DETAILED DESIGN REPORT

ODAK YAZILIM

18.01.2010

Table of Contents

1   Introduction

1.1   Project Description

In accordance with the development of the technology in the direction of mobile platforms, smart devices are introduced to users which enables people use their mobile phones for too many applications. Most of these applications are based on Bluetooth and Java technology. Taking in account these fields, we decided to integrate this technology to facilitate health sector. With the support of Turkcell we will make a communication between doctors and patients immediately. Therefore, we wanted to make possible that people could be able to get health service whenever and wherever they want using their mobile phones.

In **Turkcell Health Service (THS),** our aim is to facilitate health sector.

1.2   Definitions, Acronyms & Abbreviations

THS: Turkcell Health Service

DD: Democratic Decentralized

SMS: Short Message Service

3G: Third Generation

GPRS: General packet radio service

DFD: Data flow diagram

1.3   Project Features

THS will be a modular product which includes following features:

- Mobile consultation using 3G service

  Subscriber can access to the doctors which are provided by this service 7/24. This service will provide some specific medical monitoring devices which monitors heart rate, ECG, Blood pressure Heart rhythm regularity etc. and send this information to

an application running on a mobile phone via Bluetooth, then application sends this information to health service database provided that doctor needs while appointment. The system provides also a web page to doctors which include detailed information of patients.

- Tracking service

Chronic patients are controlled with a device which patients should use 7/24. There are some critical regions for each disease decided by the doctors. If the measurements of the patients exceed critical regions, service will inform the doctor and a person who is chosen by patients before, about patient's exact location and these measurements.

- Alarm Service

This service provides an alarm mechanism which will inform the patients about his/her medicine to be taken when its time comes. This service also sends a SMS which includes the menu of his/her diet in each meal a day.

1.4 Task Assignment to Each Group Member

In our project, each member is responsible for specific tasks as follows:

- Birkan PALA: Mobile application implementation.
- Faruk YILMAZ: Server implementations.
- Abuzer Miraç ÖZCAN: Database design and implementation.

Each member is responsible for detailing the above main tasks into minor tasks, distributing them to the group members and checking whether those minor tasks are done properly and in time.

1.5 Purpose of Document

The purpose of this document is to express the detailed design issues about our project. According to the IEEE standards the Detailed Design Report shows how the software system

will be structured to satisfy the requirements identified in the Requirement Analysis Report and to satisfy the design patterns identified in the Initial Design Report. Although design decisions that we have made are significant guidelines for our progress, they may be changeable.

This document will contain the general definition and features of the project, design constraints, architectural and procedural design diagrams, a brief explanation about our progress and schedule of the project.

2 Design Considerations

2.1 Design Constraints

Since Odak Yazılım team is not experienced in mobile application, server and database development the group may make some mistake during development process and the details of the development will be learned by the team when developing the project. Therefore, to use a structural development model like "Waterfall" seems unreasonable because risk of making an important design mistake at the very beginning of the project is really high by an inexperienced team. For this reason, the team needs a more flexible development methodology like "Spiral". It lets the development team learning by making mistakes and even an inexperienced team can form a solid structure after enough number of iterations.

Since we do not have enough time to make adequate number of iterations, taking advantage "Agile Methodology" can also be really beneficial and can speed up the development process. Because, designing and implementing a non-working prototype of the system sometimes requires lots of effort and has the risk of unsuccessfulness. Therefore instead of complex designs, producing fully tested and working parts of the project can be more rapid and can produce more reliable outcome.

To sum up, Odak Yazılım will mainly follow "Spiral Methodology" during development process. However, for the parts that needs special attention "Agile Methodology" can be plausible.
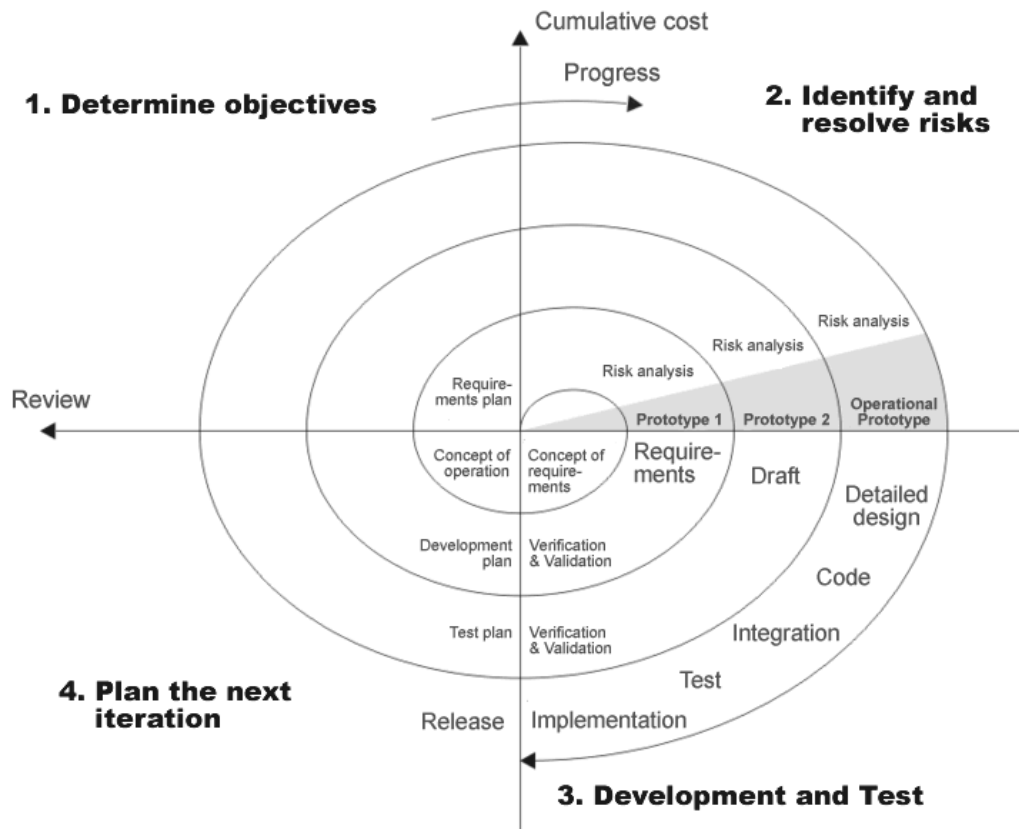
Figure 1 - Spiral Model [1]

2.2 Design Goals and Objectives

2.2.1 Portability

THS is a mobile application that runs on the mobile phones that are operated with any operating system which supports JVM. Beside from this, THS can be accessed from any place that has a Turkcell Service connection for GPRS or wireless connection. This mobility and easy accessibility result in portability which is an important feature of the new generation applications.

2.2.2 Reliability

We are planning to design THS as a fault tolerant application. After every milestone that we will reach, we will use different kinds of testing methods to improve the performance and to lessen the margin of error. Besides, in order to avoid blocking in the workflow and to supply

6

a multi-tasking application, optimization techniques will be used. This reliability property gives THS users a possibility to use the application without having difficulty.

Second issue about reliability is keeping user profiles secret. We are going to store all of them in our own database. Patient profiles will only be accessed by doctors and himself/herself and they are not accessible for any other patients.

2.2.3 User Interactivity

THS is mainly related with health sector, so we are planning it to have a very crowded user target. Designing a user-friendly application and providing an easy-use structured system will be one of our main concerns. THS user will easily perform the different kinds of actions at the same time and will have least difficulty. Besides, interactions between doctors and patients will be provided efficiently.

2.3 Development Tools

When we look at all the possible technologies, libraries, platforms that we will use in our project; we have seen that the most convenient programming language is Java.

For the mobile part of the project we will use J2ME since it includes a set of core libraries that provides most of the functionality available in the core libraries of the Java programming language.

For our database, we will use MySQL because of the advantages of it. With the help of the JBOSS, we will make our connection with database.

For the server development, we will use JSP Servlet.

As a result we use Java language as our applications default developing language with NetBeans IDE.

3 System Requirements

3.1 General Overview

The figure below shows the general overview of the THS.
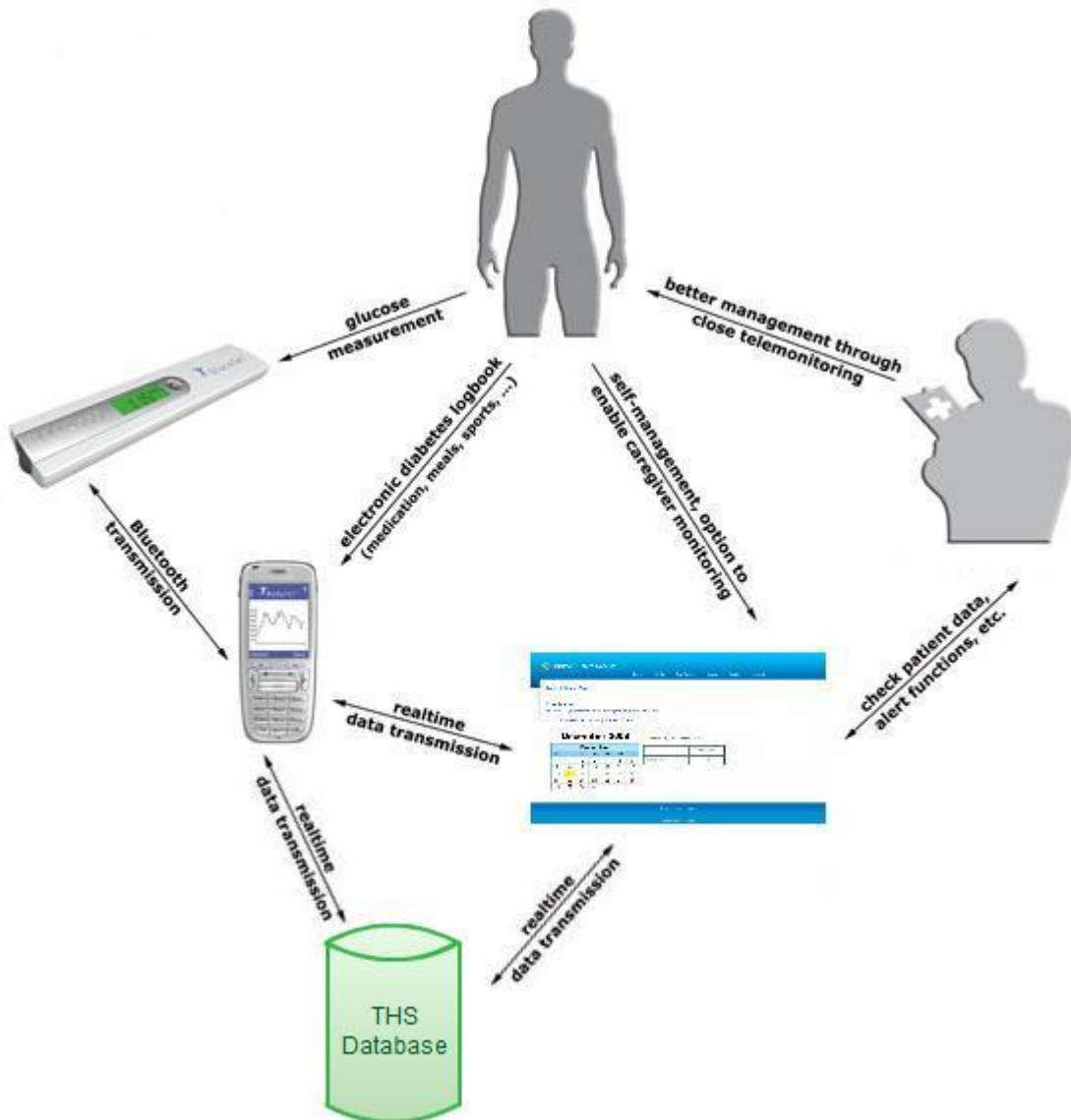
Figure 2 – General Overview [2]

3.2 Hardware Requirements

Considering our project, strictly defining hardware requirements is not wise. By the time as our number of users increase, our hardware requirements will change. As a baseline, we need a single Core2 Duo computer with 2 GB of ram. Nevertheless, as the number of users increase some enhancements on hardware configurations must be considered:

- Moving to a multi-processor and vast memory server
- Creating a clustered server architecture

3.3 Software Requirements

- Windows or Linux operating system
- JBoss
- JSP/Servlet
- J2ME
- MySQL Server as DBMS
- Turkcell API

3.4 Development Environment Requirements

- NetBeans IDE
- MySQL Administrator
- JBoss Administrator

3.5 Use Case Diagram
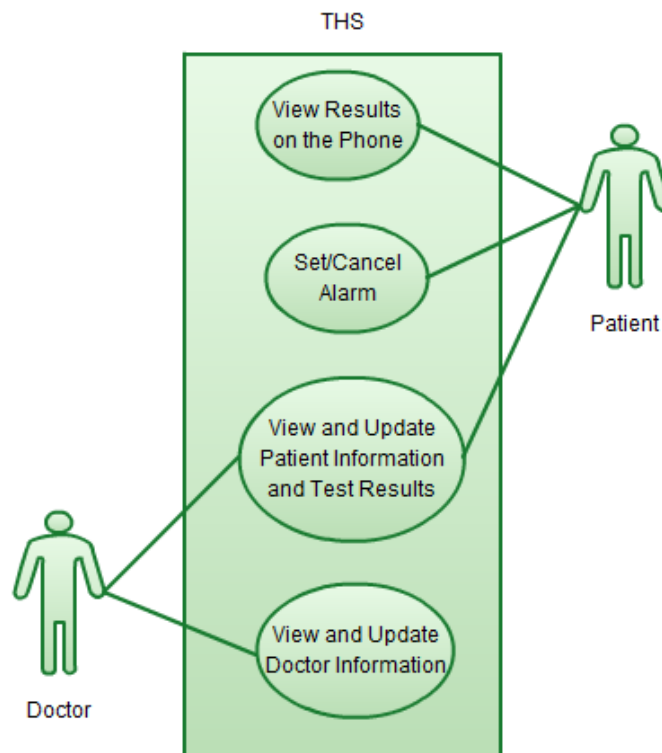
Use case diagram of the THS is shown in the figure below.

Figure 3 – Use Case Diagram

3.6   Use Case Scenarios

Actors: Doctor and Patient.

Pre-Condition: The user should subscribe THS. The patient should make the test using the device and mobile application should send this result to THS server.

Basic Flow:

1. The patient should start the mobile application.
2. Device will send test information to the mobile phone via Bluetooth.
3. After receiving mobile application will send this information to THS server.
4. To communication between mobile application and THS via GPRS, wireless, etc. :
    4.1. It will send the information coming from mobile application to THS and vice versa.
    4.2. THS will send alarm SMS to the patient via Turkcell.
    4.3. If necessary, it will send an SMS to patient's relatives and doctor.
5. It will provide video conversation between patients and doctors.
6. If necessary location of the patient will be sent to THS.
7. Keeping and updating the patients' records:
    7.1. It will keep the patients' records in THS database.
    7.2. It will update the records of patients according to the information which is measured by device.
8. It will serve a web interface to the doctors. This web interface will include:
    8.1. Patients' information
    8.2. Doctor's information
    8.3. Video conversation service
9. It will serve a web interface to the patient. This web interface will include:
    9.1. Patients' information
    9.2. Alarm service

4 System Design

4.1 Architectural Design

4.1.1 Data Flow Diagrams

4.1.1.1 Level 0

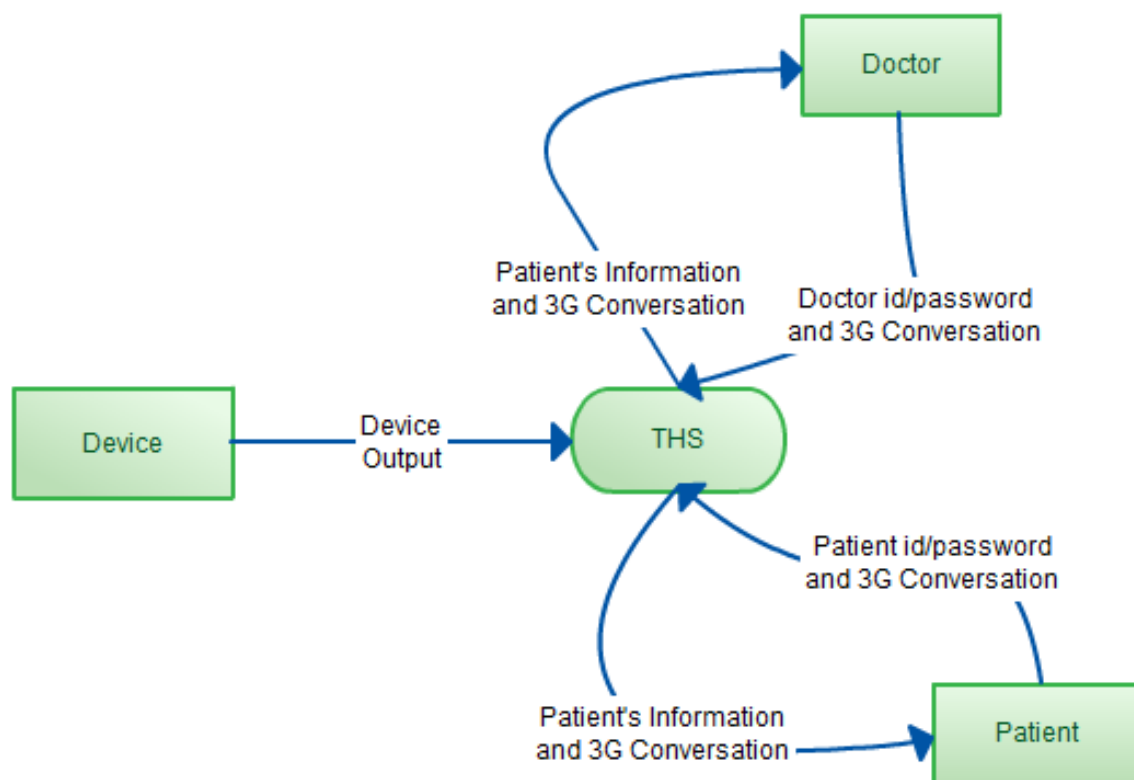Level 0 DFD is shown in the figure below.

Figure 4 - Data Flow Diagram Level 0

4.1.1.2 Level 1

Level 1 DFD is shown in the figure below.
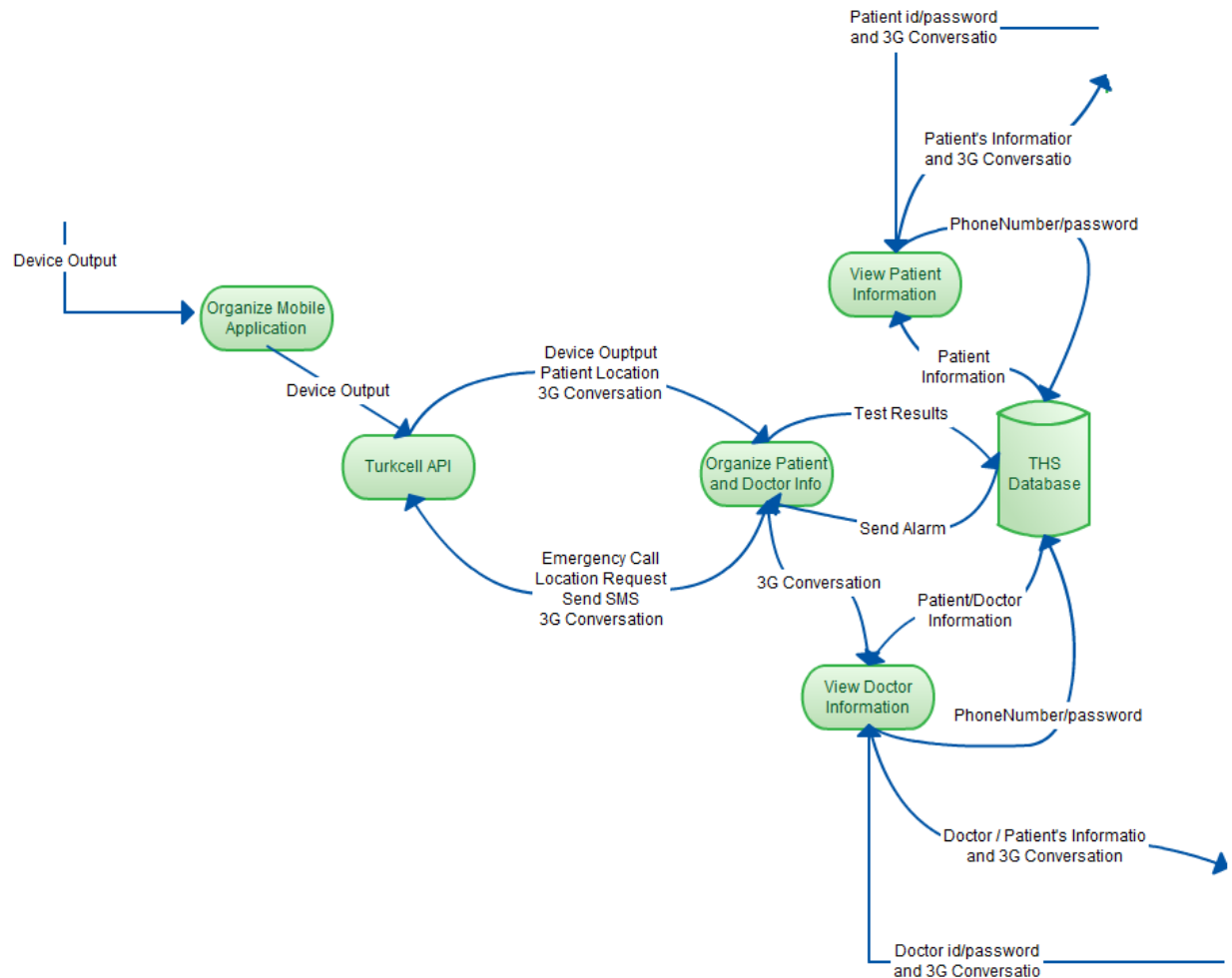
Figure 5 - Data Flow Diagram Level 1

4.1.2 Modules and Class Diagrams

According to the system design that we have made there are five modules namely:

- Mobile Module
- Server Module
- Web Page module
- Alarm Module
- Conversation Module

4.1.2.1 Mobile Module

The following class diagram shows the Mobile Module. In Mobile Module; mobile application is supposed to connect special test devices via Bluetooth and receive the patients test results. Then, it will send these results to the server using GPRS or wireless connection. Mobile module class diagram is shown in the figure below.

Figure 6 - Mobile Module Class Diagram

We would like to explain this module's classes and some of their main methods briefly.

