



**Middle East Technical University
Computer Engineering**



**CENG 491 – Computer Engineering Design – I
Fall – 2011**

Software Design Document

BEE - TECH

1. Fatih SEMİZ_1752476
2. Güner ORHAN_1631050
3. Çağlar SEYLAN_1631126
4. Tuğba DEMİR_1630722

Contents

1. Introduction	6
1.1. Problem Definition	6
1.2. Purpose	7
1.3. Scope	7
1.4. Overview	7
1.5. Definitions, Acronyms and Abbreviations	7
1.6. References	8
2. System Overview	8
2.1. Waspnote IDE	8
3. Design Considerations.....	9
3.1. Design Assumptions, Dependencies and Constraints.....	9
3.2. Design Goals and Guidelines	10
3.2.1. Performance	10
3.2.2. Reliability	10
4. Data Design	10
4.1. Data Description	10
4.1.1. Description of Data Entities	10
4.1.2. Databases and Data Storage Items	16
4.2. Data Dictionary.....	16
5. System Architecture	17
5.1. Architectural Design.....	17
5.1.1. Software Architectural Design of the Control Center	18
5.1.2. Software Architectural Design of the Sensors	20
5.2. Description of Components	20
5.2.1. Simulation Core Component	20
5.2.2. Database Component.....	22
5.2.3. Sensor Managing Component	24
5.2.4. Reporting Component	27
5.2.5. Communication Component	30
5.2.6. Timeline Component.....	33
5.2.7. Analysis Component	35
5.3. Design Rationale.....	38



5.4. Traceability of Requirements	38
6. User Interface Design	39
6.1. Overview of User Interface	39
6.1.1. Tester Interface	39
6.1.2. Admin Interface	43
6.2. Screen Images	44
6.2.1. Login Screen	44
6.2.2. Register Screen	46
6.2.3. Main Screen	47
6.2.4. Control Screen	48
6.2.5. Report Screen	49
6.3. Screen Objects and Actions	50
7. Detailed Design	53
7.1. Database Module	55
7.1.1. Classification	55
7.1.2. Definition	55
7.1.3. Responsibilities	55
7.1.4. Constraints	55
7.1.5. Compositions	55
7.1.6. Uses and Interactions	56
7.1.7. Resources	56
7.1.8. Processing	57
7.1.9. Interface / Exports	57
7.2. Simulation Core Module	57
7.2.1. Classification	57
7.2.2. Definition	58
7.2.3. Responsibilities	58
7.2.4. Constraints	58
7.2.5. Compositions	58
7.2.6. Uses and Interactions	59
7.2.7. Resources	59
7.2.8. Processing	59
7.2.9. Interface / Exports	60



7.3.	Reporting Module.....	61
7.3.1.	Classification.....	61
7.3.2.	Definition	61
7.3.3.	Responsibilities	61
7.3.4.	Constraints.....	61
7.3.5.	Compositions.....	62
7.3.6.	Uses and Interactions	62
7.3.7.	Resources	62
7.3.8.	Processing.....	62
7.3.9.	Interface / Exports	63
7.4.	Analysis Module.....	63
7.4.1.	Classification.....	63
7.4.2.	Definition	63
7.4.3.	Responsibilities	64
7.4.4.	Constraints.....	64
7.4.5.	Composition	64
7.4.6.	Uses and Interactions	65
7.4.7.	Resources	65
7.4.8.	Processing.....	65
7.4.9.	Interface / Exports	66
7.5.	Timeline Module	66
7.5.1.	Classification.....	66
7.5.2.	Definition	66
7.5.3.	Responsibilities	66
7.5.4.	Constraints.....	66
7.5.5.	Compositions.....	66
7.5.6.	Uses and Interactions	67
7.5.7.	Resources	67
7.5.8.	Processing.....	67
7.5.9.	Interface / Exports	67
7.6.	Communications Module	67
7.6.1.	Classification.....	67
7.6.2.	Definition	68



7.6.3.	Responsibilities	68
7.6.4.	Constraints.....	68
7.6.5.	Compositions.....	68
7.6.6.	Uses and Interactions	68
7.6.7.	Resources	69
7.6.8.	Processing.....	69
7.6.9.	Interface / Exports	69
7.7.	Sensor Managing Module.....	70
7.7.1.	Classification	70
7.7.2.	Definition	70
7.7.3.	Responsibilities	70
7.7.4.	Constraints.....	70
7.7.5.	Compositions.....	70
7.7.6.	Uses and Interactions	70
7.7.7.	Resources	71
7.7.8.	Processing.....	71
7.7.9.	Interface / Exports	71
8.	Libraries and Tools.....	71
8.1.	Eclipse RCP.....	71
8.1.1.	Description	72
8.1.2.	Usage in the Intruder Detection System.....	72
8.2.	Nasa World Wind Java SDK	72
8.2.1.	Description	72
8.2.2.	Usage in the Intruder Detection System.....	72
8.3.	JOGL	73
8.3.1.	Description	73
8.3.2.	Usage in the Intruder Detection System.....	73
8.4.	JFreeChart.....	73
8.4.1.	Description	73
8.4.2.	Usage in the Intruder Detection System.....	73
8.5.	Waspnote and Waspnote IDE.....	73
8.5.1.	Description	73
8.5.2.	Usage in the Intruder Detection System.....	74



8.6.	Eclipse IDE.....	74
8.6.1.	Description	74
8.6.2.	Usage in the Intruder Detection System.....	74
8.7.	PostgreSQL.....	74
8.7.1.	Description	74
8.7.2.	Usage in the Intruder Detection System.....	74
8.8.	XBee	75
8.8.1.	Description	75
8.8.2.	Usage in the Intruder Detection System.....	75
8.9.	SD Card	75
8.9.1.	Description	75
8.9.2.	Usage in the Intruder Detection System.....	75
9.	Time Planning	77
9.1.	Term 1 Gantt Chart.....	77
9.2.	Term 2 Gantt Chart.....	78
10.	Conclusion.....	79



1. Introduction

Intruder detection system is the security-based project that is essential for detection of the intruder entering to the specified confidential area. The aim of this document is that showing the more detailed representations of the requirement specifications and giving the structured data design of the project in more detailed manner.

1.1. Problem Definition

Military service of any country has confidential information about country or any other equipped military soldiers. Therefore, this secret information must be protected from any other country or person. This can be achieved by protecting the area which surrounds these places. Another use area of this intruder detection system is to protect area for soldier gunned exercises, because, entrance of the any person can cause to injure the intruder.

Due to the security necessity of the specified area, intruder detection system will cope with the intrusion detection. The most important problem is that detecting the intrusion can be difficult due to the noise which results from the animal entrance, wind, or other climate conditions.

Another important problem that our system will handle is false alarm rates. False alarm rate can occur with two ways. First one is that the system gives an alarm when there is no intruder in the detection area. Second one is that system does not give an alarm even if there is an intruder in an area. Former will cause only waste of time. However, the latter one is more crucial problem that our system must not ignore such an intrusion.

Moreover, the detection range is a problem and also the constraint of the system. The working range of the system is important, because if we can increase it, military service can protect the area with less number of seismic sensors.

Classification of the intrusion is another problem that we must handle. This problem has also relation with the decreasing the false alarm rate, because, if the intrusion classification cannot handled efficiently, this means that our system will give the intrusion detection alarm for any other kind of animal. This is also the waste of time and money for military services.

The last important problem is that power consumption of the seismic sensor and its board. These sensors will not only be placed to the smooth terrains, but also to the rough lands where the soldiers cannot reach frequently. Therefore, the power consumption of our system must be minimum as much as possible. This means that complexity of our algorithm must be very low and we can give less voltage to the seismic sensor if there is no intrusion for a long time.



1.2. Purpose

We have stated the functional and non-functional requirements of the intruder detection system in Software Requirements Specification document. At this document, we will give the detailed system design properties of these requirements.

With the help of this document, reader can understand the system and the data architecture of the system. In addition, this document includes all necessary information for a programmer to develop an intrusion detection system.

1.3. Scope

This document describes software designs and establishes the information content and organization of a software design description [1].

The scope of document includes hardware and software requirements, data design, system architecture, tools, libraries, user interface design, and time planning of our project.

1.4. Overview

We can divide the document into nine parts, including introduction part as first. In the 2nd section, we will mention about the general details of the system by giving its goals, benefits and objectives. 3rd section generally includes the design issues which are dependencies, constraints, goals, and guidelines. In section 4, we will give the data design properties which are the properties of the storage of data into the database system, or SD card with data features. 5th part includes the detailed description of program architecture. In 6th section, user interface design and its screen images will be presented. In section 7, we will give a detailed design of our entities and components. In the 8th section, we will give the required tools and libraries to develop the system. In the 9th section, we will give the finalized Gantt chart diagram for 1st and 2nd terms. In the last section, we will conclude the software design document.

1.5. Definitions, Acronyms and Abbreviations

API: Application Program Interface

DBMS: Database Management System

GPRS: General Packet Radio Service

GPS: Global Positioning System

GUI: Graphical User Interface

IDE: Integrated Development Environment



JDBC: Java Database Connectivity API

ORDBMS: Object-Relational Database Management System

SCC: Simulation Core Component

SD: Secure Digital

SDD: Software Design Document

SRAM: Static Random Access Memory

1.6. References

[1] Society, IEEE Computer. (2009). IEEE Standard for Information Technology—Systems Design— Software Design Descriptions. New York: IEEE.

[2] http://en.wikipedia.org/wiki/Flow-based_programming

[3] <http://c2.com/cgi/wiki?CouplingAndCohesion>

[4] Liang Z., Wei J., Zhao J., Liu H., Li B., Shen J., Zheng C., “The Statistical Meaning of Kurtosis and Its New Application to Identification of Persons Based on Seismic Signals”, *Sensors* 2008, 8, 5106-5119; DOI: 10.3390/s8085106

[5] <http://worldwind.arc.nasa.gov/java/>

[6] <http://www.postgresql.org/docs/current/static/intro-what-is.html>

[7] <http://itextpdf.com/>

[8] <http://code.google.com/p/java2word/>

[9] <http://www.java-tips.org/other-api-tips/jogl/what-is-jogl.html>

[10] <http://www.jfree.org/jfreechart/>

2. System Overview

We use mainly Waspnote IDE application for writing and embedding the code to the Waspnote.

2.1. Waspnote IDE

Waspnote IDE is a well-organized development environment, programmed with Java, to embed the code to the waspmote board. It contains all required libraries to handle the operations of the sensors, GPS, GPRS, or any other equipped hardware that can be integrated



on a board. Its libraries are created, effectively. All of the functions are understandable for a programmer. In figure (1), user interface of Waspnote IDE is provided.

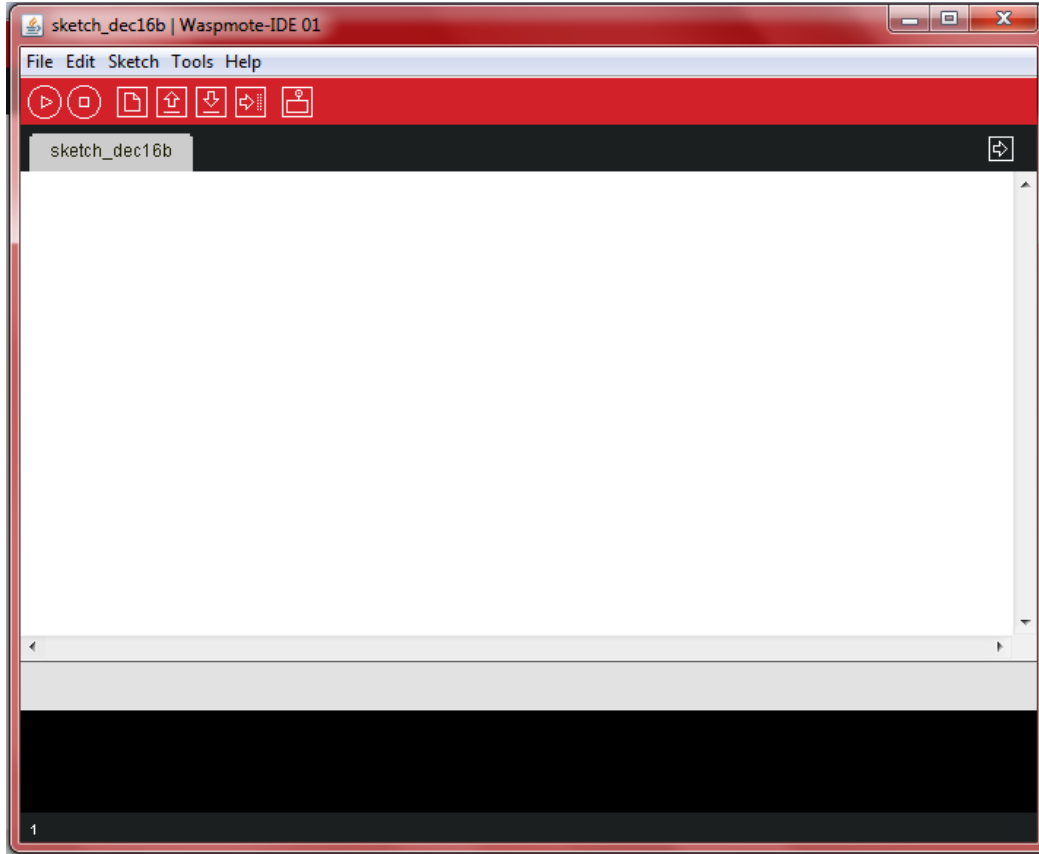


Figure 1. Waspnote IDE

It includes own grammar syntax written with Java ANTLR library.

3. Design Considerations

3.1. Design Assumptions, Dependencies and Constraints

- Intruder detection system code will be embedded to the waspmote board.
- Waspnote has ATmega1281 microprocessor with 8MHz frequency and 8 KB SRAM.
- The program must use less memory as much as possible due to the small size of memory on board.
- The microprocessor has low frequency so the time complexity of our system must be very low.
- The required database management system is PostgreSQL.



- The system has GPS to get the latitude, longitude, altitude, and the time from the satellite.
- When Wasmote is not connected to the PC, it will record the alarms to the SD card which can be integrating on it.
- Our project assumption is that each waspmote with seismic sensors will have own code inside and all seismic sensors will give an alarm independent from each other.

3.2. Design Goals and Guidelines

3.2.1. Performance

Due to the small memory size and low frequency of the microprocessor inside the board, our intruder detection algorithm must work fast. All of the calculations which decide the intrusion detected must be performed in short period of time.

3.2.2. Reliability

Important issue that we will handle is reducing the false alarm rate meaning that giving an alarm when there is no intruder or not giving any alarm that there is an intruder. False alarm rate will decrease the reliability of our system. It also causes the waste of time for military service. We can think that our system has a zero tolerance to the errors.

10

4. Data Design

4.1. Data Description

This part consists of two subparts: description of data entities, where the major data entities of the system will be identified and described one by one and database of the system where the responsibilities of the databases will be discussed and details how to create it will be given.

4.1.1. Description of Data Entities

In this part, descriptions of the data entities in the information domain of the intruder detection system are given. There are 13 data entities in the system: report, analysis, alarm, position, timeline, timelineEntry, simulationCore, database, sensor, sensorParameters, sensorManager, communication, and alarmPacket.

4.1.1.1. Report Data Entity

The Alarm data entity holds the information needed to describe a report.



Field Name	Data Type	Description
text	string	Text defining the document
documentType	bool	Type of the document. Indicates pdf or doc.
linkToCore	SimulationCore*	Pointer to simulation core component
linkToAnalysis	Analysis*	Pointer to analysis component

4.1.1.2. Analysis data entity

The Analysis data entity holds the information needed to describe statistics computed by the analysis component.

Field Name	Data Type	Description
numOfSensors	int	Number of sensors in the system
numOfAlarms	int	Total number of alarms obtained
mostActiveSensor	string	ID number of the most active sensor
leastActiveSensor	string	ID number of the least active sensor
lastAlarm	Alarm	Last alarm obtained
lastActiveSensor	string	ID number of the sensor from the last alarm obtained
mostActiveToLeast	string[]	List of IDs of sensors from most actives to least
mostRecentActiveToLeast	string[]	List of IDs of sensors from most recent actives to least
linkToCore	SimulationCore*	Pointer to simulation core component
linkToTimeline	Timeline*	Pointer to timeline component



4.1.1.3. Alarm data entity

The Alarm data entity holds the information needed to describe an alarm.

Field Name	Data Type	Description
date	date	Date when the intruder is detected
time	time	Time when the intruder is detected
alarmPos	Position	Position where the intruder is detected

4.1.1.4. Position data entity

The Position data entity holds the information needed to describe a position.

Field Name	Data Type	Description
latitude	double	Latitude value of the position
longitude	double	Longitude value of the position

12

4.1.1.5. Timeline data entity

The Timeline data entity holds the information needed to represent the timeline on the GUI.

Field Name	Data Type	Description
timeline	TimelineEntry[]	List of TimelineEntry objects to define the timeline
linkToCore	SimulationCore*	Pointer to simulation core component

4.1.1.6. TimelineEntry data entity

The TimelineEntry data entity holds the information needed to describe an entry of the TimeLine.



Field Name	Data Type	Description
sensorID	string	ID number of the sensor
alarmDate	date	Date when the intruder is detected
alarmTime	time	Time when the intruder is detected
alarmPos	Position	Position where the intruder is detected

4.1.1.7. SimulationCore data entity

The SimulationCore data entity holds the information needed to represent the simulation core component.

Field Name	Data Type	Description
linkToDatabase	Database*	Link to the database component
instructionQueue	Queue<int>	Queue of integers

13

4.1.1.8. Database data entity

The Database data entity holds the information in the database.

Field Name	Data Type	Description
sensors	Sensor[]	Array of sensors



4.1.1.9. Sensor data entity

The Sensor data entity holds the information needed to describe and mimic a sensor.

Field Name	Data Type	Description
sensorID	string	ID number of the sensor
sensorName	string	Name of the sensor
sensorPos	Position	Current position of the sensor
alarms	Alarm[]	List of alarms obtained from that sensor
parameters	sensorParameters	Parameters of the sensor
mode	int	Current mode of the sensor

4.1.1.10. SensorParameters data entity

The SensorParameters data entity holds the information needed to describe the sensor parameters.

Field Name	Data Type	Description
sensitivity	int	Sensitivity to seismic movements
power	int	Power that the sensor consumes
algorithmParameters	float[]	Parameters used in the algorithm embedded in the sensor



4.1.1.11. SensorManager data entity

The SensorManager data entity holds the information needed to describe the sensor managing component.

Field Name	Data Type	Description
linkToCommunication	Communication*	Pointer to the communication component
linkToCore	SimulationCore*	Pointer to the simulation core component

4.1.1.12. Communication data entity

The Communication data entity holds the information needed to describe the communication component.

Field Name	Data Type	Description
linkToCore	SimulationCore*	Pointer to the simulation core component

4.1.1.13. AlarmPacket data entity

The AlarmPacket data entity holds the information needed to describe the alarm packets received from the sensor.

Field Name	Data Type	Description
sensorID	string	ID number of the sensor
alarms	Alarm[]	List of alarms



4.1.2. Databases and Data Storage Items

4.1.2.1. SD Card

There will be a SD card in the sensor, more specifically, integrated with the Waspote board. With this card, the sensor will be able to store the alarms. When user wants to read the alarms from the sensor, he/she will send a command requesting the alarms in the card and the alarms will be kept in the hard disk, namely, the storage item of control center. More specific information about SD cards is given in section 8.9.

4.1.2.2. Hard-disk of the Control Center

A common hard-disk will be the main information storage item of the system. It will hold the records of the past alarms. We keep past alarms because user may want to create a document showing the statistics about the alarms kept.

The database will be created with PostgreSQL, which is an open source, object-relational database system. More specific information about PostgreSQL can be found in section 8.7.

