Degree project

VPS Administration on Android OS

Author: Tahereh Javanmardi
& Negin Malekzadeh
Date: 2012-03-29
Subject: Software Technology
Level: Master
Course code: 5DV00E
Abstract
These days using smart phones to access the internet started taking its form and eventually being a part of our life. Increasing number of people is searching for information and resources on the Internet with their mobiles. But the mobile phones have much smaller screen space. As these websites are designed for PC it is not easy to access them from mobile phones.

To solve this problem we should develop a client side android application and a server side plugin to cut down on the amount of information that can be included in the mobile pages.

The result is a user friendly application designed for mobile devices screen that let users to access the website without using the web browser.

Keywords:

Android, Webservice, REST, Jersey, WebHarvest, MySQL, Emulator
Acknowledgments
We would like to express our gratitude to our supervisor, Dr. Rüdiger Lincke for his constant encouragement, continuous support and constructive comments throughout this work.

We would also want to appreciate our families for the support they provided us throughout our entire life and particular.
Contents

1 Introduction.................................................................................................................. 1
  1.1 Problem ................................................................................................................... 1
  1.2 Motivation................................................................................................................ 1
  1.3 Goals and Criteria................................................................................................. 1
  1.4 Overview................................................................................................................ 1
2 Background ................................................................................................................ 3
  2.1 Selection of Tools .................................................................................................. 3
    2.1.1 Mobile Operating System................................................................................. 3
    2.1.2 About Android................................................................................................. 4
  2.2 Web Service ......................................................................................................... 8
    2.2.1 Selection of Web Service .............................................................................. 8
    2.2.2 About REST and RESTful Web Services.................................................... 9
  2.3 Web Scrapper Tool ............................................................................................... 11
    2.3.1 Selection of Web Scraper ............................................................................. 11
    2.3.2 About WebHarvest......................................................................................... 11
3 Features and Requirements ...................................................................................... 15
  3.1 Features................................................................................................................. 15
  3.2 Features' Tables and Fulfilled Requirements ...................................................... 17
    3.2.1 Login.............................................................................................................. 17
    3.2.2 Domain Information....................................................................................... 17
    3.2.3 Email Information.......................................................................................... 18
    3.2.4 Customer Information..................................................................................... 18
    3.2.5 Invoicing........................................................................................................ 19
    3.2.6 Payment History............................................................................................. 19
    3.2.7 Feature Invoices............................................................................................. 20
    3.2.8 Settings.......................................................................................................... 20
    3.2.9 Invoices......................................................................................................... 21
    3.2.10 Server Cloud ............................................................................................... 21
    3.2.11 IpAddresses................................................................................................. 22
    3.2.12 PTR............................................................................................................... 22
    3.2.13 Service Information ..................................................................................... 23
    3.2.14 Version History ......................................................................................... 23
    3.2.15 Service Status ............................................................................................. 24
    3.2.16 Logout.......................................................................................................... 24
List of Figure

Figure 2.1: Android OS Architecture.......................................................... 4
Figure 2.2: GridView ................................................................................. 5
Figure 2.3: Activity LifeCycle...................................................................... 7
Figure 2.4: RESTfull Web Service.................................................................. 9
Figure 2.5: One Pipeline Execution of WebHarvest....................................... 12
Figure 2.6: WebHarvest GUI......................................................................... 13
Figure 3.1: Use Case Diagram...................................................................... 16
Figure 4.1: Whole Structure of the System ................................................... 25
Figure 4.2: Client Side Class Diagram........................................................... 26
Figure 4.3: Server Side Class Diagram........................................................... 27
Figure 5.1: Login Page View in normal mobile browsers............................... 28
Figure 5.2: Login Page View using Android application ............................... 29
Figure 5.3: Notification. .............................................................................. 29
Figure 5.4: First Page View in normal mobile browsers. ............................... 30
Figure 5.5: First Page View using Android application................................. 31
Figure 5.6: Sample of Application Page View............................................... 33
Figure 5.7: Create Dynamic Project............................................................... 34
Figure 5.8: Database Structure..................................................................... 36
List of Tables

Table 2.1: Content Provider ........................................................................................................ 6
Table 2.2: JAX-RS Annotations ................................................................................................. 10
Table 3.1: Login Use Case Description ..................................................................................... 17
Table 3.2: Domain Information Use Case Description .............................................................. 17
Table 3.3: Email Information Use Case Description ................................................................. 18
Table 3.4: Customer Information Use Case Description ........................................................... 18
Table 3.5: Invoicing Use Case Description ................................................................................. 19
Table 3.6: Payment History Use Case Description .................................................................... 19
Table 3.7: Future Invoices Use Case Description ...................................................................... 20
Table 3.8: Settings Use Case Description .................................................................................. 20
Table 3.9: Invoices Use Case Description .................................................................................. 21
Table 3.10: Server Cloud Use Case Description ........................................................................ 21
Table 3.11: IpAddresses Use Case Description ........................................................................... 22
Table 3.12: PTR Use Case Description ....................................................................................... 22
Table 3.13: Service Information Use Case Description ............................................................. 23
Table 3.14: Version History Use Case Description ..................................................................... 23
Table 3.15: Service Status Use Case Description ...................................................................... 24
Table 3.16: Logout Use Case Description .................................................................................... 24
List of Charts

Chart 2.1: Share of worldwide 2011 Q2 smartphone sales to end users by operating system

3
1 Introduction

GleSYS is a company which provides virtual private servers. It was founded 1999 and today they are providing Internet solutions to customers all over the world. They started with web servers colocated in other facilities but since year 2000 they have built their own data center in their own facilities [1]. With their third generation virtual servers customers are in total control. They provide them with the tools to easily create new servers, upgrade and downgrade as well as clone servers instantly by clicking on control panel tab in GleSYS website which send the user to administration pages. Unlike traditional servers it is fast and easy to scale virtual servers with more memory, CPU or disk space. Even for a short amount of time to handle a temporary spike. GleSYS control panel makes this very easy and user has full control of the cost [2].