- **User Class**

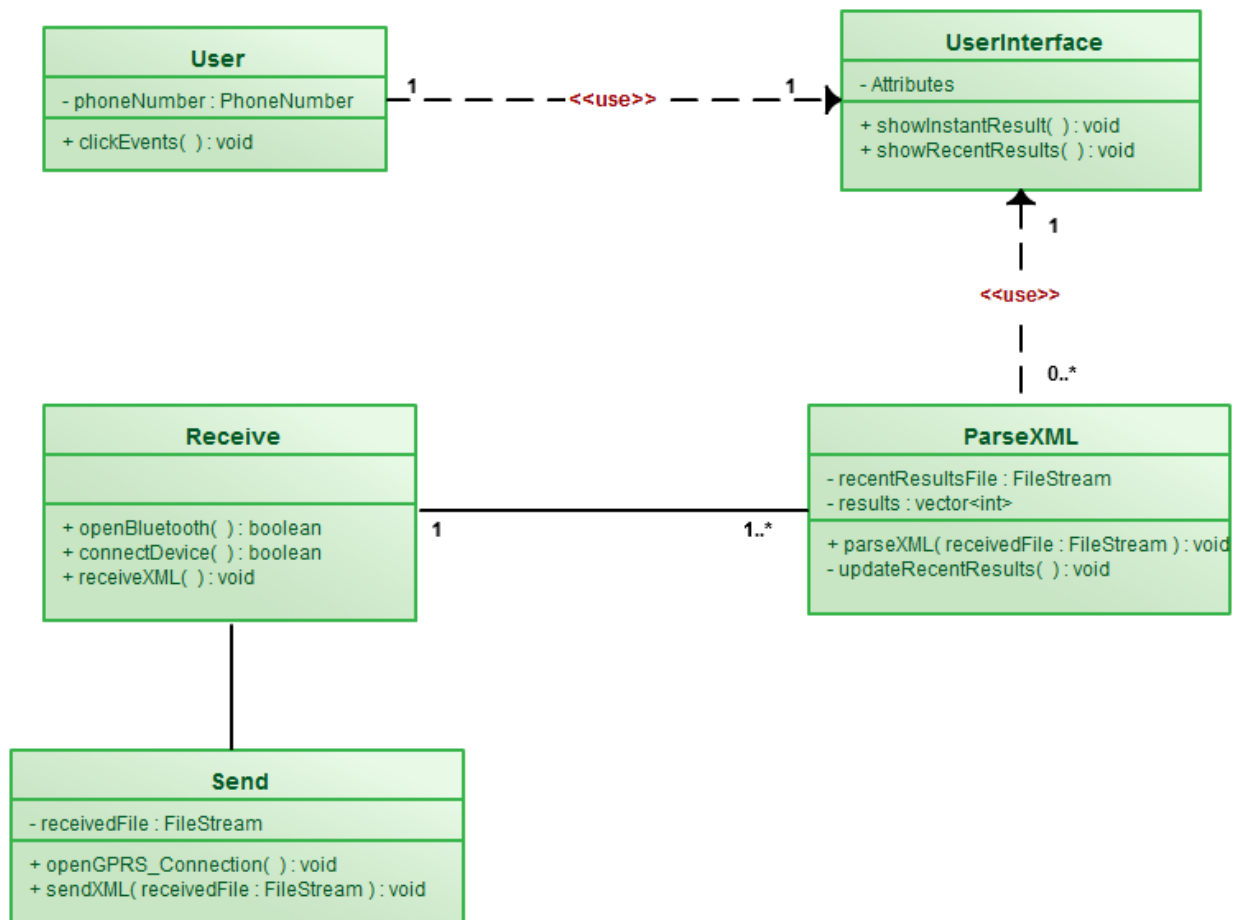 User can view his/her test results via requesting *showInstantResult()* and *showRecentResults()* methods of *UserInterface* class, using *clickEvent()* method.

- **UserInterface Class**

  This class contains methods which are related to user interface, as its name implies.

- **Receive Class**

  This class is composed of three methods namely *openBluetooth(), connectDevice()* and *receiveXML().* Whenever application starts, it will ensure whether Bluetooth is open or not. Then, checking available test device, it will establish Bluetooth connection between that device and phone. After that, it will wait for an XML file from test device and receives it.

- **ParseXML Class**

  This class is composed of three methods namely *parseXML()* and *updateRecentResults()*. After receiving XML file from test device, *parseXML()* method will parse it and *updateRecentResults()* method will update the patient's results up to 10 records and will request *showInstantResult()* of *UserInterface* class to show test results on the screen.

- **Send Class**

  This class is composed of three methods namely *openGPRS_Connection() and sendXML()*. *openGPRS_Connection()* method will establish GPRS connection between phone and *THS* server. After that, XML file consisting of test results will be sent to the server via *sendXML()* method.

4.1.2.2 Server Module

The following class diagram shows the *Server* module. It will basically be responsible for receiving XML file including patients' test results, updating the database and if the test result exceeds the limit decided by the doctor for each disease then it will send an SMS, which contains the location of the patient and test result, to a relative and to the doctor. Server module class diagram is shown in the figure below.
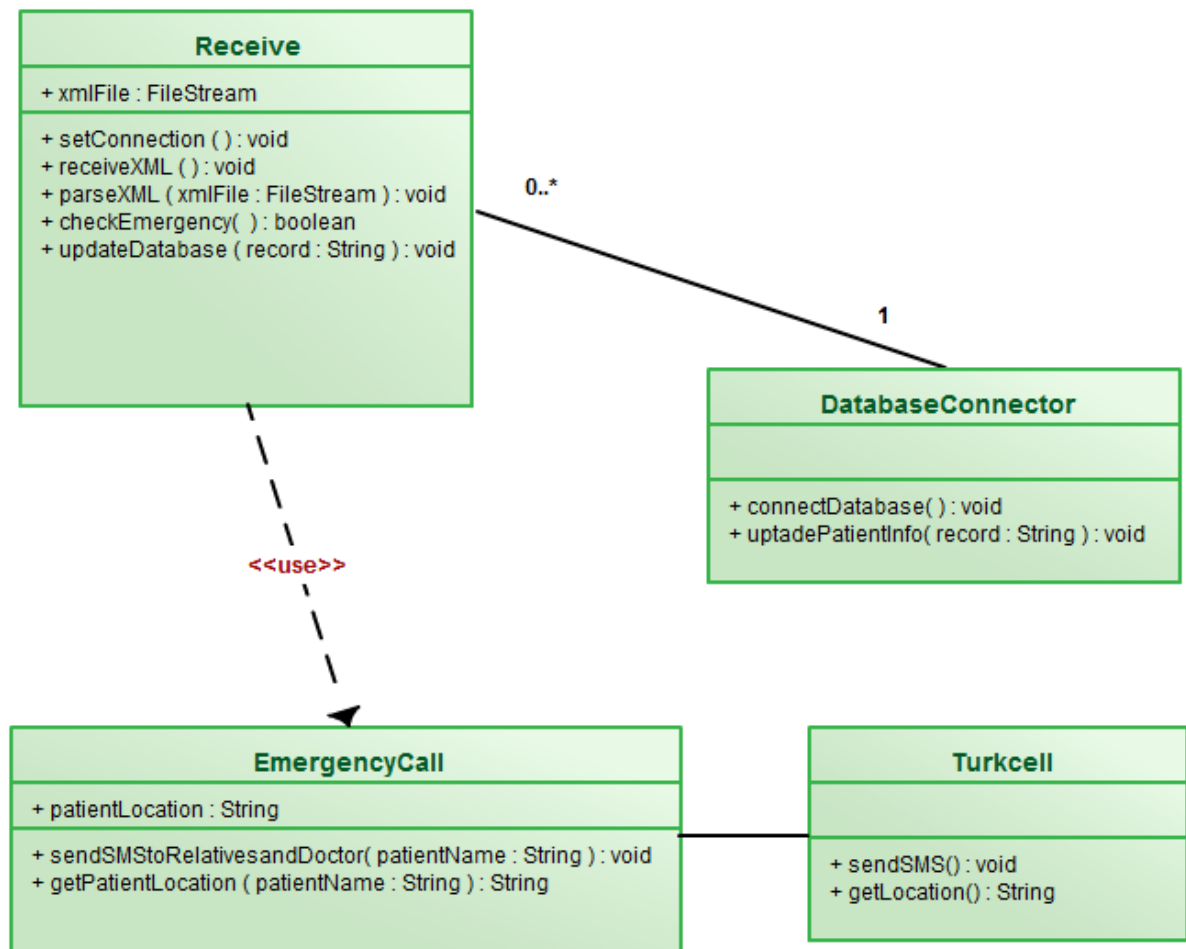

Figure 7 - Server Module Class Diagram

We would like to explain this module's classes and some of their main methods briefly.

- **Receive Class**

15

This class is composed of setConnection(), receiveXML(), parseXML(), checkEmergency() and updateDatabase() methods. setConnection() method will set up required network connections. Then, receiveXML() method will receive the XML file sent by phone. After parsing that file with parseXML() method, database will be updated with updateDatabase() method. If there is an emergency situation by using checkEmergency() it will invoke the related methods of EmergencyCall class.

- **DatabaseConnector**

This class is responsible for making a bridge between Server module and Database. It is composed of *connectDatabase()* and *uptadePatientInfo()* methods which connect the database and update the related tables.

- **EmergencyCall**

This class has a vital role in the THS, because if the test result exceeds the limit decided by the doctor for each disease then it will send an SMS, which contains the location of the patient and test result, to a relative and to the doctor. It will ensure the location of patient with Turkcell API's *getLocation()* method.

- **Turkcell API**

This is not an actual class of our module but we wanted to count it as a separate class in this module, because we will use most of the API functions in our methods during the work flow such as *sendSMS()* and *getLocation()* methods.

4.1.2.3 Web Page Module

The following class diagram shows the *Web Page* module. It will provide an interface to patients and doctors. Both of them can see their information from their related pages. In order to use this system they must sign up the system and log in. Web Page module class diagram is shown in the figure below.

Figure 8 - Web Page Module Class Diagram
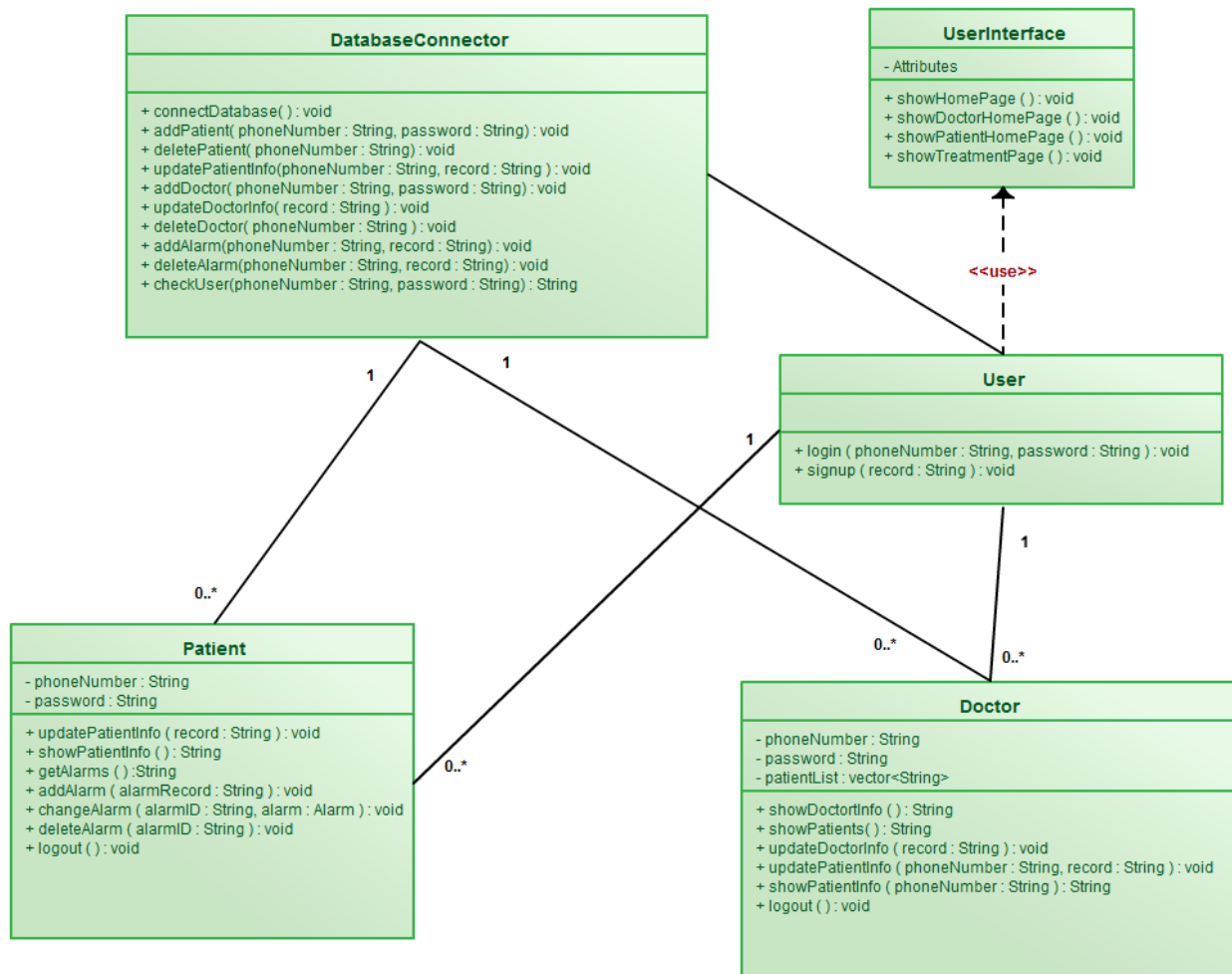
We would like to explain this module's classes and some of their main methods briefly.

