Security System with Microsoft Kinect

Detailed Design Report

-Erdinç Kaya

-İlhan Yoldaş Karabulut

-Alişan Yılmaz

-Utku Şahin

Sponsored by Simsoft

2011

Table of Contents

1. Introduction 6
1.1. Problem Definition: 6
1.2. Purpose 7
1.3. Scope 8
1.4. Overview 8
1.5 Definitions and Abbreviations 9
1.6 References 9
2. System Overview 10
3. Design Considerations 11
3.1. Design Assumptions, Dependencies and Constraints 11
3.1.1 Assumptions and Dependencies 11
3.1.2 Design Constraints 12
3.1.2.1 Time Constraints 12
3.1.2.2 Performance Constraints 13
3.1.2.3 Experience of Team Members 13
3.1.2.4. Financial Constraints 13
3.1.2.5 Device Dependent Constraints. 14
3.2. Design Goals and Guidelines 14
3.2.1 Reliability 14
3.2.2 Usability 15
3.2.3. Portability 15
3.2.4. Extensibility 15
4. Data Design 16
4.1 Data Description 17
4.1.1 Administrator Table 17
4.1.2 Movement Table 18
4.1.3 Tree Table 19

4.1.4 Log Table 20
4.1.5 Notification Table 21
4.2 Data Dictionary 21
4.2.1 Entities 21
4.2.1.1 Admin 21
4.2.1.2 Movement 22
4.2.1.3 Tree 23
4.2.1.4 Log 23
4.2.1.5 Notification 23
5. System Architecture 27
5.1. Architectural Design 27
5.2. Description of components 28
5.2.1. Database Manager 29
5.2.1.1 Processing Narrative for Database Manager 29
5.2.1.2 DatabaseManager Interface Description 29
5.2.1.3. DatabaseManager Processing Detail 30
5.2.1.4. Dynamic behavior of DatabaseManager 30
5.2.2. InputManager 31
5.2.2.1 Processing Narrative for InputManager 31
5.2.1.2 InputManager Interface Description 31
5.2.2.3. InputManager Processing Detail 32
5.2.2.4. Dynamic Behavior of InputManager 32
5.2.3. ProcessManager 33
5.2.3.1 Processing Narrative for ProcessManager 33
5.2.3.2 ProcessManager Interface Description 33
5.2.3.3. ProcessManager Processing Detail 34
5.2.2.4. Dynamic Behavior of ProcessManager 34
5.2.4. OutputManager 35
5.2.4.1 Processing Narrative for OutputManager 35
5.2.4.2 OutputManager Interface Description 35
5.2.4.3. OutputManager Processing Detail 36
5.2.3.4. Dynamic Behavior of OutputManager 36

5.3. Design Rationale 37
5.4 Traceability of requirements 40
6. User Interface Design 40
6.1 Overview of User Interface 40
6.1.1 Login Page 41
6.1.2 Administrator Page 41
6.1.3 Owner Page 41
6.1.4 New Administrator Page 41
6.1.5 Movement Defining Page 42
6.2 Screen Images 43
6.3 Screen Objects and Actions 46
6.3.1 Login Page 46
6.3.2 Administrator Page 46
6.3.3 Owner Page 46
6.3.4 New Administrator Page 47
6.3.5 Movement Defining Page 47
7. Detailed Design 47
7.1 InputHandler 47
7.1.1 Classification 47
7.1.2 Definition 47
7.1.3 Responsibilities 48
7.1.4 Constraints 48
7.1.5 Composition 48
7.1.6 Interactions 49
7.1.7 Resources 49
7.1.8 Processing 50
7.1.9 Interface/Exports 50
7.2 ProcessManager 50
7.2.1 Classification 50
7.2.2 Definition 51
7.2.3 Responsibilities 51
7.2.4 Constraints 51

7.2.5 Composition 51
7.2.6 Uses/Interactions 52
7.2.7 Resources 52
7.2.8 Processing 54
7.2.9 Interface/Exports 54
7.3 Database Manager 55
7.3.1 Classification 55
7.3.2 Definition 55
7.3.3 Responsibilities 55
7.3.4 Constraints 55
7.3.5 Composition 56
7.3.6 Interactions 56
7.3.7 Resources 56
7.3.8 Processing 57
7.3.9 Exports 58
7.4 OutputHandler Component 59
7.4.1 Classification 59
7.4.2 Definition 59
7.4.3 Responsibilities 59
7.4.4 Constraints 59
7.4.5 Composition 59
7.4.6 Uses/Interactions 60
7.4.7 Resources 60
7.4.8 Processing 60
7.2.9 Interface/Exports 61
8. Libraries and Tools 62
8.1 Microsoft Kinect SDK 62
8.2 Microsoft Visual Studio 2010 62
8.3 HTR Files 63
8.4 MySQL 66
9. Time Planning 68
9.1 Term 1 Gannt Chart 68

9.2 Term 2 Gannt Chart 69
10. Conclusion 70

1. Introduction

This Detailed Design report is a complete description of project “Security System with Microsoft Kinect”. The project will be developed as a final design project by group 213 at Middle East Technical University Computer Engineering Department. The project is sponsored by Simsoft.

1.1. Problem Definition:

Nowadays, it is easy to see security cameras everywhere. But, are there anyone watching behind them, are they recording, are they even working? These are the first questions that come to mind. The main purpose of security cameras is to discourage the criminals. There is usually no person watching behind a security camera. Because employing a person for monitoring through a camera all the time is expensive for many businesses. If a crime happens security cameras can be used for an investigation. Moreover, these systems are not really useful because of human factors. In many cases, after a security issue has occurred, the videos that have been recorded have to be watched for hours to find the exact time and event. Even it is not guaranteed that the criminal will be found. Our team intends to solve this problem in every side. The security system with kinect is a 3D movement recognition system that will respond to the criminal activities immediately and inform authorities. The system is able to adapt according to place it is setup. Moreover, there is no need someone to spend all the time monitoring the cameras since the notification system handles the work itself. As a result, the system will be able to decrease the active monitoring time and increase the reliability of the camera system. It is going to decrease the time spent in front of the monitor by catching suspicious movements and with instant reporting system. It will also Increase the reliability because our security system can work without the active help of a human.

1.2. Purpose

This document specifies the requirements of security system with kinect project. The specifications to explain our goal and the path that group 213 have drawn for the project, which means what we aim for our system to look and work like. To illustrate more clearly, this document give information about interfaces, functionalities of the system, dependencies, expected results.

This document will give an illustration about the project we aim to build. The target groups are the 213 group, Simsoft members and departmental instructors. The main purpose is to decide the requirements of the project and to decide a common ground

between these supporters and developers. The specifications of the system will help to decide that whether the system meets the needs of the customers, producers and supporters. More importantly the document will help us to make changes for better, ordered and error free.

Another purpose is the document will help us to reduce time for new specifications or modifications since it is a complete summary of the system we can see from every angle without hesitation. We can find inconsistencies and disagreements faster for the sake of our success.

1.3. Scope

This document contains the detailed description of the project Security System with Kinect. All data structures and components are explained in the document. Moreover, the interfaces are shown and explained. It can be found our expected future works in the gannt chart in section 9 in figures 21,22.

1.4. Overview

This document is a detailed explanation of the Security system with Microsoft Kinect project. In section 3, design issues such as assumptions, dependencies and constraints, design goals and guidelines that are used is explained. In section 4, information about data design and data entities in the system is provided. Section 5 is about the system's overall architecture, the architectural design of the software applications and detailed description of modules are explicated. Section 6 provides design details on the user interfaces. Section 7 gives detailed description of the components of the system. Section 8 includes the libraries and tools that will be used

during software development. In section 9, time planning and scheduling issues will be demonstrated by a Gannt Chart in figures 21 and 22. The document is ended with a conclusion.

1.5 Definitions and Abbreviations

DDR: Detailed Design Report.

SDK: Software Development Kit.

MDRS : Movement Definition and Recognition System.

HTR : Hierarchical Translate and Rotation.

N/A : Not applicable.

1.6 References

Referenced documents for this DDR document is provided in the following list. Each referenced document is listed by title, report number or version (if applicable), date, and publishing organization information. Additionally, the sources, from which the references can be obtained, are provided.

References List

[1] IEEE, “IEEE Recommended Practice for Software Requirements Specifications”, EEE Std 830-1998(Revision of IEEE Std 830-1993), June 25 1998.

[2] Microsoft, “Microsoft Kinect Official Webpage”, http://www.xbox.com/en-US/kinect.

[3] Microsoft, Kinect SDK web page, http://research.microsoft.com/en-us/um/redmond/projects/kinectsdk/

[4] Veysi İşler, Ontology-based Classification of 3D Virtual Human Motion Data, 2009

[5] Department of Computer Science, university of Sheffield, Motion Capture File Formats Explained

2. System Overview

In this project, there are mainly 2 connected hardware components. Microsoft Kinect is the one that gathers the movements and sends them to the server computer located at the place that uses this product. The server computer Works as the brain of the system since all the data is processed in this part. All data is kept in the server machine in MySQL and HTR files and incoming data from the kinect is processed by using the both existing data and incoming data together. Detection and classification operations are implemented here. The results of the implementations are saved to the database and the system sends notifications when a suspicious movement detected.