Due to advanced technology on smart phones and the fact that they are easier to move and occupy less space comparing to notebooks and laptops GleSYS users are becoming more interested in using their phones to access the information of administration website.

1.1 Problem

Administrators need to be carefully aware of the status of their servers. But they cannot be always at their computers. In comparison to laptops and notebooks, mobile smartphones are very easy to carry around and allow them to access the data from everywhere. As GleSYS administration website is designed for devices with big screens when it comes to smart phones with small screens it is very difficult to read contents until users zoom in to the part of the screen they want to view. Zooming in and out is not easy to do on all phones and can make some difficulties for users.

1.2 Motivation

As smartphones are becoming more and more popular and the usage of mobile internet is increasing day by day it is becoming more important and necessary for companies to offer their clients an alternative to their standard website that is designed to be more user friendly for mobile devices.

GleSYS is one of these companies and they are considering creating this alternative for their administration pages to provide their users more enjoyable and convenient access through mobile phones.

1.3 Goal and Criteria

Our goal is to make a possible and easy way for administrators to login to the website and have access to all features available in website using their smartphones anytime and everywhere without the need of zooming in or out for accessing contents. For this to be achieved we need to develop a mobile version of GleSYS administration website which shows the information to users on demand which is designed to look more attractive on mobile devices. In order to develop this application we create a web service. Server side saves website's information that has been requested in database and sends it to client.

The goal is reached if administrators accomplish getting requested data through their smartphones.

1.4 Overview

Chapter 2 describes special knowledge required for this project and discusses about the selection criteria and describes why special frameworks, clients and tools have been chosen. Chapter 3 defines the features and requirements of the system which can solve
the problem. Chapter 4 describes the overview of the solution for implementing the requirements and discusses about the project overall architecture and design. Chapter 5 presents both the server side module and the client implementation. Chapter 6 is the proof of concept and shows the test instances and cases attempted to assess the goal's achievement. The final chapter evaluates if the goal criteria has been met and also suggests some directions for the future work.
2 Background

In this chapter, we discuss special tools and programs which are used in this project and also the reason that we choose them. In addition, we will explain basic information about each of them.

2.1 Selection of Tools
In this section, we discuss mobile operating systems and talk about basics of Android programming language.

2.1.1 Mobile Operating System
There are several operating systems that are used for smartphones such as Android OS, Symbian, iOS, BlackBerry, etc. Android is derived from Linux and iOS is from the BSD and NeXTSTEP operating systems which all are related to Unix [3].

![Chart 2.1: Share of worldwide 2011 Q2 smartphone sales to end users by operating system [3].](image)

According to Chart 2.1 and also other data from different sources such as NPD Group, Wikimedia, Nielsen and so on, Android OS has had the highest percentage of smartphone sales in 2011. This means that Android is more popular in comparison to other operating systems. Android is growing in huge numbers that can be seen in most markets everywhere.

As Symbian is not as user friendly as Android and iOS we illuminate Symbian from our alternatives.
Moreover, programming language for Android OS is Java while the programming language for iPhone OS is Objective-C. Java programming is a common language used by developers with a large community and easy to compile on different operating systems using development environment such as Eclipse IDE. However, the most reliable way to compile Objective-C is using Mac software development tools such as Xcod on Mac computer.

Other plus points of Android OS are the alternative to change themes and custom UI's to the OS and also free development SDK.

In consideration of mentioned aspects and the fact that we did not have Mac computer, we choose Android OS.

2.1.2 About Android

Android operating system has different components that are shown in Figure 2.1.

![Android OS architecture](image-url)

Figure 2.1: Android OS architecture [7].
Below we will describe different components of Android OS in detail [7]:

- **Applications**
  Android has several core applications which are written in Java. Some examples of these core applications are client, maps, email, etc. [7].

- **Application Framework**
  Android gives the ability of developing rich applications and facilities such as access location information, run background services, device hardware, add notification to the status bar and so on to developers using open development platform.

  There are set of services and systems that are base of all applications such as:
  - An application can be created using different kinds of Views such as text boxes, lists, buttons, embedded web browser, etc. For example in Figure 2.2 an example of Grid View is shown which is scrollable and shows items in two-dimensional grid.

  ![Figure 2.2: GridView](image)

  - The only storage area that all Android packages can access is Content Provider which saves and gets data and makes it accessible to all applications. Content Provider stores data into table on a database model that like all other databases' table consists of column and row. Each row is a record and each column contains data with specific type. Table 2.1 shows how information about people and their phone numbers might look in table of Content Provider.
<table>
<thead>
<tr>
<th>_ID</th>
<th>NUMBER</th>
<th>NUMBER_KEY</th>
<th>LABEL</th>
<th>NAME</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>(425) 555</td>
<td>425 555 6677</td>
<td>Kirkland office</td>
<td>Bully Pulpit</td>
<td>TYPE_WORK</td>
</tr>
<tr>
<td>44</td>
<td>(212) 555-</td>
<td>212 555 1234</td>
<td>NY apartment</td>
<td>Alan Vain</td>
<td>TYPE_HOME</td>
</tr>
<tr>
<td>45</td>
<td>(212) 555-</td>
<td>212 555 6657</td>
<td>Downtown office</td>
<td>Alan Vain</td>
<td>TYPE_MOBILE</td>
</tr>
<tr>
<td>53</td>
<td>201.555.4433</td>
<td>201 555 4433</td>
<td>Love Nest</td>
<td>Rex Cars</td>
<td>TYPE_HOME</td>
</tr>
</tbody>
</table>

Table 2.1 Content Providers Example [9].

