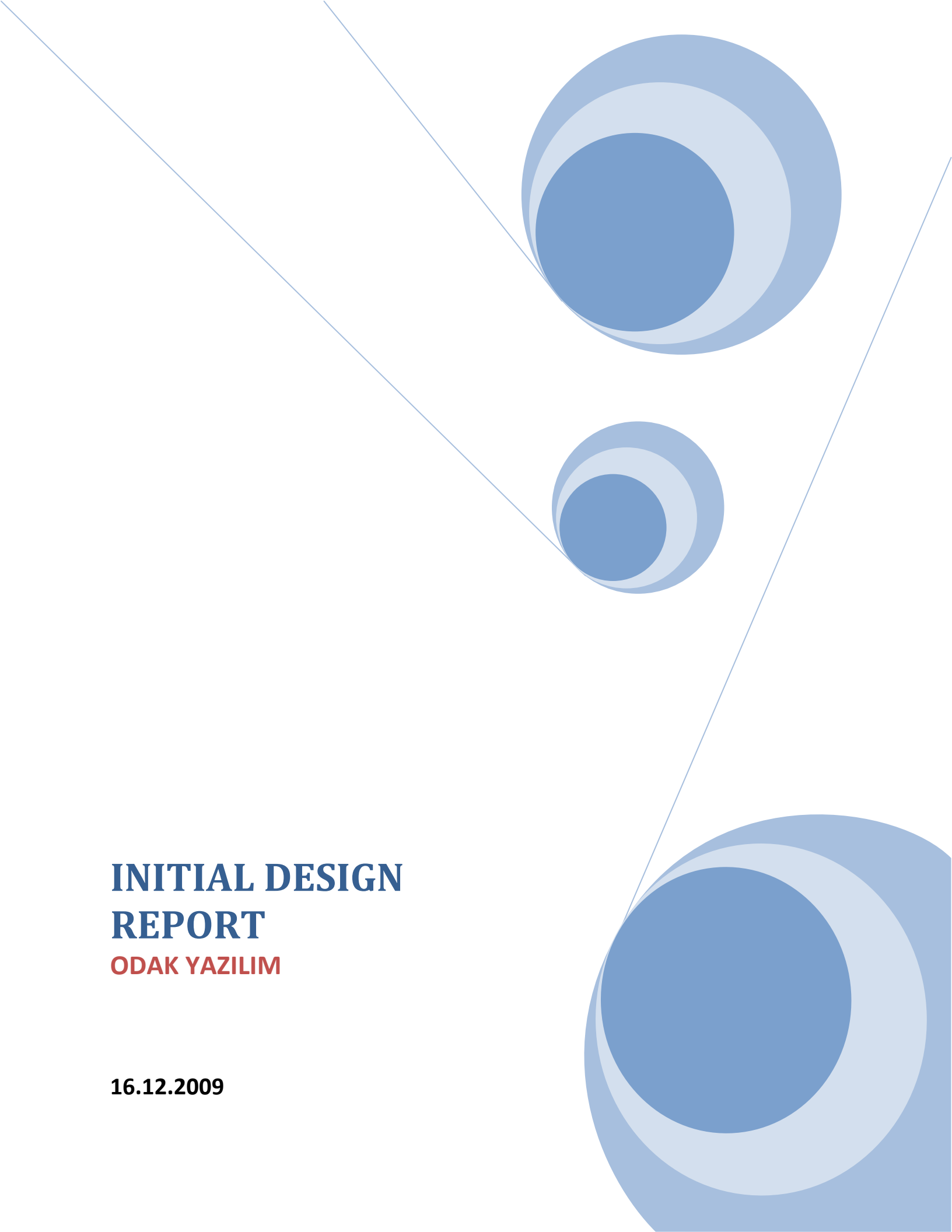

INITIAL DESIGN REPORT

ODAK YAZILIM

16.12.2009

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,*** our aim is to facilitate health sector.

1.2 Project Features

Turkcell Health Service *(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. Doctors can write out the prescription which will be sent to social security institution.

- Tracking service

  Chronic patients are controlled with a device which patients should use 7/24. If the measurements of the patients exceed critical regions, service will inform doctor and

person who is selected 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.

1.3 Purpose of Document

The purpose of this document is to express the initial design issues about our project.

According to the IEEE standards the Initial Design Report shows how the software system will be structured to satisfy the requirements identified in the Requirement Analysis Report. In other words, the requirements mentioned in the initial design report will be translated to the structural components which will be used in the implementation phase. 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.

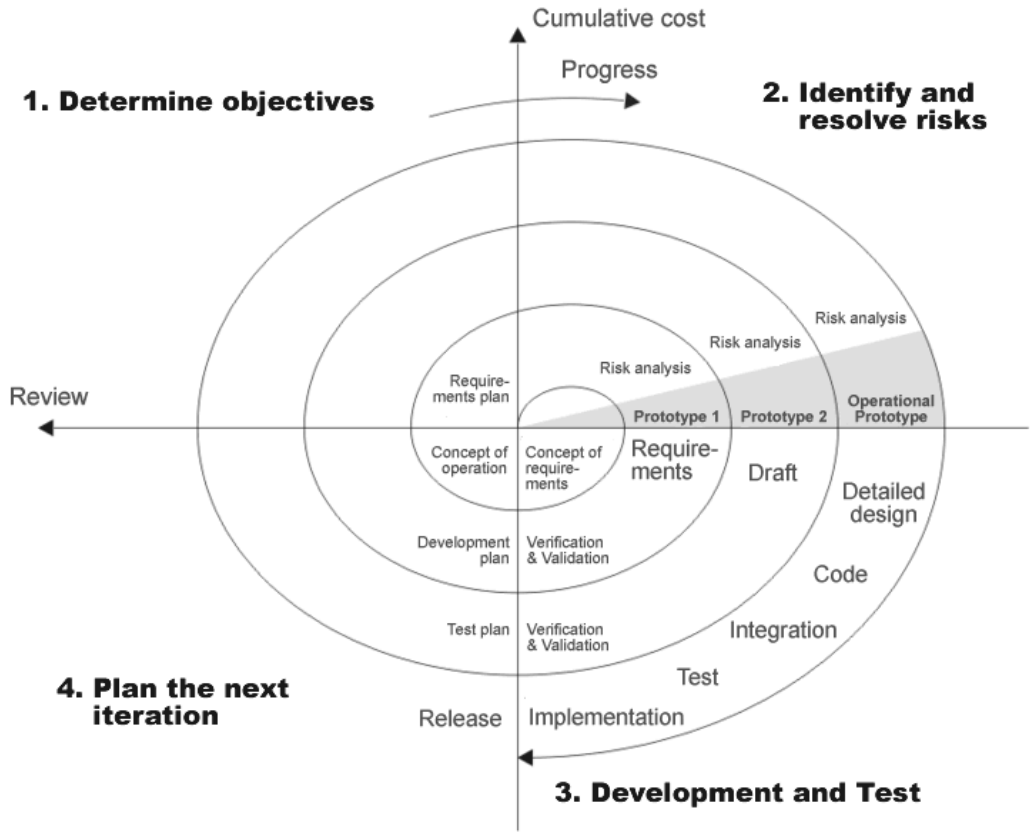

Figure 1 - Spiral Model

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. 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 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. Profiles will only be accessed by doctors and they are not accessible for any other user.

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. Because J2ME includes a set of core libraries that provides most of the functionality available in the core

libraries of the Java programming language, all these reasons led us to use Java language as our applications default developing language. And also we will develop our mobile application THS in NetBeans IDE.

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.

