a XML file format includes all the component information. Manifest should be declared at the root of the Android project. It also contains some other information about the project. Followings are the most important information that should be declared in Manifest.xml file [24].

- Any user permission such as internet access.
- API level that will be discussed in detail in the next section.
- API libraries that is used in different applications, such as Google Map.

2.5.2 Android Version

The compatibility between an application and an Android device is important to run the application. API Level is a unique integer value which identifies the framework based on the Android version [26].

Android devices with the new API level can mostly supports the applications which were developed with the old version because most of the changes in the new version are new functionalities which are added to the older version. It is also possible to have some removing or modification in the new version. As a result some applications cannot be run on the new version [26].
Each Android device keeps the API level in its system itself. Developers need to specify API level in their applications in the manifest file. This manifest file can specify the minimum, target and maximum API levels is needed for an application to run on the Android device as shown in Figure 2.5 [26].

![Manifest File Example](image)

**Figure 2.5:** Android API level in the manifest file [26].

Always it takes time for the newer version to become more well-known experimentally. According to Figure 2.6 which has been released on April 2012, the mostly used Android version is Android 2.3.3 (API level 10) nowadays. The first revision for this version has been released on February 2011.

![Android Version Popularity](image)

**Figure 2.6:** Android version popularity [28].

Android version 3.x with Honeycomb code name was the next generation of Android operating system after API level 10. It specially provided support for tablets with many new features for interface design includes [29]:

- Action bar is a pleasant feature that was introduced by API level 11. It is generally a substitution for the title bar of an activity page with some new features.
- JSON utility supports new classes in this version. It is a complement for org.json library to make parsing of a JSON string format simpler. This library has been used in the implementation of Android client for the Timelog system to communication between the Android client and the Web Service.
- Fragment was another important new feature. It can separate an activity page to make desired user interface. A fragment has its own lifecycle.

The last Generation of Android platform is Ice Cream Sandwich. It brought Android 3.x features that have been designed for tablets to the smart phones. The first and second revision of Android 4.x has been released in October and December 2011[31]. Android 4.3 has been published in March 2012 which is the third revision of Ice Cream Sandwich. It introduced API level 15 that is the newest API level up to now [32].

Action bar compatibility version is a new significant feature which has been published in this new version [3]. In the implementation of Android client for the Timelog system Action bar has been used. As a result it will be discussed in more details in Section 4.2.3.

2.5.3 Android Hardware Buttons

Smart phones with Android Operating system have three or four hardware buttons bellow their touch screen. These buttons provide some basic functionality for the Android devices. They can also be overridden by developers and some new functionality be declared for them. These buttons are Back, Menu, Search and Home button. They have been represented in Figure 2.7 respectively. Some devices do not contain a Search button.

Home button is perhaps the most usable button. It always returns the user to the home screen of device. When the user click on the Home button the current activity will be paused. Developers can override this button functionality in their application for instance when user presses the Home button in an application, it goes back to the home page of the application instead of the main home screen of Android device. Another functionality of this hardware button will appear when the user presses and holds the Home button. It opens a window for the user includes the recently used application that has been paused.

Another useful hardware button is Menu button. Users can access to the setting of their devices via the Menu button. Developers can also design different menu option in their application. Every activity in an application can include a menu definition. This menu is not directly visible for the user. User need to click on the Menu hardware button to access to them. Developers need to think about which option can be declared in the menu, especially after introducing the Action Bar because the same functionality can be define on the Action Bar. It will be discussed in Section 4.2.3 in more details.

Back button returns the user to the former page. User can click on this button multiple times to go back step by step in an application. It can be considered as a stack. If this stack is empty user goes back to the main home screen of phone.

Search button is not provided in all Android devices but it can provide some useful functionality for the user such as search in the Web, Contact list or Android Market. In some cases developers do not have to look for a place in their application for search interface because Search button can be overridden in their application [33].
Figure 2.7: Android hardware buttons [33].
3 Requirements
The first step to design the system was providing different requirements of the system. These requirements have been concluded as a result of an analysis on the Timelog web page. In the next step different features for the system have been design to fulfill these requirements.

At first Use-Cases for different features of the system will be represented. A Use-Case is the list of steps that represent the interaction between the user and the system. They can be illustrated as a diagram. Developers can use them to analyze the system requirements [34]. The basic knowledge to understand a Use-Case diagram has been supposed for the readers of this report.

Functional and non-functional requirements of the system will be represented as well. A functional requirement specifies the behavior of the system. It includes what a functionality of the system supposes to do. In the other hand a non-functional requirement includes what a functionality of the system should look like. It can be interpreted as a quality of the system. This quality can be used to investigate the usability of the system [35].

3.1 Use-Cases
The focus of Android client for the Timelog system is time tracking and statistic sections. The time tracking includes a mechanism to add a new Task. It also contains a mechanism to display the entered Tasks. Statistic section represents cost and value for different Projects based on the actual work hours. It can be used for a specific Project to display all statistic details. They will be discussed in more detail in Section 5.