- A Resource Manager contains all non-code resources that are used in the Android Project. For instance, graphics, layout files, etc.
- A Notification Manager provides the facility for applications to display an alert in the status bar which user can get access to the application by sliding down the status bar and tapping on the icon of the specific application.
- An Activity Manager that its main task is managing the lifecycle of applications and providing a common navigation back stack. The management of system's Activities works like activity stack. The running activity is the newly started activity that is located on the top of the stack. And while the new activity is running the previous activity will not come to the foreground. The state paths of an Activity are shown in Figure 2.3. The square rectangles show callback methods and the states that the Activity can be in are colored ovals [10].
Figure 2.3 Activity Lifecycle [10].
o Libraries
Android contains several libraries based on C/C++ that different components of the Android system use these libraries. Developers can access these libraries through Android application framework. Some examples of these libraries are as below:

- **System C library**: This is a Berkeley Software Distribution-derived implementation of the standard C system library which is for embedded Linux-based devices.
- **Media Libraries**: These libraries are to support audio and video formats in playback and recording part. Also supporting static image files such as MP3, MPEG4, JPG, etc.
- **Surface Manager**: Makes it possible for multiple applications to access to the display subsystem and graphic layers.
- **LibWebCore**: It is a web browser engine that supports the Android browser and an embeddable web view.
- **SGL**: A graphics engine based on 2D.
- **3D libraries**: It is an implementation based on OpenGL ES 1.0 APIs.
- **FreeType**: It provides bitmap and vector font.
- **SQLite**: It is a database engine that is accessible from all applications [7].

o Android Runtime
Most of the functionalities that are available in the core library of Java are provided by Android core libraries. A device can run multiple VMs using Dalvik virtual Machine [7].

o Linux Kernel
This is an abstraction layer between the hardware and the rest of the software stack and it includes some core system services such as process management, driver model, security memory management, etc. [7].

2.2 Web Service
In the first part of this section, we discuss different web services and give the reason for choosing RESTful web service and in the second part we speak about RESTful web service.

2.2.1 Selection of Web Service
To develop web services there are two alternatives: the traditional, standard-based approach (SOAP) and conceptually simpler and newer one (REST).

SOAP in comparison with REST is conceptually more difficult, more "heavy-weight", more verbose, harder to develop and requires tools.

RESTful stands for Representational State Transfer web services. The functionality for RESTful web services is provided by JAX-RS.

In Java EE 6, JAX-RS provides the functionality for Representational State Transfer (RESTful) web services. RESTful web services do not need XML messages or WSDL service–API definitions. Moreover it has a better integration with HTTP comparing with SOAP-based services. Jersey is an open source reference implementation for creating RESTful web services and also provides an API for developers.

In the JAX-RS specification here are annotations which are supported by Jersey. So building RESTful web services with Java and the Java Virtual Machine (JVM) becomes simpler for developers. It is inexpensive to develop RESTful web services as they have a lightweight infrastructure which lets services to be built with minimal tooling and also they use W3C and Internet Engineering Task Force (IETF) standards [12].
For mentioned reasons we have developed RESTful web services in Java with the JAX-RS reference implementation Jersey.

### 2.2.2 About REST and RESTful Web Services

REST is HTTP protocol and web-standards based architectural style. In this architecture everything is a resource. This architecture has a REST server and a REST client. REST server makes resources accessible and REST client accesses and modifies the REST resources.

REST is based on web-standards and the HTTP protocol so resources have to support the HTTP common operations and are identified by global ID's (URIs).

Resources in REST can have different representations such as json, text, XML and so on. Through HTTP protocol client can send request for specific representations [12].

REST uses following HTTP standards methods:
- GET: It is for reading the resources without any modification in resources.
- PUT: It is for creating a new resource.
- POST: It is for updating or creating resource.
- DELETE: It is for Removing the resource.

A RESTful web services represents the base URI for the services and supports HTTP standards methods such as POST, GET, PUT, DELETE. Different kinds of types like XML, TEXT and JSON are also supported by RESTful web services. In Figure 2.4 an example of RESTful web services is shown [13].

![Diagram of RESTful Web Services](http://your_domain:port/display-name/url-pattern/path_from_rest_class)

**Figure 2.4** Restful Web Services [13].

Java defines standard REST support via JAX-RS that uses annotations. "web.xml" is used to register a provided servlet by Jersey and to define REST web application's path. This servlet's base URL is:

```
http://your_domain:port/display-name/url-pattern/path_from_rest_class
```

By servlet, incoming HTTP requests are analyzed and the correct class and method will be found based on their annotation for responding to requests. In Table 2.2 some important annotations in JAX-RS are shown [12].
<table>
<thead>
<tr>
<th>Annotation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>@PATH(your_path)</td>
<td>This annotation sets the path to base URL + /path.</td>
</tr>
<tr>
<td>@POST</td>
<td>The method followed by this annotation answers to a HTTP POST request.</td>
</tr>
<tr>
<td>@GET</td>
<td>The method followed by this annotation answers to a HTTP GET request.</td>
</tr>
<tr>
<td>@PUT</td>
<td>The method followed by this annotation answers to a HTTP PUT request.</td>
</tr>
<tr>
<td>@DELETE</td>
<td>The method followed by this annotation answers to a HTTP DELETE request.</td>
</tr>
<tr>
<td>@Produces(MediaType.TEXT_PLAIN [, more-types ])</td>
<td>@Produces defines the received MIME type which is annotated with @GET.</td>
</tr>
<tr>
<td>@Consumes( type [, more-types ] )</td>
<td>@Consumes defines consumed MIME type by the followe method.</td>
</tr>
<tr>
<td>@PathParam</td>
<td>By this annotation the values from the URL will be injrcted into a method parameter.</td>
</tr>
</tbody>
</table>

Table 2.2 JAX-RS annotations [12].