3 System Design

3.1 Architectural Design

3.1.1 Data Flow Diagrams

3.1.1.1 Level 0

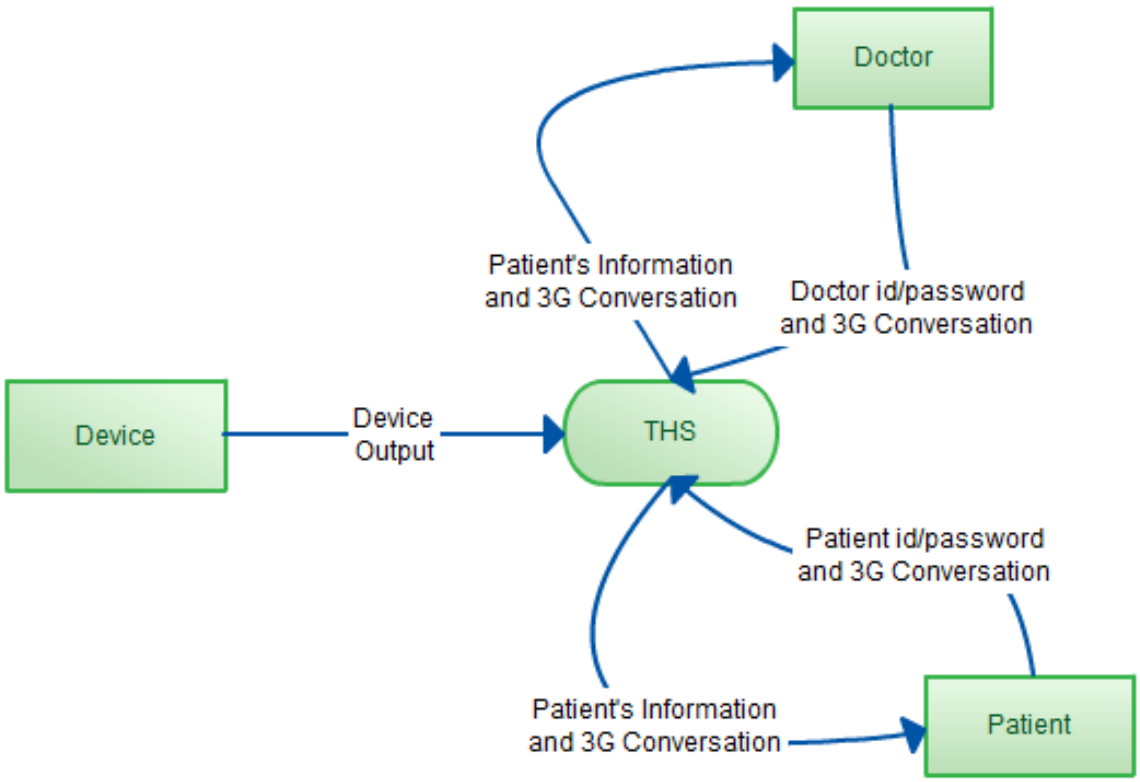

Figure 2 - Data Flow Diagram Level 0

3.1.1.2 Level 1

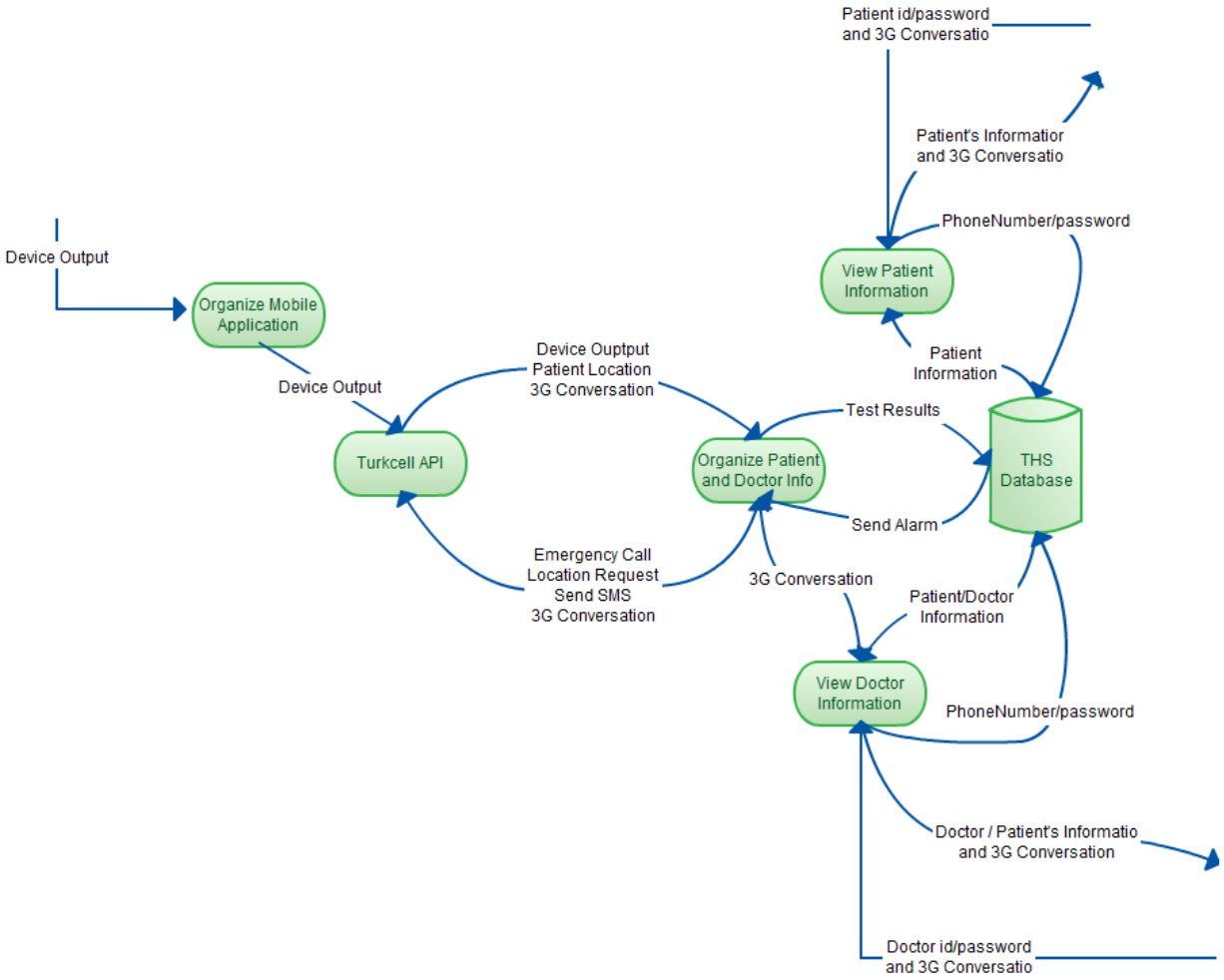

Figure 3 - Data Flow Diagram Level 1

3.1.2 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

3.1.2.1 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 3G connection.

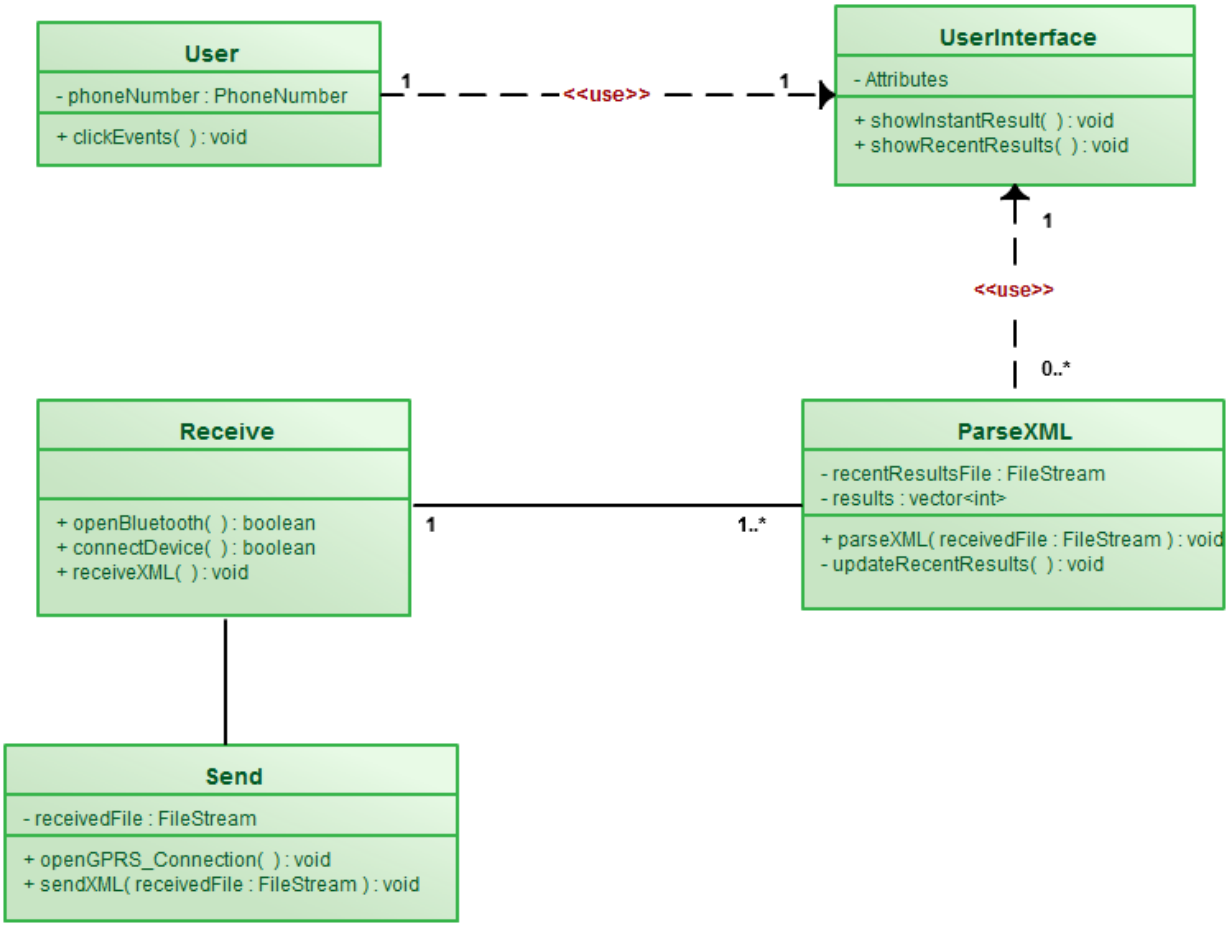

Figure 4 - Mobile Module Class Diagram

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(), updateRecentResults()* and *checkEmergency().* 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.

3.1.2.2 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 calling relatives and 112 Emergency Service in case of any unexpected situation.

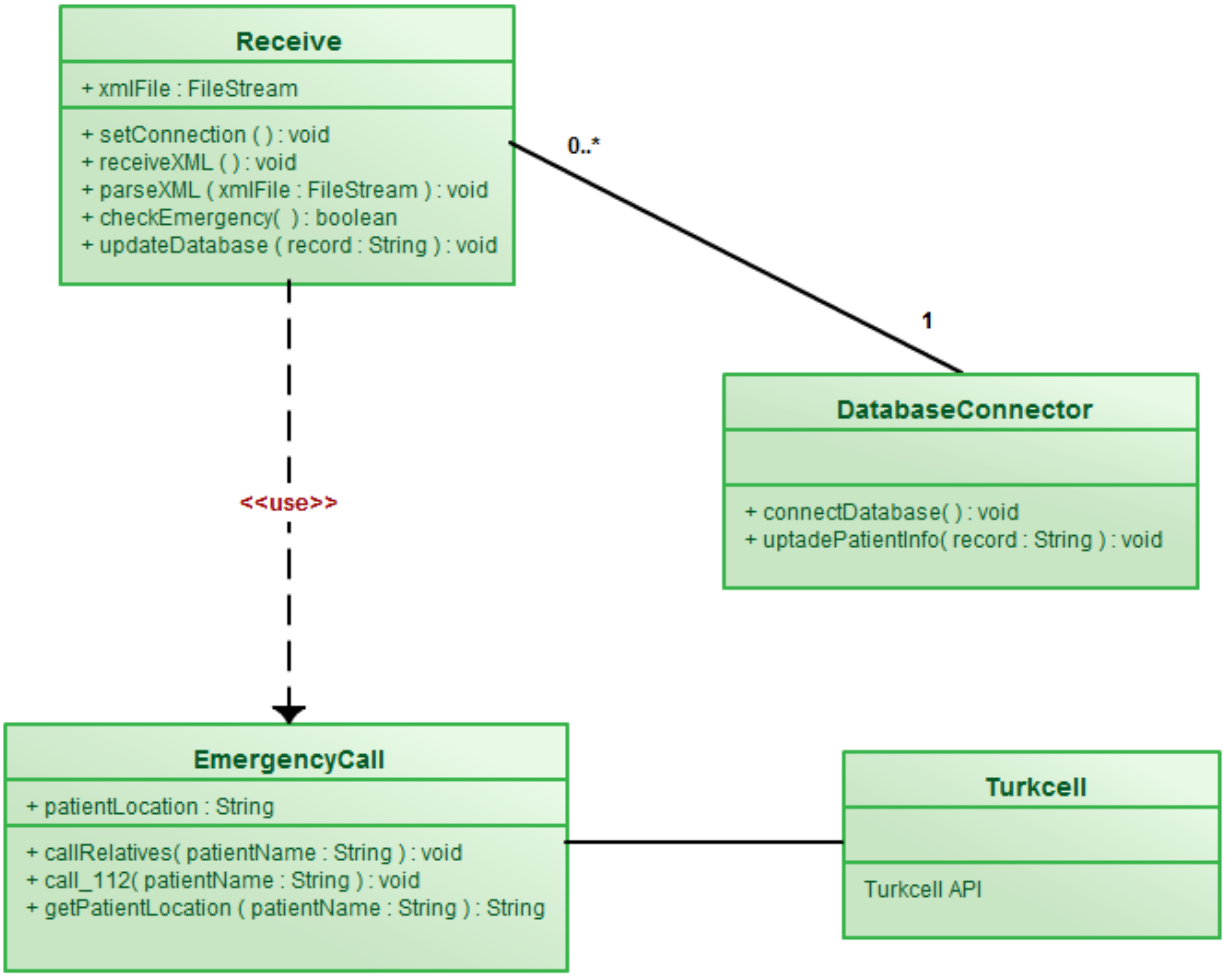

Figure 5 - Server Module Class Diagram

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

- **Receive Class**

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 it may save lives by calling relatives and 112 Emergency Service immediately in any unexpected situation. It will ensure the location of patient with Turkcell API.