The followings are the features that were required for the mobile-user of the Timelog system. They have been designed based on the different functionalities of the Timelog system. Small screen of smart phone has also been considered to design these features.

- Login
- Add Task
  - Select a Project
  - Select a Deliverable
  - Select an Activity
  - Set start and end time. Provide a short description
- View Task
  - Select all / a Project
  - Select all / a Deliverable
- Statistic
  - Select all / a Project
- Setting

In continue of this section every feature of the system will be represented in a table includes a short description, pre-condition, post-condition and the flow of events. They will be used in the next section to present functional and non-functional requirements. The
Actor for these Use-Cases is the user of the system which has a certified username and password.

### 3.1.1 Login

<table>
<thead>
<tr>
<th><strong>Description</strong></th>
<th>User should provide an appropriate username, password and server address. System should check the permission to complete the login process.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pre-conditions</strong></td>
<td>Username and password should enter correctly. Server should be reachable for the client. Default server address is provided for the user.</td>
</tr>
<tr>
<td><strong>Post-condition</strong></td>
<td>Other functionality of the system can be used if the Login process has completed successfully.</td>
</tr>
</tbody>
</table>
| **Flow of Events** | 1 – Username and password should be entered.  
2 – Default server address can be chosen. It can also be modified.  
3 – By clicking the login button, System starts to check authentication.  
4 – Login process is completed successfully or a proper error message will be provided by the system. |
| **Exception** | If entered username and/or password are incorrect, a proper message pops up regarding incorrect credential.  
If entered server address is invalid, a proper message pops up regarding invalid server address. |

Table 3.1: Use-Case for Login page.

### 3.1.2 Add Task

<table>
<thead>
<tr>
<th><strong>Description</strong></th>
<th>User can enter a new Task based on the Projects he/she has been involved in.</th>
</tr>
</thead>
</table>
| **Pre-conditions** | List of Projects that user is involved in, has been provided for the user.  
List of Deliverables and Activities based on the selected Project is also provided for the user. |
| **Post-condition** | A new Task will be entered in the system. |
| **Flow of Events** | 1 – User selects a Project.  
2 – User selects a Deliverable based on the Project that has been selected.  
3 – User selects an Activity based on the category of the Project.  
4 – User enters start and end time for the Task. A short description about the Task can also be provided by the user.  
5 – The process is completed when user clicks on the save button. A proper message pops up. |

Table 3.2: Use-Case for Add Task.

### 3.1.3 View Task

<table>
<thead>
<tr>
<th><strong>Description</strong></th>
<th>User should be able to time tracking for different Tasks that have been entered in the system.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Pre-conditions</strong></td>
<td>List of Projects that user is involved in, has been provided for the user.</td>
</tr>
</tbody>
</table>
List of Deliverables based on the selected Project is also provided.

<table>
<thead>
<tr>
<th>Post-condition</th>
<th>Flow of Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>1 – User selects a Project. User is also able to select all Projects. If all Projects are selected by the user, the system goes directly to step 3. 2 – User selects a Deliverable based on the Project that has been selected. User is also able to select all Deliverables. 3 – A view for different Tasks will be displayed based on the selected criteria in the former pages.</td>
</tr>
</tbody>
</table>

Table 3.3: Use-Case for View Task.

### 3.1.4 Statistics

<table>
<thead>
<tr>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Statistics for different Projects that user is involved in, can be represented for the user. User is also able to select a specific Project and look into statistics for different Deliverables.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pre-conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of Projects that user is involved in, has been provided for the user.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Post-condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Flow of Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – User can select a Project. User is also able to select all Projects. 2 – If a specific Project is selected by the user, different Deliverable statistics will be displayed. If all Projects are selected by the user, different Project statistics will be represented.</td>
</tr>
</tbody>
</table>

Table 3.4: Use-Case for Statistic.

### 3.2 Functional and Non-Functional

Different features of the system have been presented in the previous section. In this section different functionalities of each feature will be represented. These functionalities are the general requirements of the system. Implementation of each feature to fulfill these requirements will be discussed in Section 5. In the following tables F represents a functional requirement and NF displays a non-functional requirement.

<table>
<thead>
<tr>
<th>Login</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 A login interface should be provided for the user.</td>
</tr>
<tr>
<td>2 Username and password should be verified by the system.</td>
</tr>
<tr>
<td>3 Server Address should be confirmed by the system.</td>
</tr>
</tbody>
</table>

Table 3.5: Login requirements.

<table>
<thead>
<tr>
<th>Add Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select a Project</td>
</tr>
<tr>
<td>4 Projects that user is involved in should be verified by the system.</td>
</tr>
<tr>
<td>5 A Sorting mechanism for the Projects should be performed by the system.</td>
</tr>
<tr>
<td>6 List of Projects should be presented for the user.</td>
</tr>
</tbody>
</table>

Table 3.6: Add Task requirements.
A Search interface has been provided for the Projects.  