The complete path to a resource is based on the base URL and the @PATH annotation in the class.

```
http://your_domain:port/display-name/url-pattern/path_from_rest_class
```

For this specification Jersey is the reference implementation which includes REST server and REST client. REST client connects with server using provided library by core client.

REST web application has two parts that are data classes and services which are located separately in different packages. "web.xml" instructs Jersey servlet in order to scan specific packages for resources [12].
2.3 Web Scraper tool
In the first part of this section we give the reason for choosing WebHarvest and in the second part we explain the basics of the WebHarvest.

2.3.1 Selection of Web Scraper
There are different available tools for extracting data from websites such as Automation Anywhere, Outwit, Greasemonkey, Web-Harvest and so on. For this project we were looking for a tool that can be used from java code, be able to get data from websites using different formats, ability to save data in database, easy to learn and use and preferably free.

Between tools that we have tried, Web-Harvest was the best option that could cover our desires. If developer is familiar with HTML, JavaScript and XML it is easy to learn Web-Harvest. We can use Web-Harvest in many platforms (Windows, Linux) without any issue because it is written in Java. Every webpage has a different format which makes the data extraction very difficult. But with Web-Harvest extracting data from many websites using different format is easier as it uses Xpath and Xquery. Also it provides a user friendly GUI for developing, debugging and testing the code. And it is able to save data as XML or in database.

2.3.2 About WebHarvest
Web-Harvest is an Open Source Web Data Scraper tool. WEB-Harvest is written in Java programming language and it uses good techniques and technologies for text/xml manipulation like XQuery and Regular Expressions in order to extract useful data from desired web pages.

Different Web sites and Web pages have mixed contents and they are created with different logics. So we need to understand these logics in order to extract the demanded data. In Web-Harvest through XML-based configuration files all extraction procedures are user-defined. In order to rich the desired result; these XML-based configuration files describe the sequence of process. Each process is executed in the pipeline form which means that the output of one processor execution is input to another one. A simple configuration fragment is shown as follow:

```xml
<xpath expression="/a[@shape='triangle']/@href">
<html-to-xml>
<http url="http://www.siteName.com/"/>
</html-to-xml>
</xpath>
```

The following steps are the execution process of the mentioned configuration fragment in last page:

1. The specified URL's content will be downloaded by http processor.

2. html-to-xml processor produces XHTML content.

3. xpath processor searches specific links in XHTML and returns URL sequence.

Web-Harvest does not propose a new method for extracting data but it offers an easy way by combining existing ones. It represents several processors for data handling and control flow. Processors are considered as functions. They have zero or more input parameters and after the execution they return the result. Web-Harvest uses variable context to store named variables for easier manipulation and data reuse. The execution of one pipeline is shown in Figure 2.5 [14].
For variable manipulation Web-Harvest has several processors such as looping, file operations, exception handling, functions, etc. [15].

Web-Harvest can be used as GUI application, command line utility and from the Java code.

- Development IDE usage

It's very simple to create and test XML configuration by using Web-Harvest IDE. By double clicking on webharvest_all_xx.jar the IDE can start easily. The other way to start the IDE is typing "java -jar webharvest_all_xx.jar" in command line. IDE represents multiple document interface, hierarchical view of configuration processors, property-viewer of each executing processor and log area [16]. In Figure 2.6 screenshot of the IDE is shown.

Figure 2.5 One Pipeline Execution of WebHarvest [14].
• Command line usage

Syntax for command line use is the following [16]:

```
java -jar webharvest_all_XX.jar [-h] config=<path>
    [workdir=<path>] [debug=yes|no]
    [proxyhost=<proxy server> [proxyport=<proxy server port>]]
    [proxyuser=<proxy username> [proxypassword=<proxy password>]]
    [proxyhost=<NT host name>]
    [proxynetdomain=<NT domain name>]
    [loglevel=<level>]
    [logpropsfile=<path>]
    [plugins=<list of plugin classes>]
    [#var1=<value1> [#var2=<value2>...]]
```
• Java code usage

First it is necessary to import following Web-Harvest classes [16]:

```java
import org.webharvest.definition.ScraperConfiguration
import org.webharvest.runtime.Scraper
```

Then it's needed to create instance of ScraperConfiguration using path to the specified configuration file and also create a Scraper instance with specified working folder.

```java
ScraperConfiguration config = new ScraperConfiguration("c:/folderName/fileName.xml");
Scraper scraper = new Scraper(config, "c:/folderName/");
```

Add user object instances to the variable context if it is necessary.

```java
scraper.addVariableToContext("Var1", obj1);
scraper.addVariableToContext("Var2", obj2);
```

Set debugging if it is necessary.

```java
Scraper.setDebug(true);
```

Specify proxy server if it is necessary.

```java
scraper.getHttpClientManager().setHttpProxy("proxy.wh", 3128);
```

Specify proxy server if it is necessary. myNTHost and myNTDomain can be null.

```java
scraper.getHttpClientManager().setHttpProxyCredentials(myUsername, myPassword, myNTHost, myNTDomain);
```

Start configuration execution:

```java
scraper.execute();
```
3 Features and Requirements

In this chapter, we show the features of our system by use case diagram and the requirements related to those features.