4.2. Data Dictionary

Name	Type	Refer to Section
Alarm	Data entity	4.1.1.3
AlarmPacket	Data entity	4.1.1.13
Analysis	Data entity	4.1.1.2
Analysis	Component	5.2.7
Communication	Data entity	4.1.1.12
Communication	Component	5.2.5
Database	Database	4.1.1.8
Database	Component	5.2.2



Position	Data entity	4.1.1.4
Report	Data entity	4.1.1.1
Reporting	Component	5.2.4
Sensor	Data entity	4.1.1.9
SensorManager	Data entity	4.1.1.11
Sensor Managing	Component	5.2.3
SensorParameters	Data entity	4.1.1.10
SimulationCore	Data entity	4.1.1.7
Simulation Core	Component	5.2.1
Timeline	Data entity	4.1.1.5
Timeline	Component	5.2.6
TimelineEntry	Data entity	4.1.1.6

5. System Architecture

In this chapter, a general description about the Intruder Detection System will be given. Firstly, information about the relationships between the modules of the whole system will be provided. Then, each module will be analyzed individually. Finally, the design rationale and traceability of system requirements will be provided.

5.1. Architectural Design

The Intruder Detection System will consist of two types of devices: the control center, and sensor. There may be several sensors in the system. However, there will be only one control center managing all of these sensors. A deployment diagram for the system is provided in the figure (2).



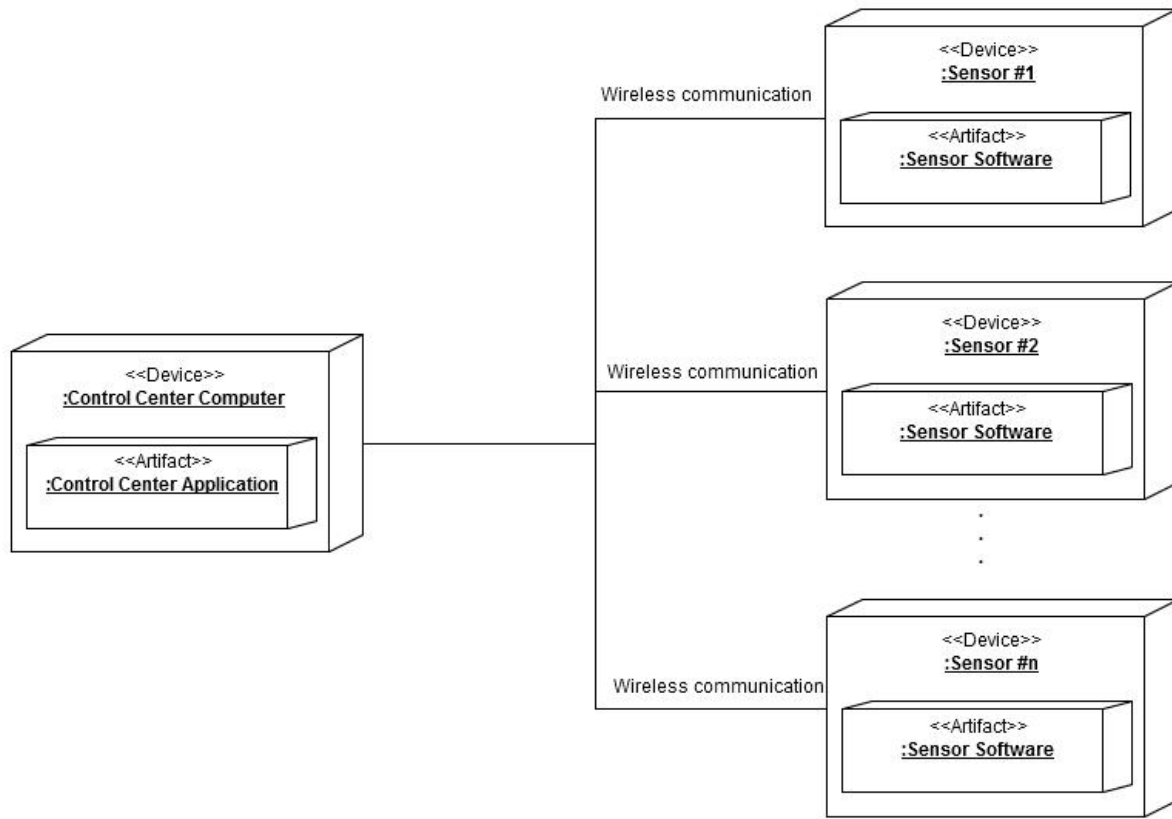


Figure 2. Deployment diagram of the system.

Software architecture design of each device will be described separately.

5.1.1. Software Architectural Design of the Control Center

The control center consists of components. Each of these components has a different task than each other. Every component in the system sees other components as black boxes. The whole system is organized with the help of data exchanges between these black boxes, namely components. This type of programming paradigm is called as flow based programming.

The system will consist of seven components. These components are;

- Simulation core component
- Database component
- Sensor managing component
- Reporting component
- Communication component
- Timeline component
- Analysis component



Component diagram of the Intruder Detection System is provided in the figure (3).

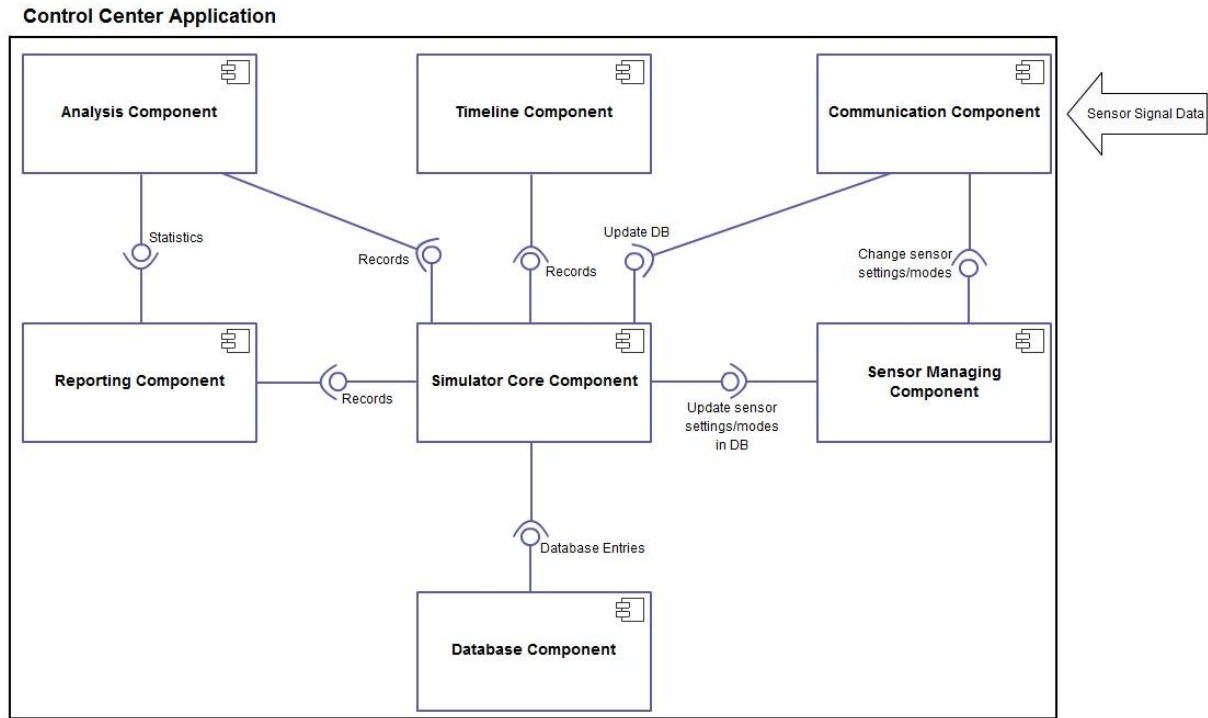


Figure 3. Component diagram of the system.

Before giving detailed descriptions of these components, we will first give short descriptions of each of these components.

Simulation Core Component: This component is the heart of the system. Only this component has access to the database component. With this, reaching of more than one operation to the database at the same time is avoided. Thus, the simulation core component makes the system more modular. Every other component will be in interaction with this component. Hence, this component can be seen as a bridge between the database component and the other components.

Database Component: This component has the responsibility of realizing the database operations. Only this component is authorized to reach the physical database. Hence, this component can be seen as a bridge between the simulation core component and the physical database.

Sensor Managing Component: This component has the responsibility of managing the sensors such as changing sensor parameters, changing mode of a sensor, registering new sensor to the system and unregistering a sensor from the system.

Reporting Component: This component is responsible from documenting the statistics related to the sensors in the system and their alarm information. The user can provide the time interval in which he/she wishes to see the statistics and the type of the document (pdf or doc).



Communication Component: The responsibility of this component is to establish communication between the control center and the sensors. Low level operations such as receiving information packets and sending them to the simulation core component are done by this component.

Timeline Component: When the user wishes to see alarms in a time interval along with corresponding sensor, position, date and time, this component takes the action. The user can be seen the information as a timetable.

Analysis Component: This component has the responsibility of computing the statistics related to the sensors in the system. Either the user can directly see the statistics or reporting component can use these statistics in the documentation.

5.1.2. Software Architectural Design of the Sensors

The sensor processes the signals related to the seismic movements and decides whether it is an intruder or not. If it is an intruder, it sends an alarm to the control center.

The responsibility of the sensors is not as complex as the control center. Thus, it is not consist of any components. There will be a detection and classification algorithm embedded in it.

We are planning to use an algorithm making use of kurtosis method. The details of the algorithm is provided in [4].

20

5.2. Description of Components

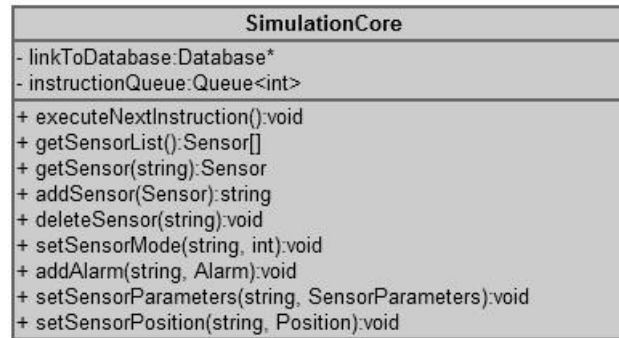
In this section, brief description of each component is given. For each component, firstly, a class diagram describing the component is provided. Then, for each component, processing narrative summarizing the responsibilities of the component, interface description describing the interfaces that the component has, processing details that describes the details related to the computations done by the component, and dynamic behavior summarizing the interactions between components and defining use case realized by the component is going to be given.

5.2.1. Simulation Core Component

Simulation core component is the heart of the system. This component is in communication with all other components. Basically, simulation core component is a bridge between other components (except database component) and the database component. The class diagram describing the simulation core component is given figure (4).



Class Diagram for the Simulation Core Component

**Figure 4.** Class diagram for the simulation core component.

5.2.1.1. Simulation Core Component Processing Narrative

We choose the design paradigm while designing the system to be flow based programming. This component is created so that the system will be more modular and more suitable for flow based programming paradigm [2].

Simulation core component can be thought as an interface between the other components (except database component) and the database component. A component can access a method of any other component via this component. This minimizes the coupling of the whole system [3].

Data transfers between the database component and the other components are realized by the simulation core component.

5.2.1.2. Simulation Core Component Interface Description

The component provides a data interface between the database component and the other components. The data communication between the database component and the other components are realized by this interface.

5.2.1.3. Simulation Core Component Processing Details

When a component requires doing an operation that makes use of database, it uses this component instead of reaching the database directly. When an operation making use of the database is requested, simulation core module puts that operation in an instruction queue. The operations in the instruction queue are done one-by-one sequentially by the simulation core component. With this kind of processing, only one operation at a time is done that makes use of the database.



5.2.1.4. Simulation Core Component Dynamic Behavior

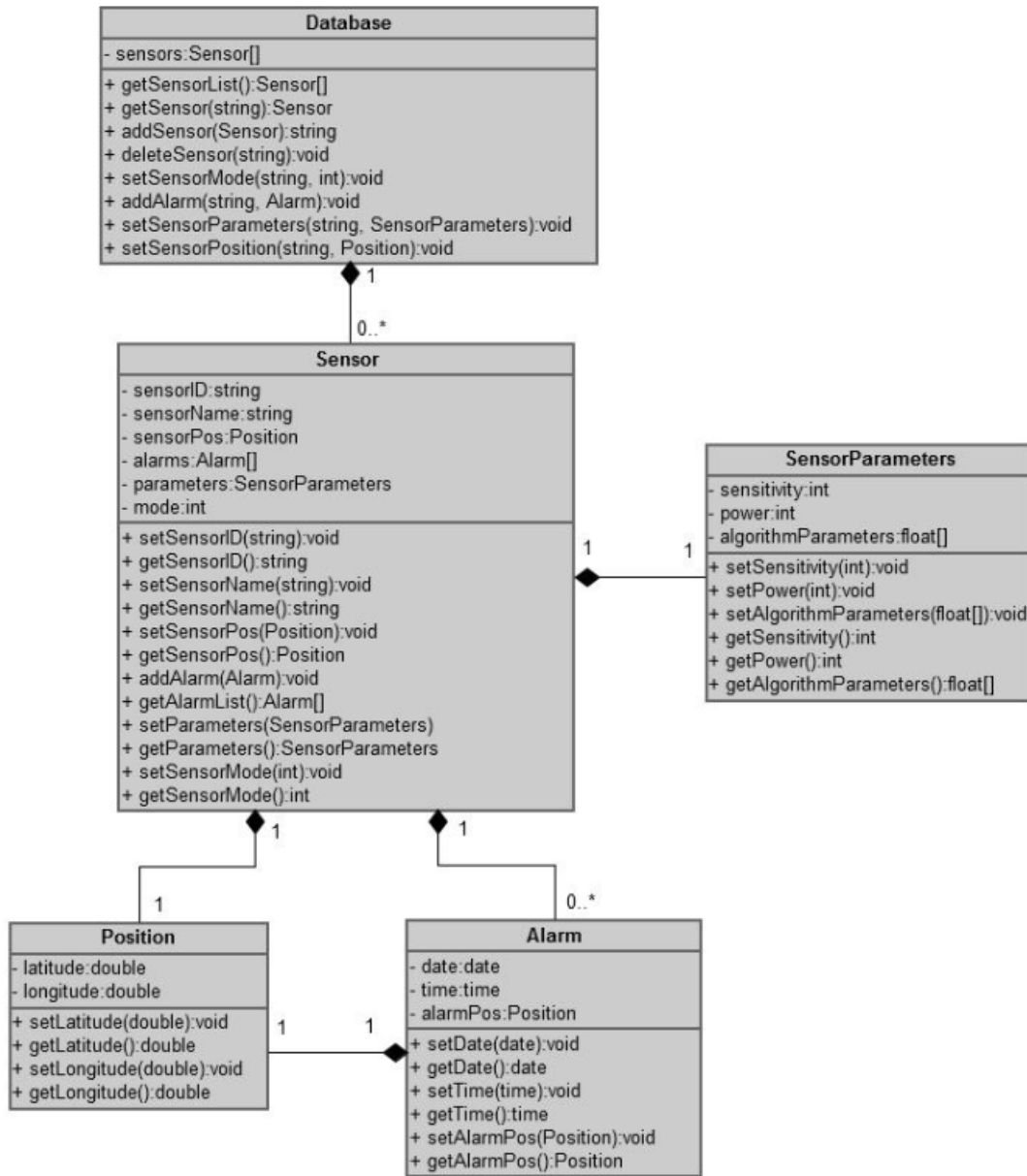
At the very basic, the responsibility of the simulation core component is to receive data from the database component and send data to the database component. More specifically, the read and write operations related to the database of timeline, analysis, reporting, communication and sensor managing components are done with the help of the simulation core component. This component does not realize any requirement directly but it helps other components to be able to make them realize the system requirements.

5.2.2. Database Component

Database component is designed to coordinate write and read operations that will be done on the database. Only this component is authorized to reach the database directly. The class diagram describing the database component is given in figure (5).



Class Diagram for the Database Component

**Figure 5.** Class diagram for the database component.

5.2.2.1. Database Component Processing Narrative

As indicated previously, the only component that interacts with the database component is the simulation core component. Thus, responsibility of this component is to realize the operations on the database desired by the simulation core component.

Indirectly and logically, it can be thought that all other components make write and read operations with the database via the simulation core component. Thus, the database component can be seen as a “bridge” between the simulation core component and the physical database.



5.2.2.2. Database Component Interface Description

The database component is an interface between the simulation core component and the physical database. The simulation core component is going to request data read and data write operations to the physical database via this component. Thus, there are two interfaces of the database component: one with the simulation core component, one with the physical database. From both of them, there will be data communication.

5.2.2.3. Database Component Processing Details

The sensors registered to the system and alarms get from SD cards of the sensors will be recorded in the database. There are two major operations with the database: data read and data write.

When the user wants a document of the alarms in the database via the reporting module, the simulation core sends a request to the database module. After that, a read operation is performed; the database module reads the alarms recorded in the database and sends them to the simulation core component.

When the user wants to get the records in the SD cards of the sensors, the alarms are obtained via the communication component. It sends them to the simulation core component and the simulation core component sends them to the database component. Then, a write operation is performed; the database module writes the alarms sent by the simulation core component to the database.

Write operations may also be requested by the sensor managing module. When a new sensor is registered or deleted, or settings of a sensor are changed, the data related to the sensors on the database are needed to be updated. These operations are redirected by the simulation core component to the database component and the database component performs the desired tasks.

5.2.2.4. Database Component Dynamic Behavior

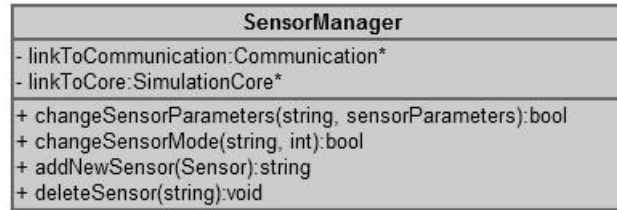
The responsibilities of the database component are described in the previous sections. At the very basic, it performs read and write operations related to the database requested by the simulation core module. This module does not realize any functional requirement directly but it helps other modules in a way that database operations are done via this component.

5.2.3. Sensor Managing Component

At the very basic, the sensor managing component will perform the tasks related to the tuning of the sensors such as changing sensor parameters etc. The details of these operations are provided in the following subsections. The class diagram describing the sensor managing component is given in figure (6).



Class Diagram for the Sensor Managing Module

**Figure 6.** Class diagram for the sensor managing component.**5.2.3.1. Sensor Managing Component Processing Narrative**

The operations related to sensors are realized by this component. The responsibilities of this component can be summarized as changing sensor parameters, changing sensor mode, adding a new sensor to the system and deleting a sensor from the system.

5.2.3.2. Sensor Managing Component Interface Description

The sensor managing module has three interfaces; one with the user via GUI, one with the communication component and one with the simulation core component.

When the user wants to change settings or mode of a sensor, it sends new parameters or mode to the sensor via the communication component and updates the parameters or mode in the database related to that sensor via the simulation core component.

When the user wants to add or delete a sensor, the sensor managing component updates the sensor list in the database accordingly via the simulation core component.

5.2.3.3. Sensor Managing Component Processing Details

The user interacts with the sensors via the sensor managing component. The computational task of the component is not heavy. It just redirects the information related to sensors from GUI to the related components, namely, to the communication component and to the simulation core component.

When the user wishes to change the settings or mode of a sensor, the components packets the related data obtained via GUI. It first sends it to the communication module. If the setting change is successful, it also calls simulation core component to update the related information in the database.



When the user wishes to add or delete a sensor, the related information is updated via the simulation core component. Thus, in such a case, this component invokes the simulation core component.

5.2.3.4. Sensor Managing Component Dynamic Behavior

The responsibilities and the processing details of the sensor managing component have already been described in the previous sub sections. As it is described, the component is in interaction with the communication component and the simulation core component.

In a parameter or mode change operation, the component first invokes the communication component. If the operation is successful, then, the component invokes the simulation core component to make the related updates in the database.

In a sensor add or delete operation, the component directly invokes the simulation core component and makes the related updates in the database.

The use cases realized by the component are changing mode or parameters of a sensor, adding a new sensor to the system and deleting a sensor from the system. The sequence diagrams summarizing these processes are given in figures (7), (8), and (9).