The goal of this application is to combine the capabilities of Microsoft Kinect and software engineering, which is simply demonstrated in **figure 1**, to make the security monitoring systems more powerful. By determining and classifying the movements, false alarm rates will decrease. Also, this project will reduce the costs for security,

because the system is almost fully automated monitoring system and it needs less human resources.

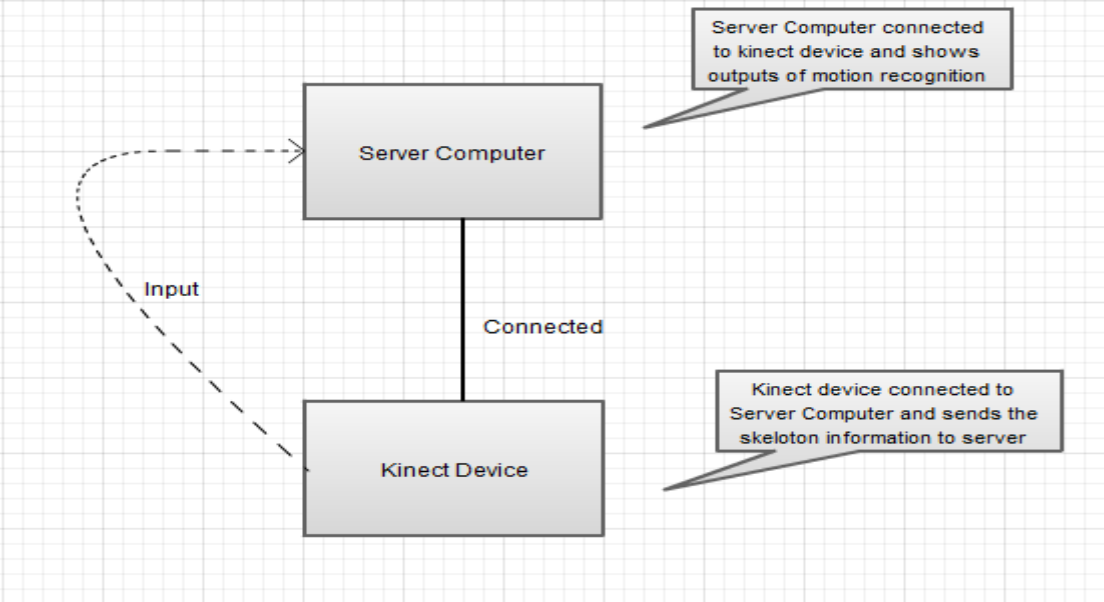

Figure 1: Overall Design

3. Design Considerations

In this section, assumptions, dependencies and constraints in the design of this system are explained. Design goals and guidelines are also clarified.

3.1. Design Assumptions, Dependencies and Constraints

3.1.1 Assumptions and Dependencies

While designing this project, we had to make some assumptions related to software, hardware and the environment. First of all, our program is intended to be run on Windows 7 OS. Since Microsoft Kinect SDK works only Windows 7 OS, that is why our program does not work on other operating systems. Another dependency is features

of the computer which runs the application. In order to make well qualified data, features of computer must be enough , approximately it must has two gigabyte ram, an intermediate level processor and also again intermediate level display graphic card. Related to dependencies, since the application can send e-mail, or short text message to related people's cell phones, the computer must have an internet connection.

Related to environment, since our aim is to detect, match and classify the movements, we have to obtain well qualified data; therefore Kinect SDK must work on good conditions, which are defined as environmental conditions. For example, lights distance etc. In order to work properly dosage of the sunlight must be low, and Kinect must be located properly, so it has to be located to the critical places, which encapsulate 3 x 3 x 3 meter area, because Kinect SDK can handle only 3 x 3 x 3 meter area. As we mentioned above target range must be in this area.

The assumed users and other people have no specific features. They are assumed to have no handicaps or genetically disordered.

3.1.2 Design Constraints

3.1.2.1 Time Constraints

The project started on September 2011, as a senior design project. The schedule of the project has been decided at the beginning of the fall semester of 2011-2012 academic year. A prototype has to be implemented at the end of fall semester of 2011-2012. The entire workload is divided into pieces for each member of the group 213 in order to be implemented efficiently, and wasting less time. After each workload part, the work will be handled by each member and will be combined and checked together by all of the group members. This project is intended to be finished at the end of May. The detailed schedule can be found in the Gantt charts in section 9 of the report. Each member will try to follow this schedule. If a member falls behind the schedule, the rest of the work for that member may be divided to other members. This

outcome of the project will not be bad, with this method the project will be able to finish at the right time.

3.1.2.2 Performance Constraints

Since detection, classifying, and matching the movements processes are very heavy, the project's algorithms must be efficient. In addition, this project is a real time project, the project is able to fetch, detect, and also classify the movement data, and also response to user in a proper time. Moreover in order to obtain high performance, environment and design dependencies must be obeyed. For example, the features of the computer have to be good enough to run this project, and also we have to locate Kinect's camera properly.

3.1.2.3 Experience of Team Members

Only İlhan Yoldaş Karabulut had experience with Kinect SDK. Rests of the members are studying on Kinect SDK. In addition to this, all of the group members are newbie to movement recognition system and processing movement files. In order to find the best algorithms for each step, we need to implement each of them by comparing the results. We are trying to find the best algorithm which also should be suitable for real-time applications. Any other tool used is a cross platform graphical user interface framework C#. The good news is all of the 213's members have a little experience on graphical user interface, and also C# framework.

3.1.2.4. Financial Constraints

Since the aim of the project is mainly to detect, classify and match movements in real time, we need to have qualified devices which have fast processors and not very

large but a little amount of memory. Moreover since the project will send short text message to authorized people's cell phone and e-mail, we have to buy short message server and mail server. Of course we need Kinect device and Kinect SDK, because Kinect is the spine of the project without Kinect and Kinect SDK the project will be useless. If we look at the top of the financial constraints, the project can be produced with a remarkable supply.

3.1.2.5 Device Dependent Constraints.

As we mentioned above, user is using a computer which runs the application. Therefore device dependent constraints are composed of a computer which runs the application and internet connection, and of course Kinect. Although user needs all of these devices, user can use only computer, in order to check out whether there is suspicious movement or not.

3.2. Design Goals and Guidelines

In this project, we try to obey the KISS (Keep it Simple Stupid) architecture. We try to avoid any complexities in our design, but while doing that, we make sure that every necessary part is included in it, and we documented all of these complexities. Our first concern is performance and speed, because detecting, classifying and matching movements must be processed in real time, and it needs to be fast and correct implementation. In order to obtain good performance and for the simplicity, we decided to use HTR files (Detailed information can be found about HTR files in section 8.3) to express movements more efficiently. In addition, through designing the project, we also take the following features into consideration:

3.2.1 Reliability

We aim that our project will run without any problems or bugs. There will be no consistency problems. Our project guarantees that response all data comes from Kinect SDK instantly. Moreover our program guarantees handling all problems comes from user. Since the system uses its own graphic user interface and it's so simple to use with only buttons, there will be no errors expected from the user side of the program.

3.2.2 Usability

Since this project intends to simplify the user's life, we designed the interfaces way too simple that can be used by any person without any education. The operations are very clear, only thing user have to do that is just sit and watch the program, if the user does not want to watch the program, they can feel free about this issue, because our application sends notifications to inform. Moreover the user may use the application anywhere needs security, with these program we can protect user's poverty, reputation and more important we can prevent life causalities.

3.2.3. Portability

This project can't be called portable at all, because one of the main constraints is place and distance. The system can be implemented to different places with new setups. But, it is not suggested since the movements are defined for specific places and there are some hardware dependencies comes with Kinect. It can be implemented with new setup of the system means the new movements should be implemented to the system. Adding new movements to the system is not a big deal even can be done from the owner of the system. But as mentioned above, the place and distance dependencies may cause some problem.

3.2.4. Extensibility

This project is designed to be extensible that means it's open to changes. Whenever a new operation is desired to be added, it can be implemented easily. For example user may want to add different movements. In such a condition we can add desired movements easily, because our application fetches movement samples from

database. If user want to extend the application, only thing we do is that adding desired movement to the database. Moreover, Kinect is a new technology which is still being developed. With these improvements we believe it will be able to make lots of changes with the hardware and software improvements.

,

4. Data Design

In this project, there are predefined interest objects such as administrators, movements, trees, logs and notifications. Admin, movement and tree objects are created before the monitoring system starts. Since movements and trees are needed to recognize and classify the captured movement, these objects are created in the movement defining part. Also, administrator objects are created before the monitoring system starts due to the fact that the system sends notification to the administrator when a suspicious movement detected. On the contrary, log and notification objects are created in the monitoring part of the system when a suspicious movement is detected. The class diagram of these objects is shown in Figure 2. Also, the ER diagram is shown in Figure 3.

Following sections give the detailed information about the mentioned data entities.

4.1 Data Description

In this project we will keep the most of information about the data objects in database management system, namely MySQL. We will have mainly five database tables which are Admin table, Movement table, Tree table, Log table and Notification table. According to the mode (either movement defining mode or monitoring mode) those database tables are reached, necessary queries are executed. Moreover, the detailed information about each movement is kept in separate HTR (hierarchical translate rotation) files, referenced by an attribute from the movement objects. Detailed information about HTR files can be found at section 8.3.