- **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.

3.1.2.3 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.

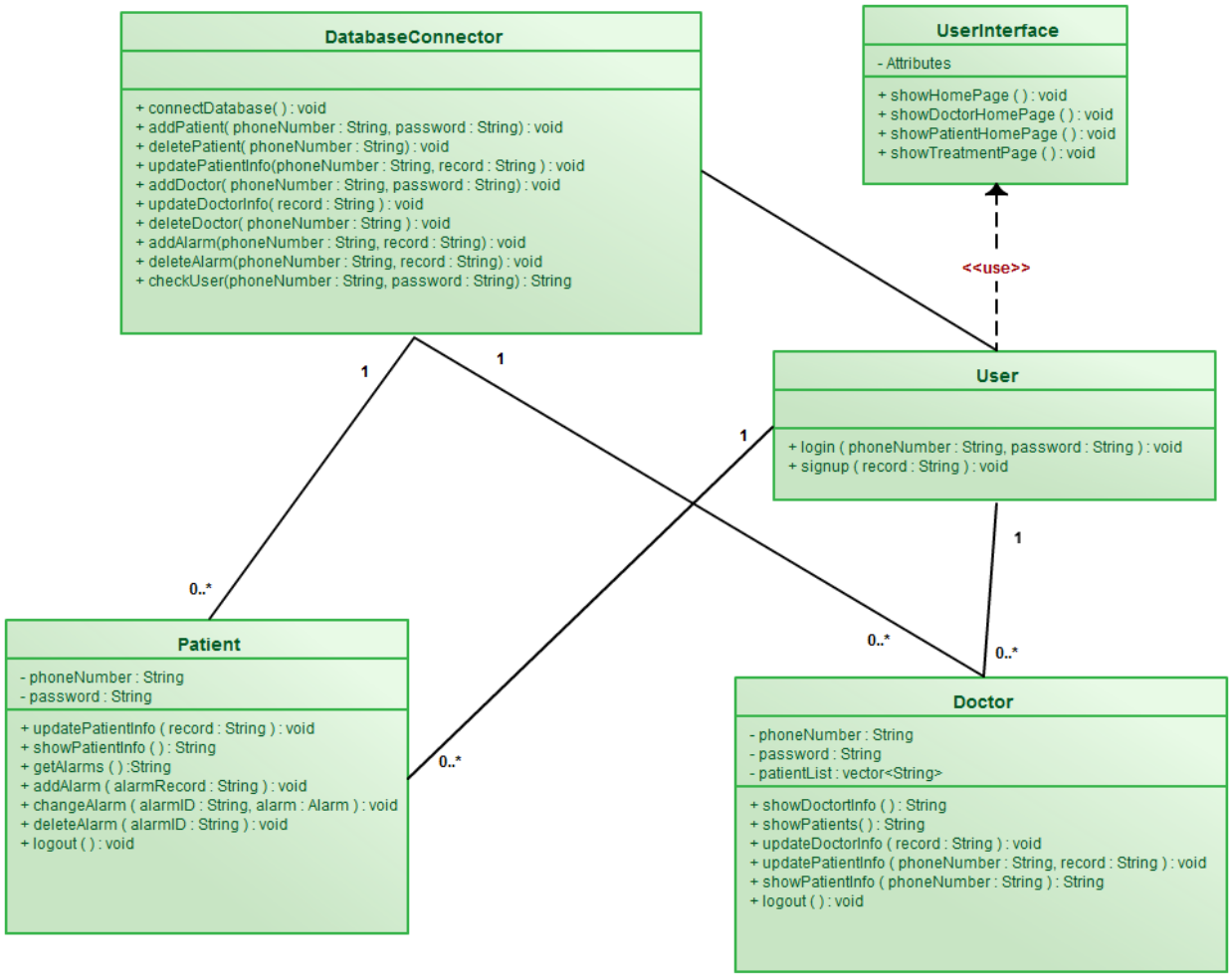

Figure 6 - Web Page Module Class Diagram

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 updates, gets, sets patient's or doctor's records. These changes are reflected to the database with *DatabaseConnecter* class methods.

3.1.2.4 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.

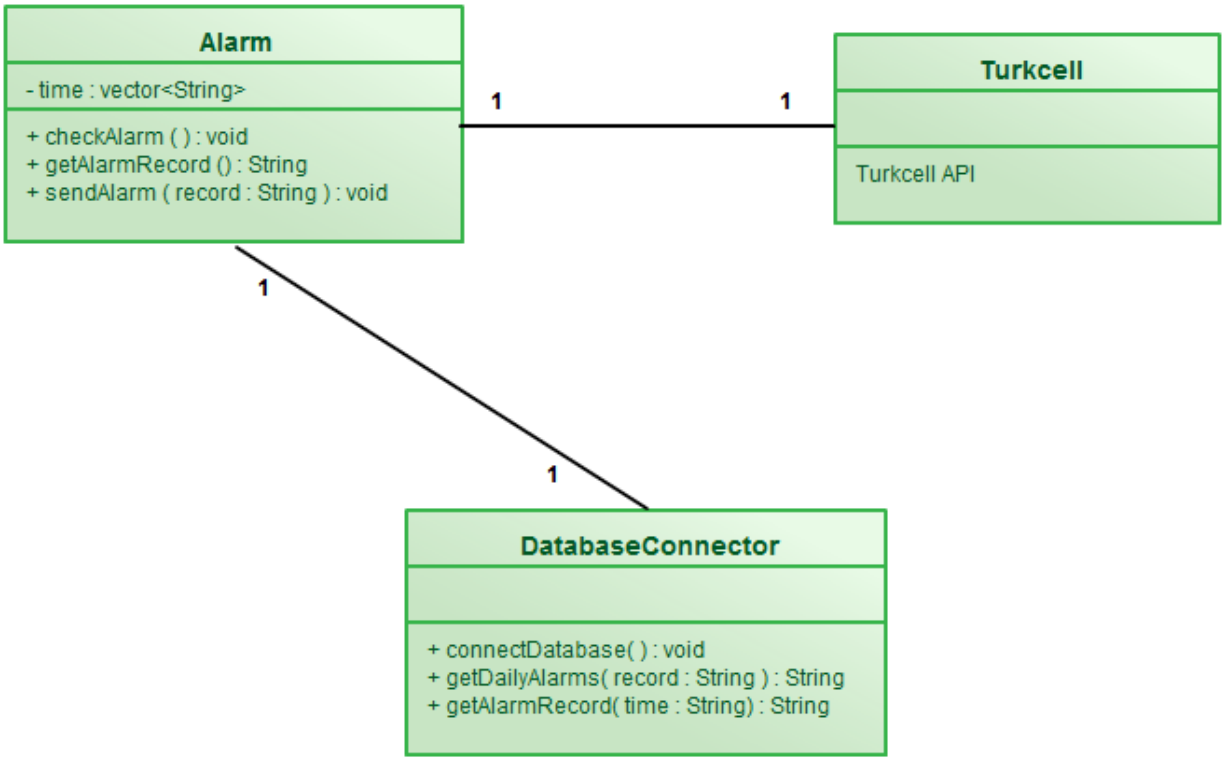

Figure 7 - Alarm Module Class Diagram

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.

3.1.2.5 Conversation Module

This module provides a communication between patient and doctors via 3G.

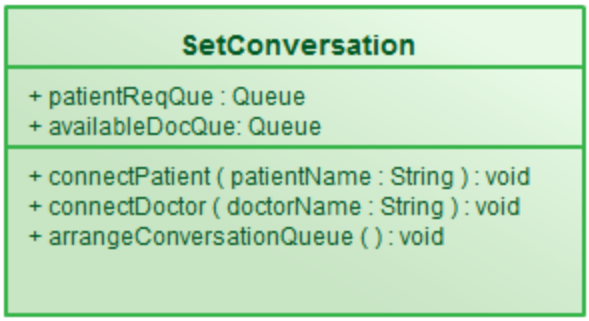

Figure 8 - Conversation Module Class Diagram

- **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.