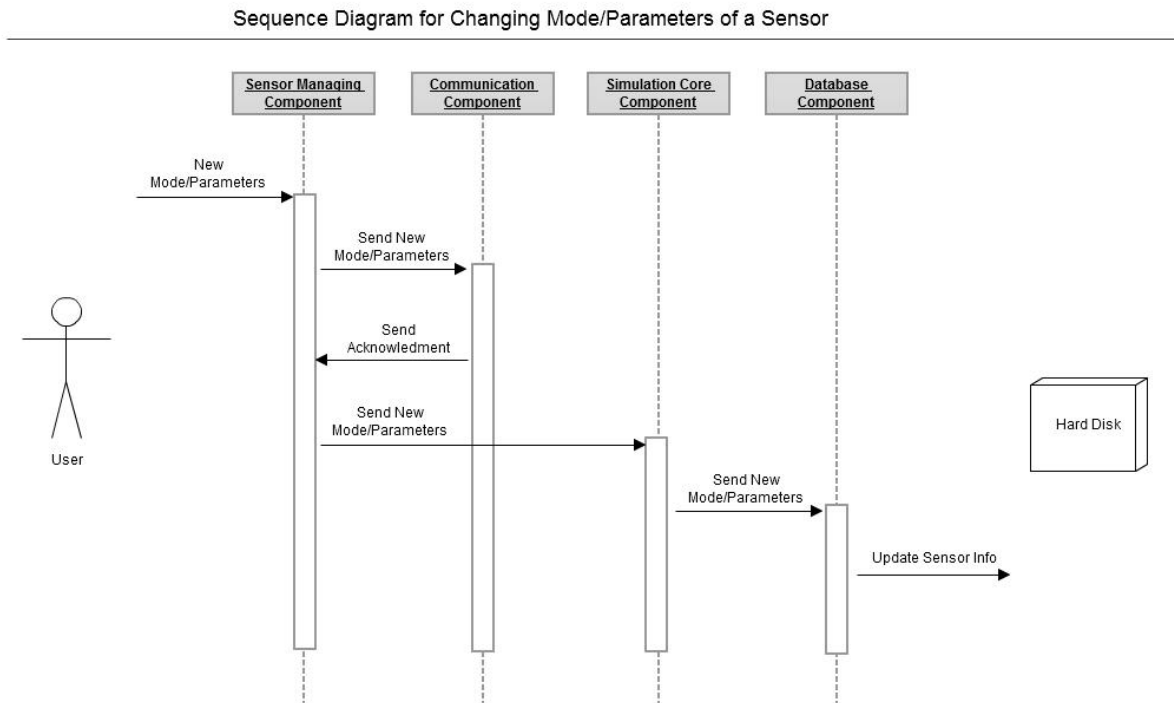


Figure 7. Sequence diagram for changing mode/parameters of a sensor.



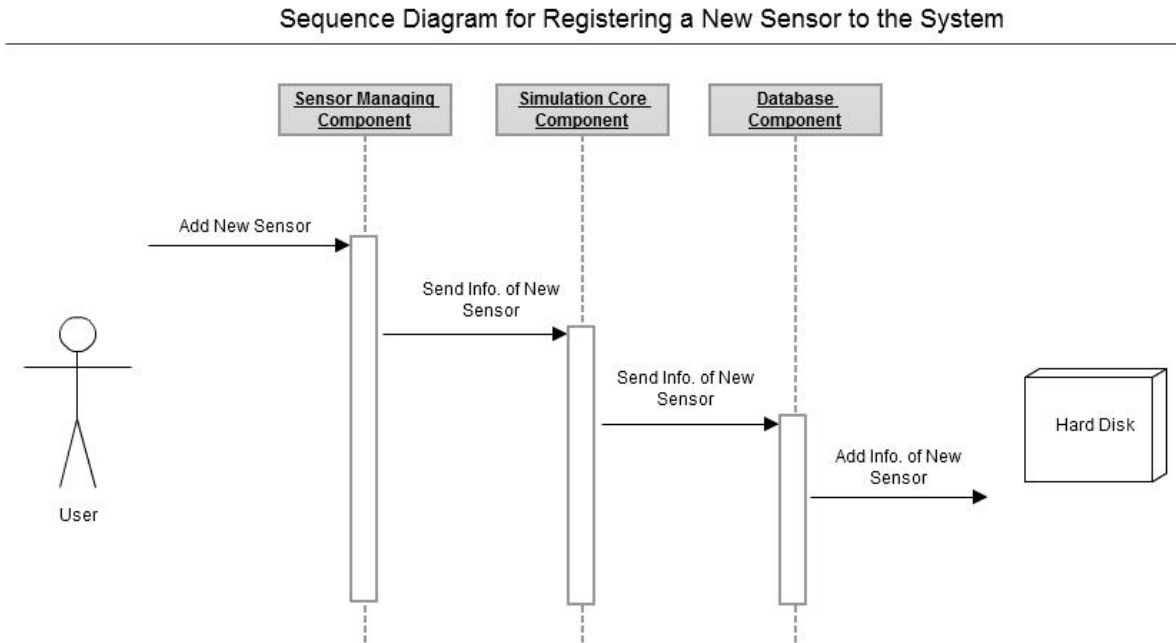


Figure 8. Sequence diagram for registering a new sensor to the system.

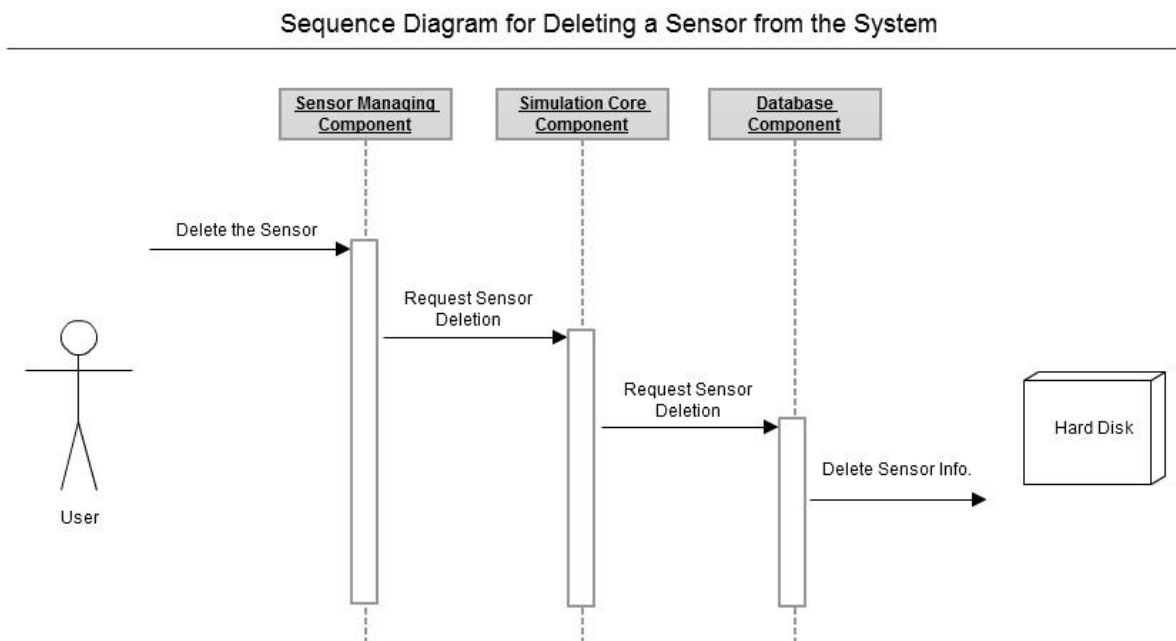


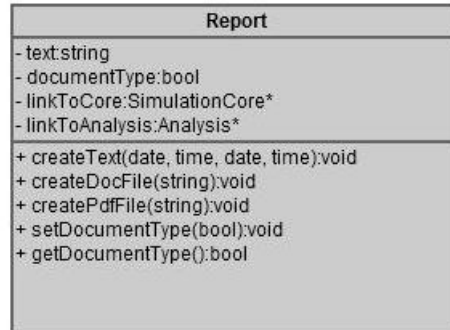
Figure 9. Sequence diagram for deleting a sensor from the system.

5.2.4. Reporting Component

With the reporting component, the user will be able get the statistics and alarm records in a document format. The details of the component are described in the following sub sections. The class diagram describing the reporting component is given in figure (10).



Class Diagram for the Reporting Component

**Figure 10.** Class diagram for the reporting component.**5.2.4.1. Reporting Component Processing Narrative**

Via this component, the user will be able to get the information about the intruder actions defined in a time interval in a report format. The user will be able to select the desired format for the document; it will be either pdf or doc.

The information on the document can be grouped into two: statistics such as most recently detected intruder, most active alarm etc., and a list that illustrates alarms obtained for each sensor along with their coordinates, dates and times.

28

5.2.4.2. Reporting Component Interface Description

The reporting component has three interfaces; one with the GUI, one with the simulation core component and one with the analysis component.

Via the GUI, the user will be able to define a time interval and type of the document. The user will get the report in the specified format. There will be alarm actions related to that time interval and statistics related to time interval on the document.

With the interface with analysis component, the reporting component will be able to get the data required for the statistics part of the document.

With the interface with simulation core component, the reporting component will be able to obtain the sensor information related to the sensors registered in the database.

5.2.4.3. Reporting Component Processing Details

From the analysis component, the reporting component will create the statistics part of the report. Please note that it is the responsibility of the analysis component to create the



statistics. Thus, the reporting component will do no computations to compute the statistics; they will be in hand already.

From the simulation core component, the reporting component will get a list of the sensors. Then, for each sensor in the list, the component will compute alarms obtained from that sensor, current position of the sensor and name of that sensor. For each alarm, the component will extract date, time and position information of that alarm.

Then, the component will create a text from these data and print the text on a document in the specified format.

5.2.4.4. Reporting Component Dynamic Behavior

The reporting component will be in interaction with the analysis component and the simulation core component.

It will obtain the information related to sensors via the simulation core component (Simulation core component is responsible from realizing the database operations.). However, it will not get all the information hand in. It will obtain the list of sensors in the database and then compute the desired information using the list. It will obtain the statistics information from the analysis module hand in.

A sequence diagram summarizing this process is given in figure (11).



Sequence Diagram for Reporting Alarms in the Database

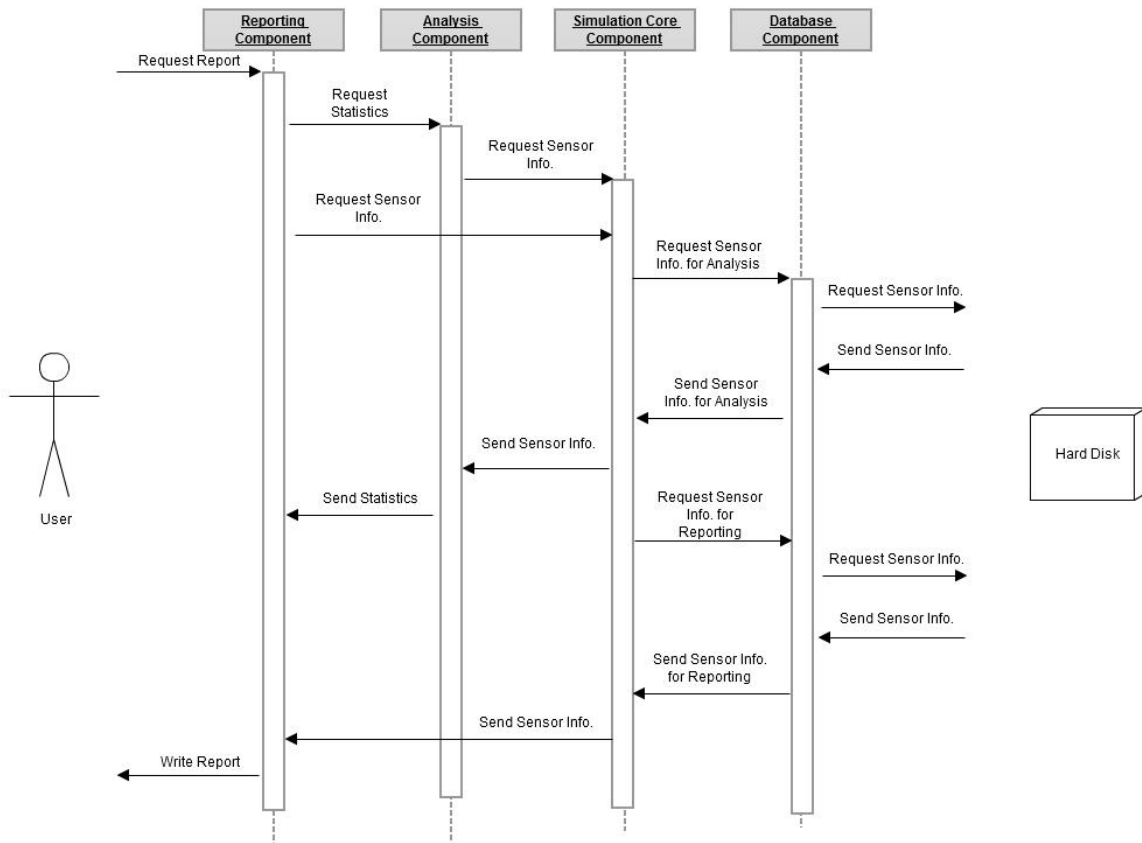


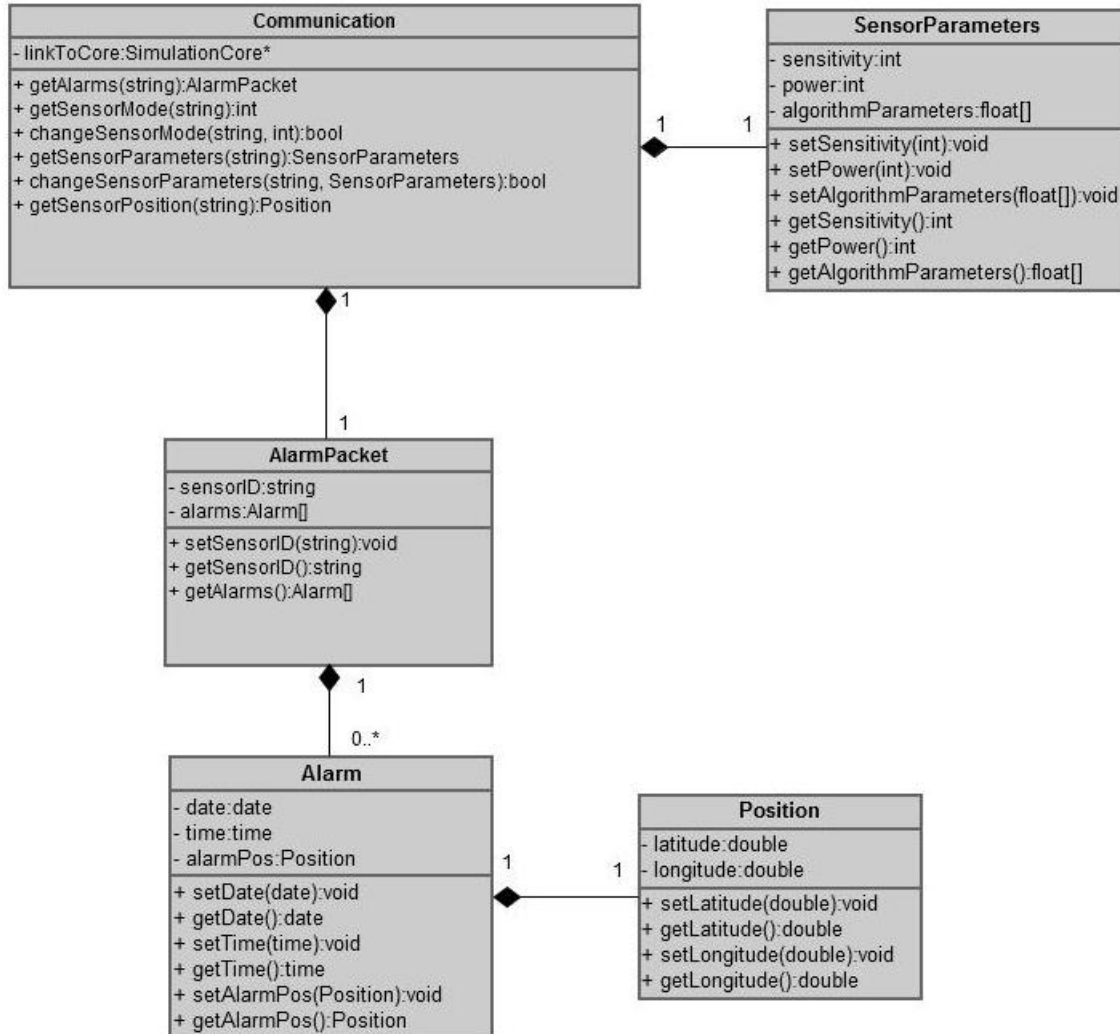
Figure 11. Sequence diagram for reporting alarms in the database.

5.2.5. Communication Component

It is the communication component that sends and receives data from the sensors. Details of the communication component are provided in the following sub sections. The class diagram describing the communication component is given in figure (12).



Class Diagram for the Communication Component

**Figure 12.** Class diagram for the communication component.

5.2.5.1. Communication Component Processing Narrative

Basically, there are two major responsibilities of this module; sending information to a sensor, getting information from a sensor.

Getting alarms from a sensor, getting mode or settings of a sensor and getting position of a sensor are getting information operations.

Changing sensor modes and parameters are sending information operations.



5.2.5.2. Communication Component Interface Description

The communication component has four interfaces; one with the sensor (via a wireless link), one with the user via the GUI, one with the simulation core component, and one with the sensor manager component.

The wireless communication interface will be provided with XBee modules. The details of these modules will not be given here. Please refer to tools and libraries section to see more details about these modules.

Via GUI, the user will be able to learn alarm information, mode, parameters and position of a sensor. The ID of the sensor will be provided in the GUI.

The communication component will be invoked by the sensor managing component. When the user wishes to change parameter or mode of a sensor, sensor managing component will pack and redirect the related information to the communication component.

The simulation core component will be invoked when the user wishes to learn alarm information, parameters, mode or position of a sensor. The related information will be sent to the simulation core component via the interface between it and the communication component.

5.2.5.3. Communication Component Processing Details

The computations performed by the communication component are not heavy. In an information getting operation, it packets the incoming packets and redirects it to the simulation core component. In an information sending operation, it packets the related information and sends it to the corresponding sensor via the wireless communication.

5.2.5.4. Communication Component Dynamic Behavior

The communication component will be in interactions with the sensor managing component and the simulation core component.

However, the communication component will not invoke the sensor managing component; it will be invoked by the sensor managing component.

The communication component will invoke the simulation core component when the user wants go get information related to the sensors. It will send these data to the simulation core so that they will be written to the database.

A sequence diagram summarizing this process is provided in figure (13).