3.1 Features
Our system features for accessing GleSYS web site using mobile devices are as below.

- **Login**: Access to the website data and features
- **Domain Information**: Access to Domain Information webpage
- **Email Information**: Access to Email Information webpage
- **Customer Information**: Access to Customer Information webpage
- **Invoicing**: Access to Invoicing webpage
- **Payment History**: Access to Payment History webpage
- **Future Invoices**: Access to Future Invoices webpage
- **Setting**: Access to Setting webpage
- **Invoices**: Access to Invoices webpage
- **Server Cloud**: Access to Server Cloud webpage
- **Ip addresses**: Access to IpAddresses webpage
- **PTR**: Access to PTR webpage
- **Service Information**: Access to Service Information webpage
- **Version History**: Access to Version History webpage
- **Service Status**: Access to Service Status webpage
- **Logout**: Disconnect from website

The use case diagram of these features is shown in Figure 3.1.
Figure 3.1 Use Case Diagram.
3.2 Features' Table and Fulfilled Requirements
In this section, we show all the future's table and describe the requirements for each feature.

3.2.1 Login

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Login</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brief Description</td>
<td>The login process gets username and password that will be checked with the system.</td>
</tr>
<tr>
<td>Flow Of Events</td>
<td>1. User inserts username and password. 2. User clicks Login button.</td>
</tr>
<tr>
<td>Actors</td>
<td>User</td>
</tr>
<tr>
<td>Preconditions</td>
<td>The User’s account should exist in system.</td>
</tr>
<tr>
<td>Post conditions</td>
<td>User will be able to access information in the website.</td>
</tr>
<tr>
<td>Exception conditions</td>
<td>If the username and/or password are not correct and if the website's server is down the user will be warned that the login process can't be done.</td>
</tr>
</tbody>
</table>

Table 3.1 Login Use Case Description.

Requirements:
- Client has to be connected to server.
- User Interface for client to insert username and password.
- Client has to send username and password to server to be checked.
- After checking username and password server sends the response to client.

3.2.2 Domain Information

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Domain Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brief Description</td>
<td>User can access the data of Domain Information page.</td>
</tr>
<tr>
<td>Flow Of Events</td>
<td>1. User clicks on a Domain Information button. 2. System displays contents of the page.</td>
</tr>
<tr>
<td>Actors</td>
<td>User</td>
</tr>
<tr>
<td>Preconditions</td>
<td>None</td>
</tr>
<tr>
<td>Post conditions</td>
<td>None</td>
</tr>
</tbody>
</table>

Table 3.2 Domain Information Use Case Description.

Requirements:
- Client has to be connected to server.
- Client should send Domain Information Identification to server.
- Server should get the Domain Information's data from related webpage.
- Server should send provided data to client.
3.2.3 Email Information

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Email Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brief Description</td>
<td>User can access the data of Email Information page.</td>
</tr>
</tbody>
</table>
| Flow Of Events      | 1. User clicks on a Email Information button. 
|                     | 2. System displays contents of the page. |
| Actors              | User             |
| Preconditions       | None             |
| Post conditions     | None             |

Table 3.3 Email Information Use Case Description.

Requirements:
- Client has to be connected to server.
- Client should send Email Information Identification to server
- Server should get the Email Information's data from related webpage.
- Server should send provided data to client.

3.2.4 Customer Information

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Customer Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brief Description</td>
<td>User can access the data of Customer Information page.</td>
</tr>
<tr>
<td></td>
<td>2. System displays contents of the page.</td>
</tr>
<tr>
<td>Actors</td>
<td>User</td>
</tr>
<tr>
<td>Preconditions</td>
<td>None</td>
</tr>
<tr>
<td>Post conditions</td>
<td>None</td>
</tr>
</tbody>
</table>

Table 3.4 Customer Information Use Case Description.

Requirements:
- Client has to be connected to server.
- Client should send Customer Information Identification to server
- Server should get the Customer Information's data from related webpage.
- Server should send provided data to client.
3.2.5 Invoicing

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Invoicing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brief Description</td>
<td>System displays the context menu which shows sub menus of invoicing webpage.</td>
</tr>
</tbody>
</table>
2. System displays context menu. |
| Actors | User |
| Preconditions | None |
| Post conditions | User will be able to select the desired menu. |

Table 3.5 Invoicing Use Case Description.

Requirements:
- Client has to be connected to server.
- Client should be able to display the context menu which consists of sub tabs of Invoicing's webpage.

3.2.6 Payment History

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Payment History</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brief Description</td>
<td>User can access the data of Payment History page.</td>
</tr>
</tbody>
</table>
| Flow Of Events | 1. User clicks on the desired menu in context menu.  
2. System displays the content of the page. |
| Actors | User |
| Preconditions | Invoicing button should have been chosen by user. |
| Post conditions | None |

Table 3.6 Payment History Use Case Description.