4.1.1 Administrator Table

Administrator table (Table 1) keeps the information about administrators. Table 1 includes 7 attributes namely, AdminID, Name, Surname, Username, Password, Email, CellPhone. Each admin has a unique AdminID and a unique Username.

Column Name	Data Type	Size	Range
AdminID	Integer	4 Bytes	
Name	String	2-20 characters	Alphanumeric characters
Surname	String	2-20 characters	Alphanumeric characters
Username	String	5-15 characters	Alphanumeric characters
Password	String	7-12 characters	Alphanumeric characters

Email	String	5-50 characters	Alphanumeric characters
Cell Phone	String	10 characters	Alphanumeric characters

Table 1: Administrator Table

4.1.2 Movement Table

Movement table keeps some information about movements and detailed information about a movement is kept in HTR files referenced in Table 2. This table includes 7 attributes, namely, MovementID, Name, Duration, FrameCount, MovementData, IsSuspicious and SecurityLevel. Each movement has a unique MovementID and a unique Name.

Column Name	Data Type	Size	Range

MovementID	Integer	4 Bytes	
Name	String	5-20 characters	Alphanumeric characters
Duration	Integer	4 Bytes	
FrameCount	Integer	4 Bytes	
MovementData	String	100 characters	
IsSuspicious	Boolean	1 Byte	True, False
SecurityLevel	Integer	4 Bytes	

Table 2: Movement Table

4.1.3 Tree Table

Tree table (Table 3) keeps the information about the tree structure of movements. Table 3 includes 2 attributes namely, ParentMovement, ChildMovement.

Column Name	Data Type	Size	Range
ParentMovement	Integer	4 Bytes	
ChildMovement	Integer	4 Bytes	

Table 3: Tree Table

Tree table keeps the information about the tree structure of movements. This table includes 2 attributes namely ParentMovement, ChildMovement and the records in this table are composed of the movements that are saved in the movement defining mode.

Using this table, a tree structure is created by means of movement type, i.e. movements are classified. While monitoring, that tree structure is used to make efficient comparisons. For example, “walking” can be considered as a parent of “running” and if a detected movement is not a “walking” type, it also won’t be a “running” type, so that path in the tree will be eliminated for comparison and that feature makes monitoring process more efficient and faster.

4.1.4 Log Table

Log table (Table 4) keeps the information about logs. Table 4 includes 5 attributes namely, LogID, AdminID, LoginDateTime, LogoutDateTime, NotificationCount. Each log has a unique LogID.

Column Name	Data Type	Size	Range
LogID	Integer	4 Bytes	
AdminID	Integer	4 Bytes	
LoginDateTime	DateTime	8 Bytes	
LogoutDateTime	DateTime	8 Bytes	
NotificationCount	Integer	4 Bytes	

Table 4: Log Table

4.1.5 Notification Table

Notification table (Table 5) keeps the information about notifications. This table includes 5 attributes namely, NotificationID, AdminID, MovementID, NotificationDateTime, RecordPath. Each notification has a unique NotificationID.

Column Name	Data Type	Size	Range
NotificationID	Integer	4 Bytes	
AdminID	Integer	4 Bytes	
MovementID	Integer	4 Bytes	
NotificationDateTime	DateTime	8 Bytes	
RecordPath	String	100 characters	

Table 5: Notification Table

4.2 Data Dictionary

4.2.1 Entities

4.2.1.1 Admin

As it is stated in 4.1.1, products have 7 attributes and below each of them will be explained in detail.

- **AdminID:** This attribute keeps the ID for each admin. Its type is integer and each admin has a unique ID.
- **Name:** This attribute keeps the name for each admin. Its type is string.
- **Surname:** This attribute keeps the surname for each admin. Its type is string.
- **Username:** This attribute keeps the username for each admin. Its type is string and each admin has a unique username.
- **Password:** This attribute keeps the password for each admin. Its type is string and each password is a 7-12 length secure word containing at least one number, one capital letter and one punctuation mark.
- **Email:** This attribute keeps the email address for each admin. Its type is string.
- **Cell Phone:** This attribute keeps the cell phone number for each admin. Its type is string.

4.2.1.2 Movement

As it is stated in 4.1.2, products have 7 attributes and below each of them will be explained in detail.

- **MovementID:** This attribute keeps the ID for each movement. Its type is integer and each movement has a unique ID.
- **Name:** This attribute keeps the name for each movement. Its type is string and each movement has a unique name.
- **Duration:** This attribute keeps the length of the recorded movement by means of seconds for each movement. Its type is integer.
- **FrameCount:** This attribute keeps the number of frames recorded in the specified duration for each movement. Its type is integer.
- **MovementData:** This attribute keeps the file path of HTR file for each movement. Its type is string.
- **IsSuspicious:** This attribute keeps the status that defines the movement is suspicious or not for each movement. Its type is boolean.

- **SecurityLevel:** This attribute keeps the importance value of the movement (low, medium, high) for each movement. Its type is integer.

4.2.1.3 Tree

As it is stated in 4.1.3, products have 2 attributes and below each of them will be explained in detail.

- **ParentMovement:** This attribute keeps the ID of the parent movement. Its type is integer.
- **ChildMovement:** This attribute keeps the ID of the child movement. Its type is integer.

4.2.1.4 Log

As it is stated in 4.1.4, products have 5 attributes and below each of them will be explained in detail.

- **LogID:** This attribute keeps the ID for each log. Its type is integer and each log has a unique ID.
- **AdminID:** This attribute keeps the ID of the admin for each log. Its type is integer.
- **LoginDateTime:** This attribute keeps the login date and time for each log. Its type is date.
- **LogoutDateTime:** This attribute keeps the logout date and time for each log. Its type is date.
- **NotificationCount:** This attribute keeps the notification count for each log. Its type is integer.

4.2.1.5 Notification

As it is stated in 4.1.5, products have 5 attributes and below each of them will be explained in detail.

- **NotificationID:** This attribute keeps the ID for each notification. Its type is integer and each notification has a unique ID.
- **AdminID:** This attribute keeps the ID of the admin for each notification. Its type is integer.
- **MovementID:** This attribute keeps the ID of the movement for each notification. Its type is integer.
- **NotificationDateTime:** This attribute keeps the notification date and time for each notification. Its type is date.
- **RecordPath:** This attribute keeps the path of the 5mins length movie that notification occurred for each notification. Its type is string.

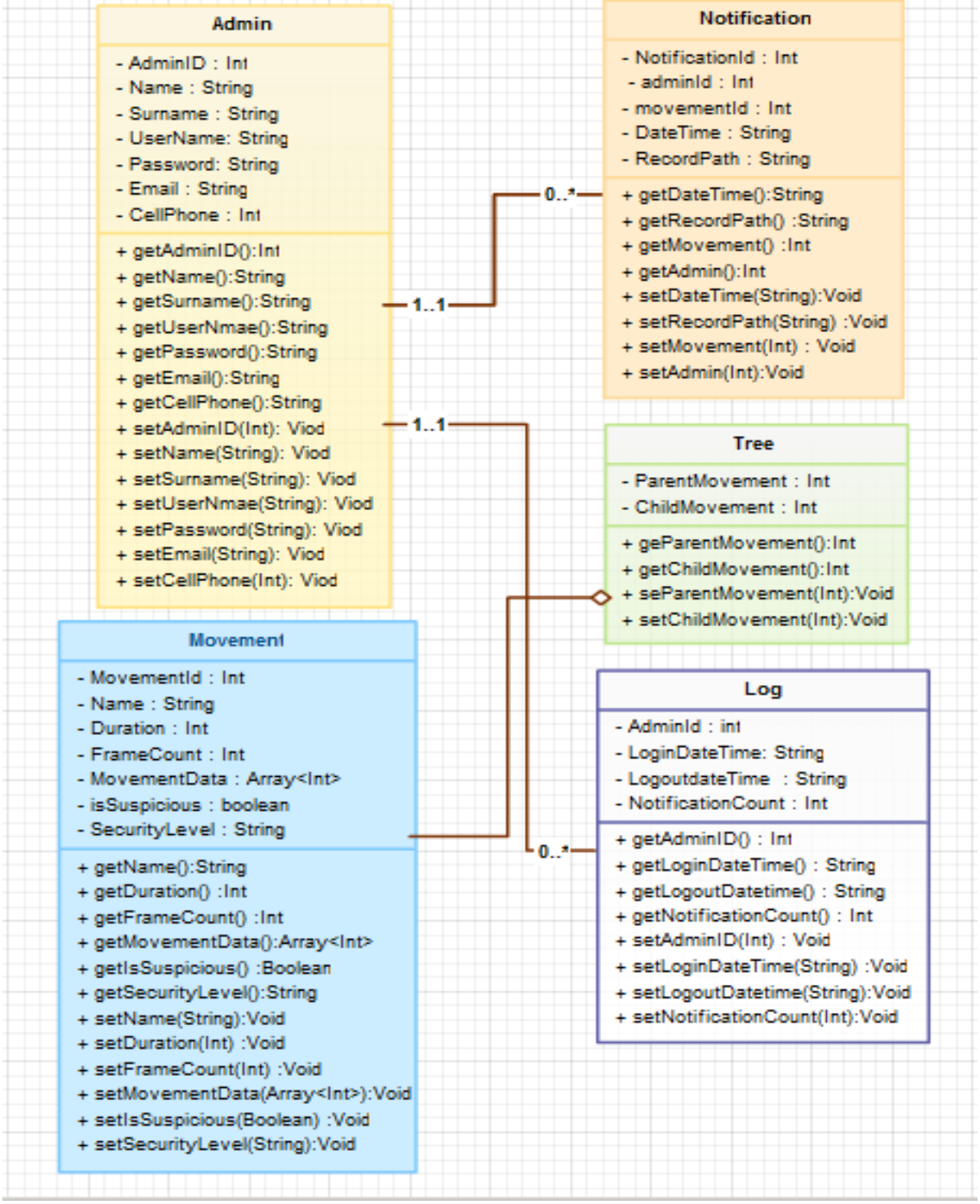

Figure 2: Class Diagram.