- **DatabaseConnector Class**

  This class is responsible for making a bridge between *Web Page* module and Database. It is composed of many methods which connect the database, add and update the related tables.

- **UserInterface Class**

  This class provides an interface to the users.

- **User Class**

  This class consists of two methods namely *login()* and *signup()* methods. *login()* method will check the user and his/her password from the Database and if success it will create patient or doctor objects. *signup()* method will create a new user, add its records to the database and create doctor or patient objects.

- **Patient Class**

  This class is used by the patient and it can updates, gets, sets patient's records and his/her alarms. These changes are reflected to the database with *DatabaseConnecter* class methods.

- **Doctor Class**

  This class is used by the doctor and it can update, gets, sets patient's or doctor's records. These changes are reflected to the database with *DatabaseConnecter* class methods.

4.1.2.4 Alarm Module

The following class diagram shows the *Alarm* module. It will provide an alarm mechanism to the patients to take their medicines or meals properly. Alarm module class diagram is shown in the figure below.
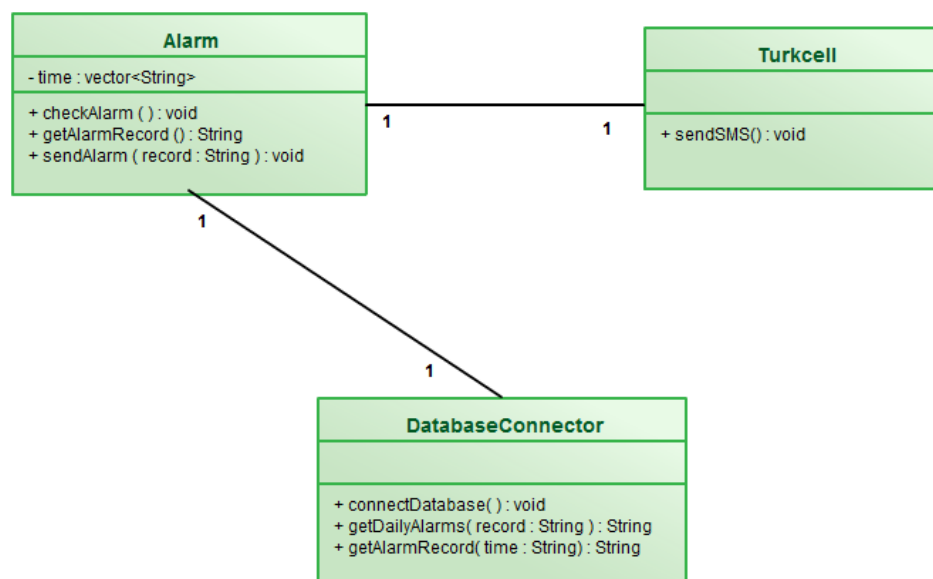
Figure 9 - Alarm Module Class Diagram

18

We would like to explain this module's classes and some of their main methods briefly.

- **Alarm Class**

  This class will get alarm times at beginning of the each day from the database by invoking DatabaseConnector class methods. Then it will store this times in the local time vector. Whenever alarm time is reached it will request other information of alarm using time variable and send an alarm SMS to the patient using Turkcell API. This SMS may consist of medicine info and meals.

- **DatabaseConnector Class**

  This class is responsible for making a bridge between *Alarm* module and Database. It is composed of three methods which connect the database and get the related tables.

- **Turkcell API**

  This is not an actual class of our module but we wanted to count it as a separate class in this module, because we will use most of the API functions in our methods during the work flow such as *sendSMS()* method.

4.1.2.5 Conversation Module

This module provides a communication between patient and doctors via 3G. Conversation module class diagram is shown in the figure below.
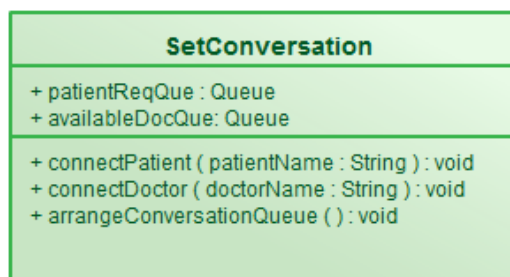
Figure 10 - Conversation Module Class Diagram

- **SetConversation Class**

This class consists of three methods namely *connectPatient(), connectDoctor()* and *arrangeConversation(). connectPatient()* will connect the patient phone to the server. *connectDoctor()* will connect the doctor to the server, then *arrangeConversation()* will arrange the communication between them.

4.1.3 ER Diagrams

Entity Relationship diagram of the THS is shown in the figure below.

Figure 11 - Entitiy Relationship Diagram

4.1.4 Database Schemas
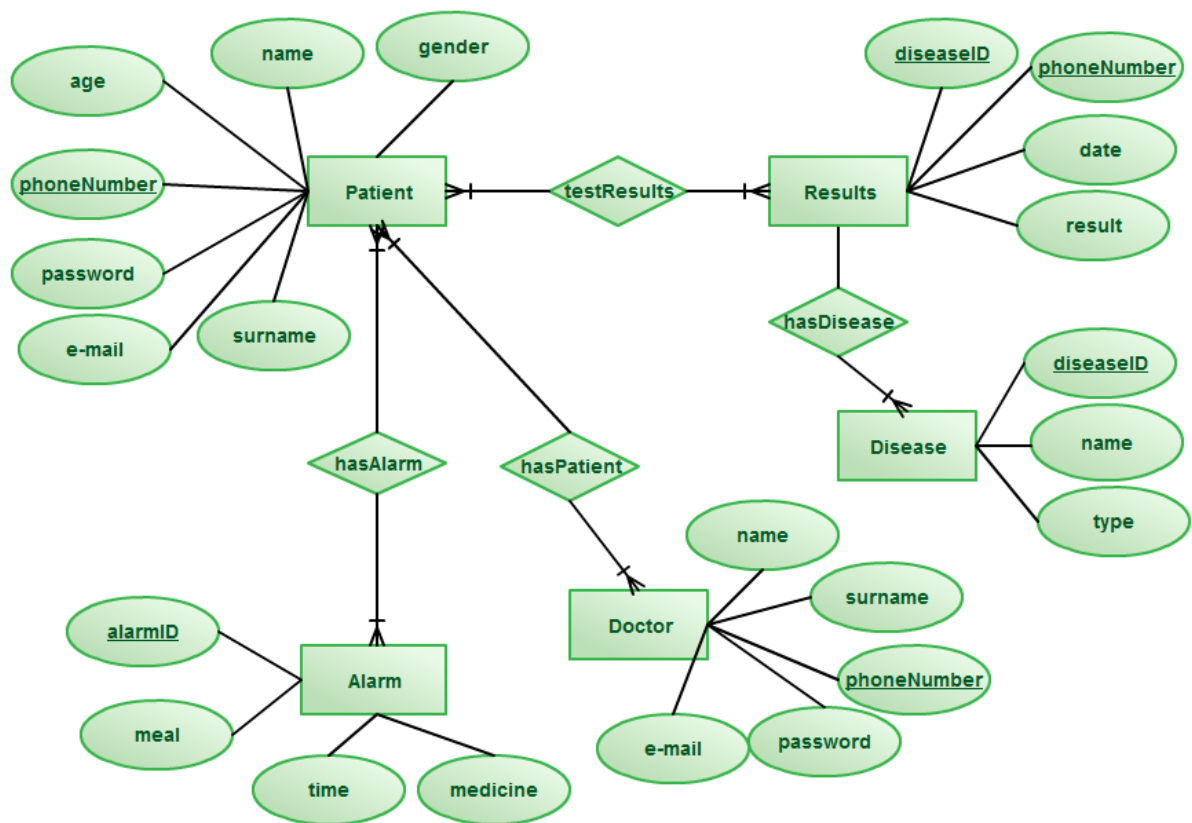
4.1.4.1 Patient Table

Field	Type	Null	Foreign Key	References
phoneNumber (P.K)	Varchar(10)	No	No	-
password	Varchar(10)	No	No	-
name	Varchar(30)	No	No	-
surname	Varchar(20)	No	No	-
age	Integer	No	No	-
gender	Varchar(5)	No	No	-
e-mail	Varchar(50)	No	No	-

4.1.4.2 Disease Table

Field	Type	Null	Foreign Key	References
diseaseID (P.K)	Integer	No	No	-
name	Varchar(20)	No	No	-
type	Varchar(30)	No	No	-

4.1.4.3 Doctor Table

Field	Type	Null	Foreign Key	References
phoneNumber (P.K)	Varchar(10)	No	No	-
password	Varchar(10)	No	No	-
name	Varchar(30)	No	No	-
surname	Varchar(20)	No	No	-
e-mail	Varchar(50)	No	No	-

4.1.4.4 Results Table

Field	Type	Null	Foreign Key	References
phoneNumber (PK)	Varchar(10)	No	No	-
diseaseID (PK)	Varchar(10)	No	No	-
date	Datetime	No	No	-
result	Varchar(20)	No	No	-

4.1.4.5 Alarm Table

Field	Type	Null	Foreign Key	References
alarmID (P.K)	Integer	No	No	-
time	Datetime	No	No	-
meal	Varchar(30)	No	No	-
medicine	Varchar(20)	No	No	-

4.1.4.6 HasDisease Table

Field	Type	Null	Foreign Key	References
diseaseID (P.K)	Integer	No	Yes	Disease
phoneNumber (P.K)	Varchar(10)	No	Yes	Results

4.1.4.7 HasAlarm Table

Field	Type	Null	Foreign Key	References
alarmID (P.K)	Integer	No	Yes	Alarm
phoneNumber (P.K)	Varchar(10)	No	Yes	Patient

4.1.4.8 HasPatient

Field	Type	Null	Foreign Key	References
patientPhoneNumber (P.K)	Varchar(10)	No	Yes	Patient
doctorPhoneNumber (P.K)	Varchar(10)	No	Yes	Doctor

4.1.4.9 TestResults

Field	Type	Null	Foreign Key	References
diseaseID (P.K)	Integer	No	Yes	Results
phoneNumber (P.K)	Varchar(10)	No	Yes	Patient

4.1.5 XML File Definition

4.1.5.1 Device XML Structure

This XML file is sent from test device to mobile phone after measurement. The structure of this file is shown below.

```
<device>

    <deviceID> </deviceID>

    <result> </result>

</device>
```

- **<device> :** Start tag of the device.xml
- **<deviceID> :** The identification number of the device that will be used to identify device the result comes from.
- **<result> :** Test result of the patient.

4.1.5.2 Result XML Structure

This XML file is sent from mobile phone to THS server. The structure of this file is shown below.

```
<result>

    <patientPhoneNumber> </patientPhoneNumber>

    <date> </date>

    <diseaseID> </diseaseID>

    <deviceID> </deviceID>

    <testResult> </testResult>

</result>
```

- **<result> :** Start tag of the result.xml
- **<patientPhoneNumber> :** Patient phone number to identify the patient.
- **<date> :** Date of the measurement.
- **<diseaseID> :** Disease identification.
- **<deviceID> :** The identification number of the device that will be used to identify device the result comes from.
- **<testResult> :** Test result of the patient.