Requirements:
- Client has to be connected to the server.
- Client should send Payment History Identification to server.
- Server should get the Payment History's data from related webpage.
- Server should send provided data to client.
3.2.7 Future Invoices

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Future Invoices</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Brief Description</strong></td>
<td>User can access the data of Future Invoices page.</td>
</tr>
<tr>
<td><strong>Flow Of Events</strong></td>
<td>1. User clicks on the desired menu in context menu. 2. System displays the content of the page.</td>
</tr>
<tr>
<td><strong>Actors</strong></td>
<td>User</td>
</tr>
<tr>
<td><strong>Preconditions</strong></td>
<td>Invoicing button should have been chosen by user.</td>
</tr>
<tr>
<td><strong>Post conditions</strong></td>
<td>None</td>
</tr>
</tbody>
</table>

Table 3.7 Future Invoices Use Case Description.

**Requirements:**
- Client has to be connected to server.
- Client should send Future Invoices Identification to server
- Server should get the Future Invoices data from related webpage.
- Server should send provided data to client.

3.2.8 Settings

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Settings</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Brief Description</strong></td>
<td>User can access the data of Settings page.</td>
</tr>
<tr>
<td><strong>Flow Of Events</strong></td>
<td>1. User clicks on the desired menu in context menu. 2. System displays the content of the page.</td>
</tr>
<tr>
<td><strong>Actors</strong></td>
<td>User</td>
</tr>
<tr>
<td><strong>Preconditions</strong></td>
<td>Invoicing button should have been chosen by user.</td>
</tr>
<tr>
<td><strong>Post conditions</strong></td>
<td>None</td>
</tr>
</tbody>
</table>

Table 3.8 Settings Use Case Description.

**Requirements:**
- Client has to be connected to server.
- Client should send Settings Identification to server
- Server should get the Settings data from related webpage.
- Server should send provided data to client.
3.2.9 Invoices

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Invoices</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brief Description</td>
<td>User can access the data of Invoices page.</td>
</tr>
</tbody>
</table>
| Flow Of Events | 1. User clicks on the desired menu in context menu.  
  2. System displays the content of the page. |
| Actors | User |
| Preconditions | Invoicing button should have been chosen by user. |
| Post conditions | None |

Table 3.9 Invoices Use Case Description.

Requirements:
- Client has to be connected to server.
- Client should send Invoices Identification to server.
- Server should get the Invoices data from related webpage.
- Server should send provided data to client.

3.2.10 Server Cloud

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Server Cloud</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brief Description</td>
<td>System displays the context menu which shows sub menus of Server Cloud webpage.</td>
</tr>
</tbody>
</table>
  2. System displays context menu. |
| Actors | User |
| Preconditions | None |
| Post conditions | User will be able to select the desired menu. |

Table 3.10 Server Cloud Use Case Description.

Requirements:
- Client has to be connected to server.
- Client should be able to display the context menu which consists of sub tabs of Server Cloud's webpage.
3.2.11 Ip addresses

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Ip addresses</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brief Description</td>
<td>User can access the data of Ip addresses page.</td>
</tr>
</tbody>
</table>
| Flow Of Events  | 1. User clicks on the desired menu in context menu.  
|                 | 2. System displays the content of the page. |
| Actors          | User                  |
| Preconditions   | Server Cloud button should have been chosen by user. |
| Post conditions | None                  |

Table 3.11 Ip addresses Use Case Description.

Requirements:

- Client has to be connected to server.
- Client should send Ip addresses Identification to server.
- Server should get the Ip addresses data from related webpage.
- Server should send provided data to client.

3.2.12 PTR

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>PTR</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brief Description</td>
<td>User can access the data of PTR page.</td>
</tr>
</tbody>
</table>
| Flow Of Events  | 1. User clicks on the desired menu in context menu.  
|                 | 2. System displays the content of the page. |
| Actors          | User                  |
| Preconditions   | Server Cloud button should have been chosen by user. |
| Post conditions | None                  |

Table 3.12 PTR Use Case Description.

Requirements:

- Client has to be connected to server.
- Client should send PTR Identification to server.
- Server should get the PTR data from related webpage.
- Server should send provided data to client.
3.2.13 Service Information

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Service Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brief Description</td>
<td>System displays the context menu which shows sub menus of Service Information webpage.</td>
</tr>
</tbody>
</table>
                        2. System displays context menu. |
| Actors              | User |
| Preconditions       | None |
| Post conditions     | User will be able to select the desired menu. |

Table 3.13 Service Information Use Case Description.

Requirements:

- Client has to be connected to server.
- Client should be able to display the context menu which consists of sub tabs of Service Information's webpage.

3.2.14 Version History

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Version History</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brief Description</td>
<td>User can access the data of Version History page.</td>
</tr>
</tbody>
</table>
| Flow Of Events      | 1. User clicks on the desired menu in context menu.  
                        2. System displays the content of the page. |
| Actors              | User |
| Preconditions       | Service Information button should have been chosen by user. |
| Post conditions     | None |

Table 3.14 Version History Use Case Description.

Requirements:

- Client has to be connected to server.
- Client should send Version History Identification to server
- Server should get the Version History data from related webpage.
- Server should send provided data to client.
3.2.15 Service Status

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Service Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brief Description</td>
<td>User can access the data of Service Status page.</td>
</tr>
</tbody>
</table>
| Flow Of Events | 1. User clicks on the desired menu in context menu.  
|                | 2. System displays the content of the page. |
| Actors | User |
| Preconditions | Service Information button should have been chosen by user. |
| Post conditions | None |

Table 3.15 Service Status Use Case Description.

Requirements:
- Client has to be connected to server.
- Client should send Service Status Identification to server
- Server should get the Service Status data from related webpage.
- Server should send provided data to client.