3.1.3 ER Diagrams

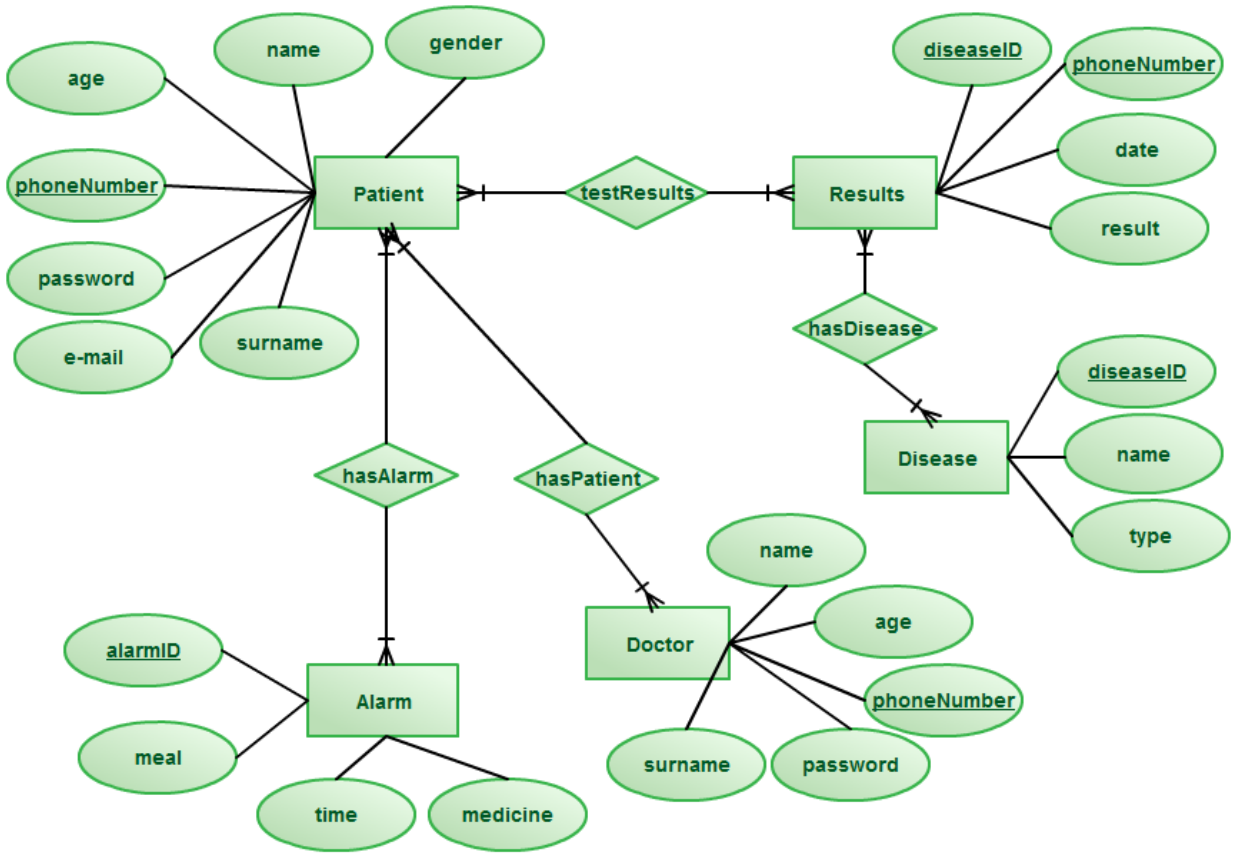

Figure 9 - Entitiy Relationship Diagram

3.1.4 Database Schemas

3.1.4.1 Patient Table

Field	Type	Null	Foreign Key	References
phoneNumber	Varchar(10)	No	No	-

(P.K)				
password	Varchar(10)	No	No	-
name	Varchar(30)	No	No	-
surname	Varchar(20)	No	No	-
age	Integer	No	No	-

3.1.4.2 Disease Table

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

3.1.4.3 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	-
age	Integer	No	No	-