4.2   Procedural Design

4.2.1   Sequence Diagrams

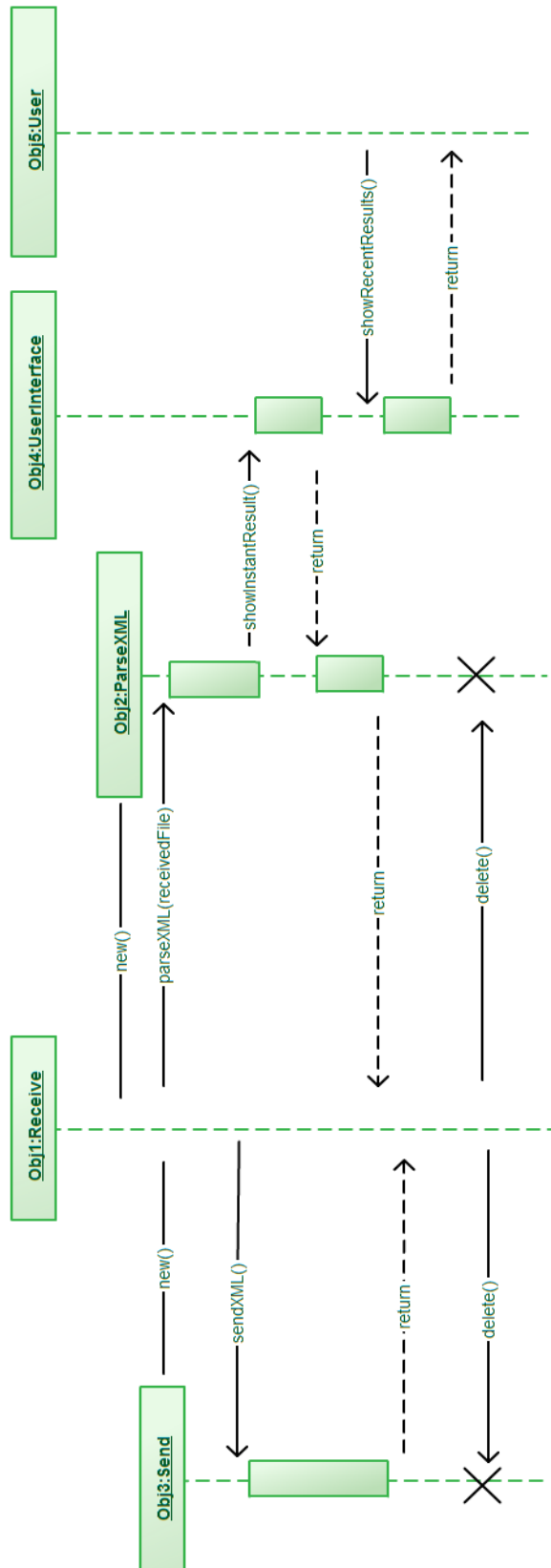The figures below show the modules' sequence diagrams respectively.

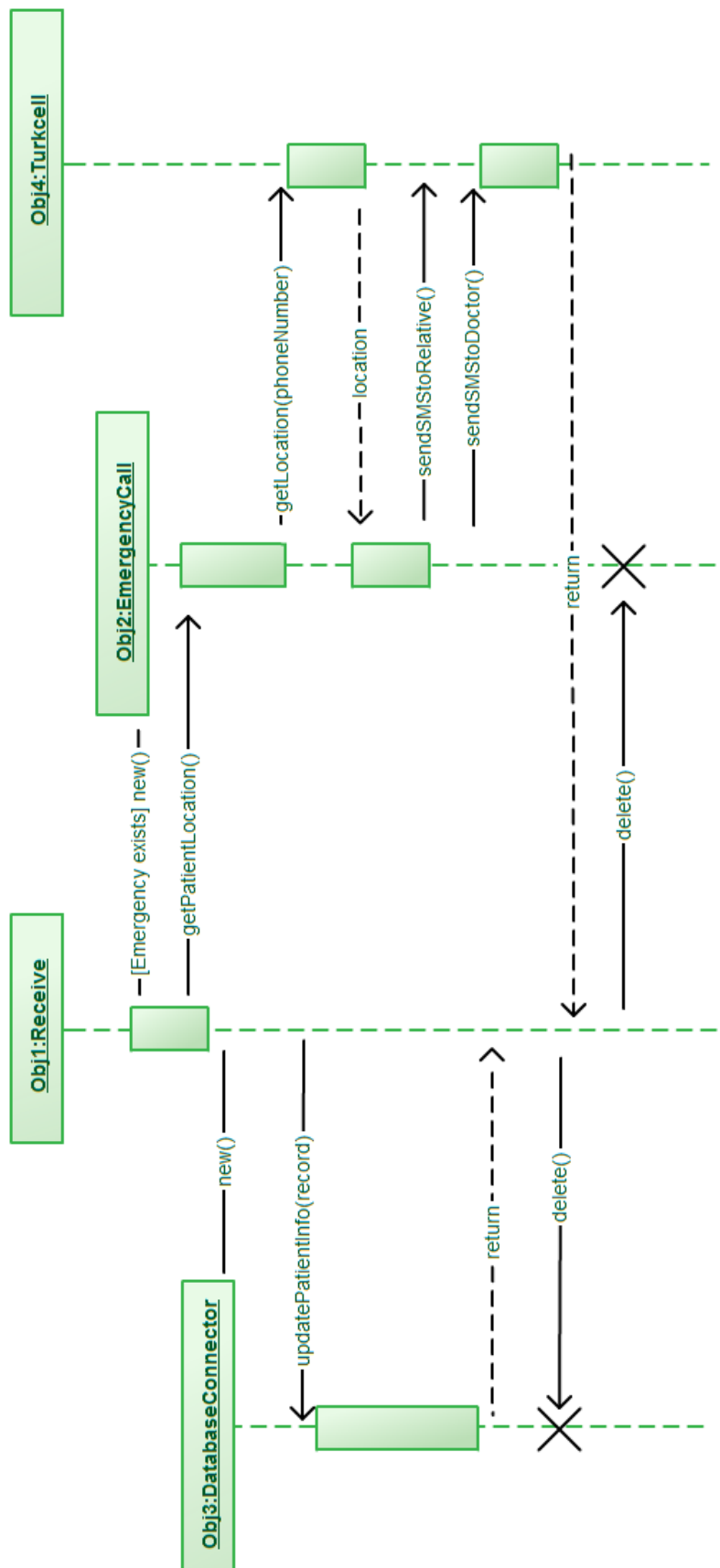#### 4.2.1.1   Mobile

Figure 12 - Mobile Sequence Diagram

Figure 13 - Server Sequence Diagram
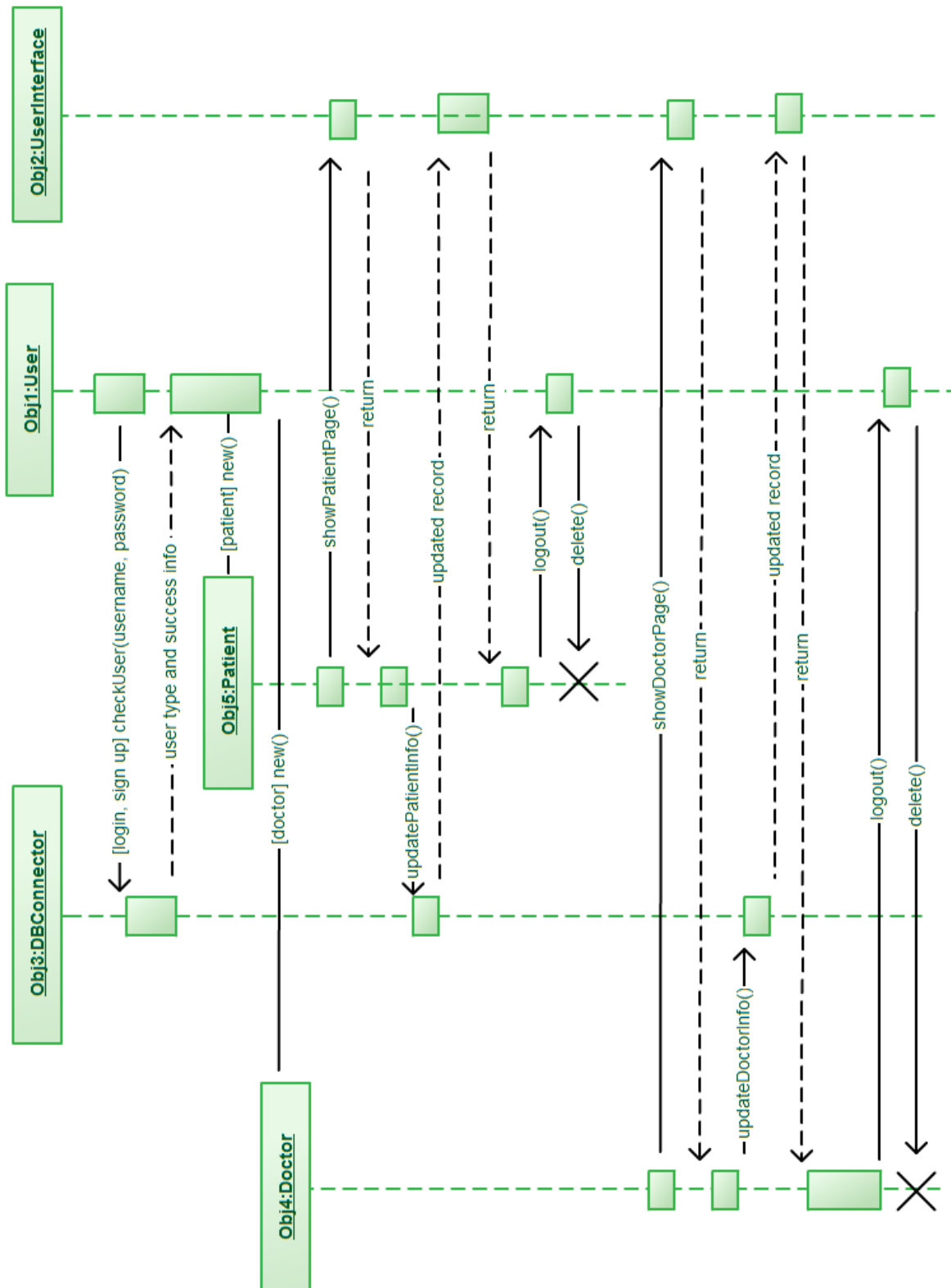
4.2.1.3　Web Page

Figure 14 - Web Page Sequence Diagram
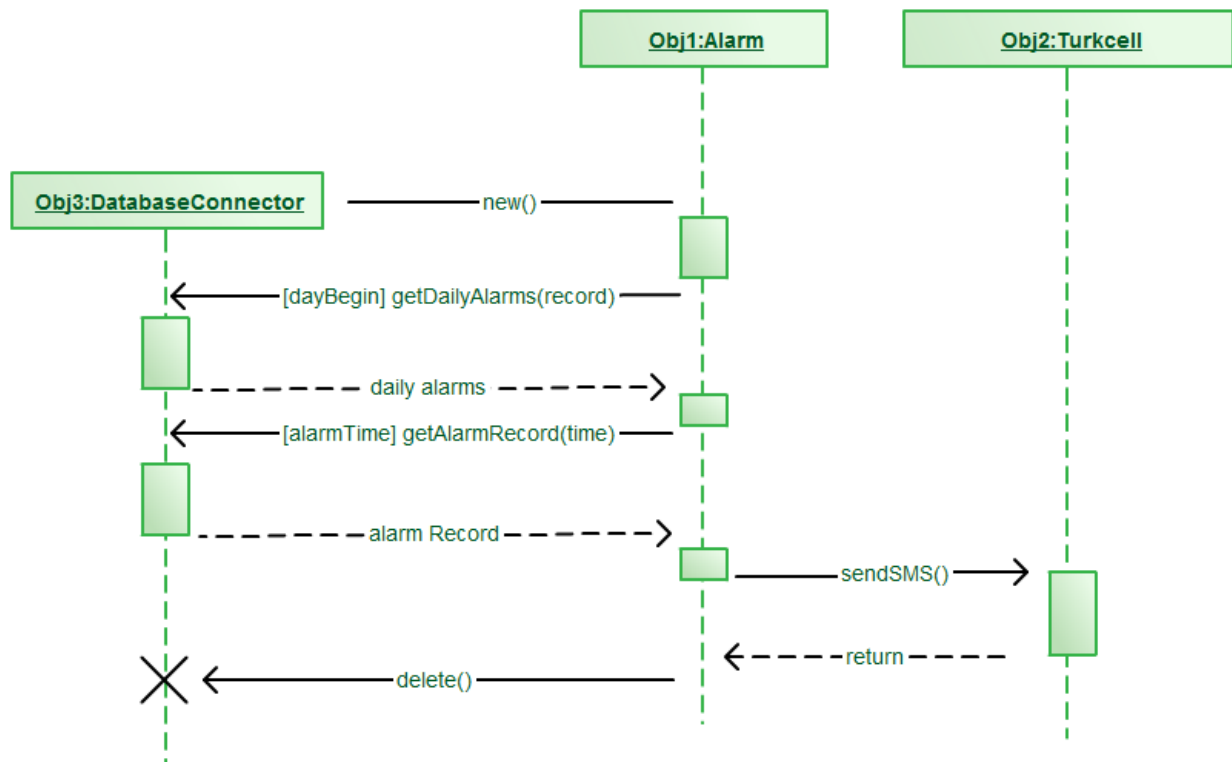
4.2.1.4 Alarm