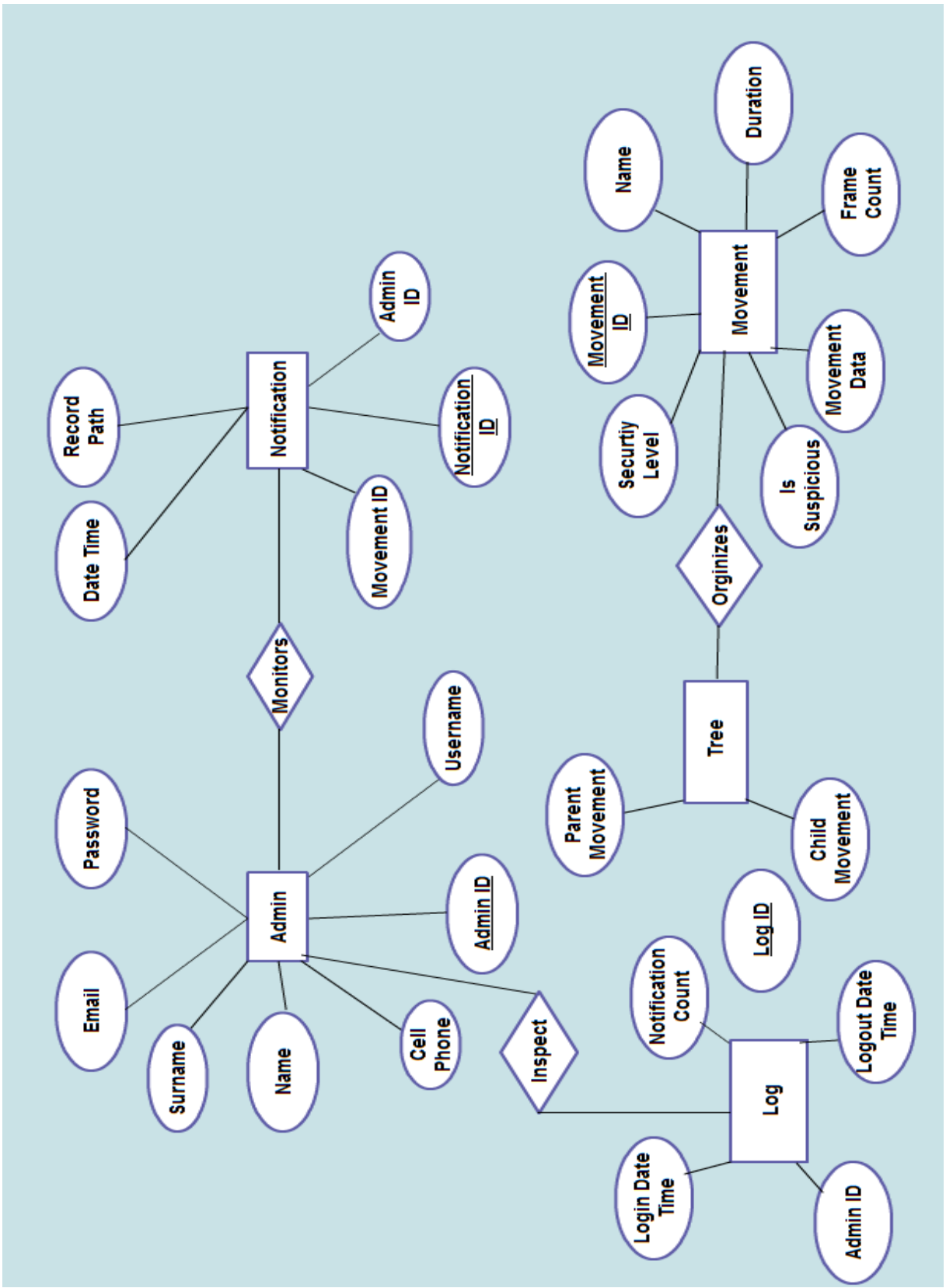

Figure 3: ER Diagram.

5. System Architecture

5.1. Architectural Design

The system consists 4 components named as DatabaseManager, InputManager, ProcessManager, OutputManager. Detailed information and sub-parts of these components will be explained in section 5.2 in detail. Generally, The DatabaseManager is used for interaction between database and ProcessManager. InputManager is responsible for getting input from both users and Kinect by using Kinect SDK. The information about current condition of the system will be given to user by using OutputManager. ProcessManager is most important part of the system which is used for organizing all other components and identifying movements. Relations of these components are shown in Figure 4.

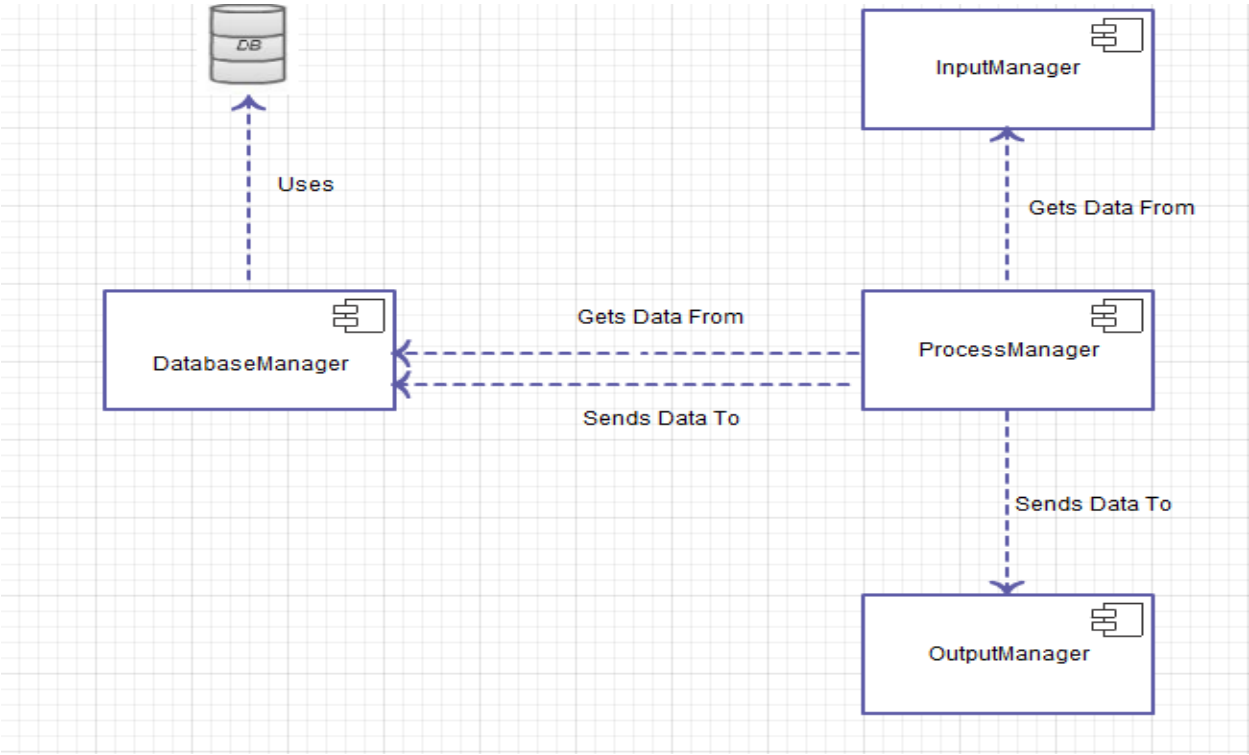

Figure 4: Component Diagram.

5.2. Description of components

In this section decomposition and interactions of the subsystems in the architectural design is explained in detail. The interaction between the components can be seen as a sequence diagram in figure 5. Use cases for the system can be seen in figure 6.