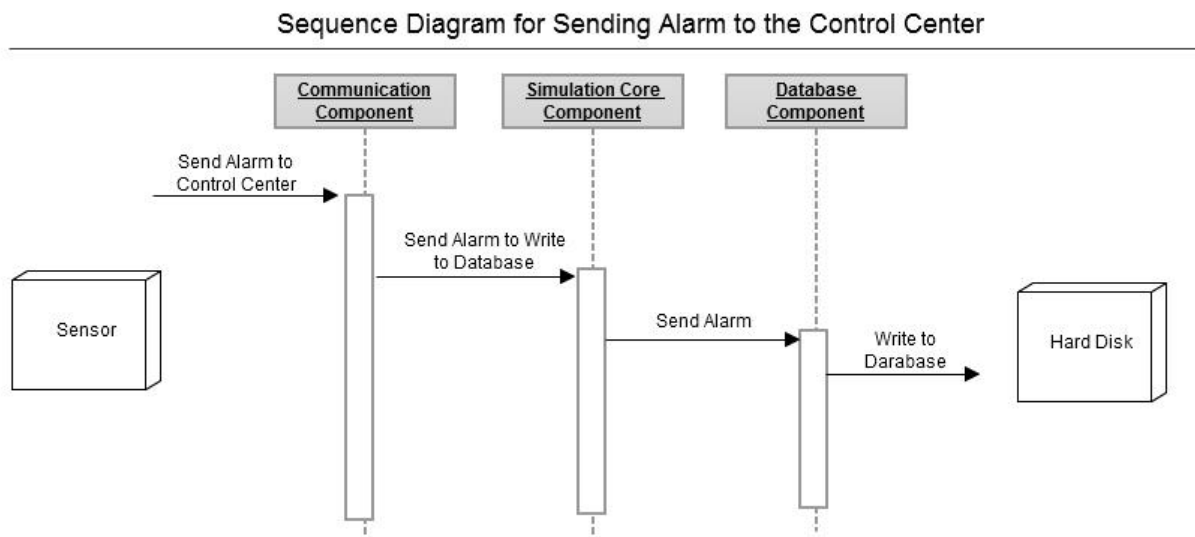
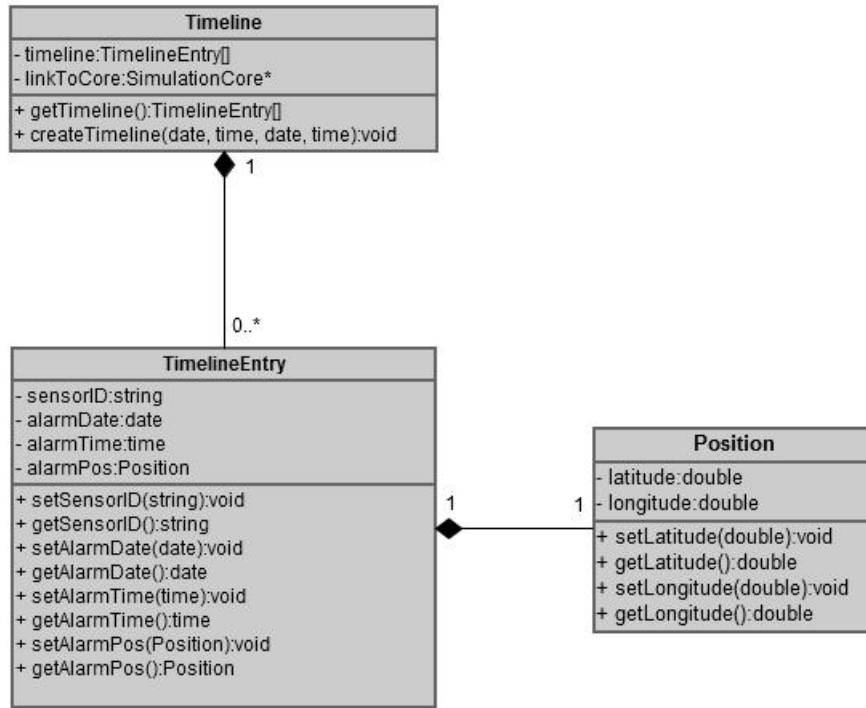


Figure 13. Sequence diagram for sending alarm to the control center.

5.2.6. Timeline Component

Timeline component is used when the user wishes to see the alarms in a time interval in the database in a list format. Details of the timeline component are provided in the following sub sections. The class diagram describing the timeline component is given in figure (14).

Class Diagram for the Timeline Component

**Figure 14.** Class diagram for the timeline component.

34

5.2.6.1. Timeline Component Processing Narrative

The responsibility of the timeline component is to create a list of alarms in the GUI for a desired time interval. It will list the alarms one-by-one. For each alarm, it will show ID of the sensor from whom the alarm is get, alarm date, alarm time, and position of the alarm.

5.2.6.2. Timeline Component Interface Description

Timeline component has two interfaces; one with the user via the GUI, and one with the simulation core component.

Via the GUI, the component will get the start time and date and ending time and date. It will create the timeline for the alarms get in this interval.

From the simulation core component, the timeline component will obtain the sensor data via the interface. By using this data, the component will compute the desired information.

5.2.6.3. Timeline Component Processing Details

The timeline component will obtain the sensor list of all sensors recorded in the database. It then processes this list and obtains the alarms and information related to these alarms (alarm



position, time, etc.) by investigating this list. It then sends these data to the GUI so that the user will be able to see the alarm list.

5.2.6.4. Timeline Component Dynamic Behavior

The timeline component will be in interaction with the simulation core component. The timeline component obtains information related to the sensor recorded in the database via the simulation core component. The simulation core component will invoke the database component to obtain the information and redirects it to the timeline component.

A sequence diagram summarizing this process is provided in figure (15).

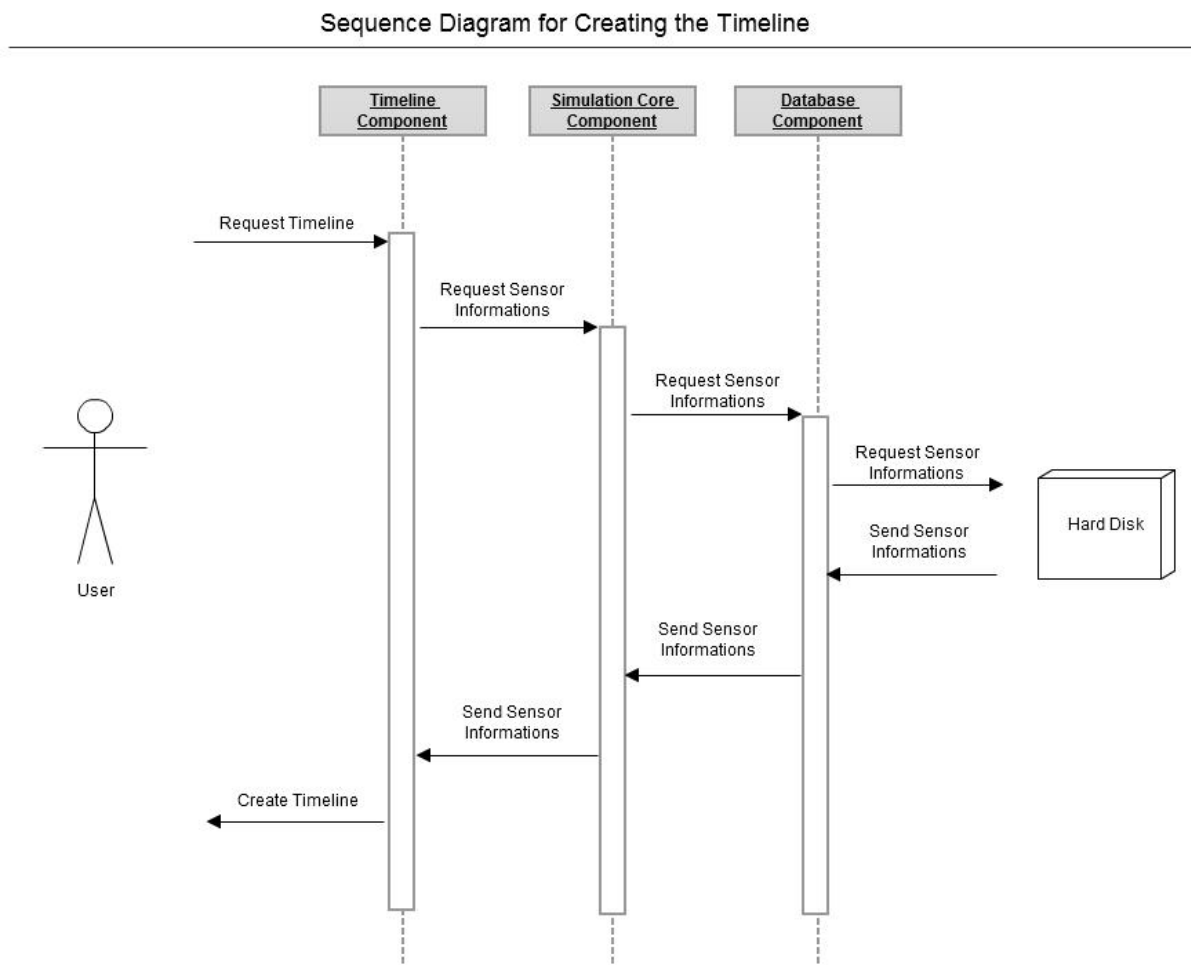


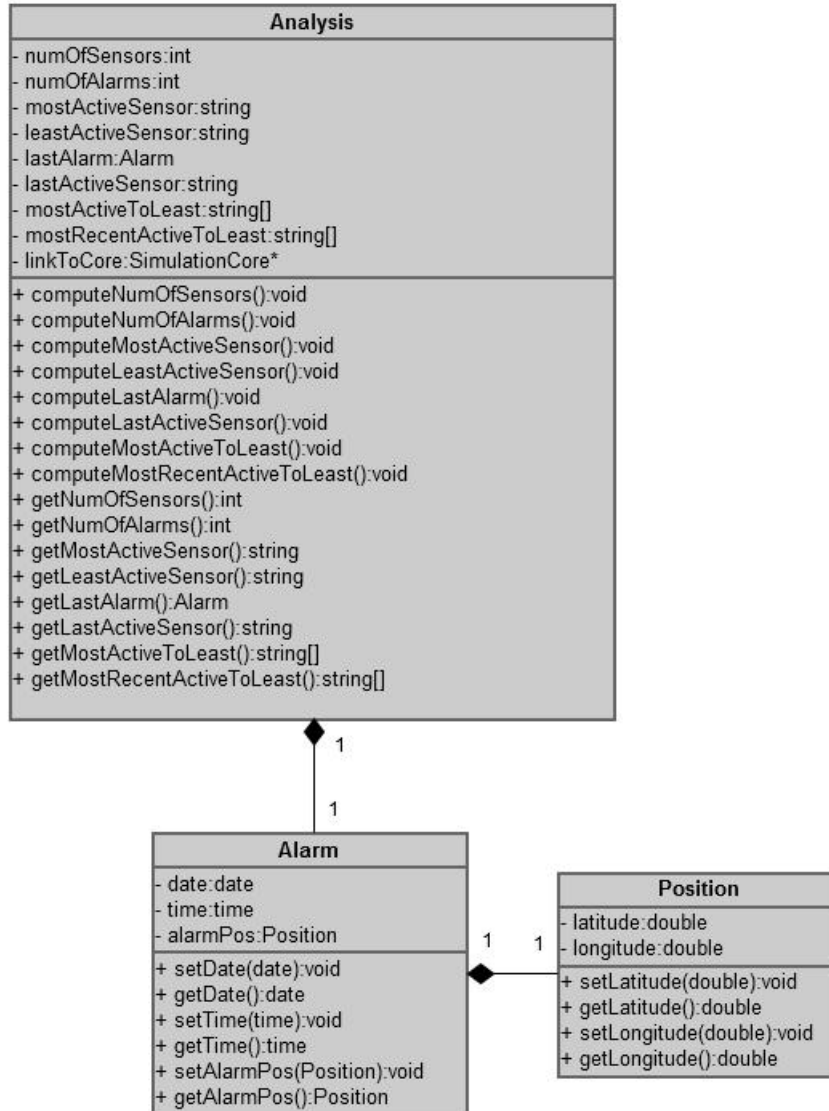
Figure 15. Sequence diagram for creating the timeline.

5.2.7. Analysis Component

With the analysis component, the user will be able to see statistics related to the sensors. Details of the analysis component are provided in the following sub sections. The class diagram describing the analysis component is given in figure (16).



Class Diagram for Analysis Component

**Figure 16.** Class diagram for the analysis component.

5.2.7.1. Analysis Component Processing Narrative

The responsibility of the analysis component is to process the information related to the sensors registered in the database, and gather statistics about them. More specifically, it will compute number of sensors, number of alarms, most active sensor, least active sensor, last alarm obtained, last active sensor, list of sensors from most active to least, list of sensors from most recent active to least recent active.



5.2.7.2. Analysis Component Interface Description

The analysis component has two interfaces; one with the simulation core component, and one with the user via GUI.

The list of sensors in the database is transferred via the simulation core component. Please note that it is the responsibility of the analysis component to process and gather information related to the sensors.

Via the GUI, the analysis component will be able to be invoked by the user.

5.2.7.3. Analysis Component Processing Details

When the analysis component gets the list of the sensors in the database, it will process the list and gather the statistics defined in the processing narrative section. After the processing, it will send the information to the GUI so that the user will be able to see the statistics.

5.2.7.4. Analysis Component Dynamic Behavior

The analysis component is in interaction with the simulation core component. When it needs the list of sensors, it invokes the simulation core component to get the list. Then, the simulation core component obtains the list and redirects it to the analysis component.

A sequence diagram related to the use case realized by this process is provided in figure (17).

37

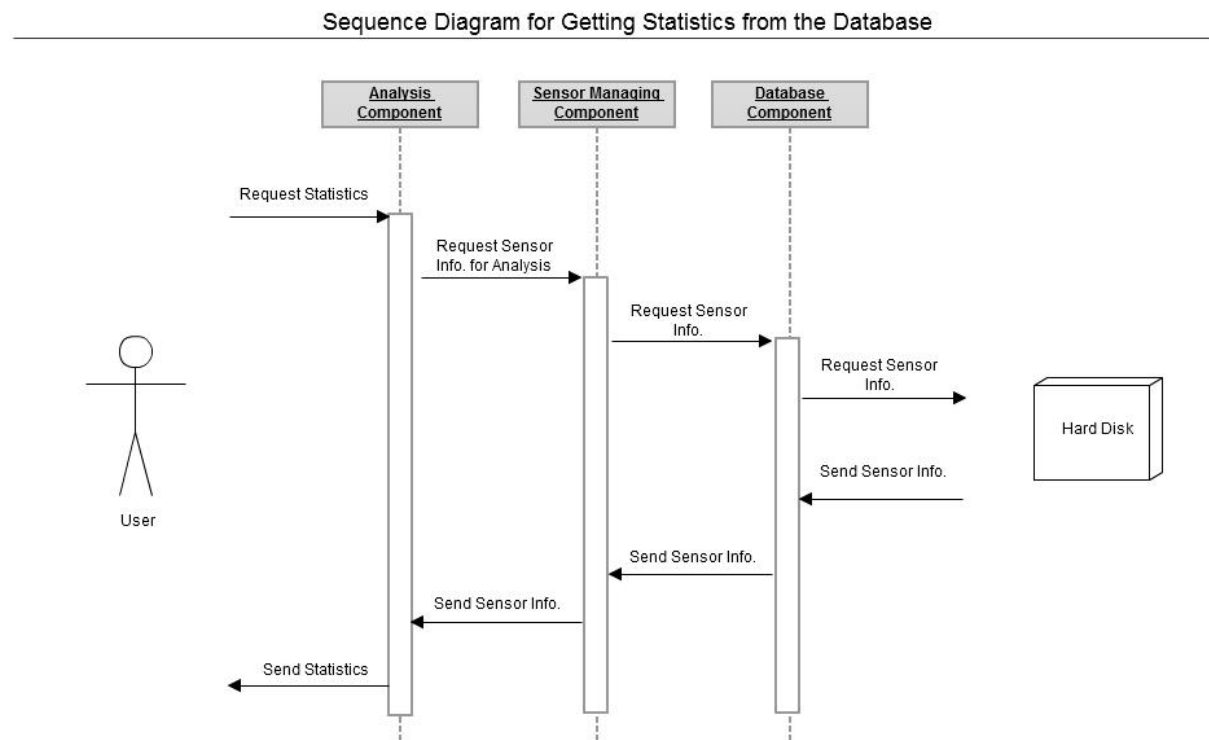


Figure 17. Sequence diagram for getting statistics from the database.



5.3. Design Rationale

The system is decomposed in this way because the requirements of the system are quite different from each other. These requirements must be realized by distinct components. Hence, decomposing the system into components in this way is intuitive.

Decomposition of the system also makes testing and debugging process easier as components can be tested and debugged independently.

Nevertheless, this approach can decrease the optimality of the system. More specifically, defining many classes with many methods such that these methods call each other can affect the performance of the system in an undesirable fashion. However, as the control center application will not be real time, this issue is not so important.

5.4. Traceability of Requirements

In the following table, each functional requirement of the system is defined in the “Requirement” column; the component that satisfies the requirement is defined in the “Component” column, the figure number illustrating the sequence diagram that summarizes the process realizing the requirement is defined in the “Figure” column.

Requirement	Component	Figure
Sending alarm to the control center	Communication	
Reporting alarms in the database	Reporting	
Getting statistics from the database	Analysis	
Changing mode/parameters of a sensor	Sensor managing	
Registering a new sensor to the system	Sensor managing	
Deleting a sensor from the system	Sensor managing	
Creating timeline	Timeline	



6. User Interface Design

6.1. Overview of User Interface

This part contains explanations about usage of the system. In this part, users are provided enough information about interfaces and possible actions can be made. We will have two types of users, namely Admin and Tester.

Tester: Checks data sent by sensor and gets reports.

Admin: Checks data sent by sensor, gets report, adds new geophone and user to the system. Admin has more authority than testers.

6.1.1. Tester Interface

6.1.1.1. Login Page

After starting application, this page is opened in front of user. User is asked to enter a valid username and password to login to the system. If the username/password combination is not valid, user is warned about that error. User must check for his login information and must provide new combination. For new users, there is a Register link in this page. After clicking this link, user will be directed to Registration Page where new users register themselves to the system. Activity diagram for Login is shown in figure (18).



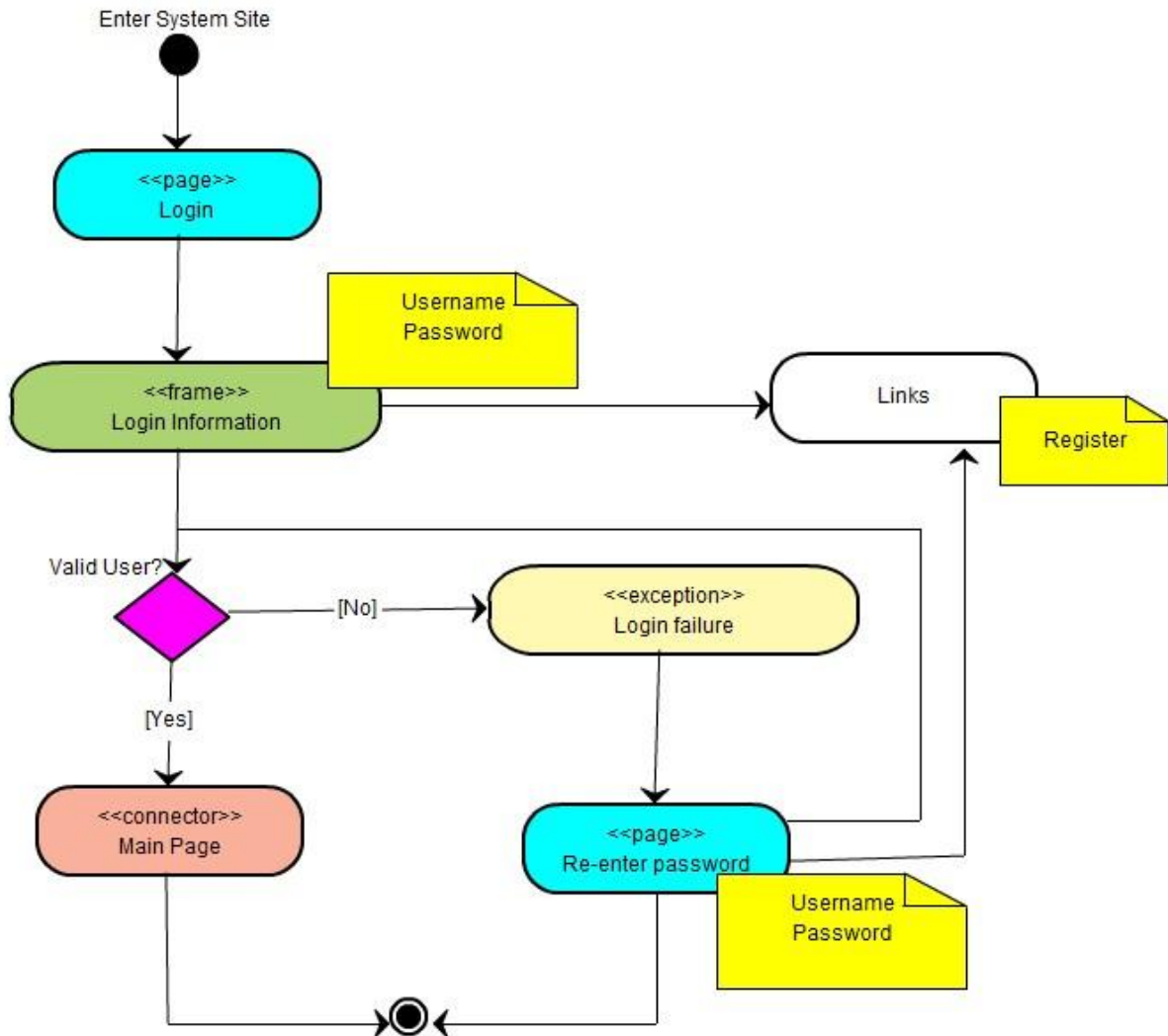


Figure 18. Activity diagram for the login page.