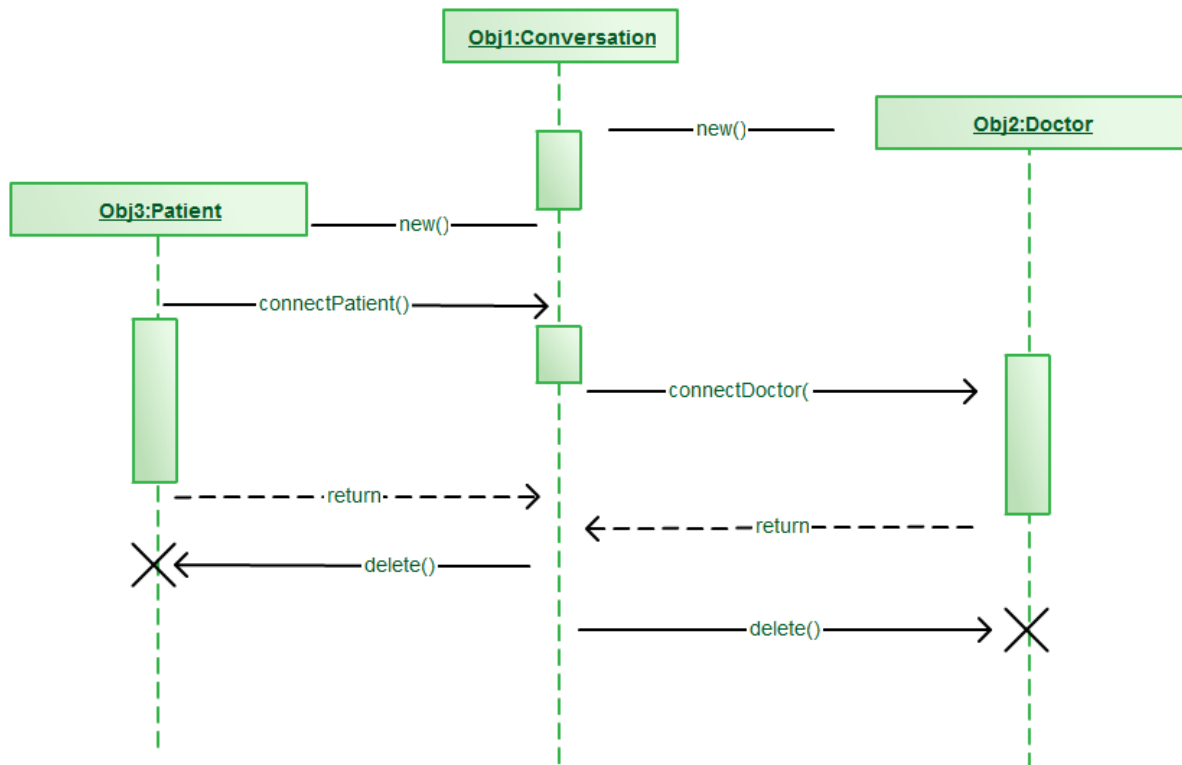**Figure 15 - Alarm Sequence Diagram**

4.2.1.5 Conversation

Figure 16 – Conversation Sequence Diagram

4.2.2 Activity Diagrams

Activity diagrams of the THS are shown in the figures below respectively.
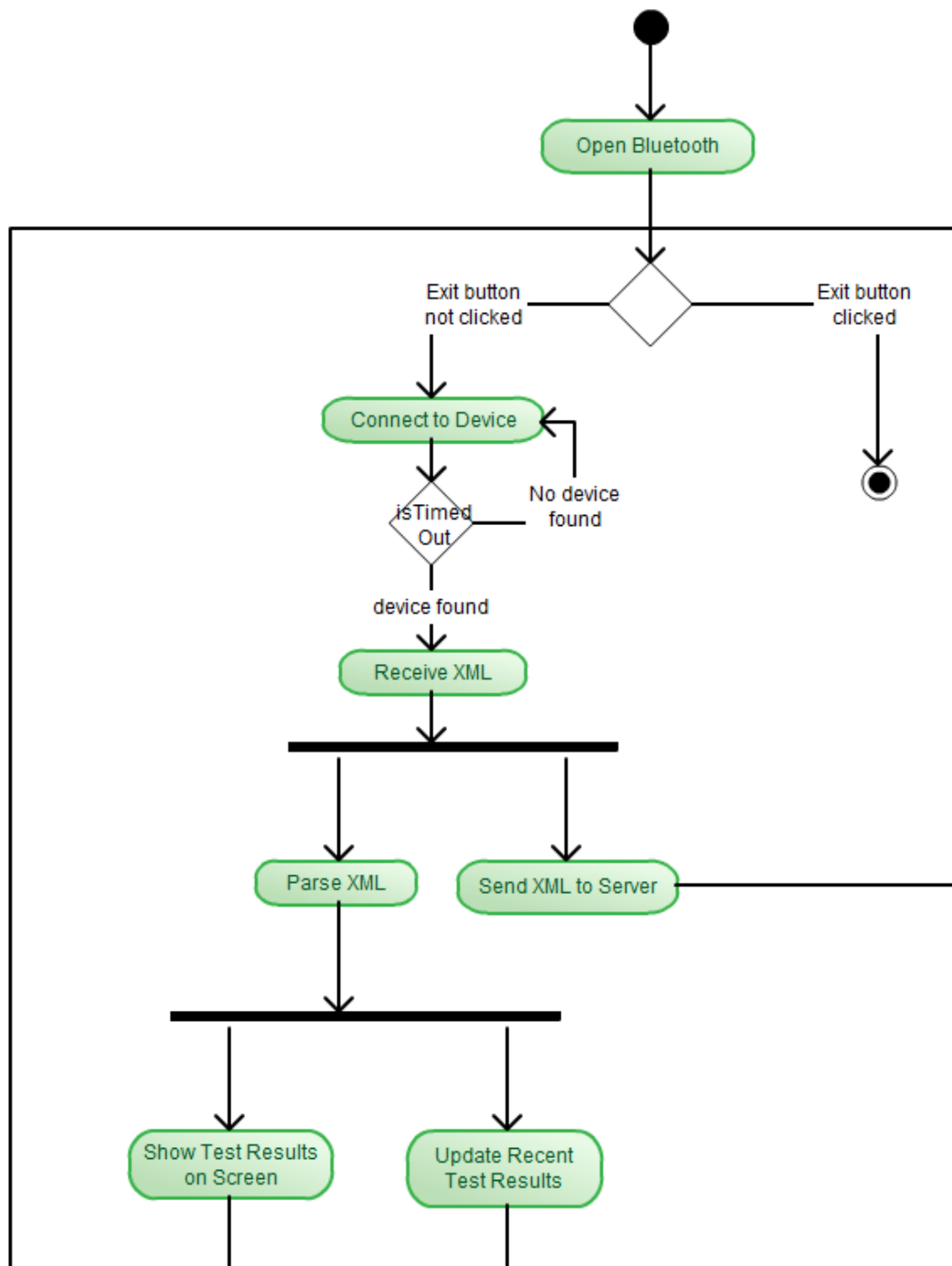
4.2.2.1 Mobile