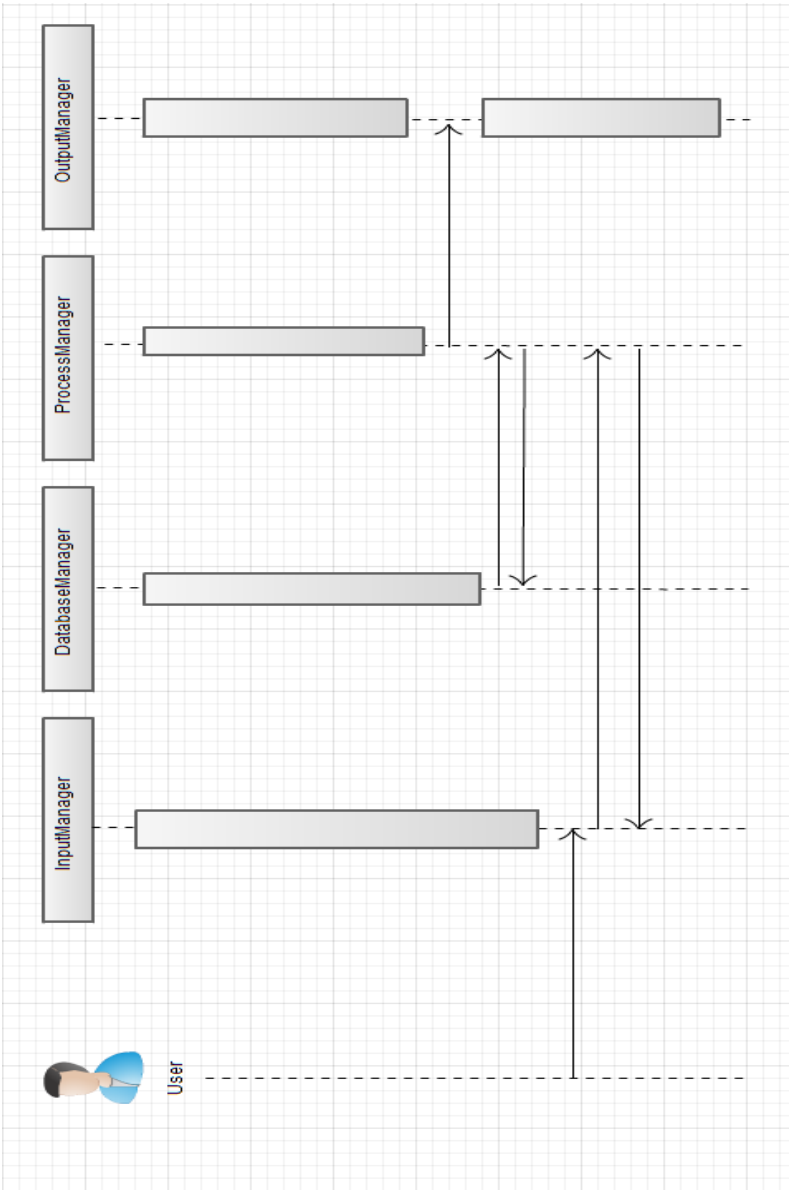

Figure 5: State Diagram.

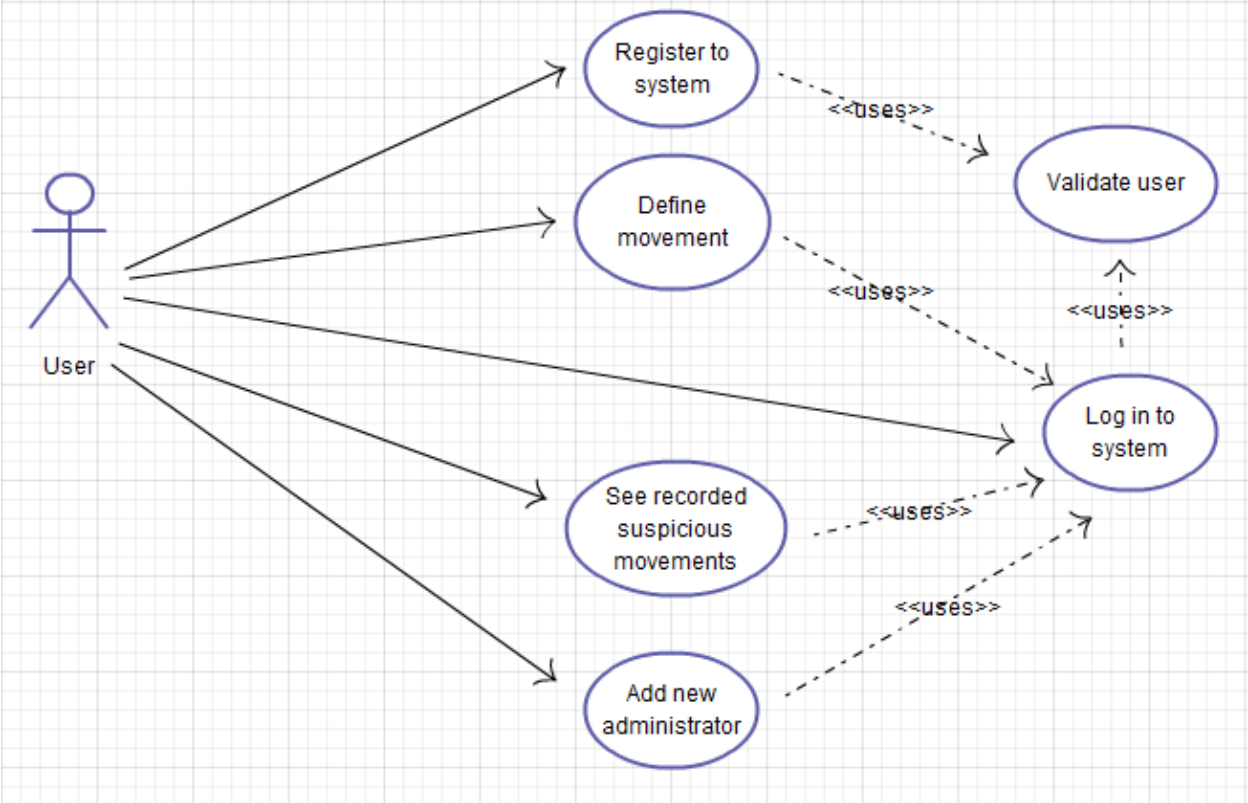

Figure 6: Use Case Diagram.

5.2.1. Database Manager

5.2.1.1 Processing Narrative for Database Manager

DatabaseManager is used for communication with database .Database is a sub-component of the DatabaseManager when a user wants to see administrator information, log information of the movements or previously recorded camera ,this component gets request from the ProcessManager ,turns this request to queries, make database execute these queries and sends the resulting data to ProcessManager.

5.2.1.2 DatabaseManager Interface Description

Although, It seems to be database is an input interface of the DatabaseManager, actually database is a sub-component of the databaseManager. ProcessManager sends

data to DatabaseManager in order to query to database, hence ProcessManager is input interface of DatabaseManager. Database manager sends the results of the queries to the ProcessManager, hence ProcessManager is output interface of the Database Manager.

5.2.1.3. DatabaseManager Processing Detail

- ProcessManager request for data.
- DatabaseManager uses the request from Process Manager as input.
- DatabaseManager queries input gathered from Process Manageer to database.
- DatabaseManager sends the resulting datasets to ProcessManager.

5.2.1.4. Dynamic behavior of DatabaseManager

DatabaseManager has only interaction with ProcessManager. The dynamic behavior can be explained as ProcessManager request data from DatabaseManager. After that DatabaseManager queries this request to database, sends resulting data to ProcessManager.

The DatabaseManager component is shown in Figure 7.

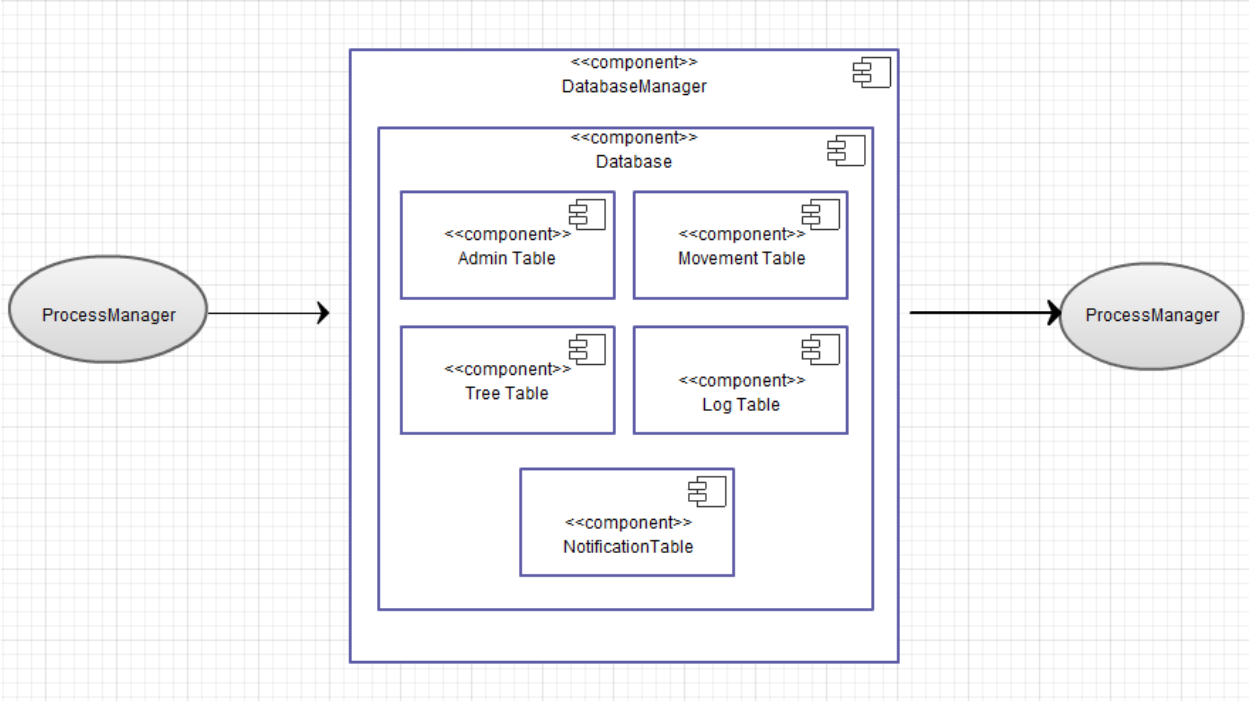

Figure 7: DatabaseManager Component

5.2.2. InputManager

5.2.2.1 Processing Narrative for InputManager

InputManager component is responsible for getting input from Kinect device and user. The user can interact with the system only using this component. Interaction with user is handled by UserInput sub-component of the system. The kinectInput sub-component manages the data coming from kinect.