Select a Deliverable

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>A Search interface has been provided for the Projects.</td>
</tr>
<tr>
<td>8</td>
<td>Deliverables based on the selected Project should be verified by the system.</td>
</tr>
<tr>
<td>9</td>
<td>A Sorting mechanism for the Deliverables should be performed by the system.</td>
</tr>
<tr>
<td>10</td>
<td>An interface for the list of Deliverables should be displayed for the user.</td>
</tr>
<tr>
<td>11</td>
<td>A search interface has been provided for the Deliverables.</td>
</tr>
</tbody>
</table>

Select An Activity

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Activities based on the Category of selected Project should be verified.</td>
</tr>
<tr>
<td>13</td>
<td>Activities should be sorted alphabetically by the system.</td>
</tr>
<tr>
<td>14</td>
<td>List of activities should be displayed for the user.</td>
</tr>
<tr>
<td>15</td>
<td>A search interface has been provided for the user.</td>
</tr>
</tbody>
</table>

Start and End Time

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>An interface should be provided to enter start and end time for the Task. A brief description for the Task can also be entered by the user.</td>
</tr>
<tr>
<td>17</td>
<td>A new Task should be added to the system by clicking on the save button.</td>
</tr>
</tbody>
</table>

Table 3.6: Add Task requirements.

View Task

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>18</td>
<td>In addition an item should be designed to able the user to select all Projects.</td>
</tr>
</tbody>
</table>

Select all / a Project

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>19</td>
<td>In addition an item should be designed to able the user to select all Deliverables.</td>
</tr>
</tbody>
</table>

Select all / a Deliverable

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td>List of Tasks should be verified by the system based on the selected criteria in the former step.</td>
</tr>
<tr>
<td>21</td>
<td>An interface to display the Tasks should be presented for the user.</td>
</tr>
</tbody>
</table>

Table 3.7: View Task requirements.

Statistics

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>An interface to represent the statistics should be provided.</td>
</tr>
<tr>
<td>23</td>
<td>All statistics should be calculated for by the system.</td>
</tr>
</tbody>
</table>

Table 3.8: Statistic requirements.
4 Architecture and Design

General architecture of the Timelog system will be discussed at the start of this section. Web Service Architecture is presented in the next section. In Section 4.2 the most important controls to design the Android client will be represented.

4.1 Overall architecture

As discussed in Section 2.1 connecting to the database from the Android client directly is not a good solution. Direct connection does not provide any security mechanism. Web Service is an appropriate solution to guarantee the security for us. As it discussed, this Web Service can be a generic protocol for all mobile-users of the Timelog system.

A Restful Web Service can be a good solution for the Timelog system since this system does not include any enterprise scenarios. Security mechanism that can be provided by a SOAP approach is not necessary for the Timelog system. A Restful Web Service has less bandwidth usage in compare with SOAP. We can transfer JSON and Plain Text media type in the Restful Web Service. They can be parsed faster in comparison with XML and speed up our client. Cashing can also be performed in the Restful Web Service. In Figure 4.1 an overall architecture for the system has been represented. As you can see in this figure Web Service maps the data to MySQL server using Hibernate. In the Web Service data is serialized and desterilized to JSON format. Android App is using HTTP Client library to send and receive data in JSON format.

![Overall architecture of the system.](image)

The main Timelog system is a web based application that has been developed by Google Web Toolkit (GWT). It used the Hibernate Technology to connect to the database. The database of this system is MySQL Server. As a result we need to provide a mechanism in our Restful Web Service to connect to this database. Hibernate that has been discussed in Section 2.2, is a proper approach to connect to the database. It provides a mapping from
our object-oriented models to the database. These object-oriented models will be presented in the beginning of Section 4.1 as part of Web Service Architecture.

4.2 Web Service Architecture
One of the most important packages in the Web Service contains the object-oriented models for mapping to the database.

They have been concluded based on the requirements of the system presented in Section 3.2. These models are represented in Figure 4.2.

When the company receives a Project, managers need to break it down to smaller assignments. These assignments are Deliverable in the Timelog system. Every Project belongs to a Category. Categories includes one or several Activities such as understand the system, development, meeting, consulting and etc. Each Activity has a cost. Statistics in the Timelog system are based on the cost of different Activities in a Project.

![Figure 4.2: Hibernate object-oriented models for the Timelog.](image)

When a new Task wants to be started, user needs to specify the Project, Deliverable and the Activity at first, and then start and end time with a brief description about the Task should be entered by the user.

Another important package in the Web Service includes the controller classes to transfer the information between the Android client and database. They have been implemented via Jersey which is an implementation of JAX-RS Project.
The Java API for Restful Web Services (JAX-RS) project was started in 2007 and the first revision of it was released in October 2008 [36].