3.1.4.4 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	-

3.1.4.5 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	-

3.1.4.6 HasDisease Table

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

3.1.4.7 HasAlarm Table

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

3.1.4.8 HasPatient

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

3.1.4.9 TestResults

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

3.2 Procedural Design

3.2.1 Sequence Diagrams

3.2.1.1 Mobile

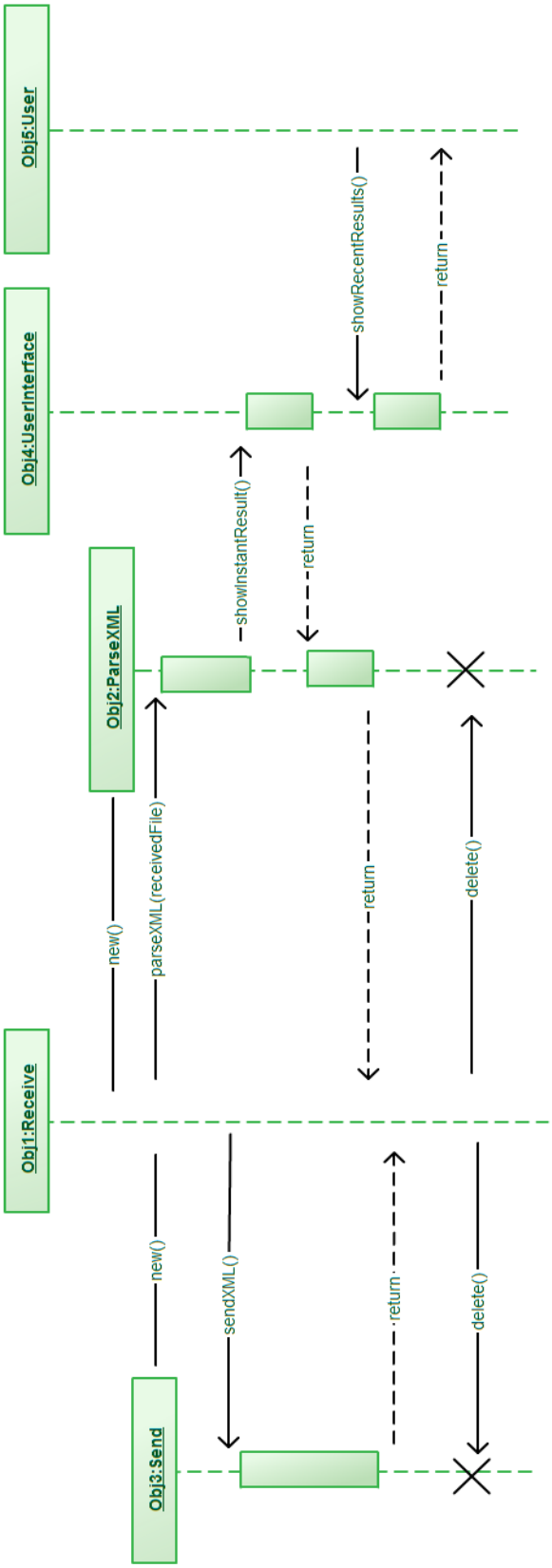

Figure 10 - Mobile Sequence Diagram

3.2.1.2 Server

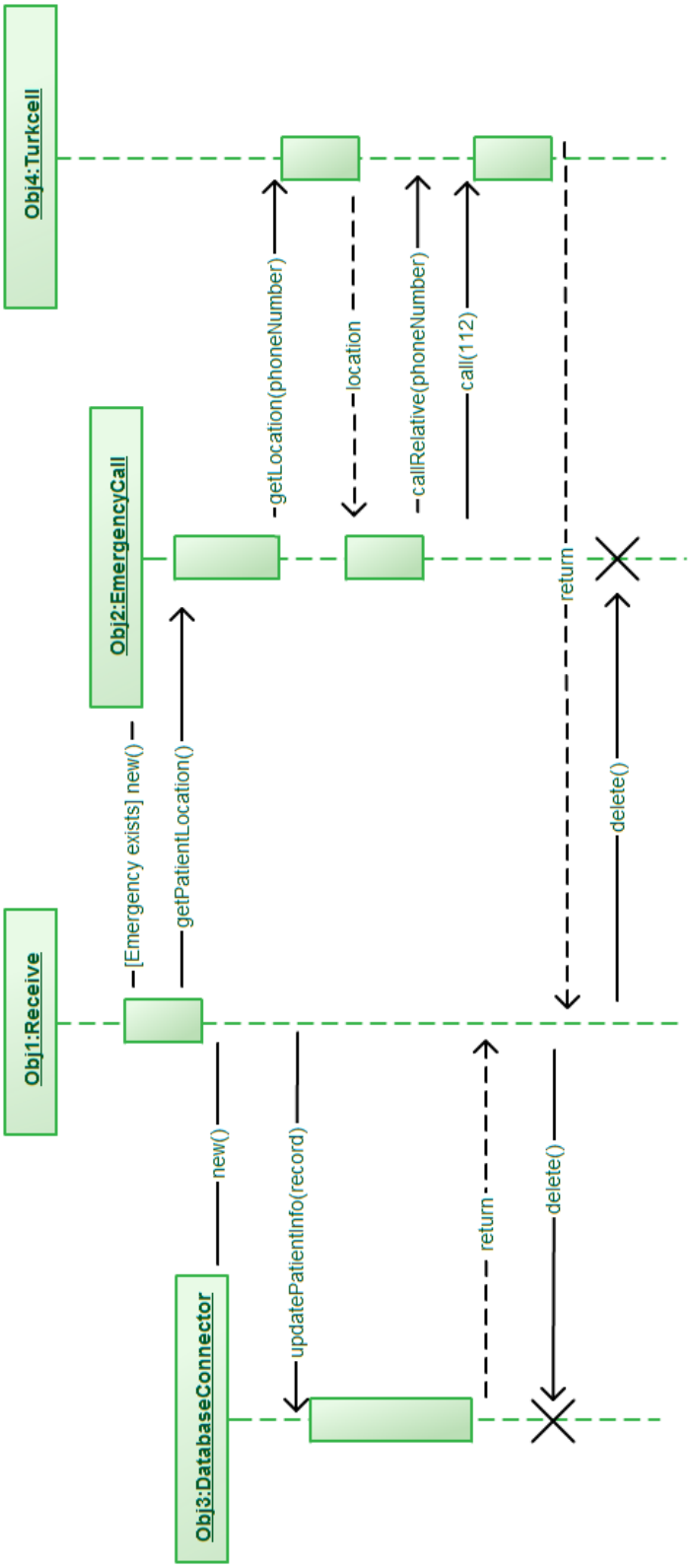

Figure 11 - Server Sequence Diagram

3.2.1.3 Web Page

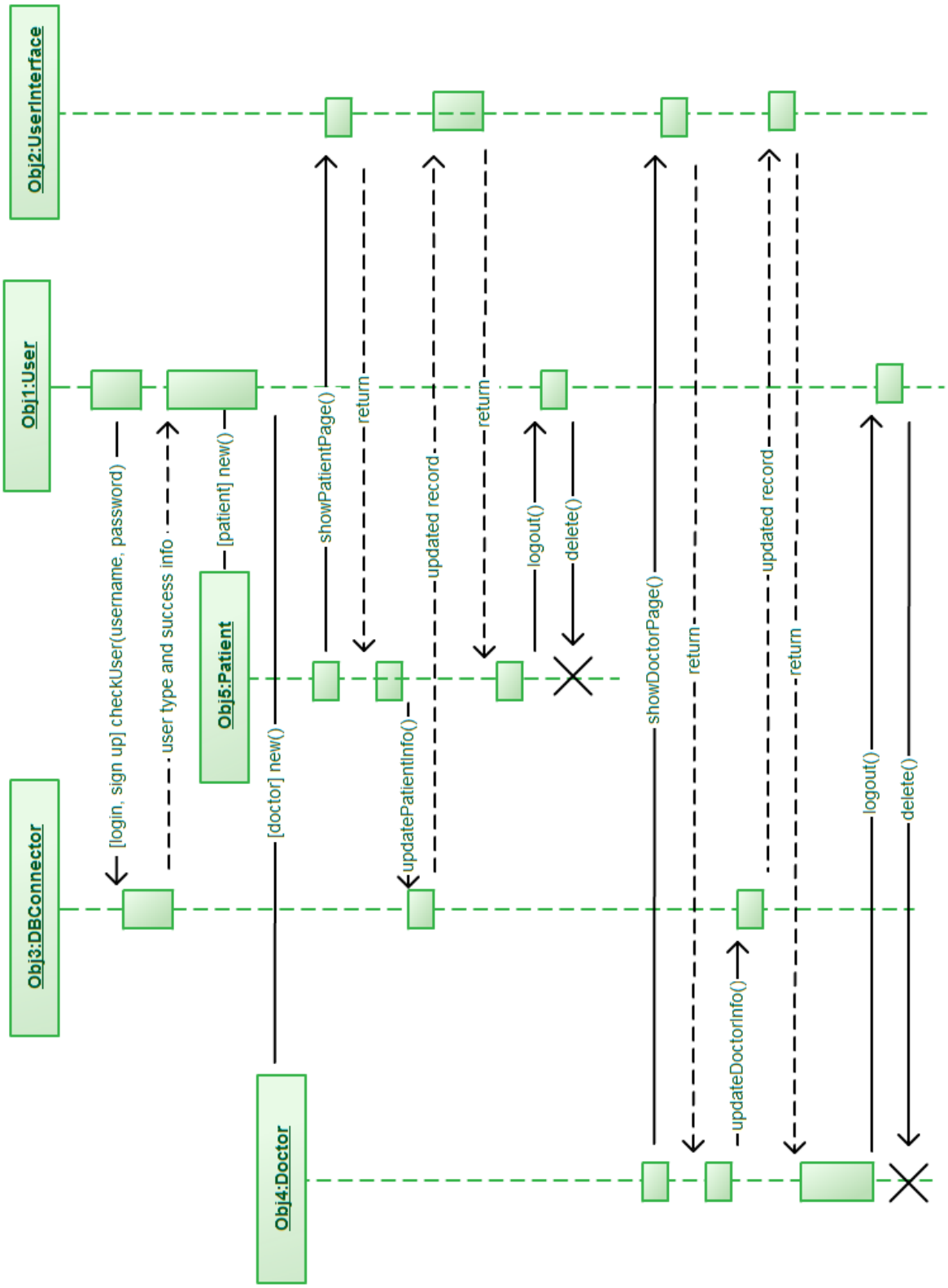

Figure 12 - Web Page Sequence Diagram

3.2.1.4 Alarm

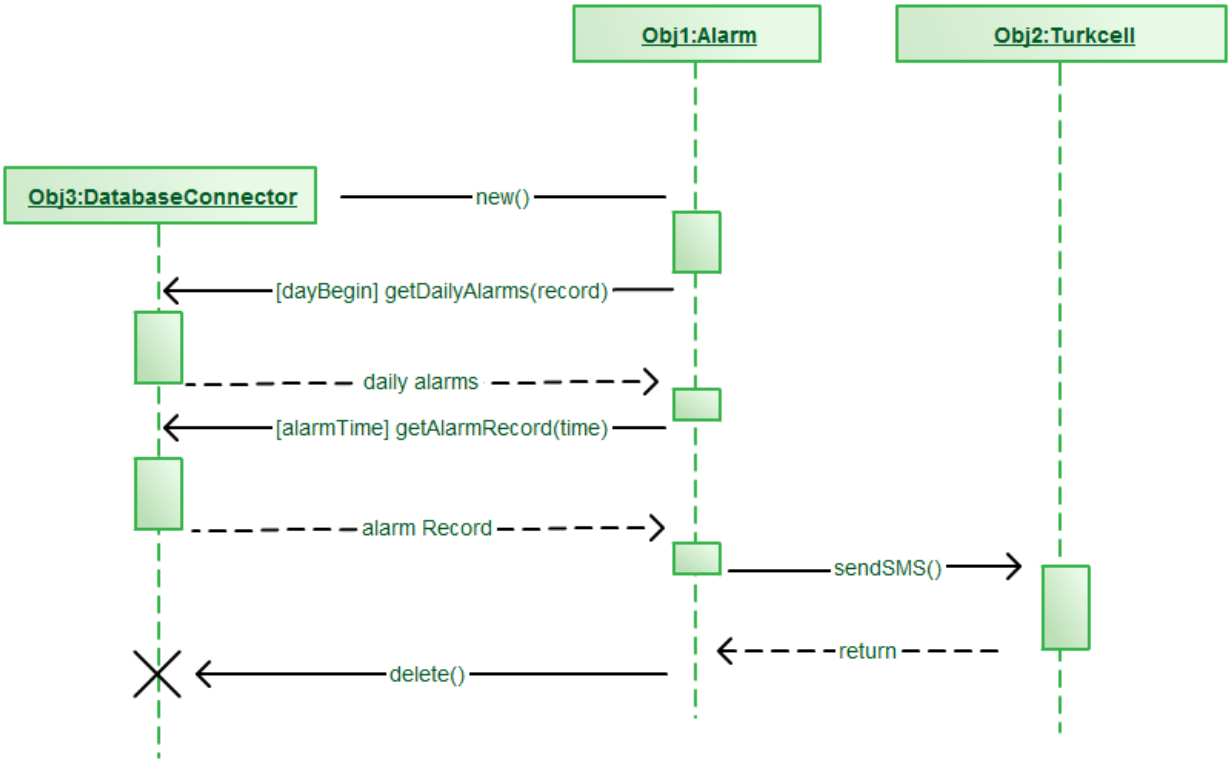