6.1.1.2. Registration Page

In this page, there is a registration form. User must fill all the fields before submitting the form. If there exists a username or an email address with selected ones user is warned about these errors (Figure (24) in part 6.2). User is asked to provide new username/email address. When successful registration is achieved, user will be directed to Login Page (as explained in diagram in Figure (19)). After approval by admin, user can enter the system freely using his username and password.

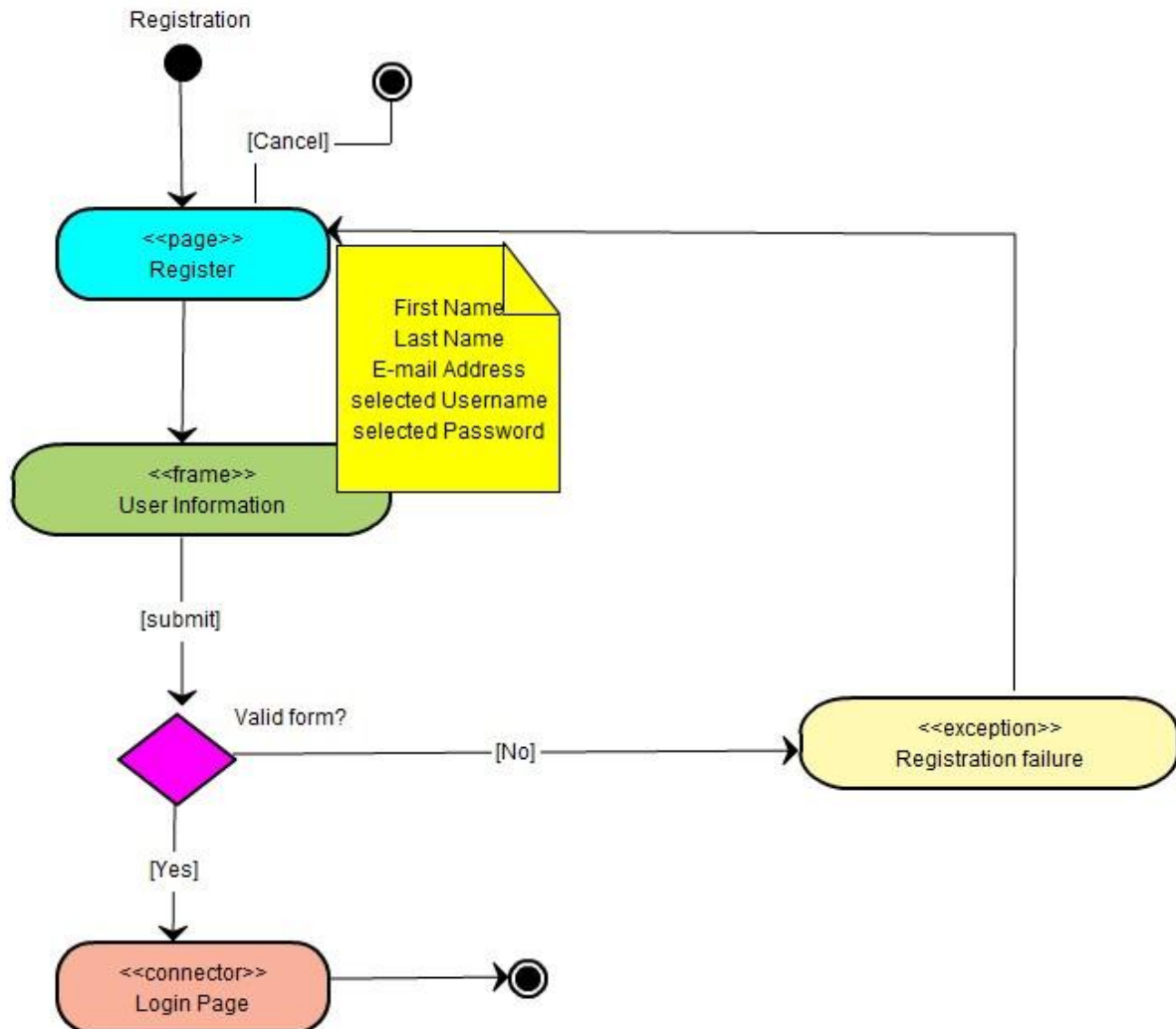


Figure 19. Activity diagram for the registration page.

6.1.1.3. Main Page

After successful login, user will be directed to this page. This page contains a map which is produced using Nasa World Wind Java SDK ([5]). This map shows the place of each geophones in a certain region which are signed by rectangular-shaped marker. Colors of markers display important information about the state of geophones.



: Last alarm time is less than 15 minutes



: Last alarm time is less than 24 hour but more than 15 minutes



: Last alarm time is more than a day

User can get information about geophones clicking on colored markers (look at Figure (27) in part 6.2.3). This information is ID, latitude, longitude and last data time of geophone. By using “click here” link on opened information window, user will be directed to Report Page. Activity diagram for operations in Main Page is given in figure (20).

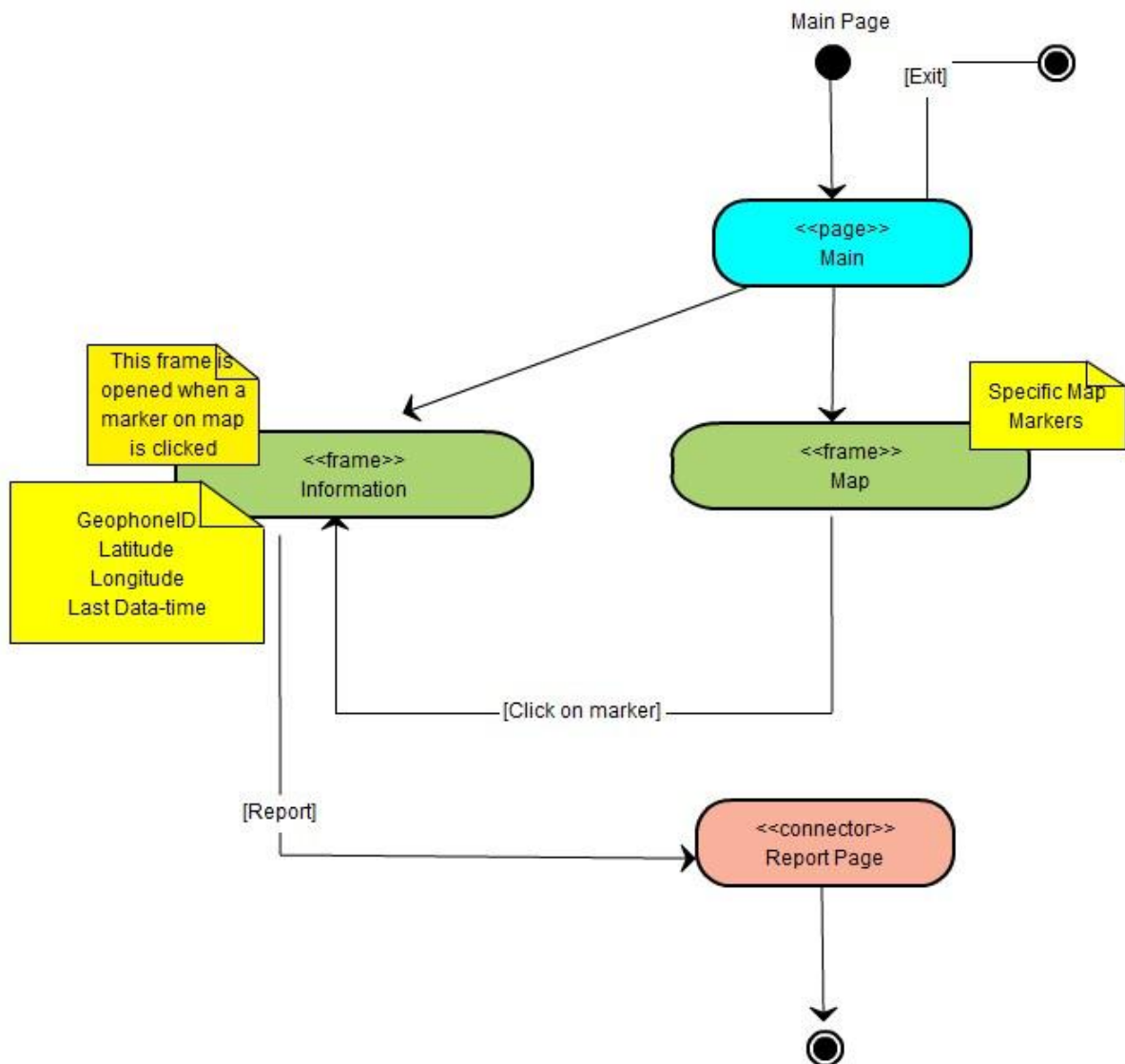
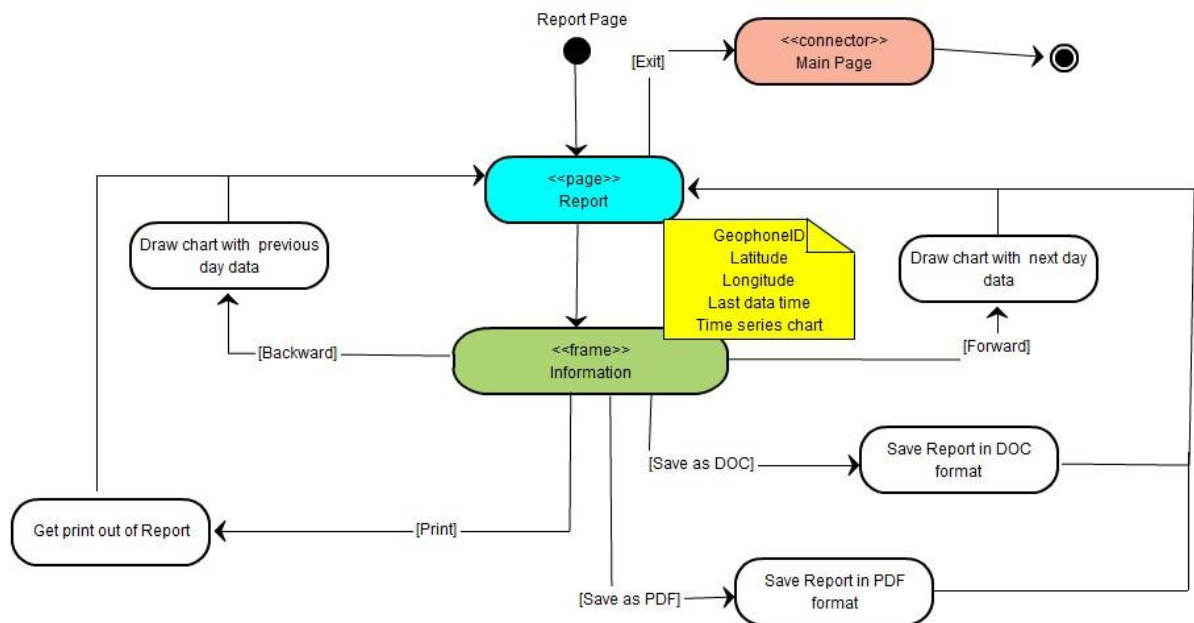


Figure 20. Activity diagram for the main page.



6.1.1.4. Report Page

User can see report about sensor which is clicked lastly. GeophoneID, latitude, longitude and last data time is contained in this page. User also will be able to see time series chart of geophone. This chart shows last 24 hour flow of incoming signals. Below chart, there are forward and backward buttons. Clicking on these buttons, user can see the charts of previous and next days (look at Figure (21) to see activity diagram). There will be Print button to print out the report. In addition, there are two buttons to save Report on computer, namely “Save As DOC” and “Save As PDF”.



43

Figure 21. Activity diagram for the report page.

6.1.2. Admin Interface

6.1.2.1. Login Page

Login page of admin will be the same as testers (explained in part 6.1.1.1). The only difference is that admin will not need to register to system.

6.1.2.2. Control Page

There are two frames in this page. In the first frame, users who want to register to the system will be displayed. Admin can select people and after pressing “Add Selected Users” button, users will be added to the database of the system. There is also “Delete Selected Users” button. Admin can reject requests of some users by selecting them and pressing that delete button. Second frame will be used for adding new sensors to the system. There will be a form to be filled. This form asks user to enter latitude, longitude and geophoneID. After filling this



form, user will click on “Add Geophone” button and geophone will be added to the system. Result of each activity can be made is shown in figure (22).

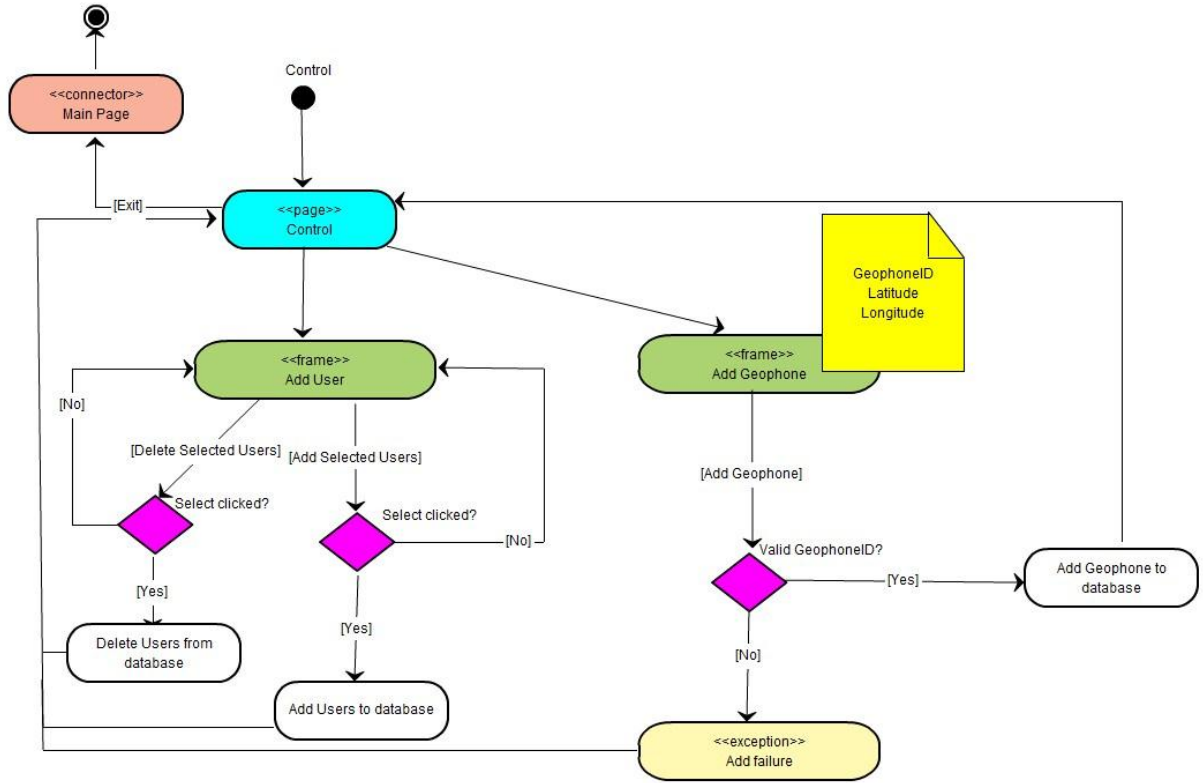


Figure 22. Activity diagram for the control page.

6.1.2.3. Main Page

This page will be the same as tester's Main Page (explained in part 6.1.1.3).

6.1.2.4. Report Page

This page will be the same as tester's Report Page (explained in part 6.1.1.4).

6.2. Screen Images

6.2.1. Login Screen

Login screen can be seen in figure (23).





Figure 23. Login screen

When wrong username/password is entered, User will be warned about this by screen as shown in figure (24).

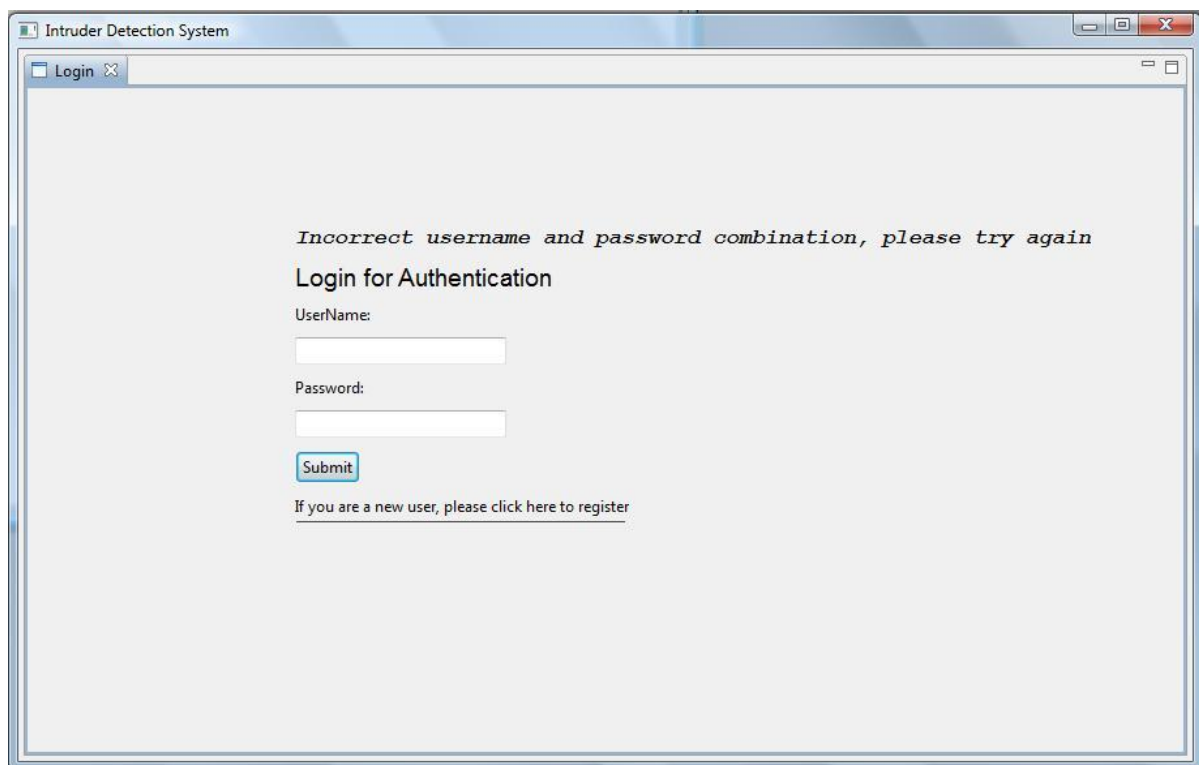
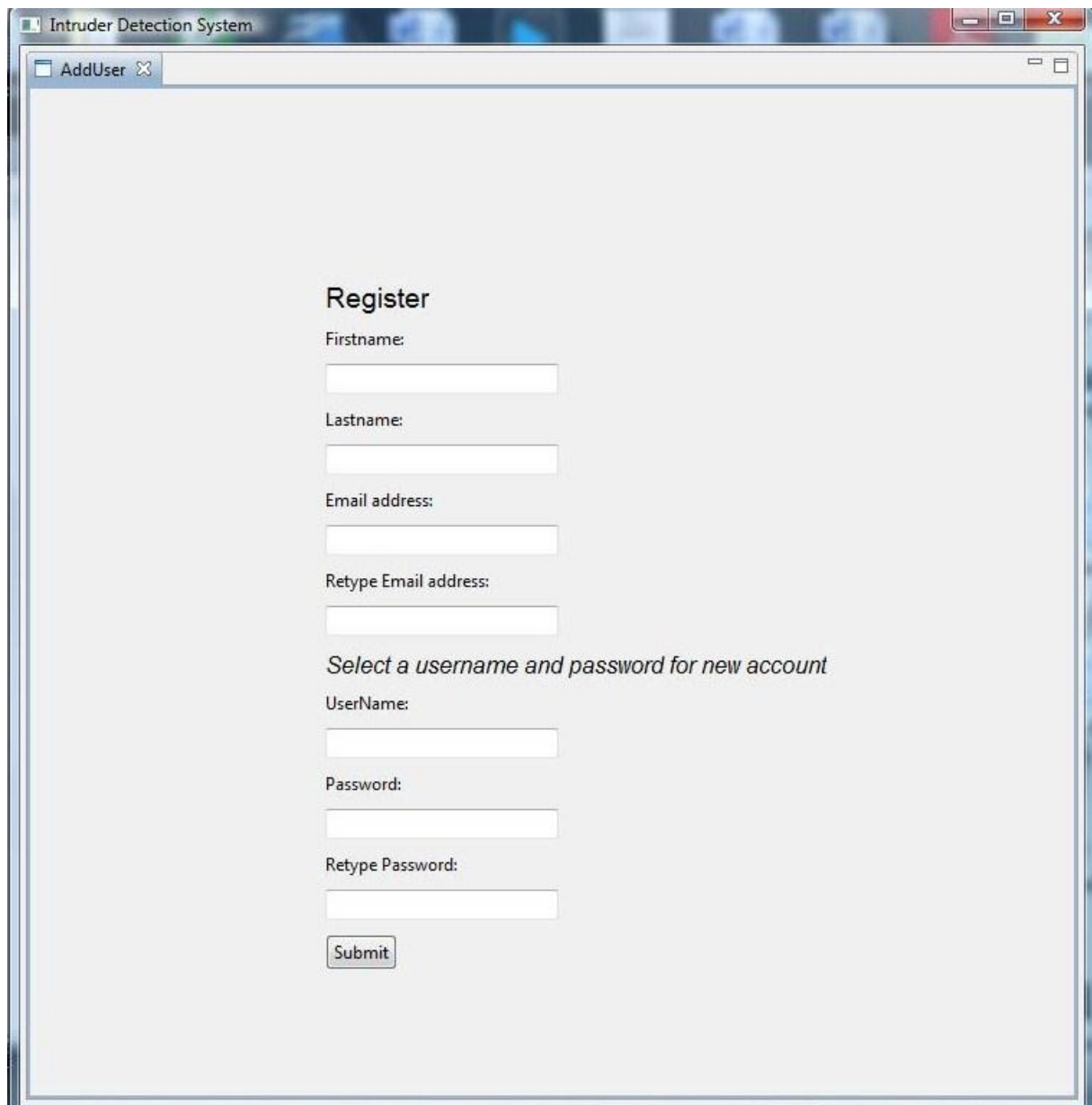


Figure 24. Error-login screen



6.2.2. Register Screen

Register screen can be seen in figure (25). Moreover, error register screen can be seen in figure (26).