The goal of the project was to implement a mechanism to develop Restful Web Services in Java Environment. Intuitive way to develop a Restful Web Service is an important characteristic of this project. "Jersey is the reference implementation for JAX-RS [36]." It includes three major components:

- **Core Server**: We can implement the Rest Services via the libraries provided by this component.
- **Core Client**: We can test our Rest Services easily via the provided libraries by this component.
- **Integration**: An application that is developed by Jersey can integrate with a Java-Servlet such as Apache Tomcat.

Jersey can transfer different media type such as XML, JSON or Plain Text (String format). It provides some notations to implement Rest Services. Followings are a brief overview for these notations.

- @Path: It provides the URL address to transfer the information for instance it can be: http://localhost:8080/rest/timelog.
- @GET: It is used to retrieve the information and send them to HTTP Client.
- @POST: It is used to create a new data and sent it to HTTP Client.
- @PUT: It is used to modify the information and reply them to HTTP Client.
- @DELETE: It is used to remove information and reply to HTTP Client.
- @Consumes: It includes the content of Media Type that receives from client such as XML, JSON or Plain Text.
- @Produces: It includes the content of Media Type to respond to the client such as XML, JSON or Plain Text.

There are several kinds of client that can be developed to communicate with Jersey such as client with java.net.URL library, Apache HttpClient library and Jersey client (that provided by Client Core of JAX-RS).

Apache HTTPClient is the most common library to communicate with an Android application [37].

### 4.3 Android Design

Saving battery life is important for any mobile-user. People like the live wallpapers but they consume the battery life continuously [38]. As a result black background has been selected for the Android client of Timelog system in order to save the battery life. In the following of this section some of the building features that have been used in the development of our Android client will be presented.

#### 4.3.1 List View and Grid View

A very common pattern to present the information in Android applications is a list.
List View is the provided control by Android to display items in a list. Lists display items in a vertical arrangement. User can scroll through these items [39]. Grid list is an alternative for common list views. They are an appropriate view to show a list of images. They can be scrolled either vertically or horizontally. In Figure 4.3 a simple example for a List View and Grid View has been presented [40].

![Figure 4.3: List View in comparison to Grid View.](image)

To implement a List View programmatically, ArrayAdapter class can be extended. But to implement a Grid View BaseAdapter class should be extended [41].

### 4.3.2 Progress Dialog
Progress Dialog is another feature that has been used in the development of this application. As it discussed before Android client needs to fetch the information from the database via the Web Service. We do not know how long we have to wait every single time that data transfers from the Web Service. As a result showing a Progress Dialog can be a good idea until all data prepare to display. It is extended from Alert Dialog class that can present a progress animation in the form of a spinning wheel [42]. In Figure 4.4 an example for the Progress Dialog has been represented.

### 4.3.3 Action Bar
Another feature in the design of this application is Action Bar. It has been introduced as one the most important structural element in design of an Android application [43]. It can be used in most of activities in an Android application that a kind of functionality is needed such as search. It can also provide the user location in the application via navigation bar. In Figure 4.5 a view for an Action Bar has been displayed.
To use all features that have been designed for the Action Bar such as navigation mode and modify action bar styles the API level should be equal or greater than level 11. By the way in the last revision of Android in March 2012 the Action Bar compatibility version has been released [3]. It helps us to use the Action Bar with the older version. Followings are the main reasons to use an Action Bar in design of an Android application [43].

![Progress Dialog in Android](image)

**Figure 4.4: Progress Dialog in Android.**

![Action Bar Demo](image)

**Figure 4.5: Action Bar in Android [3].**

- Create an appropriate place for the logo of the application. It can also present a title for the activity. As a result it can be considered as a feature to provide an identity for the application.
- Provide a proper place for the most essential key actions such as search, refresh or create new. Right corner of the Action Bar is the location of these controls.

  Developers can declare an item in the option menu or as an action item in the Action Bar. It is important to make a proper choice. The followings are the important reasons to display the item on the Action Bar.

- An item that is frequently used such as Search in the items of a list. It is recommended to use Search button in the Action Bar since not all Android phones have a
hardware search button. It is also recommended to declare it as the first item in the action view [44]. It has been used in the Action Bar which is provided for the Timelog system.

- An item that is not frequently uses but it is an important item such as “Add network in Wi-Fi settings and Switch to camera in the Gallery application” [44].
- Typical items that is common in different Android applications such as refresh button. Refresh item is also used in the Timelog application.

It is recommended to use the minimum number of items in the Action Bar. It does not suppose to look like a desktop toolbar. Four items in an Action Bar is the maximum number that has been suggested. Developers need to consider the level of importance for the items in the Action bar. More important items should be declared at first.

A nice feature in the Action Bar is “ifRoom” value when an item is declared. If screen does not have enough space for an item in the Action Bar, it will appear in the overflow menu. The overflow menu is appeared when user click on the Menu hardware button. In the cases that an Android device does not provide a Menu hardware button, an additional button will be presented on the Action Bar. An example for the overflow menu in the Action Bar has been displayed in Figure 4.6. It is suggested to use Settings, Help, and Feedback actions in the overflow menu [44].