5.2.1.2 InputManager Interface Description

Input interface of the InputManager is user interfaces of the system because the actions done by users via user interfaces are directly processed in InputManager. Kinect device is another input interface of the InputManager.

Output interface of the InputManager is MovementDefinition and MovementRecognition sub-components of ProcessManager. InputManager sends the data which is gathered from user and Kinect device to MovementDefinition and MovementRecognition components.

5.2.2.3. InputManager Processing Detail

When InputManager is in User interaction mode;

- User enters the inputs via user interfaces.
- InputManager gets user inputs.
- InputManager sends data to ProcessManager.

When InputManager is in motion detection mode;

- Data is gathered via Kinect device.
- Data is sent to MovementDefinition and MovementRecognition sub-components of ProcessManager.

5.2.2.4. Dynamic Behavior of InputManager

InputManager is in interaction only with ProcessManager. InputManager gets information from both user interfaces and Kinect device and sends this data to MovementDefinition and MovementRecognition sub-components of ProcessManager.

The sub-components are detailed in section 7.

The InputManager component diagram is shown in Figure 8.

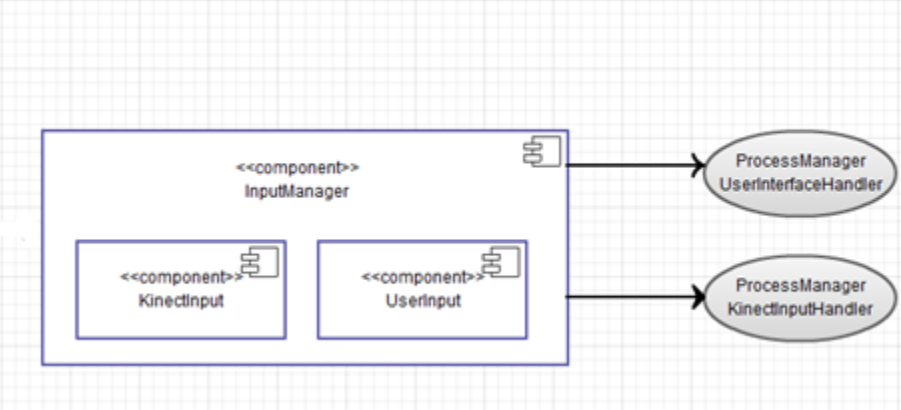

Figure 8: InputManager component diagram.

5.2.3. ProcessManager

5.2.3.1 Processing Narrative for ProcessManager

ProcessManager is responsible for getting input from InputManager and DatabaseManager and processing data to motion definition task. ProcessManager gets data from Kinect device by using kinectInput sub-component of the InputManager process this data by comparing previously stored data and sends the output to the output manager. ProcessManager gets user related data from UserInput sub-component of the Input handler and sends this data to DatabaseManager to store or query about some modification.

5.2.3.2 ProcessManager Interface Description

Input interface of ProcessManager is DatabaseManager, kinectInput sub-component of InputManager and UserInput sub-component of InputManager. Those components send data to MovementDefinition and MovementRecognition sub-components of ProcessManager.

Output interfaces of ProcessManager are DatabaseManager and OutputManager . DatabaseManager waits request for data from ProcessManager so this is output interfaces of ProcessManager. The OutputManager is main output interface of ProcessManager because the outputs of the processed data are sent to OutputHandler by ProcessManager.

5.2.3.3. ProcessManager Processing Detail

When ProcessManager is in motion definition mode;

- ProcessManager demands for data from KinectInput sub-component of InputManager.
- Process the kinect input to define a motion.
- Sends defined motion information to DatabaseManager.
- Sends related motion information to OutputManager.

When ProcessManager is in motion recognition mode;

- Gets kinect data as HTR from KinectInput sub-componet of InputManager.
- Compare this HTR file with previously recorded HTR files.
- According to similarity thresholds defines motion.
- Send Log information to DatabaseManager.
- Sends information about movement to OutputManager.

5.2.2.4. Dynamic Behavior of ProcessManager

ProcessManager is in interaction with all components of the system . ProcessManager gets data from DatabaseManager and InputManager , process this data and sends output to OutputManager. This dynamic behavior can be seen in diagram below.

The sub-components are detailed in section 7.

The ProcessManager component is shown in Figure 9.

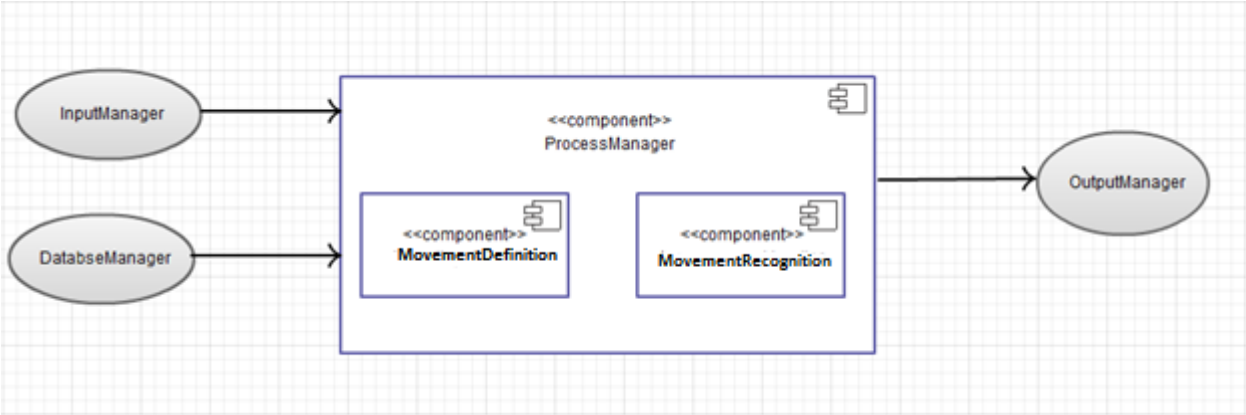

Figure 9: ProcessManager component.

5.2.4. OutputManager

5.2.4.1 Processing Narrative for OutputManager

The OutputManager is responsible for showing processed data in user interfaces. When the program is in Motion recognition state MotionRecognition sub-component of ProcessManager sends the output of motion detection to the OutputManager and when current state is motion definition, MotionDefinition outputs the data to OutputManager.

5.2.4.2 OutputManager Interface Description

The OutputManager has no output interface. MotionRecognition sub-component and MotionDefinition sub-component of the ProcessManager is the output interfaces of OutputManager.

5.2.4.3. OutputManager Processing Detail

When ProcessManager is in motion definition mode;

- MotionDefinition sub-component of ProcessManager sends data to OutputManager.
- OutputManager shows the related information via user interfaces.

When ProcessManager is in motion recognition mode;

- MotionRecognition sub-component of the ProcessManager sends data to OutputManager.
- OutputManager shows the motion information via user interfaces.

5.2.3.4. Dynamic Behavior of OutputManager

The OutputManager is only interacting with sub-components of the ProcessManager. Dynamic behavior is getting data from these components and displaying them in user interfaces.

The sub-components are detailed in section 7.

The OutputManager component diagram is shown in Figure 10.

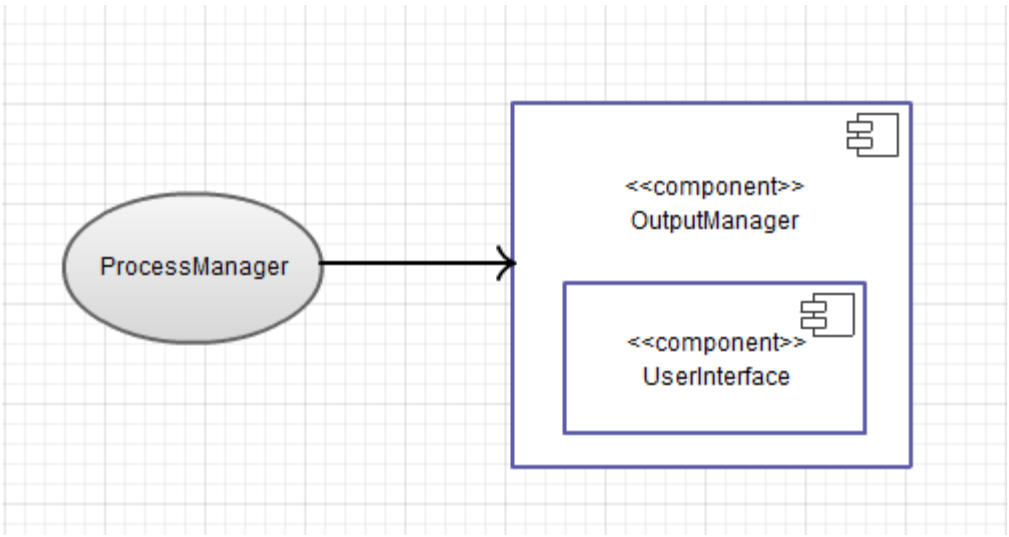

Figure 10: OutputManager component diagram.