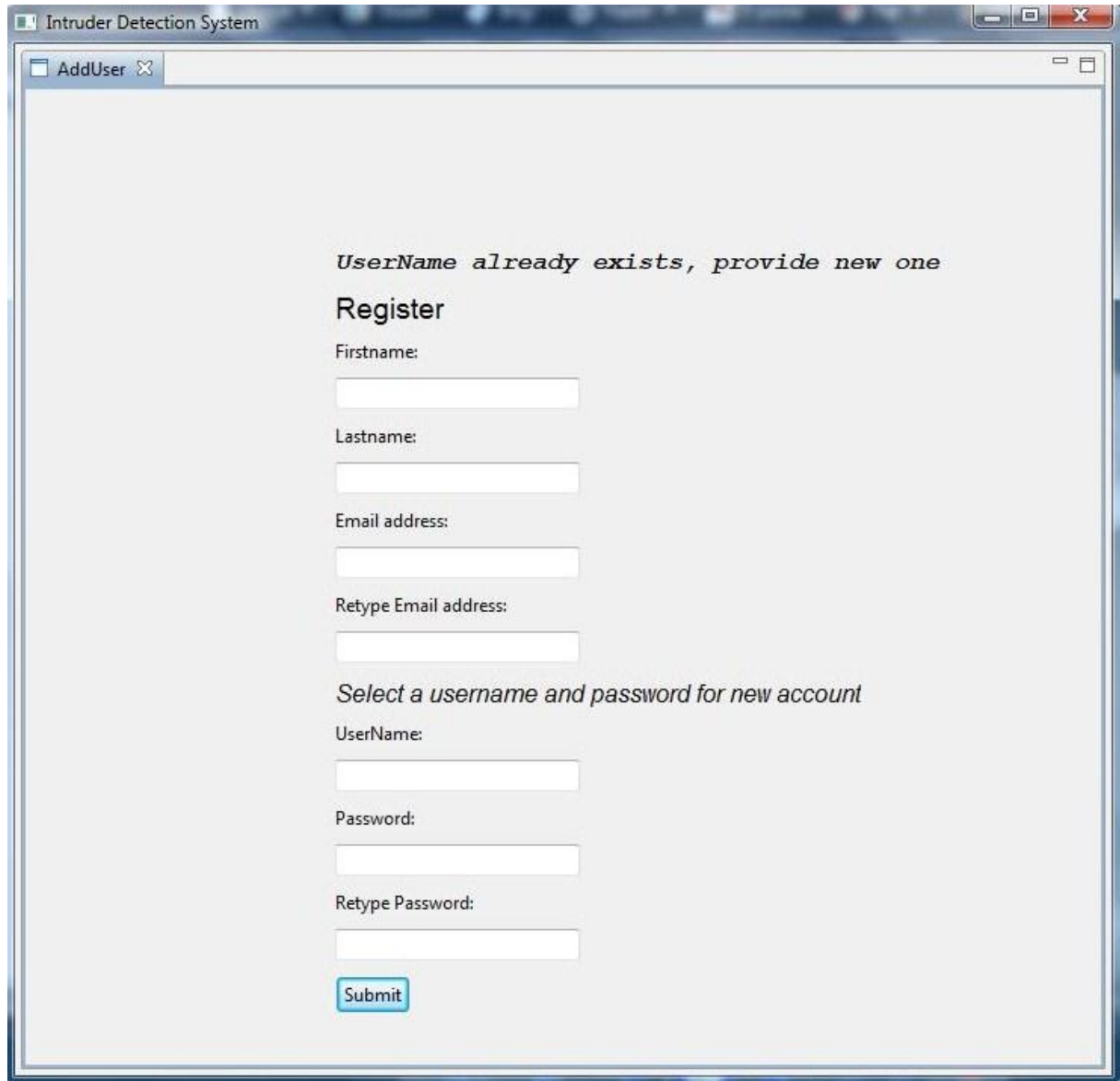


The screenshot shows a web application window titled "Intruder Detection System" with a sub-tab labeled "AddUser". The main content area is titled "Register" and contains the following form elements:

- Firstname:
- Lastname:
- Email address:
- Retype Email address:
- Select a username and password for new account*
- UserName:
- Password:
- Retype Password:
-

Figure 25. Register screen





The screenshot shows a window titled "Intruder Detection System" with a tab labeled "AddUser". The window contains an error message in a monospace font: "UserName already exists, provide new one". Below the message is a "Register" section with the following fields: "Firstname:", "Lastname:", "Email address:", "Retype Email address:", "Select a username and password for new account", "UserName:", "Password:", and "Retype Password:". Each field is followed by a text input box. At the bottom of the form is a "Submit" button.

Figure 26. Error-register screen

6.2.3. Main Screen

Main screen can be seen in figure (27).



Red :

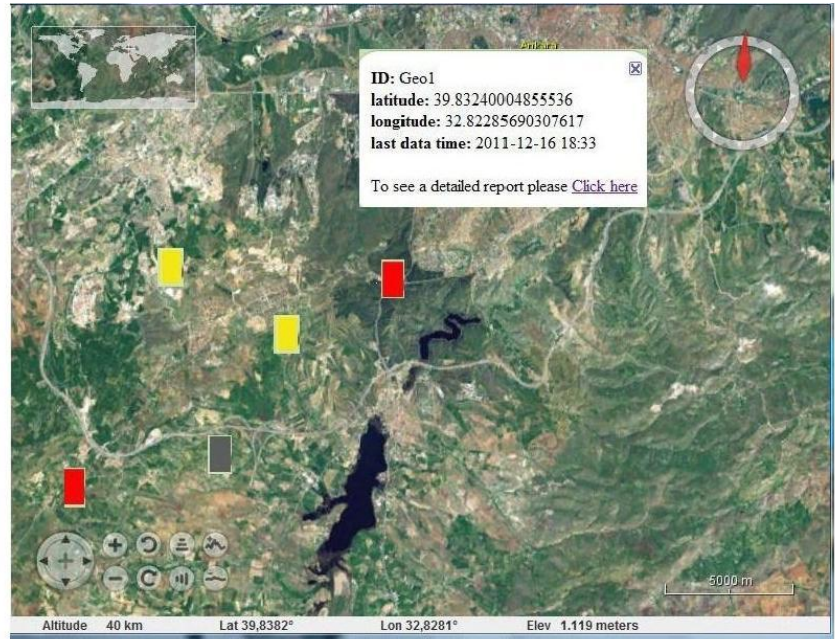
last data in less than 15 minutes

Yellow:

last data in less than 24 hours but more than 15 minutes

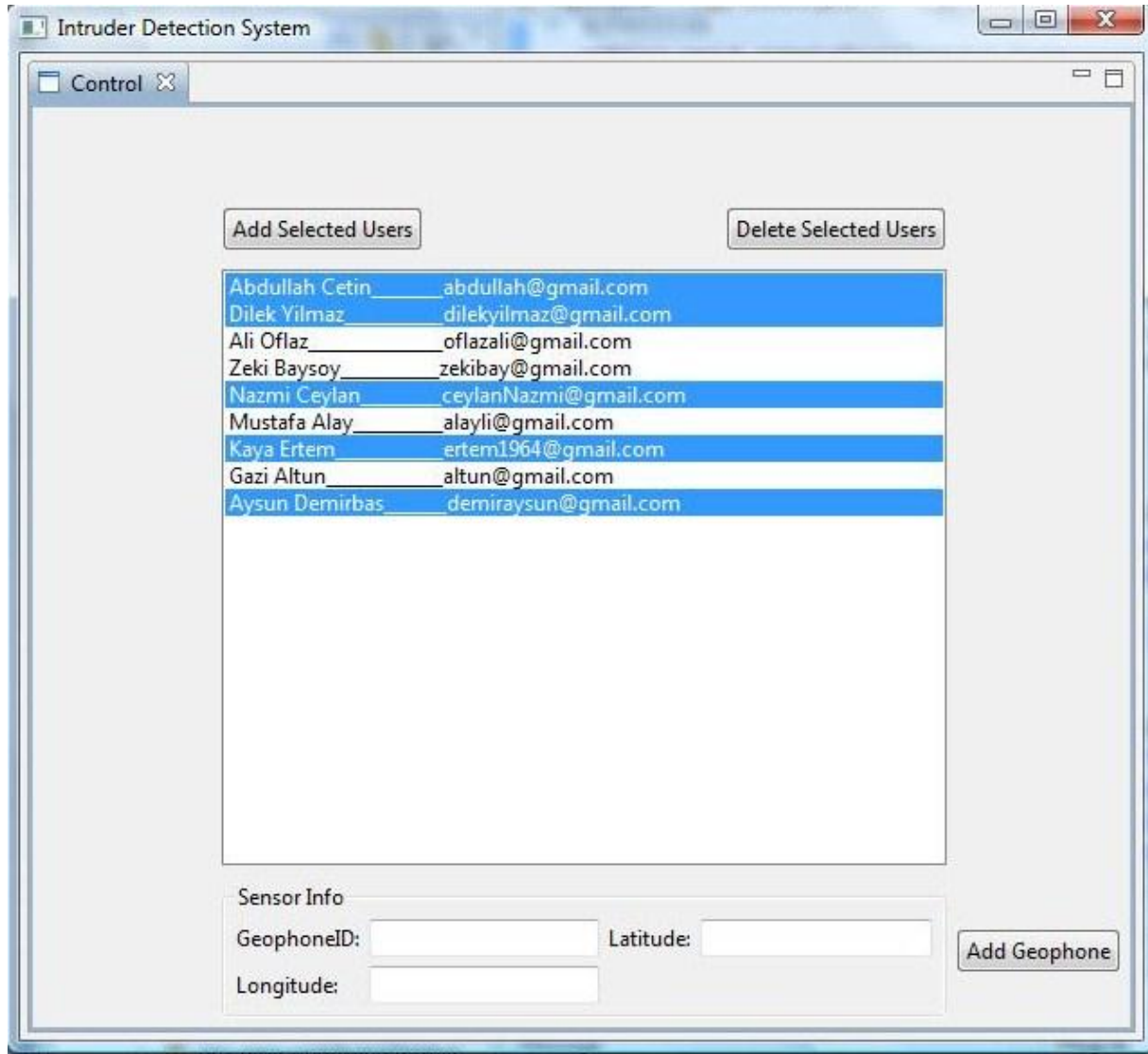
Gray :

last data in more than 24 hours

**Figure 27.** Main screen**6.2.4. Control Screen**

Control screen is given in figure (28).



**Figure 28.** Control screen

6.2.5. Report Screen

Report screen can be seen in figure (29).

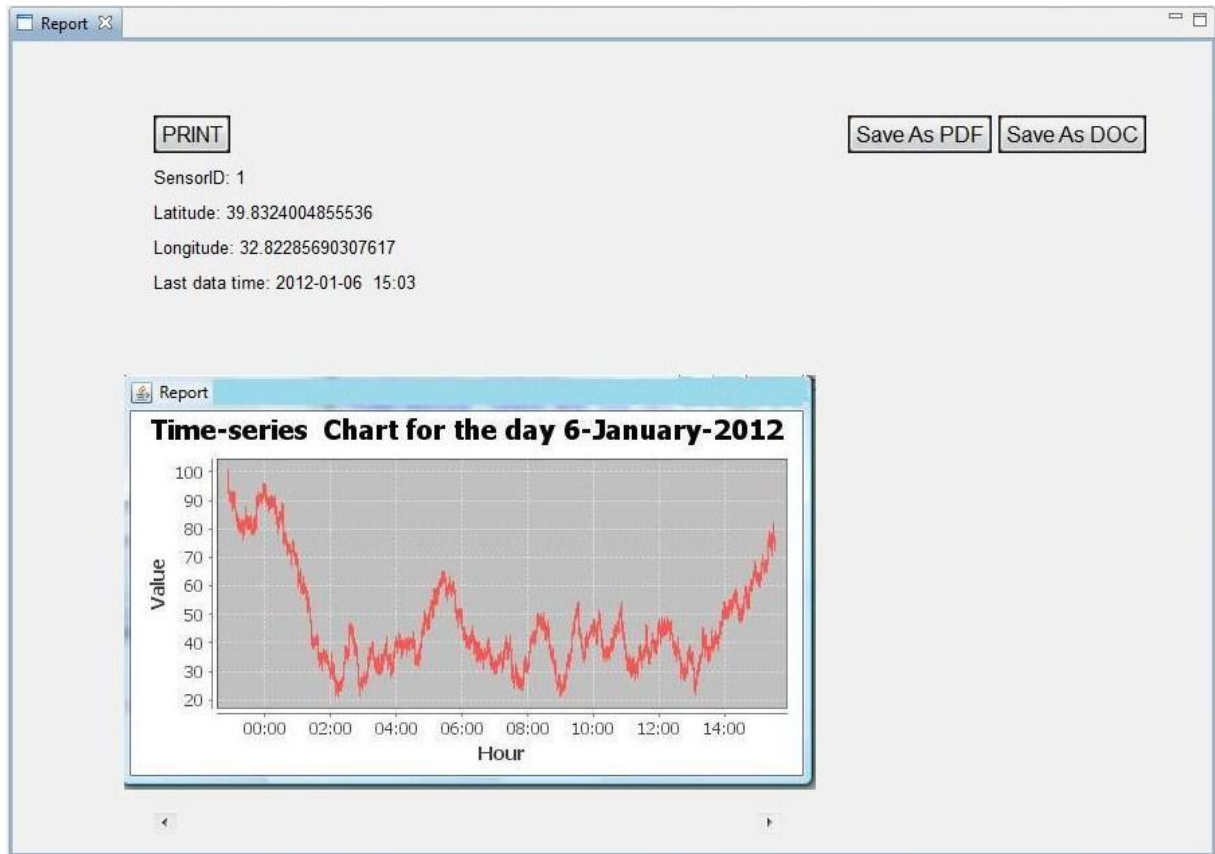


Figure 29. Report screen

50

6.3. Screen Objects and Actions

While on Control screen or Report screen, if user presses the close marker (Figure (30)), user will be directed to Main Page.



Figure 30. Close markers on Control and Report screens

There will be a map on Main screen as explained in part 6.1.1.3. By clicking on middle of rotate marker, user will be able to rotate around world origin. Using left, right, down and up signs exist on rotate marker, user will be able to move to left, right, up and down.(look at figure (31)).



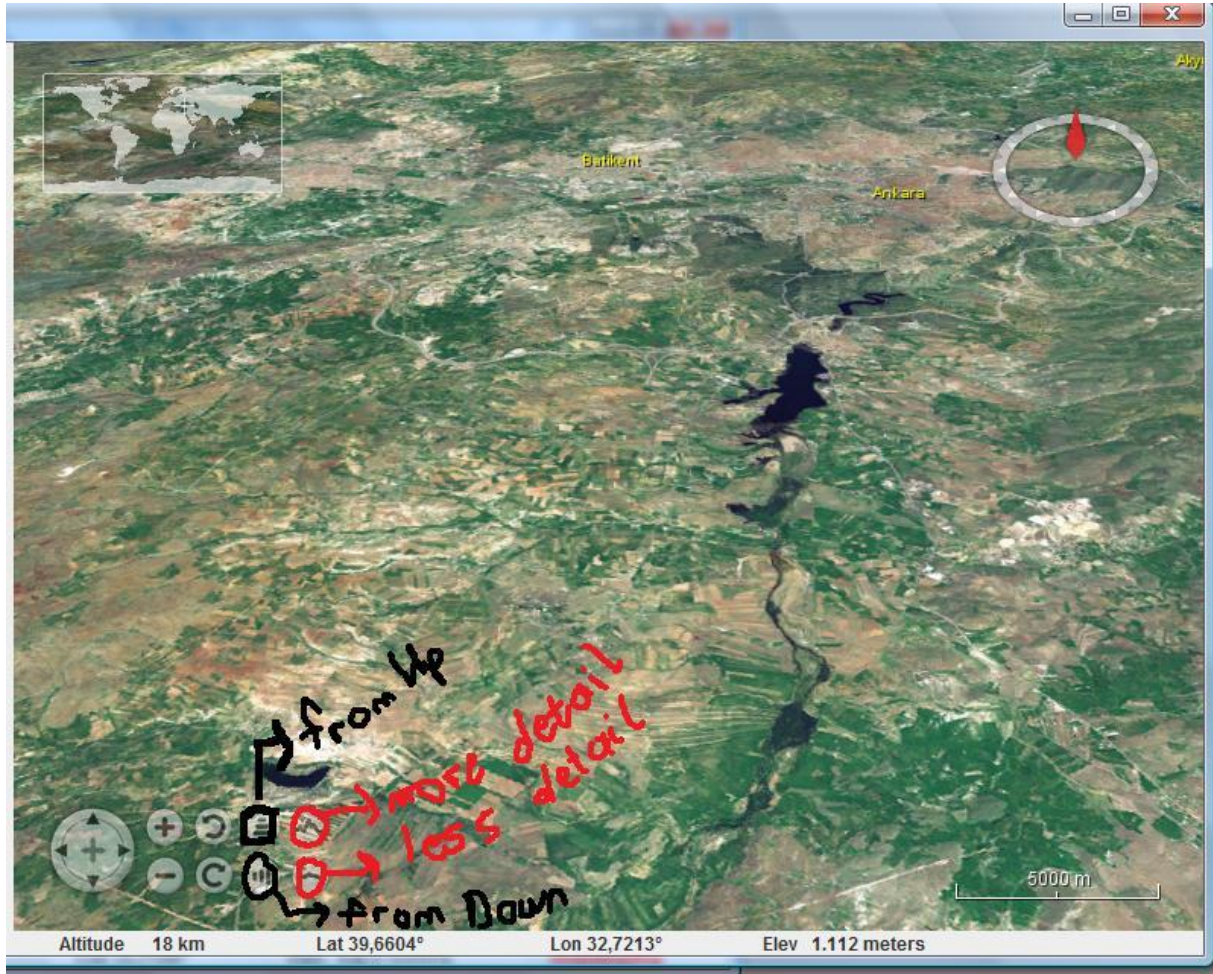


51

Figure 31. Rotate and zoom actions on Main Screen

Using + / - signs on map as shown in figure (31), user can zoom in/ zoom out.