![Figure 4.6: Overflow menu on Action Bar in Android [44].](image)

### 4.3.4 Search Interface

Android helps developers to add search functionality to their application [45]. Two kind of search interface has been introduced by Android. They have same functionality with a little difference in UI and implementation.

- **Search Widget** is a kind of search view that can be displayed anywhere in an activity. It appears like a standard Edit Text in Android. Developers need to configure it and show the result of search with an appropriate way in an activity. It supports the search suggestion. It can also used on an Action Bar. By the way it can be used in the Android devices with API level 11 or greater. In Figure 4.7 a Search Widget has been presented.

![Figure 4.7: Search Widget on an Action Bar in Android [45].](image)
• **Search Dialog** is an extension of UI feature on the Android system. It always appears at the top of an activity. The Android system handles all events of this Search Dialog by returning the result of query to the activity. It contains the application icon on the left side. It can also support search suggestion for the user. Figure 4.8 is an example for the Search Dialog.

![Search Dialog in Android](image)

**Figure 4.8**: Search Dialog in Android [45].
5 Implementation

Implementation of every feature including a brief description and Web Service specification will be presented in this section.

5.1 Overall View

The overall implementation of the Android client is represented in Figure 5.1.

Figure 5.1: Overall view of the Android client.
If user has just one option to select Project, Deliverable or Activity, system jumps automatically to the next page as show in Figure 5.1 by arrows.

If connection to the server fails in every one of these activities, a proper error message pops up and system goes to Setting page automatically.

A progress Dialog is shown in all pages until system is getting ready to display Information. In continue implementation of every feature of the system will be discussed.

5.2 Login

![Login Screen](image)

**Figure 5.2: Login Screen.**

**Description**

- A default server address has been provided for the application.
- Failure: If User enters an incorrect Server Address, a message pops up regarding connection failed to the Server.
  If username and/or password are entered incorrectly, a message pops up regarding incorrect credential.
- Success: Home page of the application will be displayed (Section 5.2).

**Web Service API**

- Get the List of Users in the system from database.

```java
@GET
@Produces(MediaType.APPLICATION_JSON)
Session session = HibernateUtil.getSessionFactory().getCurrentSession();
session.beginTransaction();
```
Collection<User> user =
(Collection<User>)session.createQuery("from User").list();
session.getTransaction().commit();

Navigation
Action Bar
  • Refresh item → the default value for the server address will be represented.
Hardware Button
  • Home button → Main Screen Page of phone

5.3 Home

![Home Screen](image)

Figure 5.3: Home Screen.

Description
  • In all items except setting the next step is to select Project (Section 5.3). If user has just one Project, system jumps to next pages automatically.
  • The Logout item has been put on Option Menu (hardware menu button) as shown in Figure 5.3.

Navigation
Action Bar
  • Refresh item → the view of the list changes from List View to Grid View and vice versa. The last selected view is saved for the next use.
Hardware button
  • Home button → Main Screen Page of phone
• Menu button ➔ Logout from the system and go to the login page. (Section 5.1)

5.4 Select Project

![Select Project Screen](image)

Figure 5.4: Select Project Screen.

**Description**

- To add a new Task user must select one Project.
- If user wants to display Tasks or view statistics, he should also be able to select all Projects. It has been added as a special item at top of the list.
- If a Project is selected, user needs to specify the Deliverable in the next page (Section 5.4).
- If all Projects are selected, system displays the Tasks or statistics directly.

**Web Service API**

- Get the list of all Projects that user is involved in.

```java
@GET
@Consumes(MediaType.APPLICATION_JSON)
@Produces(MediaType.APPLICATION_JSON)
Projects = session.createQuery("select Project from Task as t where t.user.username = :name").setString("name", username).list();
```

**Special Requirement**

- The three recent used Projects should be found and displayed at the top of the list.
- Other items of the list should be sorted alphabetically (A to Z).
Navigation
Action Bar
- Logo item ➔ User returns back to the Home page of the application (Section 5.2).
- Search item ➔ the search interface comes up as shown in Figure 5.4.

Hardware button
- Home button ➔ Home Page of the application (Section 5.2)
- Search button ➔ the search interface comes up as shown in Figure 5.4.

5.5 Select Deliverable

![Select Deliverable Screen](image)

Figure 5.5: Select Deliverable Screen.

Description
- At the top of the page the selected Project is represented as shown in Figure 5.5.
- To add a new Task one Deliverable must be selected by the user.
- User is also able to select all Deliverable to view Tasks.
- If user wants to add a Task, the next step is to specify the Activity (section 5.5). Otherwise Tasks will be displayed (Section 5.7).
- A Deliverable can be as a child or parent. The parent Deliverables are distinguished by an icon and bold font as shown in Figure 5.5.