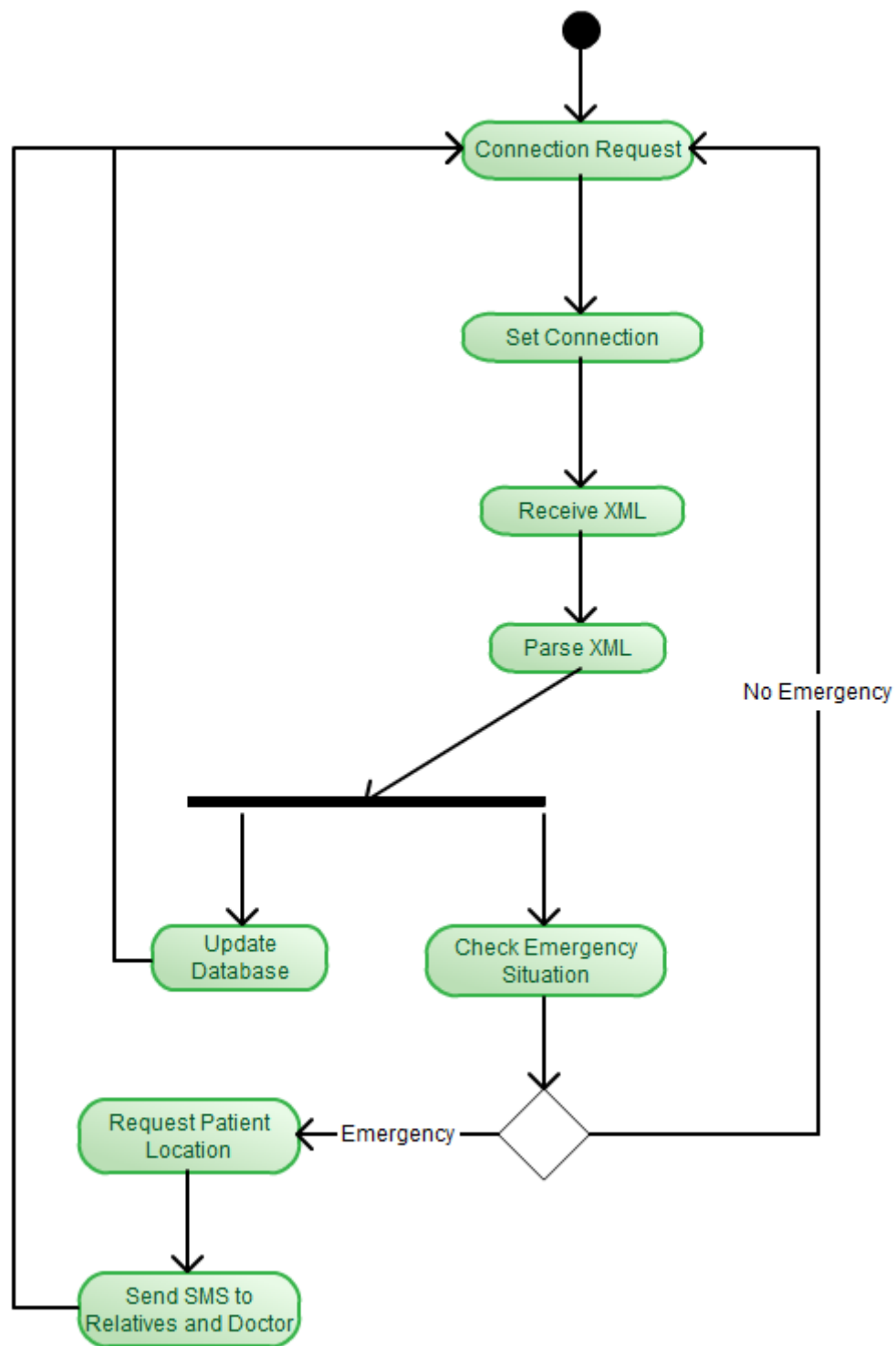**Figure 17 - Mobile Activity Diagram**

4.2.2.2 Server

Figure 18 - Server Activity Diagram

4.2.2.3 Web Page

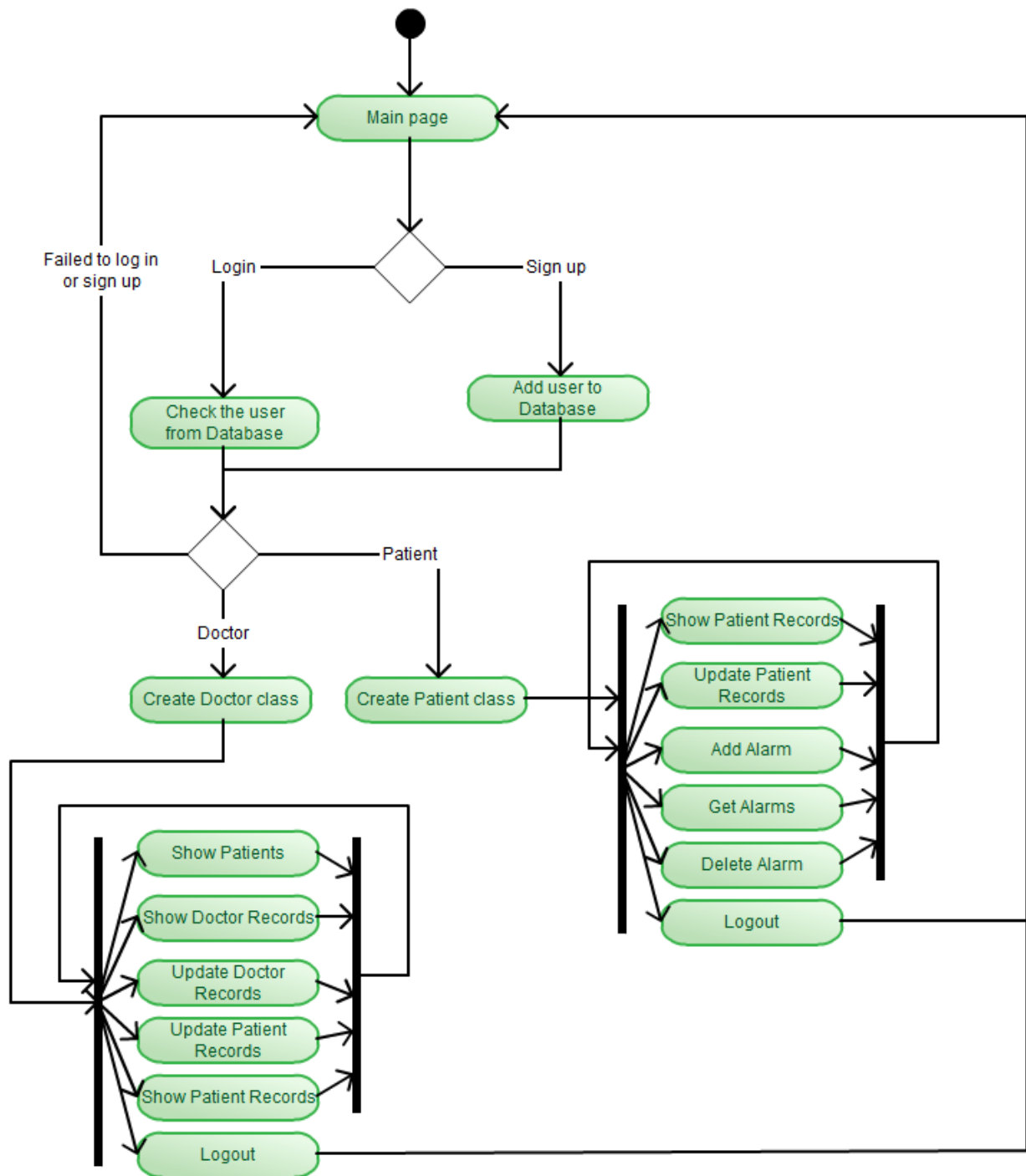**Figure 19 - Web Page Activity Diagram**

4.2.2.4 Alarm
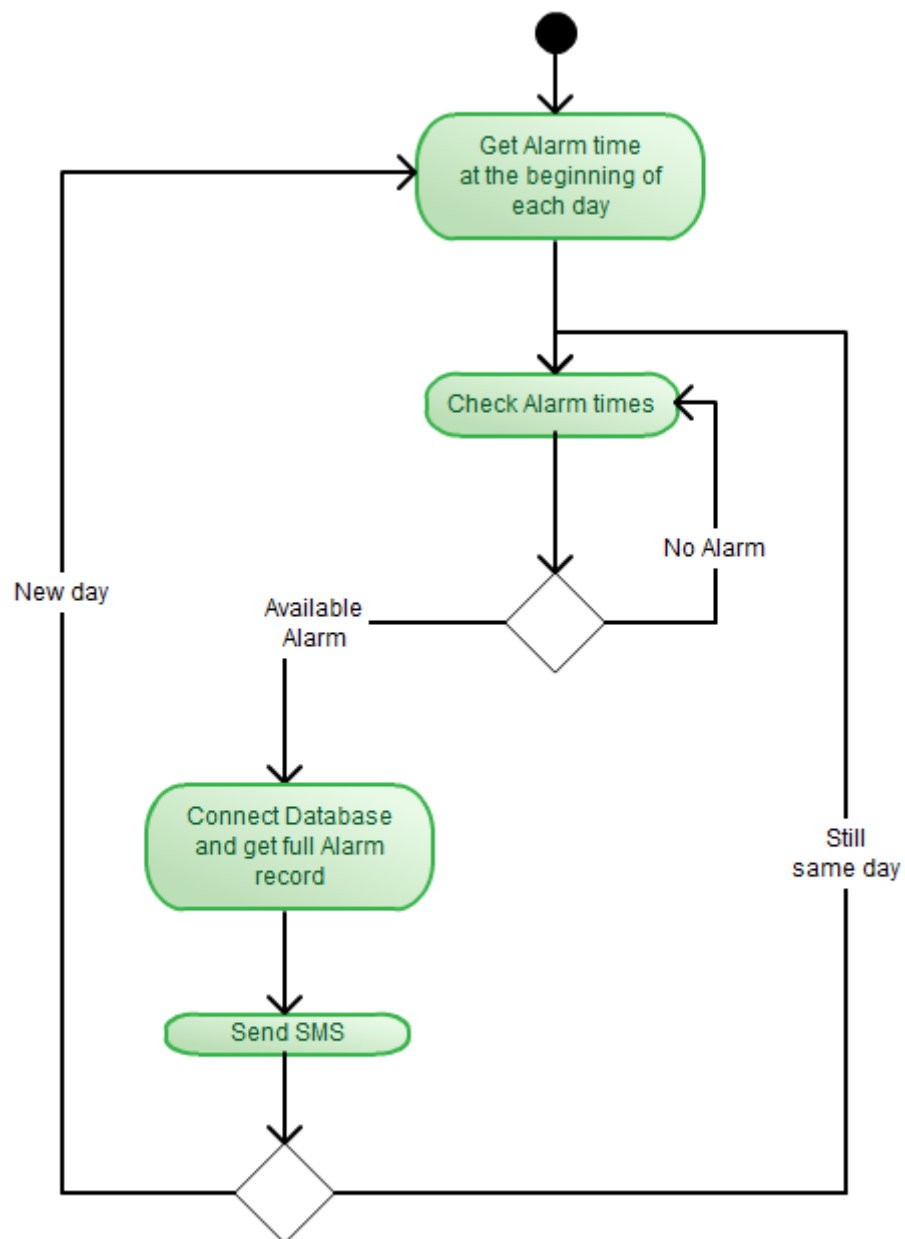
Figure 20 - Alarm Activity Diagram

4.2.2.5 Conversation
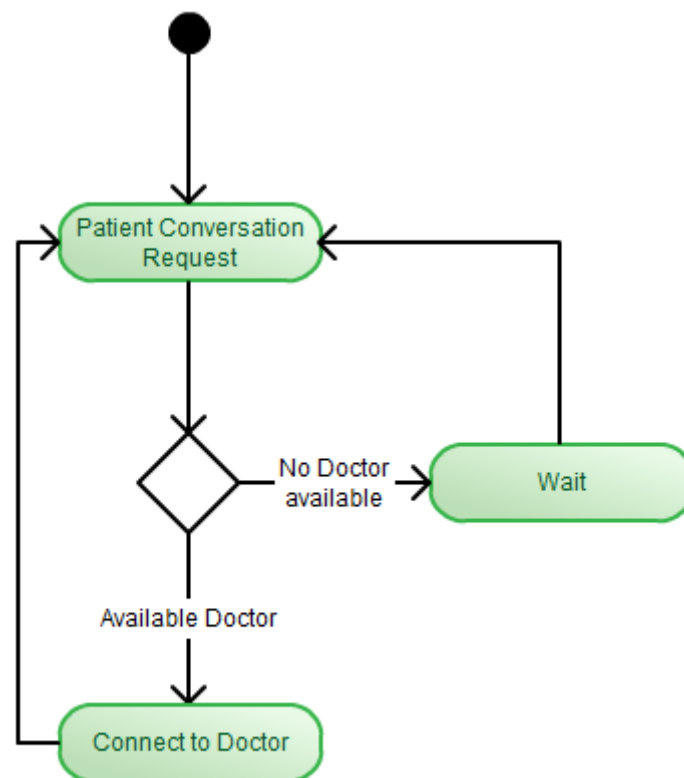
Figure 21 - Conversation Activity Diagram

5 User Interface

5.1 Web Page

Web Page interfaces are shown in the figures below respectively.

5.1.1 Login / Signup Page

Figure 22 - Login / Sign up Page

5.1.2 Patient Home Page

Figure 23 - Patient Home Page

5.1.3 Doctor Home Page
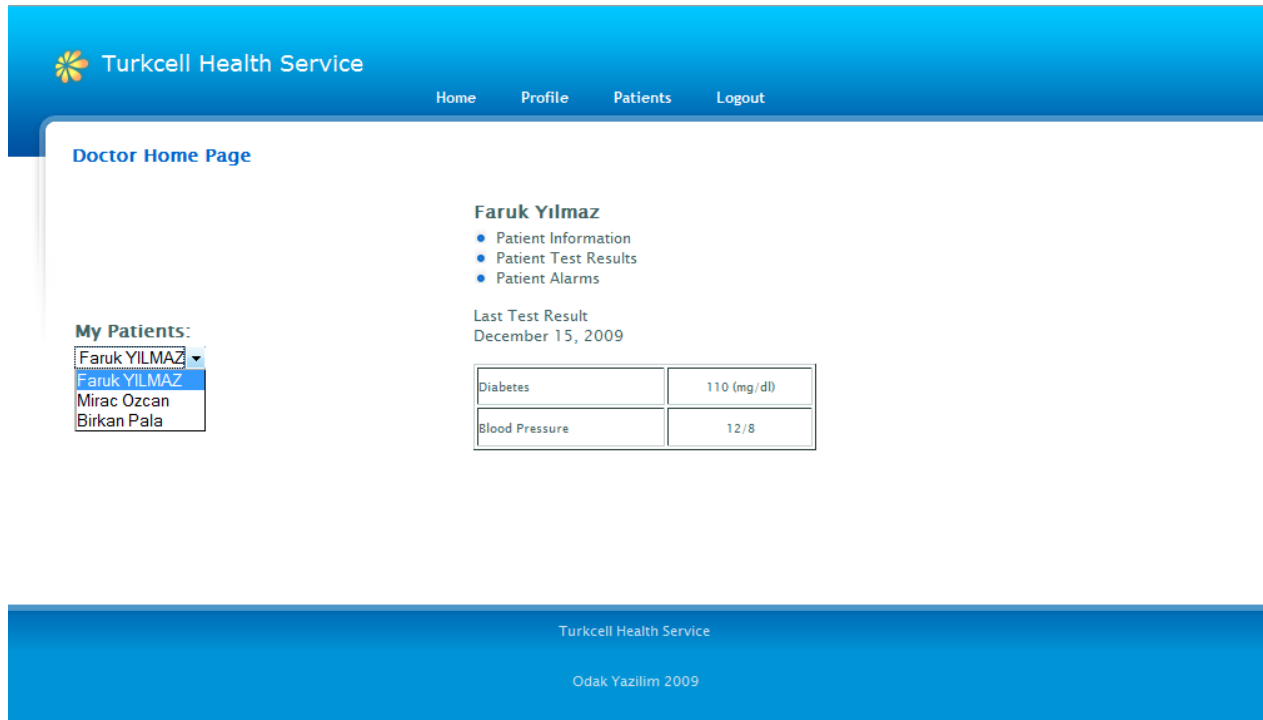
Figure 24 - Doctor Home Page

5.2 Mobile Application

Mobile application interfaces are shown in the figures below respectively.