3.2.16 Logout

<table>
<thead>
<tr>
<th>Use Case Name</th>
<th>Logout</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brief Description</td>
<td>The logout process terminates user's access to the system.</td>
</tr>
</tbody>
</table>
|                | 2. System displays the menu item.  
|                | 3. User clicks Logout item.  
|                | 4. System displays the login page. |
| Actors | User |
| Preconditions | The active page should be the main page. |
| Post conditions | User will be able to login again. |

Table 3.16 Logout Use Case Description.

Requirements:
- Client has to be connected to server.
- Client should be terminated from system.
4 Architecture and Design

In this chapter, we describe the overall architecture of our Android application which is shown in Figure 4.1.

According to Figure 4.1
- The Android client connects to the RESTfull web service using the HTTP Client methods.
- Client sends the request for specific data to web service.
- Web service connects to the web scraper (webHarvest) to extract the requested data and saves it in the database.
- Web service reads the requested data from database and sends it in JSON format to the client.

Below the client side and the server side will be described in detail.

Client side:
Client connects to the server side to send and get the requested data. Then client rearranges received data and makes a user friendly interface.

As it is shown in Figure 4.2 the "Connection" class sends the request to the server side and receives requested data. "Deserializer" class deserializes data and sets them in the relevant class. At the end view classes will make user interface. The detailed explanation for this part is described in Chapter 5, Implementation.
Figure 4.2 Client Side Class Diagram.
Server side:
On the server side the main thing is the connection between web service, web scraper and database. After receiving the request and identification from client side, RESTfull web service executes the web scraper which extracts the demanded data and saves it in MySQL. Then web service reads data from database and produces JSON object for sending to the client side.

As it is shown in Figure 4.3 the "Glesys" class receives the login information and requested data identification. "DataHandler" class extracts data from webpage and saves it in MySQL using "ExecuteWebHarvest" method and reads data from MySQL and sets data to relevant class using "readData" method. "Serializer" class produces JSON object for sending to client side.

![Figure 4.3 Server Side Class Diagram](image-url)
5 Implementation

In this chapter, we explain the implementation of client and server side. For more detail some codes are shown in Appendix.

5.1 Client Side
In this section we describe the client side implementation and describe some comparison between accessing data using Android application and normal mobile browser which is called microbrowser.

5.1.1 Login Page
Figure 5.1 shows how the login page looks in mobile devices using normal mobile browser. The problem with using normal browsers is that the contents are displayed too small and it is hard for user to access the webpage.

![Figure 5.1 Login Page View in normal mobile browsers](image)
Figure 5.2 shows how the login page looks in mobile devices using developed application. This page consists of two Android <EditText> for inserting username/password and Android <CheckBox> that provides the alternative of saving username/password. Also there is an Android <Button> which logs in the user to system. If login information is correct then Status Bar Notification's icon will be appeared to the system's status bar as it is shown in Figure 5.3.

![Figure 5.2 Login Page View using Android Application.](image1)

Figure 5.2 Login Page View using Android Application.

![Figure 5.3 Notification.](image2)

Figure 5.3 Notification.
5.1.2 First Page After Login
The first page that will be shown after logging in looks like Figure 5.4 in the normal mobile browsers and Figure 5.5 in mobile devices screen using developed Android application.

Figure 5.4 First Page View in normal mobile browsers.
In Figure 5.4 there are several tabs which consist of some sub tabs. These sub tabs contain information that users would desire to access. Each icon in Figure 5.5 corresponds to a tab in Figure 5.4. As it is shown in Figure 5.4 these tabs and sub tabs are very small and it is hard to access them through smartphones screen while in the developed application as it is shown in Figure 5.5 tabs looks bigger and it is easier to access them.

The XML codes used for implementing the interface of Figure 5.5 is as below which is consists of an Android <GridView> for displaying icons.
5.1.3 Other Pages Implementation

The classes mentioned in Diagram 4.1 in Chapter 4, Architecture and design can be categorized to four kinds based on their role in the system.

1. The class that handles the communication between client part and server part ("Connection" class).

2. In this class objects, instance of data classes (No.3), restores the contents of each field to the value and type it had before serialization in server ("Deserializer" class).

3. Data classes which contain the received data from server side. For each sub tab one specific class is created. The attributes of these classes will be set by "deserializer" class. ("DomainInformation", "CustomerInformation", "PTR", "IpAddresses", …).

4. View classes which make interface using saved data. For each sub tab one specific class is created ("DomainInformationView", "CustomerInformationView", "PTRView", "IpAddressesView", …).
Below No.1 and No.2 implementation way is explained in more detail.

1. "Connection" class:
   To make communication between client and server side, HTTPClient method has been used. A part of the applied code is as following.

   ```java
   DefaultHttpClient httpclient = new DefaultHttpClient();
   HttpPost httppost = new HttpPost("url address");

   httppost.setEntity(new UrlEncodedFormEntity(username, password, request identification));
   HttpResponse response = httpclient.execute(httppost);
   data = EntityUtils.toString(response.getEntity());
   ```

2. View classes:
   Each of these classes should have an individual design based on the web page interface. To fulfill this task different kinds of Android view groups are used. For example In order to design the page that is shown in Figure 5.6, tab layout and table layout has been applied.

   ![Figure 5.6 Sample of Application Page View.](image)
5.2 Server Side
This section consists of three parts that construct the server part. In each part the way of implementation is discussed.

5.2.1 Web Service
As it is mentioned in Chapter 4, Architecture and Design RESTfull we use web service in this project which is described in detail in Chapter 2, Section 2.2.2.