Web Service API
- Get the list of all Deliverables based on the selected Project.

```java
@GET
@Consumes(MediaType.APPLICATION_JSON)
@Produces(MediaType.APPLICATION_JSON)
```
Deliverables = (Collection<Deliverable>)
session.createQuery("from Deliverable as d where 
d.Project.id= :id").setLong("id", ProjectId).list();

Special Requirement
- The three recent used Deliverables by the user should be found and displayed at the top of the list.
- Other items of the list should be sorted first numerically and then alphabetically.

Navigation
Action Bar
- Logo item ➔ User returns back to the Home page of the application (Section 5.2).
- Search item ➔ the search interface comes up.

Hardware button
- Home button ➔ Home Page of the application (Section 5.2)
- Search button ➔ the search interface comes up.

5.6 Select Activity

Figure 5.6: Select Activity Screen.

Description
- At the top of the page the selected Project and Deliverable are represented as shown in Figure 5.6.
- User need to specify the Activity when he wants to add a new Task.

Web Service API
- Get the list of all Activities based on the category of selected Project.

```java
@GET
@Consumes(MediaType.APPLICATION_JSON)
@Produces(MediaType.APPLICATION_JSON)
activities = (Collection<Activity>)
session.createQuery("from Activity as a where " + 
"a.category.id=" + categoryId).list();
```

**Special Requirement**
- Items of the list should be sorted alphabetically (A to Z).

**Navigation**

**Action Bar**
- Logo item: User returns back to the Home page of the application (Section 5.2).
- Search item: The search interface comes up.

**Hardware button**
- Home button ➔ Home Page of the application (Section 5.2)
- Search button ➔ the search interface comes up.

**5.7 Add Task**

![Add Task Screen]

**Description**
- At the top of the page selected Project, Deliverable and Activity are represented as shown in Figure 5.7.
- The Default values have been provided for the date, start time and end time. Default date is today. Start time is the current time and end Time is a quarter later.
- User adds a new Task to the database by click at the Save button.

**Web Service API**
- A new Task is added to the database.

```java
@POST
@Consumes(MediaType.APPLICATION_JSON)
@Produces(MediaType.TEXT_PLAIN)
public Response addTask(Task Task) {
    Session session = HibernateUtil.getSessionFactory().getCurrentSession();
    session.beginTransaction();
    session.save(Task);
    session.getTransaction().commit();
    ...
}
```

**Navigation**

**Action Bar**
- Logo item → User returns back to the Home page of the application (Section 5.2).
- Refresh item → Date and Times are updated to the current time.

**Hardware button**
- Home button → Home Page of the application (Section 5.2)

**5.8 View Task**

**Description**
- This page is based on selected options in the former pages. It can show Tasks for all Projects, all Deliverables of a Project or specific Deliverable of a Project. In Figure 5.8, Tasks for all Deliverables of a Project are represented as an example.

**Web Service API**
- Get the list of Tasks for all Projects

```java
Tasks = (Collection<Task>)session.createQuery("from Task as t where t.user.username = :name")
    .setString("name", username).list();
```

- Get the list of Tasks for all Deliverables of a Project

```java
Tasks = (Collection<Task>)session.createQuery("from Task as t where t.user.username = :name" + " and t.Project.id = :id order by t.Deliverable.id")
    .setString("name", username).setLong("id", ProjectId).list();
```

- Get the list of Tasks for a specific Deliverable of a Project

```java
Tasks = (Collection<Task>)session.createQuery("from Task as t where t.user.username = :name" + " and t.Project.id = :id and t.Deliverable.id = :delivetable_id")
    .setString("name", username).setLong("id", ProjectId)
    .setLong("delivetable_id", DeliverableId).list();
```
Figure 5.8: View Task Screen.

Special Requirement
- To display Tasks as shown in Figure 5.8, list of Tasks should be sorted by Project, Deliverable and Activity respectively.

Navigation
Action Bar
- Logo item  User returns back to the Home page of the application (Section 5.2).
- Refresh item  if other clients of the system make any change in the database, they will be visible after refreshing.

Hardware button
- Home button  Home Page of the application (Section 5.2)

5.8 Statistic
Description
- Statistics can be displayed for a specific Project or all Projects as are shown in Figure 5.9 respectively.
- Statistics are calculated based on the cost of Activities. A Project belongs to a category that can have different kind of Activities.
- A different horizontal view is also provided for the statistics when user keeps the phone horizontally as shown in Figure 5.10.

Web Service API
- Get the list of Tasks for all Projects
Tasks = (Collection<Task>)session.createQuery("from Task as t where t.user.username = :name")
 .setString("name", username).list();

- Get the list of Tasks for a specific Project

Tasks = (Collection<Task>)session.createQuery("from Task as t where t.user.username = :name" + " and t.Project.id = :id order by t.Deliverable.id").setString("name", username).setLong("id", ProjectId).list();

Navigation
Action Bar

- Logo item ➔ User returns back to the Home page of the application (Section 5.2).
- Refresh item ➔ if other clients of the system make any change in the database, they will be visible after refreshing.