5.3. Design Rationale

As it is stated before, our design is composed of 4 main components, namely DatabaseManager, InputManager, ProcessManager and OutputManager. Since we keep the information about data objects in database, we needed a component for it and we named it DatabaseHandler. Our data objects, whose detailed information is kept in the database tables, are Admin, Movement, Tree, Notification and Log, so we make them sub-components of database.

Gathering information operation is implemented by kinect device and user interfaces. The data gathered from these components is sent to ProcessManager , so we decided to have a component called InputHandler and make KinectInput and UserInput its sub-components.

The most important part of the system is processing data which comes from Kinect device, in order to deal with motion recognition and notification preparing tasks we thought that we should have a component and ProcessManager designed for this purpose. Other important task of the ProcessManager is organizing user interaction. Because ProcessManager have 2 important tasks to do we construct

MovementDefinition and MovementRecognition sub-components of the ProcessManager .

Finally, we decided to have an OutputManager component since the results coming from both parts of the Processor component need to be shown on user interfaces.

While defining sub-components of the system we try to make them related to their main components and for different purposes we try to define separate sub-components. For Example KinectInput and UserInput sub-components of InputManager are used for gathering data from real life to system while they gather information from different sources such as user and kinect device. Components and subcomponents are shown in Figure 11 in a hierarchical way.

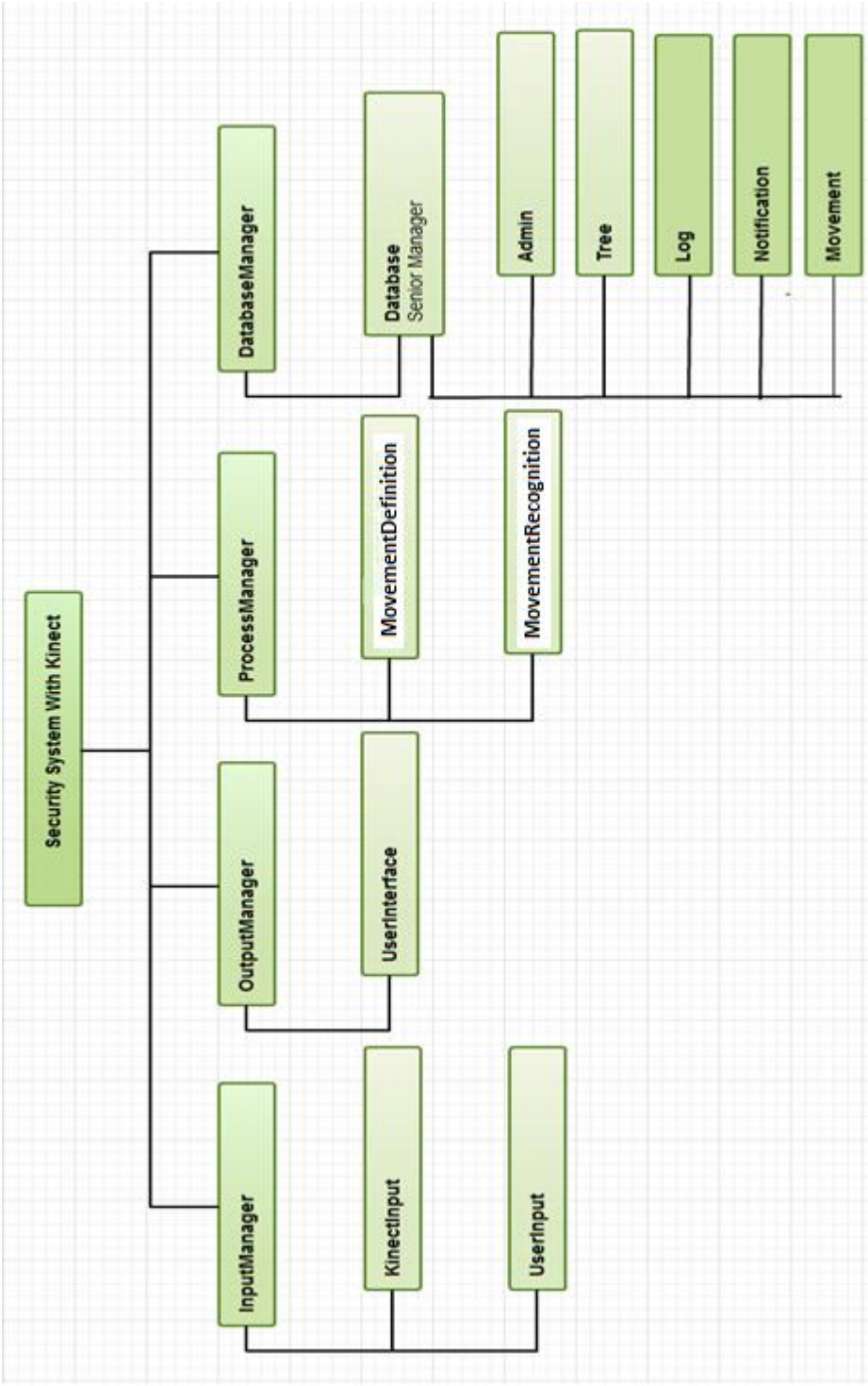

Figure 11: Aggregation Hierarchy Diagram.

5.4 Traceability of requirements

The use case for overall system can be seen in figure 6. In the table 6 component diagrams for every component are shown with in a table.

Components	Component Diagrams
InpunManager	InputManager component diagram (Figure 7)
DatabaseManager	DatabaseManager component diagram (figure 6)
ProcessManager	ProcessManager component diagram (figure 8)
OutputManager	OutputManager component diagram (figure 9)

Table 6: Component, diagram linkage.

6. User Interface Design

6.1 Overview of User Interface

There are 5 different pages in our user interface and 3 different users authorized to use them. The first user is the owner of the product. The owner's main duty is to add new administrators to the system. Moreover he/she has all the authorizations that administrators have. The second user is the administrator that has been assigned by the administrator. Administrators' main duty is to check the notifications. The notifications provided by the system show the time information of the incident and a 5 minute video of the incident. With the simple designed Administrator Page, it will be pretty easy for the administrator to check the incidents. The last user is the product provider. The provider is supposed to implement new movements according to the

place that our product is implemented. For ease of understanding, I will explain every page respectively.

6.1.1 Login Page

This is the very first page the user reaches after starting program. The users will be directed to the pages according to their usernames and for security, passwords. The page is shown in Figure 12.

6.1.2 Administrator Page

There are 4 main parts in this page. The first one is “live video streaming part”. As can be understood from the name of it, this part shows an instant video captured by Kinect. The second part is notification history. This part shows the incidents recently happened. The third part is the last notification part. Apart from the notification history the last notification is showed at another part. This part is made to administrators’ job easier. The last part shows the information of notification selected. The page is shown in Figure 13.

6.1.3 Owner Page

The only difference of the owner page from administrator page is a button that directs to new administrator page in order to add new administrators. The page is shown in Figure 14.

6.1.4 New Administrator Page

The only purpose of this page is to add new administrators only can be reachable by the owner. The page is shown in Figure 15.

6.1.5 Movement Defining Page

The purpose of this page is to implement new movements to the system. There is a skeletal view that kinect created. The name of the movement and security level of it will be defined at that part. Then the new movement will be added to movements list. The page is shown in Figure 16.

6.2 Screen Images

Figure 12: Login Page.

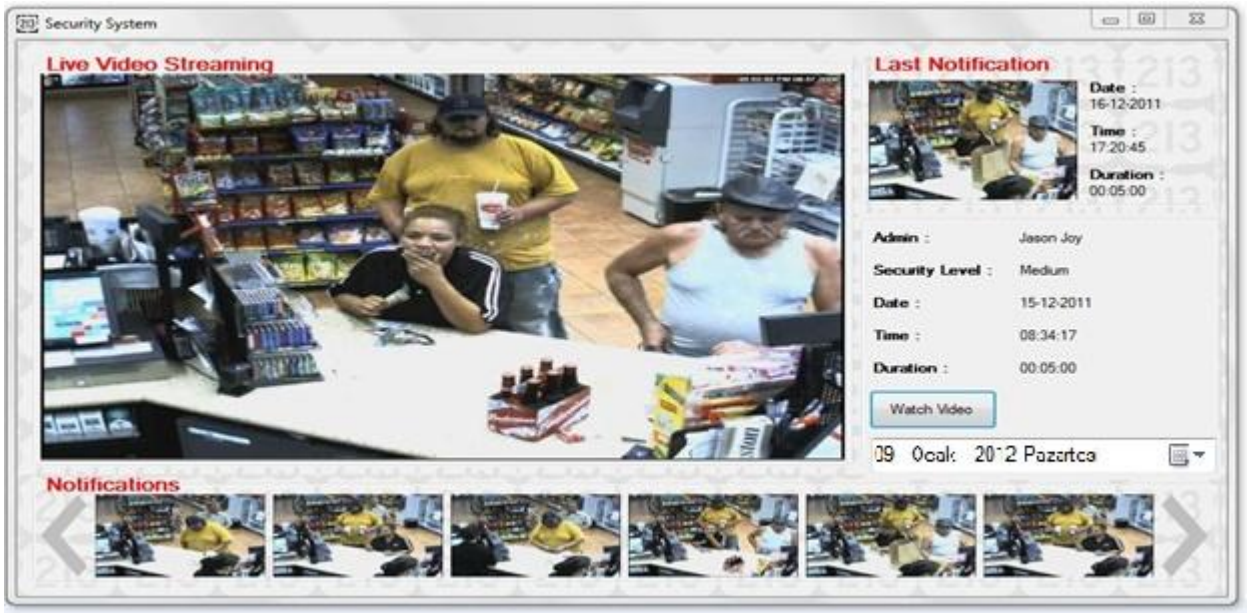

Figure 13 : Administrator Page.