User can see map from up and down by clicking from Up and from Down markers as shown in figure (32). User can make choice about the details of location which is seen. User can see more detail or less by using markers on map (look at Figure (32)).



52

Figure 32. Up, down and detail actions on Main screen

Highly detailed map scene can be seen in Figure (33).





Figure 33. Detailed Map

7. Detailed Design

The overview of each component was provided at System Architecture section. In this section we will provide the internal details of these components. By and large we will provide information about each of the components classification, definition, responsibility, constraint, composition, interaction, resource and processing details. The aim of this part is to contain all the details that will be needed for implementation phase.

Please note that the class diagrams of the components are given individually in the system architecture chapter. In this part, instead of giving them again, we provide the class diagram of the whole system. Class diagram of the whole system can be seen in figure (34).



Class Diagram of the Intruder Detection System

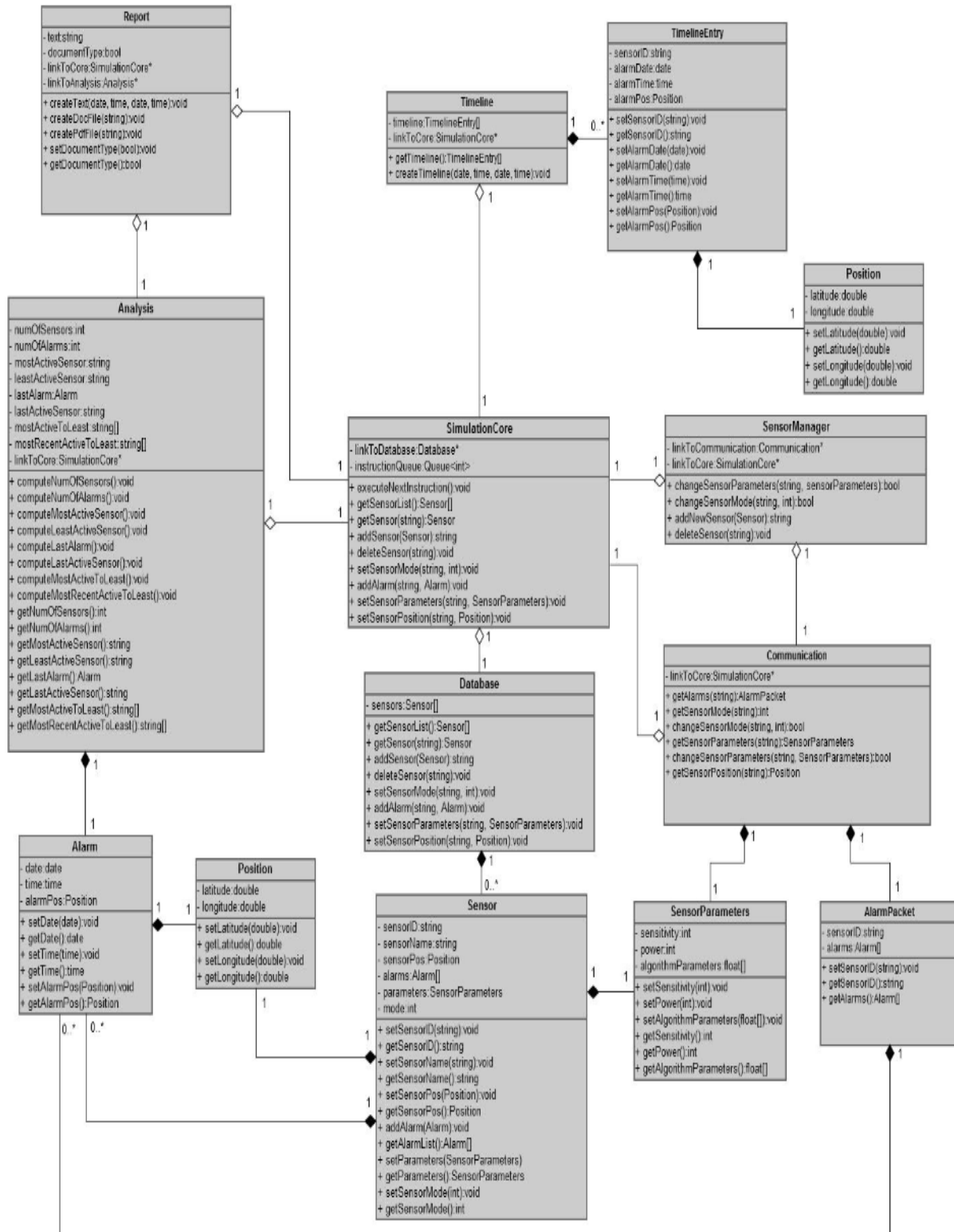


Figure 34. Class diagram of the whole system.



7.1. Database Module

In this section detailed design of Database Module is provided.

7.1.1. Classification

Database Module is a module of the Intruder Detection System which interacts with the Simulation Core Component and Database System.

7.1.2. Definition

This module is the place where the interaction with the database and the whole system is held. Database module is the interface between the Database Management System (DBMS) and the Simulation Core Component. This module organizes the reading and writing process on the database. No other module has direct access to database. This module has communication with Simulation Core Component. And the whole database accesses are carried out with the help of this communication.

7.1.3. Responsibilities

The main responsibility of this component is to manage the requests come from the Simulation Core Component. Any of the other components makes their requests on Simulation Core Component. These requests are generally about reading from database or writing to database. For each of the request coming from a different component, performing the corresponding reading and writing operations is also a responsibility of this component.

7.1.4. Constraints

The constraints of the Database component are:

Database unit must interact with Simulation Core Component without serious time-delays to be able to get and send the information accurately.

The module is completely dependent to other modules. If a wrong information transfer is made between the Simulation Core Component and the other modules. There is no way to handle that situation.

It should take the information to be written to database without much time-delay because other modules also can make data reading request and if the data supposed to be written to data base is not written yet, then serious problems like not reporting the alarms could occur.

7.1.5. Compositions

The Database Module is composed of one sub component. That sub component is composed of two sub components. One can see the class diagram at the Database Module part at the



System Architecture part (Part 5) to see the details of these subcomponents. The descriptions of these sub components are provided below:

Sensor Sub Component:

This sub component is used for reaching the information related to sensors. When writing data to the database, data shall be written in a structured manner. Similarly when reading information from database structured data shall be parsed and hold into relevant data structures. For this job sensor sub component has two sub components namely Alarm and Position sub components.

Alarm Sub Component:

This component holds the date, time and alarm positions. Organizations of these particles are done by this sub component. This sub component is responsible from reading and writing of these particles.

Position:

This component holds the latitude and longitude information. Organization of this information is done on the Position Sub Component. This sub component is responsible from reading and writing of these particles.

7.1.6. Uses and Interactions

Database component has Sensor sub component for performing the interactions with the database. This sub component is responsible for communicating with database. It can perform operations like inserting data to database, deleting, updating or retrieving. This sub component consist classes namely Alarm and Position classes. These classes are used to represent the entity structure of the database. When there is a request to Simulation Core Component for reaching the database, the related class instances will be created. If the request is a reading request associated class will be filled with requested information from database tables. If the request is a writing request the associated class will be filled with the information send from the Simulation Core Component. Then this information will be written to relevant tables in the database tables.

7.1.7. Resources

Our Database Module reaches to Database System with the help of PostgreSQL DBMS. PostgreSQL, is an object-relational database management system (ORDBMS) available for many platforms including Linux, FreeBSD, Solaris, Microsoft Windows and Mac OS X [6].

Because of we are implementing Database interaction module of our system with Java, we use the industry standard for database-independent connectivity with our program and the



database. This industry standard is Java Database Connectivity API (JDBC), it provides an appropriate connection between the system and the database.

7.1.8. Processing

Database component carries on its operations over Sensor sub component. In this sub component sensor information is processed with the help of Alarm and Position objects. These objects have some variables for placing the relevant values. Furthermore they have some functions to process data. The latitude and longitude values taken from GPS are stored at Position object. These values can be read, written, deleted or updated by the help of `setLatitude(float)`, `getLatitude()`, `setLongitude(float)` and `getLongitude()` functions provided by this object. On the other hand; date, time and position values of the alarms are stored at Alarm object. This object provides set and get functions for the specified values. Time and date information provides information about the times of the Alarms. These values will be used by many modules like Timeline module, Reporting Module and Analysis module. Position values are also important for Reporting and Analysis modules. Furthermore position information may be used to determine the direction of the intruder as a future study. In addition to these, Sensor sub component provides `setSensorMode(int)` and `getSensorMode()` functions to read and update sensor status in the database. This function is needed by Sensor Managing Module to update sensor information. `deleteSensor(string)` method finds the sensor defined by its ID given as string to it, and deletes the information related to that sensor from the database.

57

7.1.9. Interface / Exports

As a result of all of its duties this component makes it creates an interface called Sensor. This interface is provided to Simulator Core Component.

So now Simulator Core Component is aware of the alarm times, dates and positions and also all of the sensor settings. With the help of this interface Simulator Core Component may change, delete or retrieve information from the database.

7.2. Simulation Core Module

In this module detailed design of Simulation Core Module is provided.

7.2.1. Classification

Simulation Core Module is a module of the Intruder Detection System which interacts with all of the modules of the system. It is actually main component of the system. All the data flow is done on this module.



7.2.2. Definition

This module is the place where the whole data flow of the system is organized. The only module that has interaction with database module is Simulation Core Module. It creates an interface for all of the other modules of the system.

This module takes many requests for reaching the Database Module. Mainly what this module does is to arranging the requests, putting them in order and processing these requests as fast as possible.

After fulfilling the requests of the relevant modules, it notifies them that the work requested is performed.

7.2.3. Responsibilities

In general main responsibility of this component is to organize the data flow of the system. This module is always open to requests. When a new arrived request is come it must be remembered. So this modules main work is to remember the requests make a list of them and fulfill the requests.

To give better illustration of the responsibilities of this module following can be listed as the main responsibilities of this component.

- Make polling to see that if there is an incoming request to reach the Database Module.
- Put the requests to the, request list.
- Perform the requests.
- Notify the relevant module after the requests performed.

Another responsibility of this module is to provide a communication interface to the other modules of the Intruder Detection System. So, receiving data from the modules and transmitting data to the modules is a vital responsibility of this module.

7.2.4. Constraints

By and large there are two main constraints of this module.

The requests taken from the other modules must not be lost. Putting them to a list and processing from that list is so much important for the reliability of the data. If one of the requests is lost then this would cause to missing an intruder. Or if one of the requests are done more than once by mistake, this would slow down the system as well as having an improper result.

Another constraint of this module is the timing constraint. It must response in a small amount of time to the requested components. If it does not answer in a short amount of time, then the requests will accumulate and system will slow down which means having unnoticed intruders.

7.2.5. Compositions

This module is composed of one sub component which is called as Simulation Core.



Simulation Core: This object has the work to hold the request list. It has a queue in it to hold the requests. Furthermore it makes communication with Database Module with the help of linktoDatabase member. In addition to that it has some functions like executeNextInstruction(), getSensorList(), addAlarm(string, Alarm) to read data from Database Module and send data to Database Module. Details of these functions will be provided at processing section.

7.2.6. Uses and Interactions

This module has communications with, Database module, Sensor managing module, Timeline module, Reporting module, Analysis module and Communication Module. Basically it has interactions with all of the modules of the system. It provides this interface with the sub component Simulation Core. It can take requests, make a list of them and perform one by one. After performing the requests it sends notification to the modules that the request is taken.

The only module that does not send requests to Simulation Core Module is the Database Module. Simulation Core Module makes data reading or writing request according to the incoming request. And when the Database Module makes the work it also notifies Simulation Core Component. When a notification comes from Database Module, it communicates with the module that sent the last request and sends a notification with the relevant data if it was needed.

7.2.7. Resources

This module needs some connection rules to construct the communication protocols of the system which will be provided to the system before the run.

Furthermore this module only needs C++ data structures for having the queue structure. Apart from that there is no resource is needed for this part.

7.2.8. Processing

This module is responsible from data flow of the system. There is no interaction between the other modules of the system. They only interact with the Simulation Core Module. This provides the modularity of the system.

This module continues its process with polling for the incoming requests. When a new request comes the module is notified. And it puts the request to instructionQueue member. Then it starts new instructions to follow from instructionQueue with the help of executeNextInstruction. If the instruction is a Database read request, make the request to Database Module, save the read data and send it to the module which makes the request. Then notify that module. If there is a Database write request, read the data that will be written, make the request to Database Module, send the data that will be written. Notify the module that made the request. The other functionalities that provided by this module is the following:



getSensorList(): This is a read request. When this request is comes from the other modules. Then this module send a Database read request for sensor list. Database module returns the sensor list to Simulation Core Module.

getSensor(string): This is a read request. When this request is comes from the other modules. Then this module send a Database read request for specified sensor. Database module returns the specified sensor to Simulation Core Module.

addSensor(Sensor): This is a write request. When this request is comes from the other modules. Then this module send a Database write request for adding the specified sensor information to the sensor list. Database module returns successful operation notification to the Simulation Core Module when the writing process finishes.

deleteSensor(string): This is a delete request. When this request comes, it invokes delete request of database module to delete the intended sensor defined as its ID in a string given as the argument.

setSensorMode(string, int): This is a write request. When this request is comes from the other modules. Then this module send a Database write request for changing the sensor mode of the specified sensor. Database module returns successful operation notification to the Simulation Core Module when the writing process finishes.

addAlarm(string, Alarm): This is a write request. When this request is comes from the other modules. Then this module send a Database write request for adding an alarm to alarm list with the specified date, time and position information. Database module returns successful operation notification to the Simulation Core Module when the writing process finishes.

setSensorParameters(string, SensorParameters): This is a write request. When this request is comes from the other modules. Then this module send a Database write request for changing the sensor parameters of the specified sensor. Database module returns successful operation notification to the Simulation Core Module when the writing process finishes.

setSensorPosition(string, Position): This is a write request. When this request is comes from the other modules. Then this module send a Database write request for changing the position values of the specified sensor. Database module returns successful operation notification to the Simulation Core Module when the writing process finishes.

7.2.9. Interface / Exports

As a result of all of its duties this component makes it creates an interface called Simulation Core. This interface is provided to all of modules except Database Module. With the help of this interface all of the modules can make the changes on the database, like setting the alarm times, dates and positions, changing sensor settings, reading sensor settings, changing sensor positions etc.



7.3. Reporting Module

In this module detailed design of Reporting Module is provided.

7.3.1. Classification

Reporting Module is a module of the Intruder Detection System which interacts with the Simulation Core Module and Analysis Module.

7.3.2. Definition

This module is the place where reporting process is handled. Reporting module take the necessary data from the Analysis module and Database. When some data from Database is needed it makes communications with Simulation Core Component. It can provide reports on PDF or Doc format. Also it has some functions to organize the document that will be printed. The details of these functions will be described at Processing part.

7.3.3. Responsibilities

This module has responsibility to print a report in a specified document type. This document type will be determined by the user. The responsibilities of this module can be summarized as the following:

- Take type of document from the user.
- Request necessary information from the Analysis Module, and the Database via Simulation Core Component.
- Create a text and fill it with the information taken from Analysis Module and the Database.
- Create the document in the specified document type. Namely Pdf or Doc formats.

7.3.4. Constraints

This module is totally dependent to Analysis Module. Because the information that is present in the documents is directly taken from Analysis Module. For this reason, if there is a mistake at the data taken from Analysis Module, there is no way to handle it.

This module is also indirectly dependent to all the other modules that are present in the system. The reason of it is, this module uses the information from the database. This information is coming from all of the other modules of the system. So, the reliability of the data that is presented in documents is depends on the correctness of the data that is send from the other modules.



7.3.5. Compositions

The Report Module is composed of one sub component. The name of this sub component is Report sub component.

Report: This object has the work of holding the document to be printed and document type. Furthermore it has links to Analysis Module and Simulation Core Module. The names of the links are linktoCore and linktoAnalysis. Furthermore it has functions like createText(date,time), createDocFile(string), createPdfFile(string), setDocumentType(bool) and getDocumentType() functions. The details of these functions will be described at processing part.

7.3.6. Uses and Interactions

This module has communications with, Analysis Module and Simulation Core Module. It does not provide an interface to other modules.

It makes communications with Analysis module to take the analyzed data. With this it does not need to make mathematical calculations on the data. It presents this analyzed data on the documents.

Report Module also makes communications with the Simulation Core Module to reach to the Database. It can request Alarm Times, Positions, the times that the sensor settings changed etc.

With this kind of information it can put an action table on the report data. It can mark the times that alarms occur in the schedule that will be present in the document.

7.3.7. Resources

When creating Pdf or Doc documents we need some libraries to make this work easier. We will work with Java, so we needed java libraries for creating Pdf or Doc documents.

We will use iText pdf creation library for creating pdf files [7]. It is a library that allows the user to create and manipulate PDF documents. Furthermore it is free software, so reaching it is so easy. Further details of this library can be found at our 2nd reference.

For doc file creating there some libraries for Microsoft Word. We will use a free software tool which uses these libraries and creates word files so easily. The name of the tool is java2word [8]. It is free software and it simply creates a doc file from the java code. The details of this product can be found at our 3rd reference.

7.3.8. Processing

This module is responsible from creating report documents when the user requests. It simply takes the necessary data from Analysis Module and Simulation Core Module. It takes



document type information from the user. When the user requests the report of the system this module starts to work. It has a text member for writing the text. Then it converts this text item to either doc or pdf according to the document type information. The detailed information about the functions of Report object is provided below:

createText(date,time): This function is used to create a new Text item and initialize it with the given date and time items. When this function is called, the requests are made by the calls from the linktoAnalysis and linktoSimulationCore . When these data arrives the data is also written to the text item. Then, when creating doc or pdf files, it will be converted to doc or pdf.

createDocFile(string): This function is used to create a new doc file with the given name. It simply reads the current text item. It calls java2word function and creates a new doc file with the given name.

createPdfFile(string): This function is used to create a new pdf file with the given name. It simply reads the current text item. It uses iText library to create a new pdf file with the given name.

setDocumentType(bool): This function is used to change the current document type. When the value of this is true the document will be created will have a type pdf. When the value of this is false the document will be created will have a type doc. The default value of the documentType member is true. This function can also be called by the user to change the document type.

getDocumentType(): This function is used to read current document type information.

7.3.9. Interface / Exports

As a result of all the duties this component creates a pdf or doc document which is prepared to be printed immediately. By interface, it does not provide an interface to other modules.

7.4. Analysis Module

7.4.1. Classification

Analysis module is a module of the system.

7.4.2. Definition

This module is created to make some statistical calculations about the alarm times, sensors, detection for reporting the events. It has a connection to DBMS to get the values from the database and calculate the statistical values. It also has a connection with reporting module which commands it to get the statistical values about alarm timestamps and sensor properties.



Basically, the duty of this module is getting computing some values to give the results to the operator.

7.4.3. Responsibilities

We have mention about the responsibilities or duties of this module on definition part. However, we can give the more detailed information about this module. The general responsibilities of our module are:

- The number of sensors
- The numbers of detected intrusion
- Last alarm time
- The sensor id which has the most active sensor
- The sensor id which has the least active sensor
- The sensor id that gives an alarm last

7.4.4. Constraints

The constraints of the Analysis module are:

- When report module commands a statistical value from this module, it must reply the command in a very short time.
- It does not require any synchronization.
- It must always be on waiting state for requests not to miss them.

64

7.4.5. Composition

This module is composed of one sub module named Alarm. This Alarm sub module has also one sub structure which holds the position information of the alarm.

Alarm Sub-module:

This object has a private property in analysis module. Alarm sub module holds the time position and date of the alarm which was last detected. This object is set whenever the report module wants the statistical calculation value. When this command comes, the analysis module gives a request to take the alarm times from database by using the database module and finds the last alarm from the taken alarm from database.

Position:

This object holds the latitude and longitude values of the alarm position. We mean that the sensor position by saying the alarm position.



7.4.6. Uses and Interactions

This module directly works in collaboration with the reporting module. Basically, it has a connection with the database module for getting the information from database. Since, it is an intermediate module between database and the reporting module; it does not any interaction with any other module.

7.4.7. Resources

Due to the fact that this module is only for statistical calculation, it does not need any resource except math library of C++ programming language. It uses the square root, power and any other functions of this library only.