Hardware button

- Home button ➔ Home Page of the application (Section 5.2).

![Figure 5.9: Statistic Screen.](image-url)
<table>
<thead>
<tr>
<th>Deliverable</th>
<th>Actual(h)</th>
<th>Cost(SEK)</th>
<th>Value(SEK)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Analyze Vehicle)</td>
<td>20:55</td>
<td>4180.00</td>
<td>14421.00</td>
</tr>
<tr>
<td>19.8 - Add pie chart</td>
<td>01:45</td>
<td>350.00</td>
<td>1207.50</td>
</tr>
<tr>
<td>19.9 - Driver names below</td>
<td>03:15</td>
<td>650.00</td>
<td>2242.50</td>
</tr>
<tr>
<td>cm - Status and planning meetings with customer</td>
<td>14:20</td>
<td>2866.67</td>
<td>14190.00</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>109:46</strong></td>
<td><strong>21950.00</strong></td>
<td><strong>84107.50</strong></td>
</tr>
</tbody>
</table>

Figure 5.10: Horizontal Statistic Screen.
6 Case Study
In this section the improvement process of the Android application during the thesis will be presented. Most of the features of the system that has been presented in Section 5 have a kind of progress during the time. A view for these features of the web application in the browser of Android device will also be presented in this section.

The first change in the Android client was the background color of the pages. First the white background was selected but later we decide to use black color in order to save the battery life of the phone.

Action Bar compatibility version was introduced by the revision of Android 4.0 in March 2012. We decide to use Action Bar in our application after this release. Search interface was another feature that has been added to the application after Action Bar. Progress Dialog has been added to the application in order to improve the UI design.

6.1 Login, Home Page
We decided to make the Login and Home page as simple as possible. The small screen of smart phone devices is not good to show the Logo in big size. A small logo icon on the Action Bar which is suggested by Android developer GUI guidelines is a better option as shown in Figure 6.1. Server address was hard coded at first but in the final version user is able to modify the server address.

Display the username at the top of the Home page is not necessary. Logout should not display as a very small button. This view is more proper for a Web Page as shown in Figure 6.2. In final version we put the Logout item in the option menu as discussed in Section 5.2.

6.2 Select Project
At first we decide to use the Grid View control to display the Projects but the view looked not nice. List View is a better option since we want to show the Project in the sorting order. Also we used icon for every Project based on their category but finally we decide to
remove the icon. The improvement of this page is presented in Figure 6.3. The same progresses have been done for Select Deliverable and Activity pages.

![Figure 6.2: Home Page Progress.](image)

![Figure 6.3: Select Project Progress.](image)

**6.3 Add Task**

The improvement in Add Task Page is represented in Figure 6.4. The first attempt was more proper to edit the information. It was not a good design to add a Task in a small screen.

The second attempt was more proper for the small screen of the phone but there were some unnecessary stuff on the page such as big logo icon, back to menu button and End Date item. In the Final versions these extra items have been removed. Logo icon was placed on the Action Bar. Hardware Home button was overridden therefore we did not need any extra button in the page to return back to home page of the application. System could handle the End Date based on the entered data for start and end time, so it could be eliminated. The selected Project, Deliverable and Activity in the former pages could also be displayed at top of the page with removing the extra stuff.
6.4 View Task
The design progress to Display Tasks is represented in Figure 6.5.

The first attempt was proper for a Web Page but not a smart phone client. The next attempt was ok for the small screen but there was lots of free space that could be used. In the last version the Duration item is also added to the view since we try to use the free spaces in the best form.

6.5 Statistic
The design progress to Show Statistics is represented in Figure 6.6. The final version is more proper for an Android application. The free spaces are used in a better way. Also a horizontal view Designed for the statistic when user keeps the Android device horizontally.
6.6 Sorting

We need to sort the data to make our application more user-friendly. It is important to sort the data as much as possible in the Server-side because the memory of phone is limited and it can be time consuming to implement different sorting instructions there. We do not have this memory limitation problem in the server-side.

In the not final attempts we had some sorting operations on the client-side. Table 5.1 displays the improvement in the speed of application when we transfer all the sorting instructions to the server-side as much as possible. Sorting in the client-side can be worsened when the size of list become bigger.

The Size column in Table 5.1 displays the number of data which have been read from database. The Time is also calculated based on the average of three tests on each page. These tests have been done by an Android device with version 2.3.3 (API level 10) which is the most common Android phone nowadays as shown in Figure 2.6.

There is not much difference in the Select Activity and View Task pages as shown in Table 5.1 because we just implement one sorting operation in them but when we implement two different sorting operations the difference become considerable.