For implementing RESTful webservice the following processes is necessary.
1. Download Jersey jars from Jersey homepage.
2. Install Tomcat as servlet container and Eclipse WTP as development environment.
3. Create new dynamic web project in eclipse environment by selecting File->New->Other->Web->Dynamic Web Project as Figure 5.7 [17].

![Create Dynamic Project](image)

Figure 5.7 Create Dynamic Project [17].

4. Copy all jars from downloaded Jersey zip file into folder "WEB-INF/lib".
5. Create a Java class to get request from client side and execute WebHarvest for extracting demanded data from web page.
6. Register Jersey as the server dispatcher for REST requests by copying following XML code in "web.xml" file.
5.2.2 Web Scraper
WebHarvest tool is used as a web scraper in this project. WebHarvest is explained in Chapter 2, Section 2.3.

Every extraction process for each page is defined in one configuration file using XML based language. In line with the requested data from client side the relevant configuration file will be executed by server side using java code.

By executing the configuration file, WebHarvest logs in to the GleSYS website using received login information from client side and extracts data on demand. Thereupon WebHarvest connects to database and saves fetched data. Due to connecting to database following code has been used.

```xml
<database
  connection="jdbc:mysql://localhost/Gleysis"  
  jdbcclass="com.mysql.jdbc.Driver"
  username="root" password="root">

  <!--Access to database -->

</database>
```
5.2.3 Database
Utilized database in this project is MySQL. For each sub tab in GleSYS website specific tables has been created. In these tables classified extracted data will be saved in order to make easier access. Our database structure is as follow.

Figure 5.8 Database Structure.
6 Case Study

The main purpose has been to develop an application that makes the access to GleSYS website easier and more user friendly from smart phone devices. To achieve this goal a service has been created. The system has been tested from different user based aspects in order to confirm its practicalities and capabilities.

Different mentioned aspects are tested as follow.

• Installing and uninstalling the application on mobile device which worked properly.

• The functionality of client side and server side on mobile device has been checked. For example if user inserts the correct login information, the requested data is shown in expected way otherwise the logging process would not happen and user will be warned about inserting wrong information.

• The view of the website in mobile devices screen has been checked. Comparing to before it has been improved a lot as it is shown in Figure 5.4/5.5 in Chapter 5. Also the contents of the web page are displayed in the suitable size for the mobile devices screen.

• The compatibility of the application has been checked with Android OS version 2.2 and 2.3 which has been working successfully.
7 Conclusion and Future Work

In the first part of this chapter our conclusion from this whole project is described and in the second part, the possible future works are recommended.

7.1 Conclusion
As described in Chapter 1 the problem is that administrators want to be aware of the status of their servers everywhere at any time. However, because administration website is designed for much bigger screens than mobile smartphones screen, they have some difficulties accessing website through their smartphones.

The goal was to develop a user friendly and attractive Android application which makes it very easier for administrators to login to the website and have access to all features available in website using their smartphones everywhere at any time without the need of zooming in or out for accessing contents.

As it is mentioned in Chapter 6 we have tested the installing and uninstalling the application, the functionality of client side and server side on mobile device, the view of the website in mobile devices screen and compatibility of the application with Android OS version 2.2 and 2.3. As a result everything is working properly and administrators' desire is satisfied so we can say that we have reached our goal.

7.2 Future Work
The future possible works to improve this application are as follow.

- Extract data from Server Overview page that we couldn't handle in this thesis.
- Add the facilities to the application such as adding/removing server or modifying the information in the website through mobile devices.
- Develop application for other mobile OS such as iOS, Blackberry and so on using created RESTful service in this project.
References

Appendix

Sample Code:

A part of login page codes is as follow:

```java
1. SharedPreferences pref =
   2. getSharedPreferences(PREFS_NAME, MODE_PRIVATE);
3. username = pref.getString(PREF_USERNAME, null);
4. password = pref.getString(PREF_PASSWORD, null);
5. if (username != null && password != null) {
   6. UN.setText(username);
   7. PW.setText(password);
   }
8. }
9. checkbox = (CheckBox) findViewById(R.id.checkbox);
10. final Button LogButton = (Button)
   11. findViewById(R.id.login);
12. LogButton.setOnClickListener(new LoginClick());
13. }

14. @Override
15. public void onPause() {
16.   super.onPause();
17.   save(checkbox.isChecked());
18. }
19. @Override
20. public void onResume() {
21.   super.onResume();
22.   checkbox.setChecked(load());
23. }

24. private void save(final boolean isChecked) {
25.   SharedPreferences sharedPreferences =
26.     getPreferences(Context.MODE_PRIVATE);
27.   SharedPreferences.Editor editor =
28.     sharedPreferences.edit();
29.   editor.putBoolean("check", isChecked);
30.   editor.commit();
31. }

32. private boolean load() {
33.   SharedPreferences sharedPreferences =
34.     getPreferences(Context.MODE_PRIVATE);
35.   boolean checkboxState =
36.     sharedPreferences.getBoolean("check", false);
37.   if (checkboxState == false){
38.     getSharedPreferences(PREFS_NAME, 
39.         MODE_PRIVATE).edit().clear();
40.     UN.setText("");
41.     PW.setText("");
42. }
43.   return checkboxState;
44. }
```
The explanation of above code is as follow:

1-13:
If CheckBox is checked the value of username and password setts in EditText.

14-18:
When another activity comes in front of the activity the save method (9-10) is called.

19-23
When the activity comes foreground the load method (11-12) is called.

24-31
Saves the status of CheckBox.

32-44
Returns the CheckBox status.