Figure 13 - Alarm Sequence Diagram

3.2.2 Activity Diagrams

3.2.2.1 Mobile

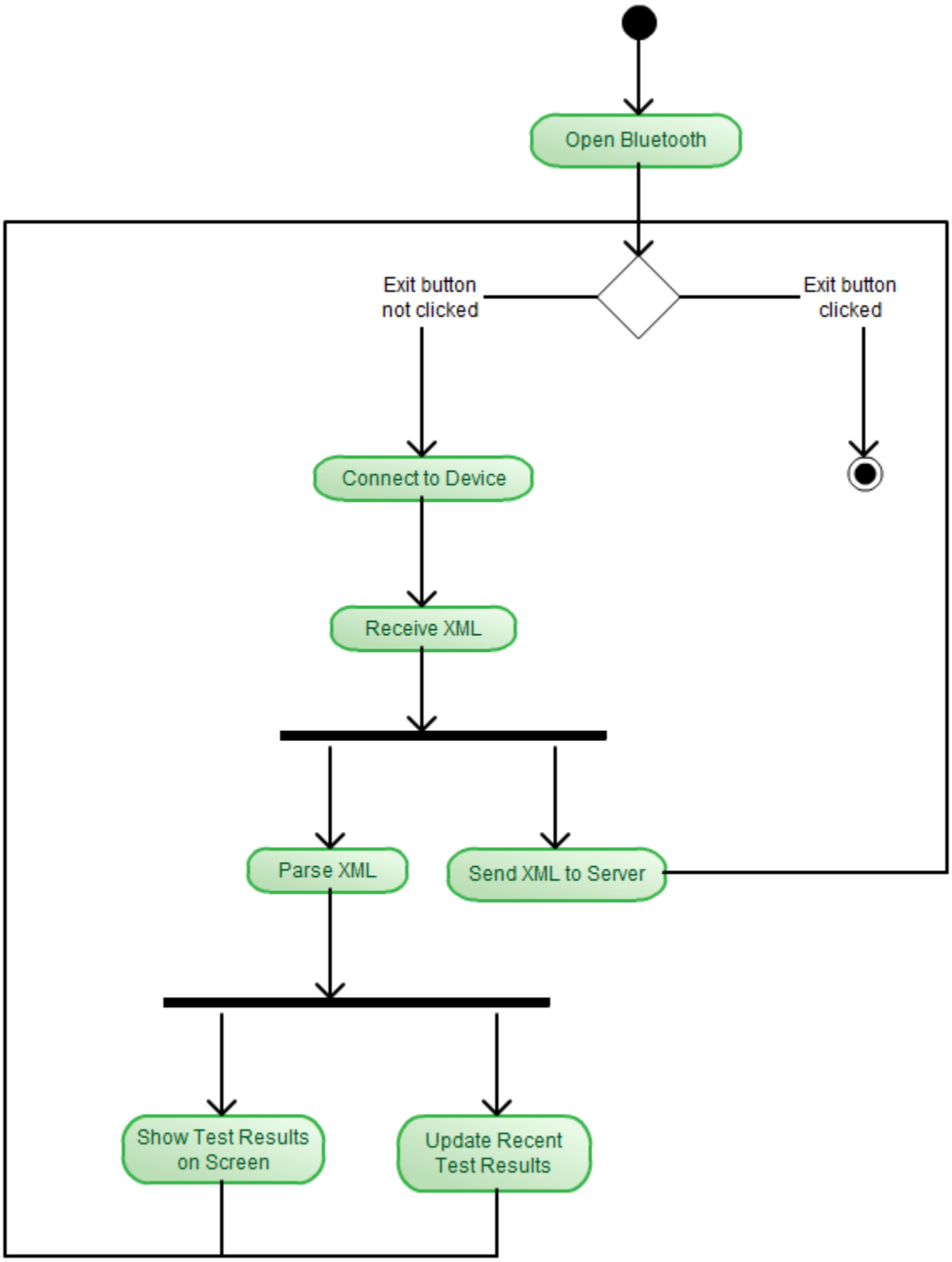

Figure 14 - Mobile Activity Diagram

3.2.2.2 Server

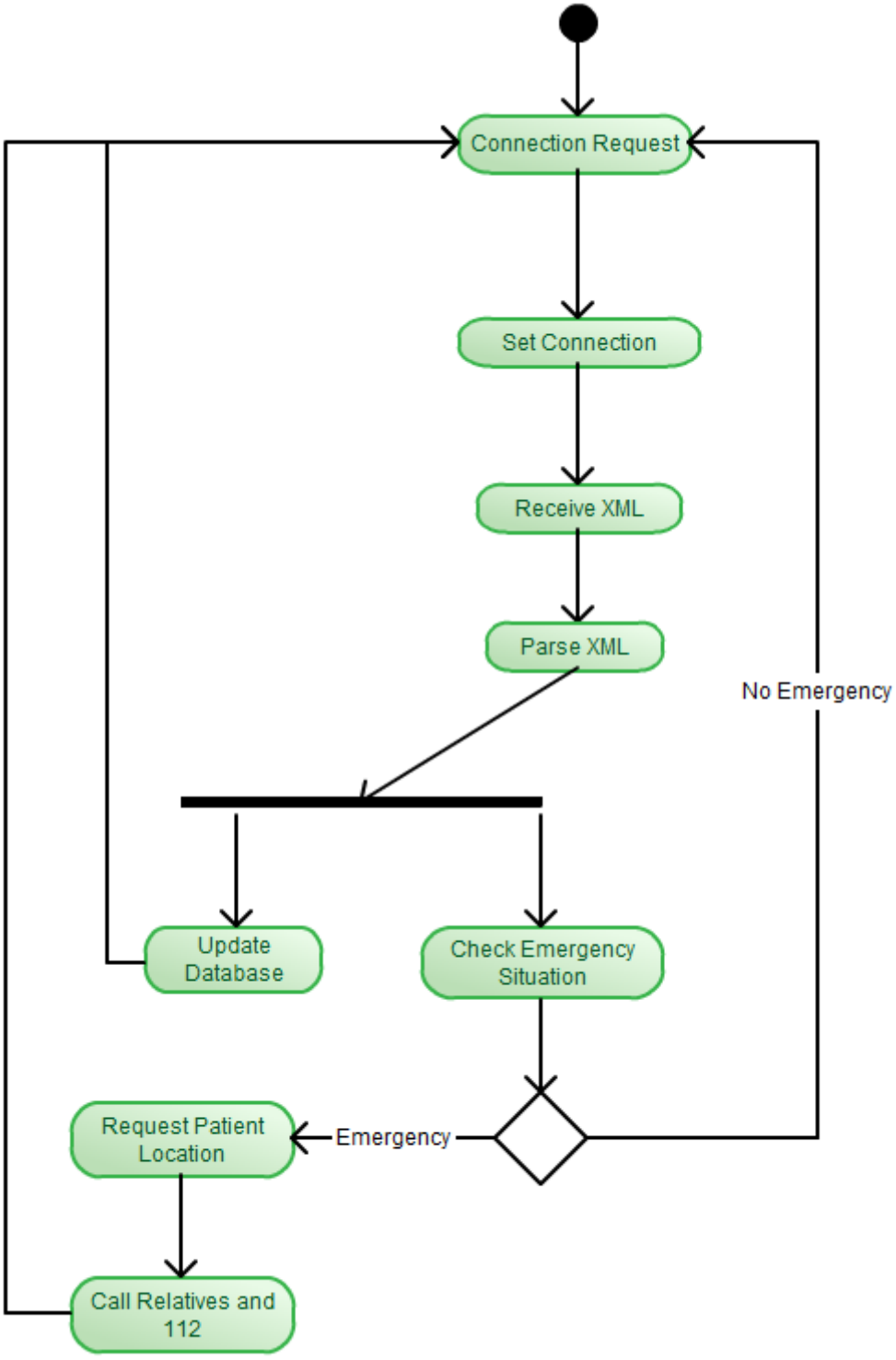

Figure 15 - Server Activity Diagram

3.2.2.3 Web Page

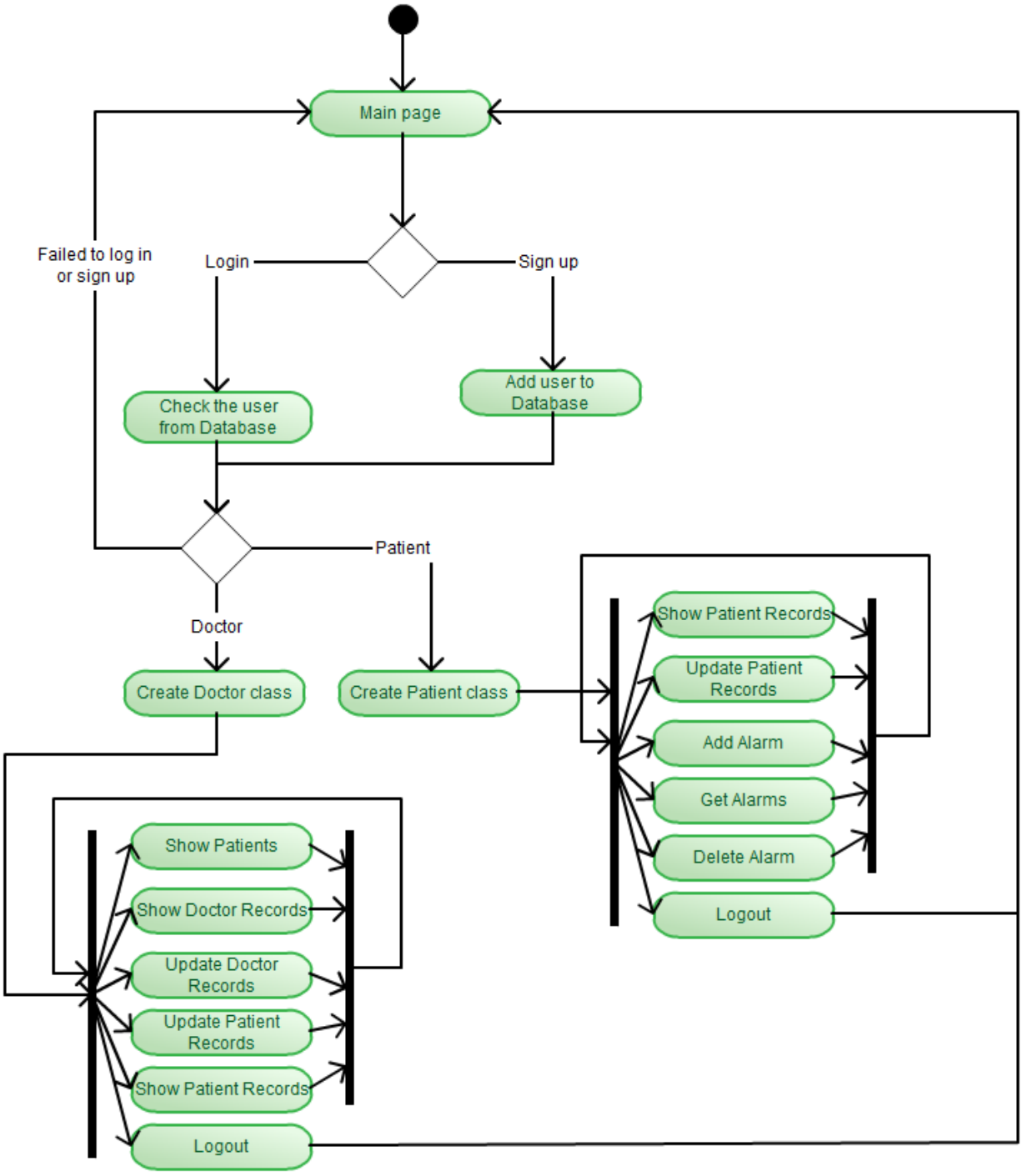

Figure 16 - Web Page Activity Diagram

3.2.2.4 Alarm

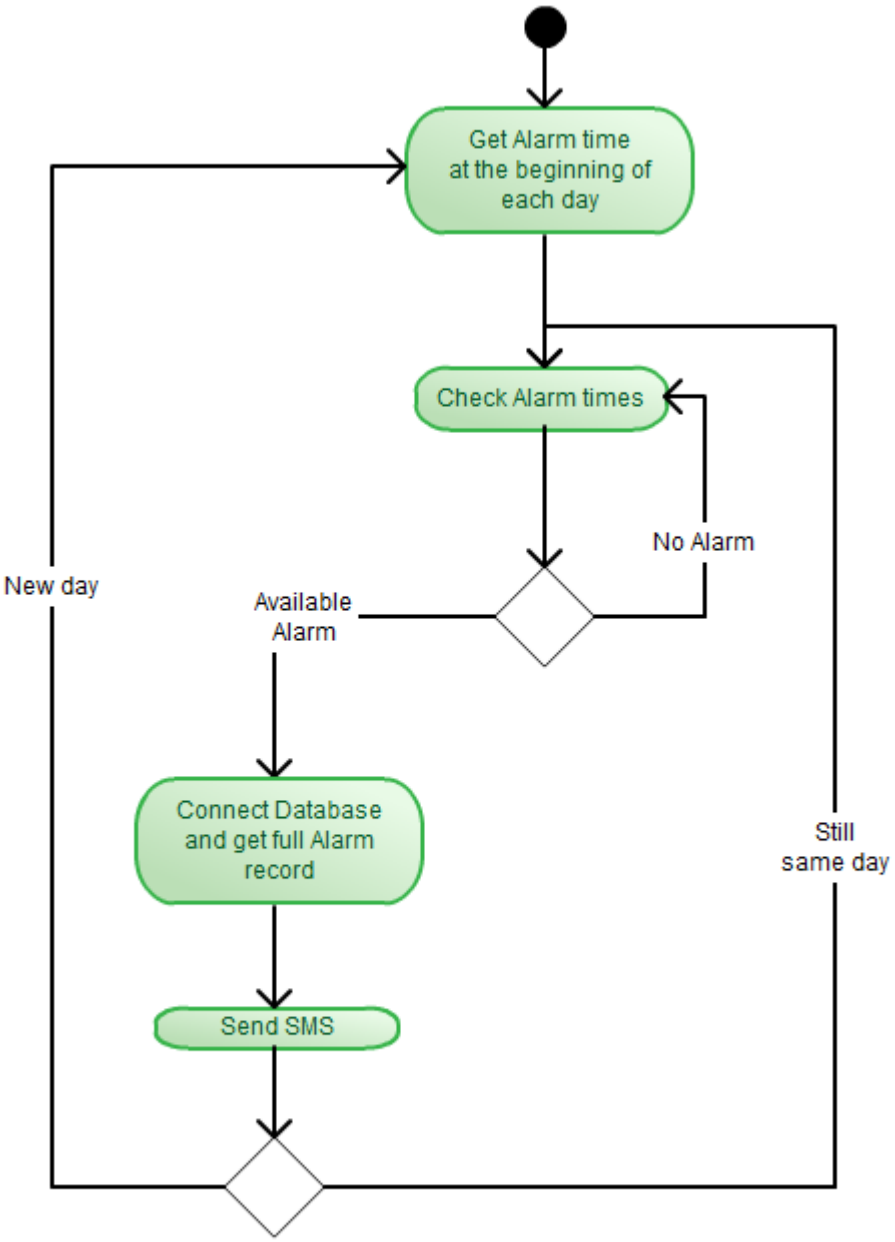

Figure 17 - Alarm Activity Diagram

3.2.2.5 Conversation

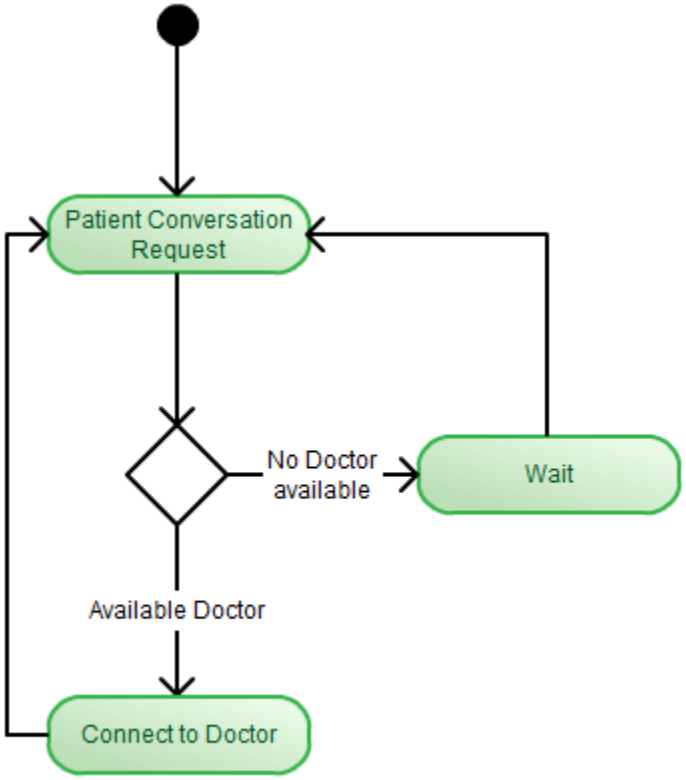

Figure 18 - Conversation Activity Diagram

4 User Interface

4.1 Web Page

4.1.1 Login / Signup Page

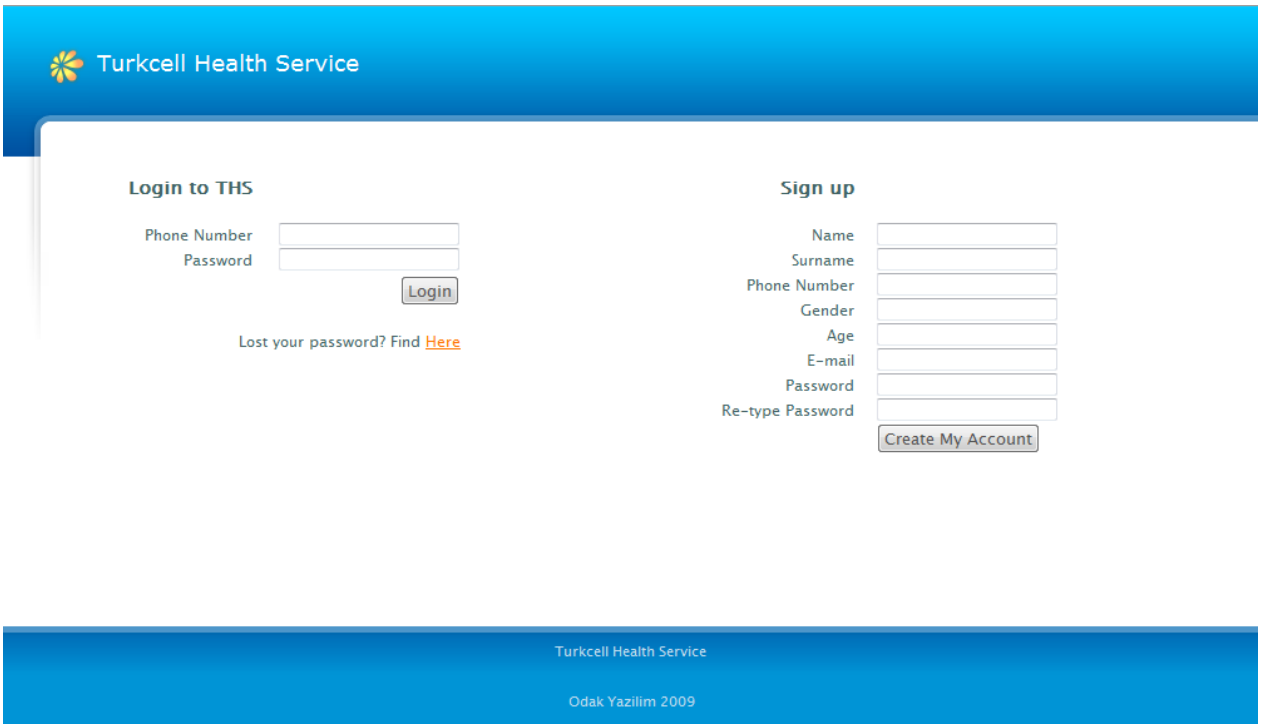

Figure 19 - Login / Sign up Page

4.1.2 Patient Home Page

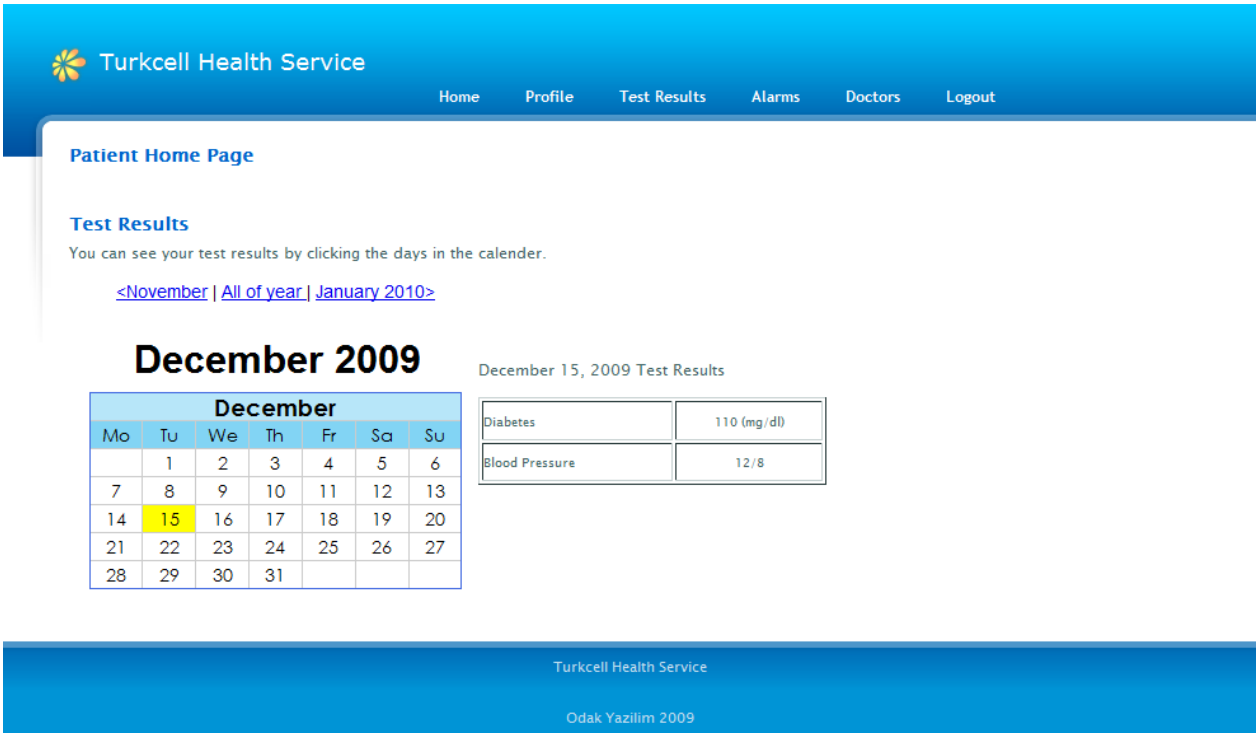

Figure 20 - Patient Home Page

4.1.3 Doctor Home Page

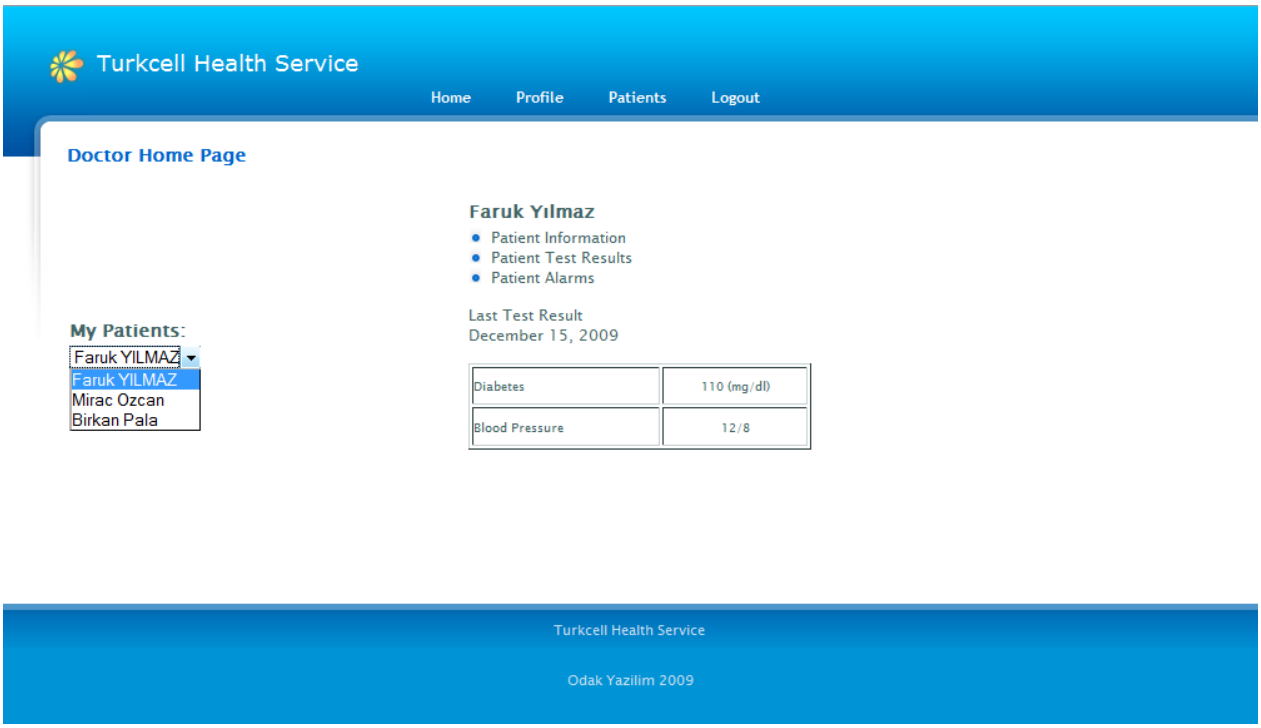