5.2.1 Startup and Instant Result

Figure 25 - Mobile Application Startup
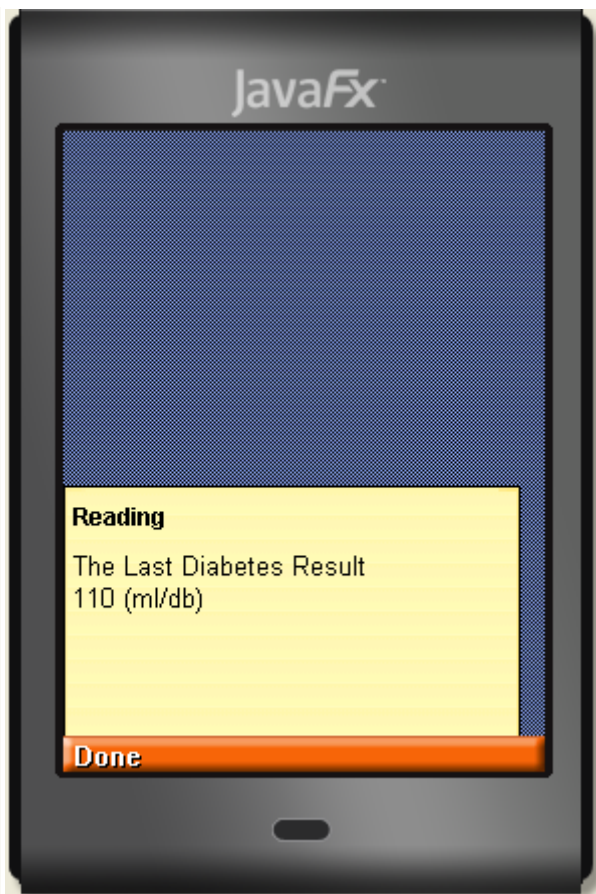
Figure 26 - Mobile Application Instant Result Page

5.2.2 Recent Results
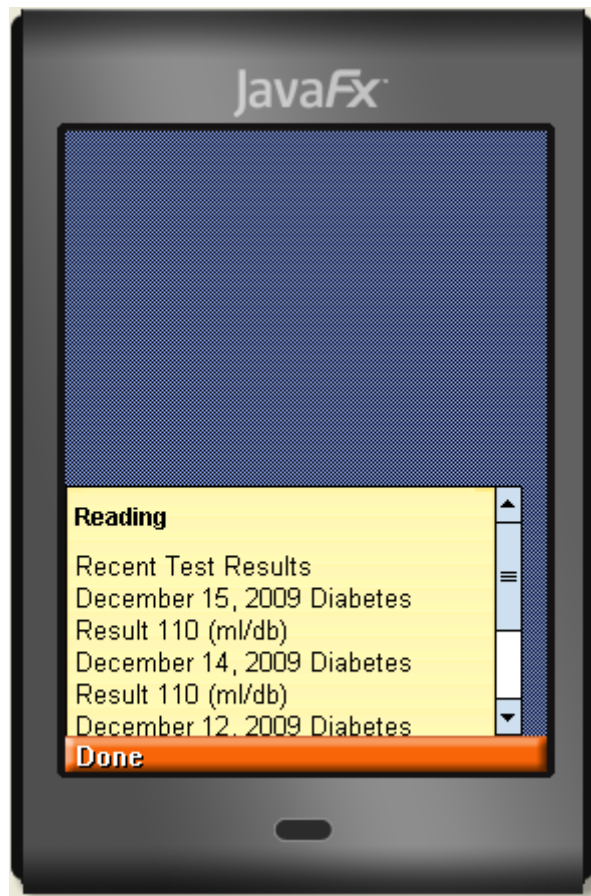
Figure 27 - Mobile Application Recent Results Page

6 Testing

As Odak Yazılım, we believe that testing is very crucial for the project to be completed until the deadline and for the THS to be an error free application. For this reason, we will follow a testing strategy which we think lead us to the success.

The steps we will follow are:

- Unit Testing
- Integration Testing
- System Testing
- System Integration Testing

6.1 Unit Testing

Unit testing will be held at each prototype implementation. During the implementation, after the completion of each module, these modules will be tested. If we find an error or missing, that will be fixed immediately.

6.2 Integration Testing

After each module is tested, these modules will be integrated. And integration testing will be done to expose defects in the interfaces and interaction between integrated modules. After the errors and missing things are detected, we will fix the errors so that our prototype is error free and complete.

So, each integration testing means a prototype is completed. Then, we can start to design the next prototype, implement and start from unit testing.

6.3 System Testing

We will do system testing when our system is complete, that is all prototypes are implemented gradually. System testing will help us to learn if THS has met all its requirements. If there are some errors and missing things, they will be resolved as soon as possible.

6.4 System Integration Testing

When we are ready to integrate our system to a third party system, we need to do system integration testing. After the complete testing, our THS is ready to meet its users.

7 Project Schedule

The schedule of the THS project is shown in the figure below.

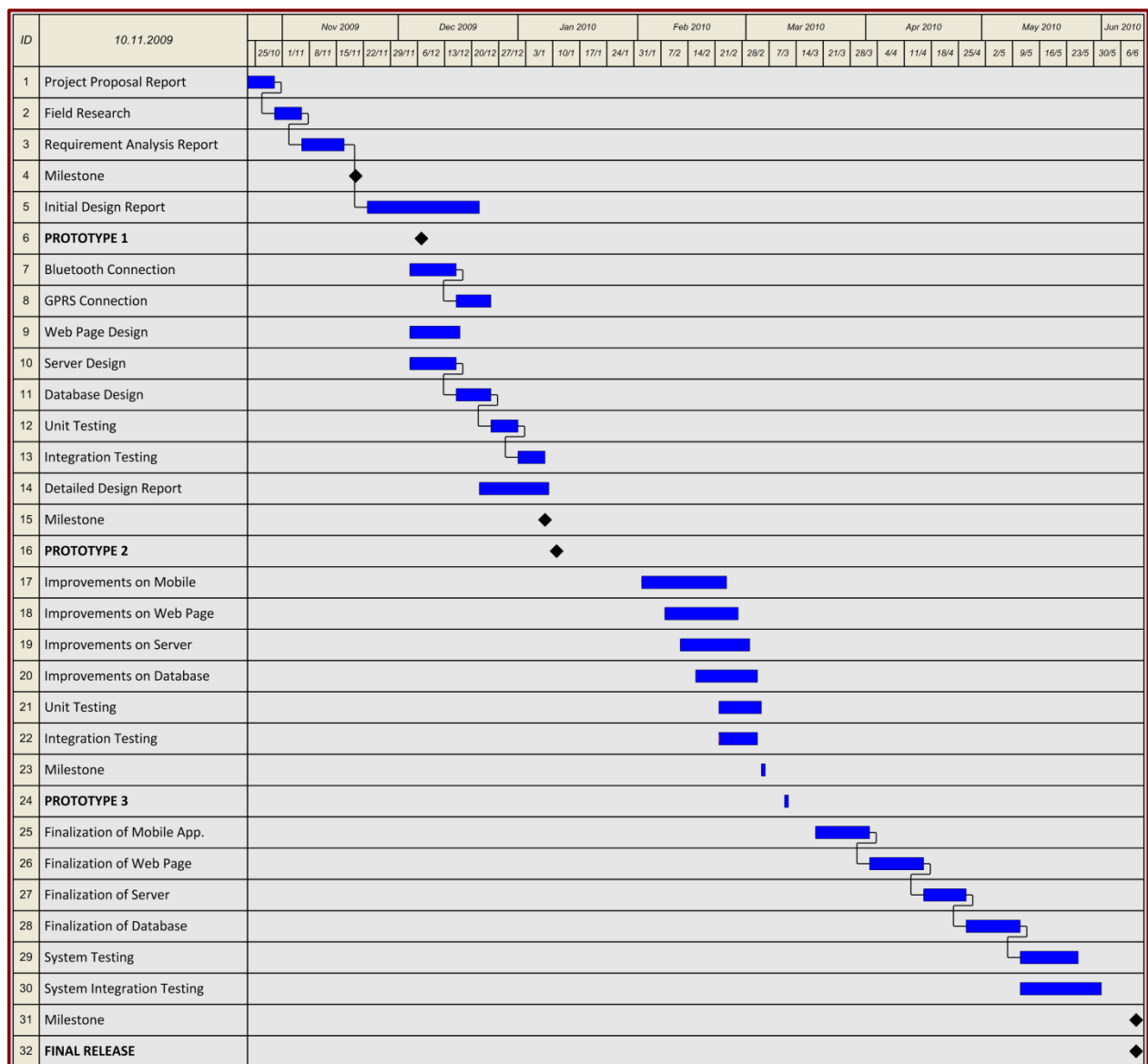| ID	10.11.2009	Nov 2009	Dec 2009	Jan 2010	Feb 2010	Mar 2010	Apr 2010	May 2010	Jun 2010
1	Project Proposal Report								
2	Field Research								
3	Requirement Analysis Report								
4	Milestone								
5	Initial Design Report								
6	**PROTOTYPE 1**								
7	Bluetooth Connection								
8	GPRS Connection								
9	Web Page Design								
10	Server Design								
11	Database Design								
12	Unit Testing								
13	Integration Testing								
14	Detailed Design Report								
15	Milestone								
16	**PROTOTYPE 2**								
17	Improvements on Mobile								
18	Improvements on Web Page								
19	Improvements on Server								
20	Improvements on Database								
21	Unit Testing								
22	Integration Testing								
23	Milestone								
24	**PROTOTYPE 3**								
25	Finalization of Mobile App.								
26	Finalization of Web Page								
27	Finalization of Server								
28	Finalization of Database								
29	System Testing								
30	System Integration Testing								
31	Milestone								
32	**FINAL RELEASE**								

Figure 28 - Project Schedule

8 Conclusion

8.1 What has been done so far?

Until now, we set up the J2ME SDK, NetBeans IDE and other necessary environment. We try to figure out how a mobile application is implemented and develop small J2ME applications. We developed an application which has server and client sides that can be connected to each other via Bluetooth. Server side is the computer and the client side is the mobile phone and the server can send an XML file to the client. Moreover, we implemented another J2ME

application which can parse an XML file and display on the phone screen. Then we combined the client and XML parser applications.

In addition, for the server and web page module, we set up JBOSS, MySQL, Apache and Macromedia Dreamweaver IDE. We examined how they work and developed a small part of our project. In detailed, we construct our database, designed our web interfaces and connected them to each other using JSP Servlet technology.

8.2   Future Work

The main flow of the project can be examined in the Gantt chart section. In this section, near future works for the prototype implementation will be discussed.

In the following weeks, till the day which the demo will take place; we will be working on the prototype. We will improve the applications we have done so far and make suitable them to the demo. The target prototype will be developed by integration of separate prototypes. The team has currently developed some applications for the *Mobile* module and *Server* module. Implementing this prototype, we will certainly face with some problems; those will help us to clarify the final construction elements of our system. The necessary changes will be done and the details of the classes, sequence etc. will be determined. Later after the first run of our system prototype, we will be focusing to the details of the interface of the web page and mobile application. This part is also important to reach and interact with the users. Debugging and reimplementation will be done to maintain a consistence and stability for our system. All these works will help us to see problems may appear and to get experience on the project. After all, the final design report will be written that will provide us to start implementing the project in the second term.

References

1. Spiral Model, Wikipedia. http://en.wikipedia.org/wiki/Spiral_model
2. Bodytel Scientific Inc. http://www.bodytel.com/en/patient/products/the_system.html