<table>
<thead>
<tr>
<th>Page</th>
<th>Size</th>
<th>Sort</th>
<th>Client-side (s)</th>
<th>Server-side (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Project</td>
<td>11</td>
<td>1 - Find three recent used Projects 2 - Sort other elements alphabetically</td>
<td>6.47</td>
<td>3.17</td>
</tr>
<tr>
<td>(Section 5.3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Select Project</td>
<td>18</td>
<td>1 - Find three recent used Projects 2 - Sort other elements alphabetically</td>
<td>10.24</td>
<td>3.34</td>
</tr>
<tr>
<td>(Section 5.3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Select Deliverable</td>
<td>8</td>
<td>1 - Find three recent used Deliverables 2 - Sort other elements numerically</td>
<td>4.87</td>
<td>2.83</td>
</tr>
<tr>
<td>(Section 5.4)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Select Deliverable</td>
<td>195</td>
<td>1 - Find three recent used Deliverables 2 - Sort other elements numerically</td>
<td>50.34</td>
<td>8.64</td>
</tr>
<tr>
<td>(Section 5.4)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Select Activity</td>
<td>31</td>
<td>1 - Sort all elements alphabetically</td>
<td>3.20</td>
<td>3.14</td>
</tr>
<tr>
<td>(Section 5.5)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 6.1: Compare sorting in the Server-side and Client-side.

<table>
<thead>
<tr>
<th>View Task (Section 5.7)</th>
<th>72</th>
<th>1 - Sort by Deliverable and Activity respectively</th>
<th>5.25</th>
<th>4.83</th>
</tr>
</thead>
<tbody>
<tr>
<td>View Task (Section 5.7)</td>
<td>166</td>
<td>1- Sort by Deliverable and Activity respectively</td>
<td>8.18</td>
<td>7.37</td>
</tr>
<tr>
<td>View Task (Section 5.7)</td>
<td>132</td>
<td>1- Sort by Project, Deliverable and Activity respectively</td>
<td>7.34</td>
<td>6.37</td>
</tr>
<tr>
<td>View Task (Section 5.7)</td>
<td>366</td>
<td>1- Sort by Project, Deliverable and Activity respectively</td>
<td>17.43</td>
<td>15.68</td>
</tr>
</tbody>
</table>
7 Conclusion and Future Work

In this section we will summarize how the problem has been solved and the goals have been achieved. Some suggestion for the future work has also been provided in the second section.

7.1 Conclusion

Small screen of Android Phones is not capable of displaying different functionalities of the Timelog web application. An Android client could be a proper solution to help the users of Timelog system when they want to access to the system with their Android devices. We decide to create a mechanism to Login, Add Task, View Task and display Statistic for the Android client as it has been discussed in Section 3.

To create a connection between database of Timelog system and Android phone, a Web Service has been designed. It has been discussed in Section 4.1.

Create a user-friendly GUI for this client based on the latest guidelines of Android 4.0 was another sub goal. Section 4.2 includes the features that have been used in the development of this client to meet this goal. Implementation of each feature of the system has been discussed in detail in Section 5.

The requirement regarding Login page in Section 3.1.1 is fulfilled and described in Section 5.2.

A mechanism to add a new Task in Section 3.1.2 is defined. The implementation of this requirement is described in Section 5.4 (select a Project), Section 5.5 (select a Deliverable), Section 5.6 (select an Activity) and finally add the Task in Section 5.7.

Section 3.1.3 presented the requirements to display Tasks. They have been implemented and described in Section 5.4, 5.5 and 5.8.

We also have a mechanism to display statistics. The requirements are discussed in Section 3.1.4. They have been implemented in Section 5.4 in order to select a Project and Section 5.9 to show Statistics in vertical and horizontal view.

At the moment users of Timelog system can be accessed to the system using our native Android App. As a result we can claim that the requirements in Section 3 have been fulfilled and the main goal that was an Android client for the Timelog system is accomplished and we have solved the problem.

7.2 Future Work

Other smart phone clients can also be developed for this system since the Web Service has been developed and they can use the same Web Service. iPhone client can be more preferable. It had a very good progress in market recently. Figure 7.1 which is the latest research from Nielsen Company and released on 18 January 2012 displays this progress. Windows phone, BlackBerry or other clients can be developed later.
Figure 7.1: Operating system popularity for the most well-known smart phones [2].

A mechanism to edit the data has been developed in the first attempt that has been presented in Section 6. It can be improved to the latest Android UI guidelines and used in the next version of the system. It is suggested to use the contextual Action Bar for API level 11 or greater as shown in Figure 7.2.

Figure 7.2: Floating context menu and the Contextual Action Bar [46].

Fragment as shown in Figure 7.3 and Search Widget are two of the nice features introduced by API level 11. They can be used in the design of Android client for devices with this API level or greater.

The Timelog web application includes other functionalities that can be added to the client.

- A simpler mechanism to add a new Task.
- View Tasks and Statistics for a specific month, week and date.
- A mechanism to cut down a Project to Deliverables.
- Create a new Project in the system.
Figure 7.3: Fragment in Android [47].
References