Figure 21 - Doctor Home Page

4.2 Mobile Application

4.2.1 Startup

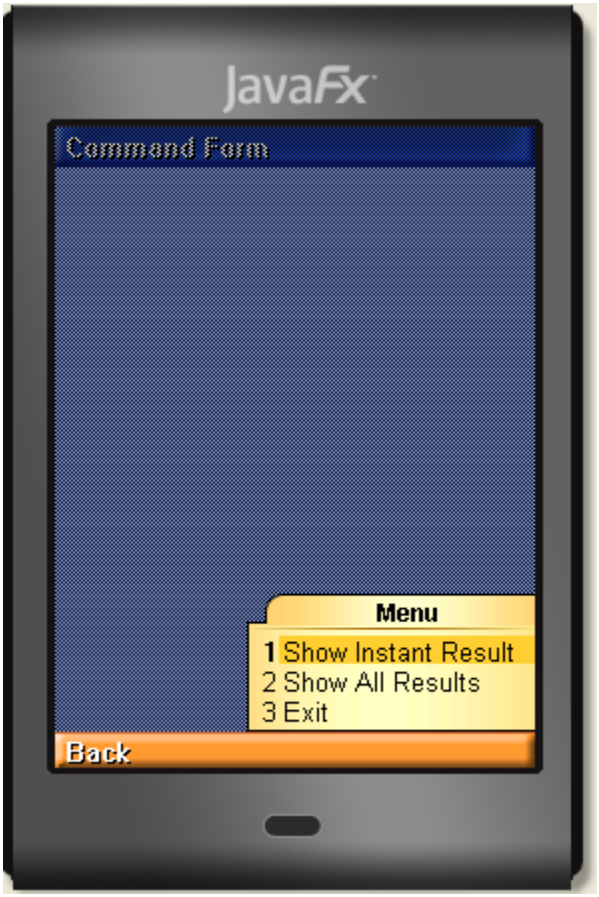

Figure 22 - Mobile Application Startup

4.2.2 Instant Result

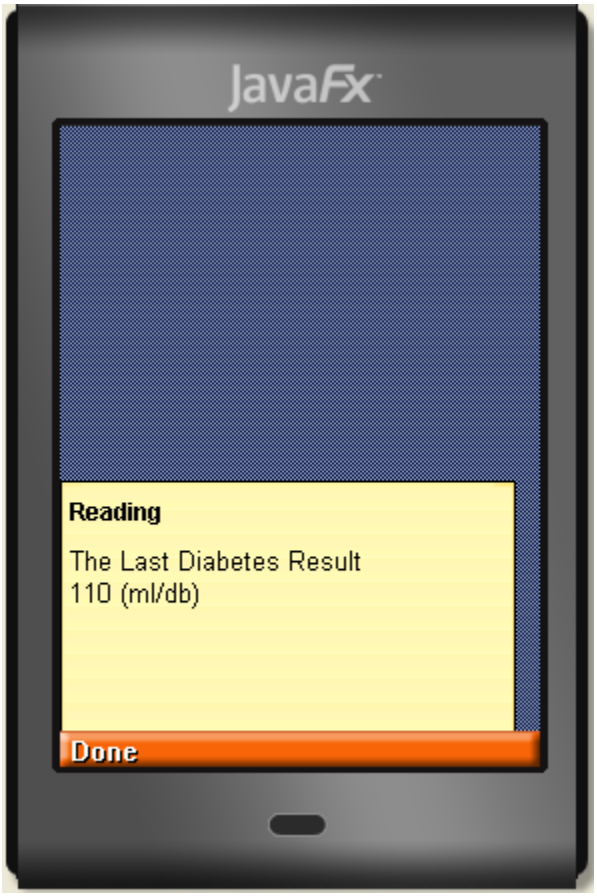

Figure 23 - Mobile Application Instant Result Page

4.2.3 Recent Results

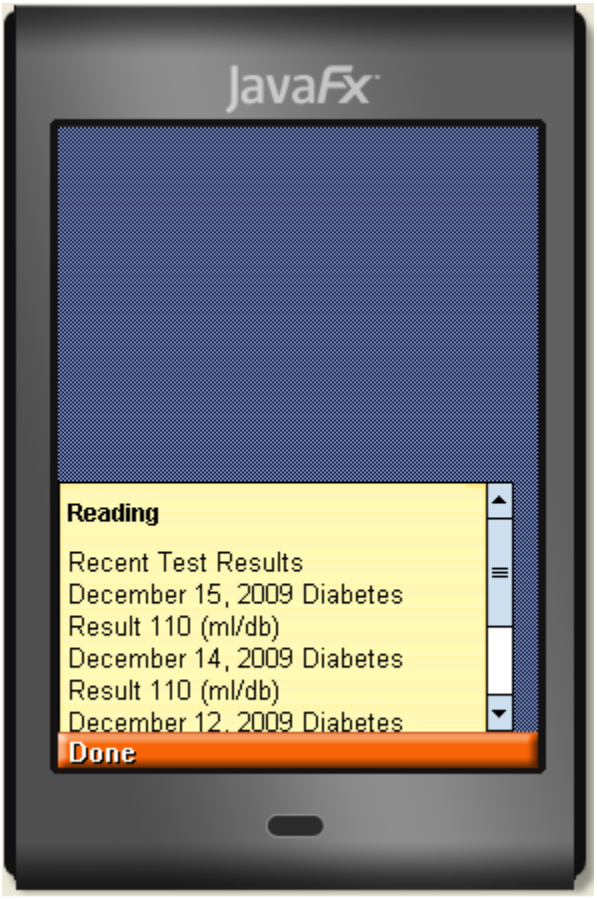

Figure 24 - Mobile Application Recent Results Page

5 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

5.1 Unit Testing

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

5.2 Integration Testing

After each module is tested (unit test), 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.

5.3 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.

5.4 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.

6 Project Schedule

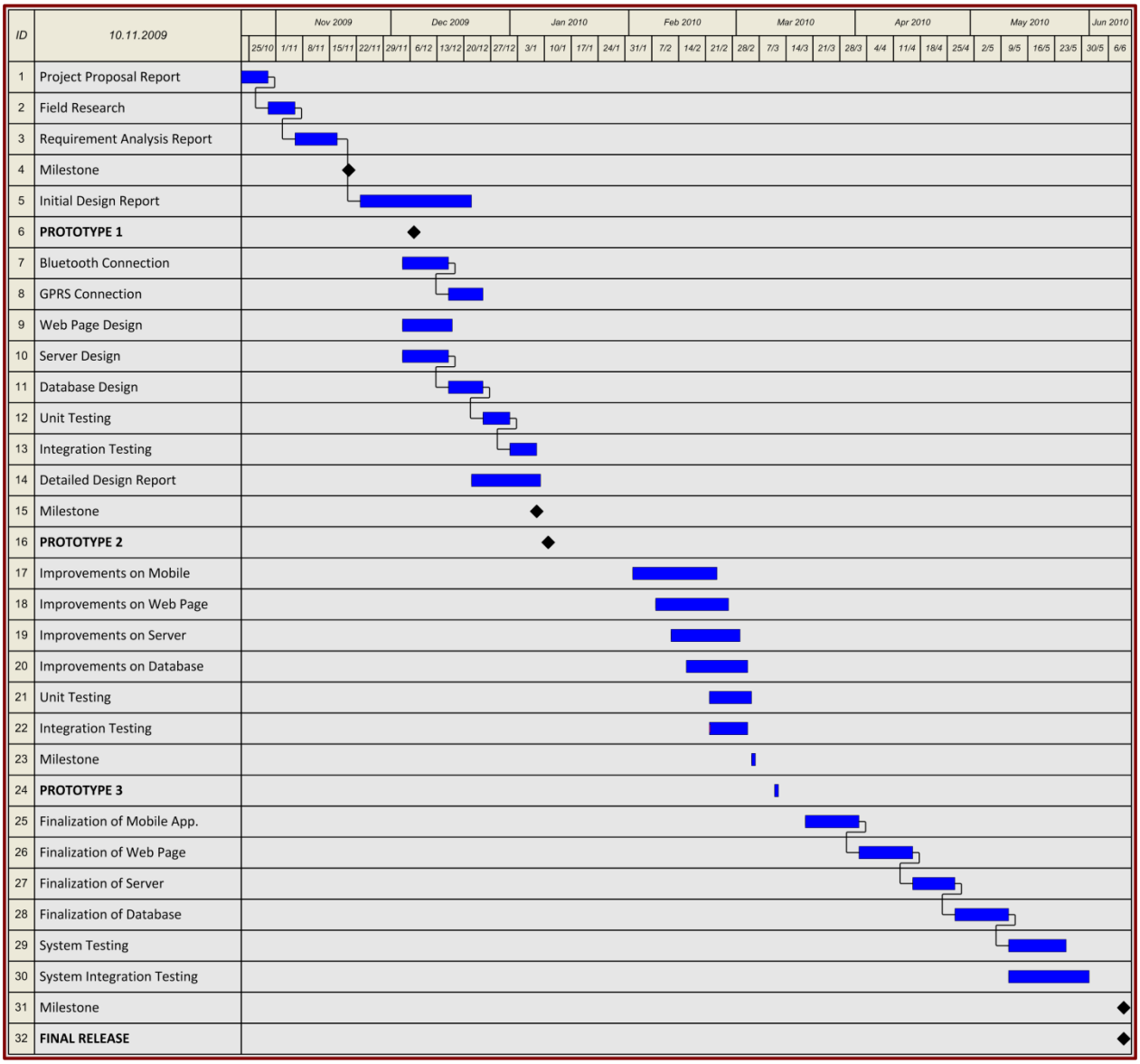

Figure 25 - Project Schedule

7 Conclusion

7.1 What has been done so far?

We decided that we should develop small parts of our application and at the demo time, at the end of the semester, we will join these parts.

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. For example, we developed an application which has server and client sides that can be connected to each other via Bluetooth. Moreover, we implemented another J2ME application which can parse an XML file and display on the phone screen.

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.

7.2 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 first prototype. We will improve the small 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 small applications for the *Mobile* module and *Server* module. Implementing this first 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.