7.4.8. Processing

We want to give the processing details of this module by explaining all methods of it.

computeNumOfSensors(): it takes the sensor id from the database and counts it to find the number of sensors and stores the information to the numOfSensors class variable.

computeNumOfAlarms(): it takes the alarm information from the database and counts the alarms and stores the number of alarm to the numOfAlarm class variable.

computeMostActiveSensor(): it takes the sensors and alarm values of each sensor. It finds the sensor which gives the most number of alarms in a predefined time period and stores the mostActiveSensor class variable

*computeLeastActiveSensor():*it takes the sensors and alarm values of each sensor. It finds the sensor which gives the least number of alarms in a predefined time period and stores the leastActiveSensor class variable.

computeLastAlarm(): it takes the alarms from the database and finds the alarm whose time is last and stores the lastAlarm class variable.

computeLastActiveSensor(): it takes the sensor and its alarm whose time is the last one and stores the sensor information to the lastActiveSensor class variable.

computeMostActiveToLeast(): it sorts the sensors from the most active to least active with respect to their number of alarms of each.

computeMostRecentActiveToLeast(): it sorts the sensors from most recent active to recent least with respect to the last alarm time of each sensor.

This module includes also get version of all of these compute methods and they returns the class variables which have been calculated with this compute methods.



7.4.9. Interface / Exports

With the help of this module, reporting module does not need any method to make mathematical operations. As we have said earlier, this module is an interface between reporting and database modules.

7.5. Timeline Module

Timeline module is a module for displaying the alarm information of the intruder detection system.

7.5.1. Classification

This component is a module of our system. The output of this module is directly related with the graphical user interface of our system.

7.5.2. Definition

The timeline module is directly related with the output to the user. It uses the database records to mark the corresponding point on the timeline graph. This timeline graph gives the alarm timestamps and the total number of alarms that detected on corresponding sensor.

Mainly, this module does not require any connection to the database module; instead, it has a link to the simulation core module to get the required information from the database.

66

7.5.3. Responsibilities

The main responsibility of this module is marking the specified timestamp on the timeline graph. This accomplished by giving a request database management system to get the alarm information of selected sensor by using the simulation core module.

7.5.4. Constraints

When timeline screen is active, this module must take the new alarm dates of the selected sensor, synchronously, and mark the timestamp with this value.

7.5.5. Compositions

The timeline module includes one sub-component. This is the TimelineEntry.

TimelineEntry: This component includes the sensorID, alarmDate, alarmTime, and alarmPos variables. Timeline entry component is essential, because, with the help of this component, we can take all alarm information for only one specific sensor which is assigned to the



sensorID. Taken alarm day information is stored to the alarmDate variable, taken alarm time information is stored to the alarmTime and specific sensor identification is stored the sensorID.

7.5.6. Uses and Interactions

This module uses the simulation core module to get the required information from database. Therefore, we can say that this module has a communication with SCC.

7.5.7. Resources

This module only requires the graphical user interface libraries of Java programming language to draw the graph and point out the alarm timestamp on it.

7.5.8. Processing

The timeline module includes some methods to fulfill its duty. These methods are:

createTimeLine(date, time, date, time): This method create a timeline chart with the given date and time values. First two date and time variable is the starting time to draw the chart towards to the second date and time variable values. After setting the bounding values of timestamp, it takes the alarm information from the database by using the linkToCore pointer whose type is SCC, and stores the alarm values to the timeline class variable whose type is timelineEntry array.

getTimeline(): this methods returns the timeline class variable.

7.5.9. Interface / Exports

This module creates a visual representation of the alarm information for the sake of user-friendly interface. This module does not provide any interface to the other modules.

7.6. Communications Module

Communications module is the module which is responsible from data communications between the control center and the sensors.

7.6.1. Classification

Communications component is a module of the system.



7.6.2. Definition

The purpose of this module is to provide communication between the control center and the sensors.

The information to be sent comes to this module at the control center side lastly. This module packets the information to be sent and sends it to the sensor.

Also, the information received firstly comes to this module at the control center side. This module unpacks the received information and redirects it to the related component.

7.6.3. Responsibilities

There are two major types of responsibilities of this module: sending information and receiving information.

It sends new sensor mode or settings to the sensor if the user wishes to change them. Before sending them, it takes related data from the sensor managing module.

It unpacks and redirects the received data to be recorded in the database to the simulation core component. Namely, when it gets alarms from a sensor, gets mode or parameters of a sensor, or gets position of a sensor.

7.6.4. Constraints

For receiving operations, it is assumed that the data obtained from the XBee wireless communication unit is correct.

7.6.5. Compositions

It uses two sub components, namely, SlarmPacket and SensorParameters.

SensorParameters holds settings related to the sensor and parameters related to the algorithm embedded in the sensor. It is used when receiving or sending related to the sensor to packet the data.

AlarmPacket holds information related to the alarms obtained from the sensors. More specifically, it holds alarm list of the intruders detected by the sensor and ID number of that sensor.

7.6.6. Uses and Interactions

This module makes use of simulation core module to send the information to be written to the database.



This module is used by sensor managing module to get the information intended to be sent to a sensor.

7.6.7. Resources

This module needs XBee wireless communication unit to send information to a sensor or receive information from a sensor. Also, it needs GUI so that the user can invoke this module to get alarms.

7.6.8. Processing

The communications module has six methods to fulfill its duty:

changeSensorParameters(string, sensorParameters): The function sends new parameters to the sensor in sensorParameters packet. It finds the intended sensor with its ID number that is given by the first argument to the method as string. It returns true if the operation is successful, false otherwise.

changeSensorMode(string, int): The function sends new sensor mode to the sensor defined by an int. It finds the intended sensor with its ID number that is given by the first argument to the method as string. It returns true if the operation is successful, false otherwise.

getAlarms(string): It simply obtains alarms in the intended sensor and packets these alarms to an AlarmPacket object and returns that object. It finds the intended sensor via its ID number that is provided in a string as an argument.

getSensorMode(string): It just learns the mode of the intended sensor and returns it in an int. It finds the intended sensor via its ID number that is provided in a string as an argument.

getSensorParameters(string): It just learns the parameters of the intended sensor and returns it in a SensorParameters object. It finds the intended sensor via its ID number that is provided in a string as an argument.

getSensorPosition(string): It just learns the position of the intended sensor and returns it in a Position object. It finds the intended sensor via its ID number that is provided in a string as an argument.

7.6.9. Interface / Exports

This module establishes an interface between the other modules and XBee wireless communication unit. It provides interface to sensor managing unit. Also, via a GUI, the user will be able to get alarms from a sensor, learn mode of a sensor etc.



7.7. Sensor Managing Module

Sensor managing module is responsible from operations related to the sensors.

7.7.1. Classification

Sensor managing component is a module of the system.

7.7.2. Definition

The purpose of this module is to redirect the operations related to the sensor to the communication module. When the user wishes to do an operation related with the sensors, it first handled by this module. It sends the information required to be sent to the sensor to the communication module and invokes the simulation core module to make the required updates in the database.

7.7.3. Responsibilities

When the user wishes to change the sensor mode or parameters, it uses this module. This module then redirects the parameters to the communications module so that they will be sent to the sensor. If the change operation is successful, then it invokes the simulation core module to make the required updates in the database.

Moreover, when a new sensor is added to the system or a sensor is deleted from the system, it invokes the simulation core module to make the required updates in the database.

7.7.4. Constraints

It is assumed that the acknowledgment signal received by the communications module is correct so that when the user wishes to change mode or parameters of a sensor it does not updates the database in a mistaken way.

7.7.5. Compositions

The module does not have any sub components.

7.7.6. Uses and Interactions

This module makes use of simulation core module to update the information in the database related to a sensor whose mode/parameters is/are changed successfully. Moreover, this module makes use of the communication module to send information to a sensor.



7.7.7. Resources

This module needs the GUI from where the user will be able to define the new sensor mode, parameters or add/delete a sensor.

7.7.8. Processing

The sensor managing module has four methods to fulfill its duty:

changeSensorParameters(string, sensorParameters): This method is used to change parameters of a sensor. The first argument is the ID number of the intended sensor and the second argument is the parameter packet of the sensor. It calls *changeSensorParameters()* method of the communication module.

changeSensorMode(string, int): This method is used to change mode of a sensor. The first argument is ID number the ID number of the intended sensor and the second argument is the new mode of the sensor. It calls *changeSensorMode()* method of the communication module.

addNewSensor(Sensor): The method is used to register a new sensor to the system defined by the Sensor object given as an argument. It calls *addSensor()* method of the simulation core module.

deleteSensor(string): The method is used to delete a sensor from the system defined by the ID number given as the argument in a string. It calls *deleteSensor()* method of the simulation core module.

71

7.7.9. Interface / Exports

The module provides service to users so that they will be able to define new parameters, mode for a sensor, add or delete a sensor. It is reached by the user via a GUI. It provides no interfaces to any other modules. It makes use of the communications module and the simulation core module.

8. Libraries and Tools

The libraries and tools that will be used while developing the intruder detection system are given in this section. For each library and tool, we first give its description and then explain its usage in developing the system.

8.1. Eclipse RCP



8.1.1. Description

Eclipse RCP is used for plugin development. It is a subset of Eclipse Platform. It is created as plugin project in Eclipse IDE. Eclipse RCP uses the components shown in Figure (35). The WorkbenchPage contains Parts, which can be Views or Editor. Views extend the abstract class “ViewPart “. They are used to display information in an RCP application.

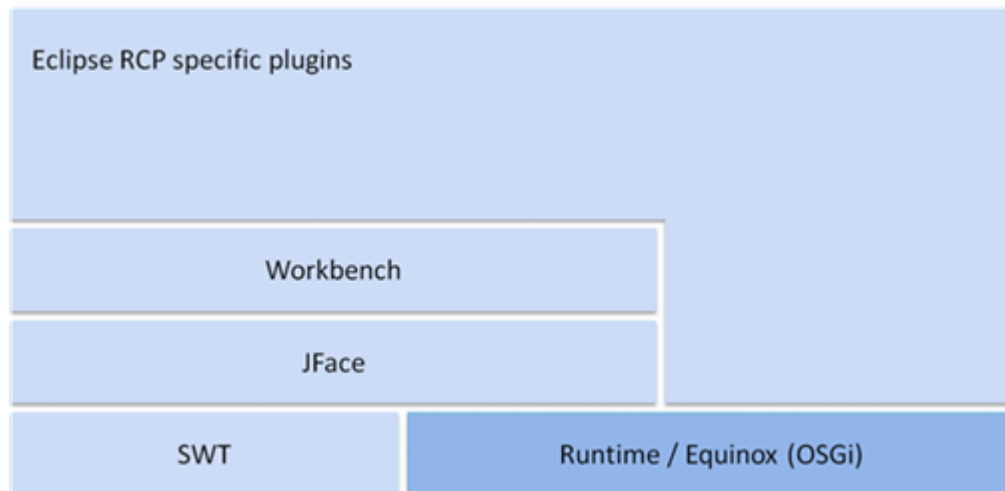


Figure 35. Components of Eclipse RCP

72

8.1.2. Usage in the Intruder Detection System

We use Eclipse RCP to develop desktop application. In addition, we will use Views in eclipse RCP to create screen displays.

8.2. Nasa World Wind Java SDK

8.2.1. Description

World Wind is open-source software, developed by NASA that allows you to zoom from satellite altitude into any place on earth. Leveraging Landsat satellite imagery and SRTM data, World Wind lets you experience any part of earth's terrain in visually rich 3D form, just as if you were really there.

8.2.2. Usage in the Intruder Detection System

We will use Nasa World Wind Java SDK to use World Wind technology in our application. We will draw our maps on GUI using World Wind technology.



8.3. JOGL

8.3.1. Description

JOGL is a Java API which provides bindings to the OpenGL libraries for the Java Virtual Machine. This allows computer graphics programmers to use the object-oriented tools for Java with hardware-accelerated 2D and 3D graphics while leveraging their existing knowledge of OpenGL [9].

8.3.2. Usage in the Intruder Detection System

We will need it to draw 2D graphics and 3D map in our system.

8.4. JFreeChart

8.4.1. Description

JFreeChart is a free 100% Java chart library that makes it easy for developers to display professional quality charts in their applications [10]. JFreeChart's extensive feature set includes:

- a consistent and well-documented API, supporting a wide range of chart types;
- a flexible design that is easy to extend, and targets both server-side and client-side applications;
- support for many output types, including Swing components, image files (including PNG and JPEG), and vector graphics file formats (including PDF, EPS and SVG);

8.4.2. Usage in the Intruder Detection System

We will use it to draw our timeline diagrams in the Report screen.

8.5. Waspnote and Waspnote IDE

8.5.1. Description

Waspnote is an integrated board developed by Libelium Company on which sensors can be integrated. Its usage is extremely wide, it can be used in agriculture, health, industrial processes, marketing etc.



Wasmote IDE is the environment to write the code which will be embedded into the Wasmote. It is developed with Java ANTLR Library. Although it has its own syntax and libraries, its usage is quite similar to C programming language.

8.5.2. Usage in the Intruder Detection System

We are going to use Wasmote integrated board in our design along with a GPS. The GPS is going to tell us the position of the sensor in terms of latitude, longitude and altitude. The main algorithm (The algorithm which determines whether a signal is due to an intruder or not) is going to be embedded in the Wasmote.

8.6. Eclipse IDE

8.6.1. Description

The Eclipse IDE provides libraries and tools for Java developers to build Java applications. It provides validation, incremental compilation, cross-referencing, code assist etc. It also provides debugging tools.

8.6.2. Usage in the Intruder Detection System

The control center part of the intruder detection system will be developed with Java programming language. Thus, we will use Eclipse IDE which is considered by many developers to be one of the best Java development environments.

8.7. PostgreSQL

8.7.1. Description

PostgreSQL is a powerful, open source object-relational database system. It has been developed actively for 15 years. It runs on all major operating systems including Linux, UNIX, and Windows.

It has full support for foreign keys, joins, views, triggers and stored procedures (in multiple languages). It includes most SQL: 2008 data types. It supports storage of binary large objects, including pictures, sounds or videos. It has native programming interfaces for C/C++, Java, .Net etc.

8.7.2. Usage in the Intruder Detection System

We are going to use PostgreSQL while creating the database of the system. We are going to take advantage of it by using its programming interface with Java.



8.8. XBee

8.8.1. Description

XBee is a radio module is designed for point-to-point and point-to-multipoint communications by Digi International Company. It is mainly used for wireless communications. An image of XBee Pro is provided in figure (36).



Figure 36. XBee Pro

75

8.8.2. Usage in the Intruder Detection System

The sensor for intruder detection will send the alarms to the control center via wireless communication. Because of its practical and simple usage, XBee will be used for wireless communication infrastructure. It will be integrated with Waspote board.

8.9. SD Card

8.9.1. Description

SD is a non-volatile memory card format developed for use in portable devices. As its practicality and small dimensions, it is being used for data transfer between devices like flash disks.

8.9.2. Usage in the Intruder Detection System

Besides sending the alarm to the control center, they will also be recorded in the sensor. The alarms will be saved in SD card which is also going to be integrated with Waspote board.



The user will be able to read the alarms saved in the SD card at a later time. In other words, it will be used as the main storage device by the sensor itself.

In the following figures, we present two images of the sensor (figures (37) and (38)). Please note that, the hardware consists of Wasmote with GPS, XBee and an SD card.



Figure 37. The sensor (not hidden)

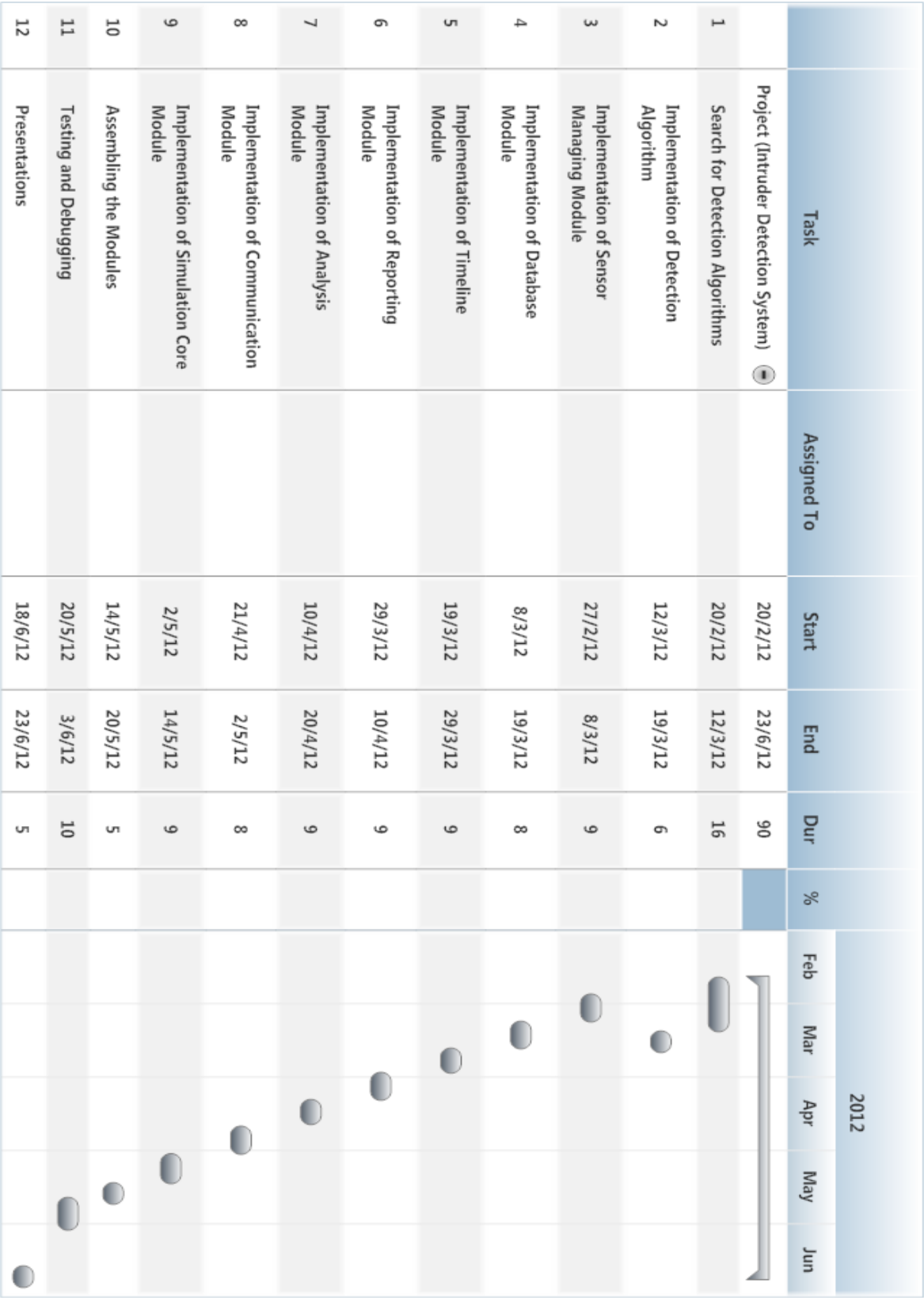
9. Time Planning

9.1. Term 1 Gantt Chart

	Task	Assigned To	Start	End	Dur	%	2011				2012
							Oct	Nov	Dec	Jan	
	Project (Intruder Detection System)		1/10/11	25/1/12	83						
1	Topic Selection		1/10/11	10/10/11	6						
2	Literature Search		10/10/11	17/10/11	6						
3	Marketing Search		10/10/11	17/10/11	6						
4	Project Proposal		17/10/11	31/10/11	11						
5	Software Requirements Spec.		17/11/11	25/11/11	7						
6	Testing with Sensor Equipment		25/11/11	20/12/11	18						
7	Initial Design Report		8/12/11	20/12/11	9						
8	Designing User Interfaces		17/12/11	9/1/12	16						
9	Detailed Design Report		20/12/11	9/1/12	15						
10	Creating the Database		17/12/11	9/1/12	16						
11	Prototype Demo		21/1/12	25/1/12	3						



9.2. Term 2 Gantt Chart



10. Conclusion

This document specifies the design details of the Intruder Detection System taken by the team BeeTech. More specifically, in this document, design details such as data design of the system, system architectural design, user interface design and detailed design of the modules are provided.

An experienced programmer can build the system by following the directives and architecture described in this document.