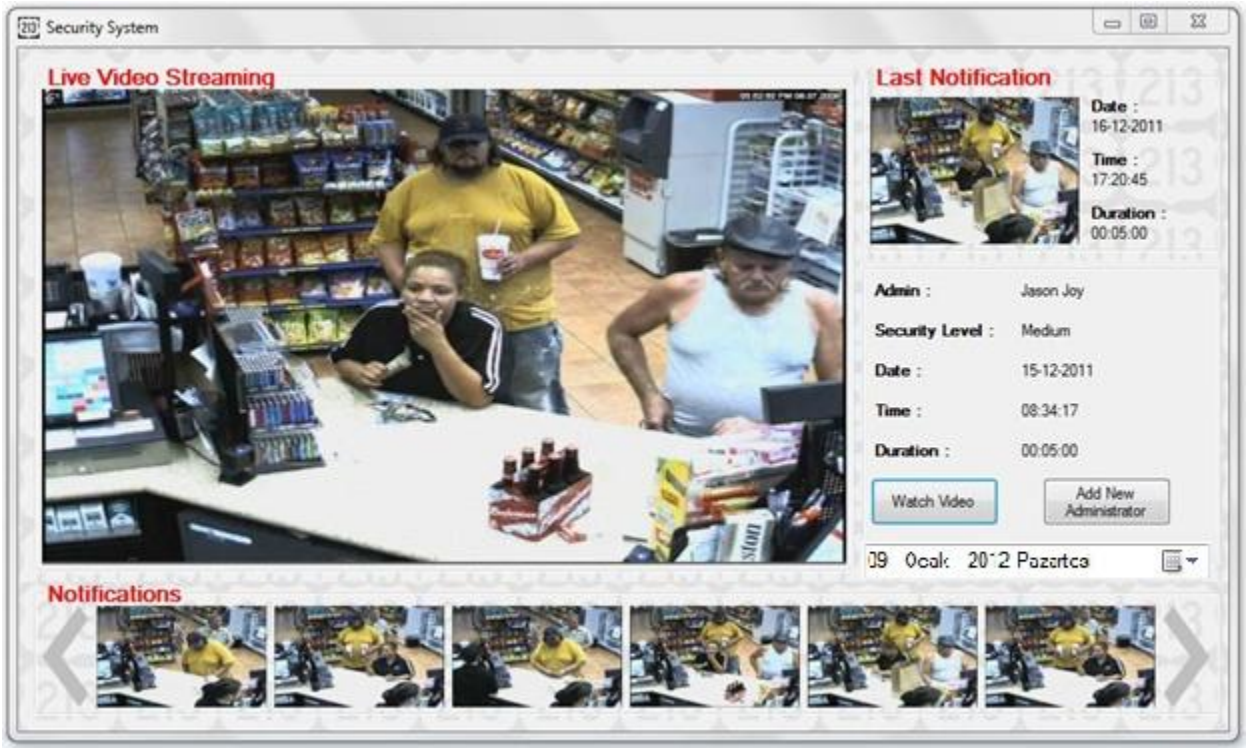

Figure 14: Owner Page.

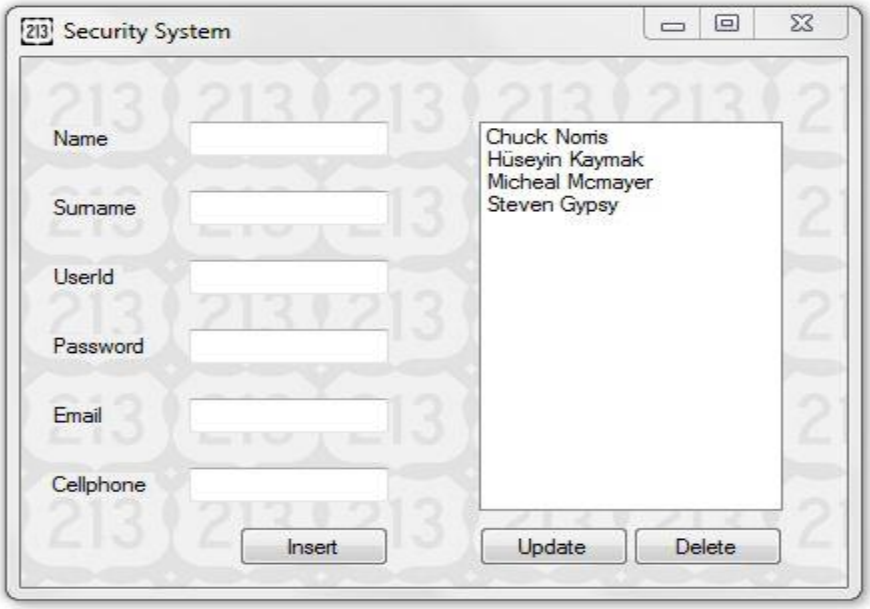

Figure 15: New Administrator Page.

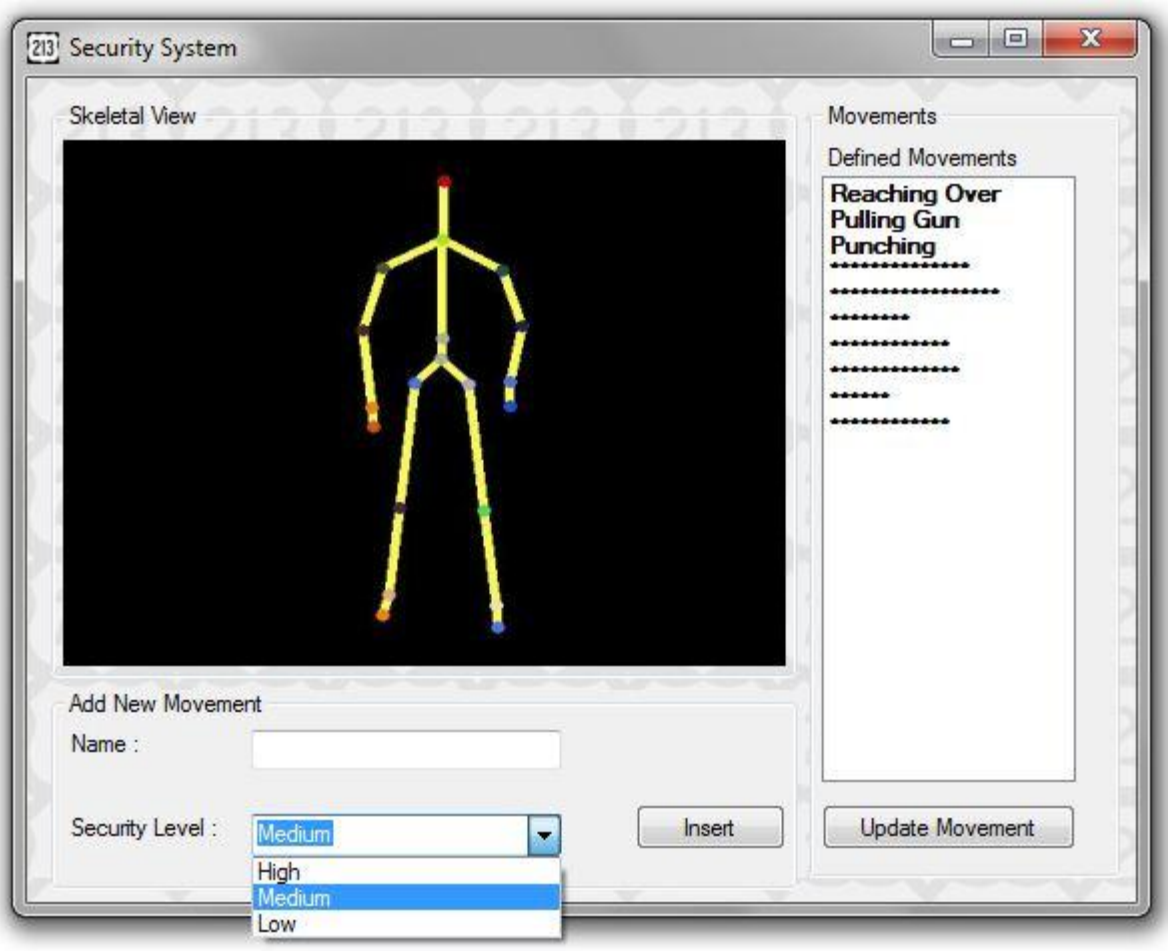

Figure 16: Movement Defining Page.

6.3 Screen Objects and Actions

6.3.1 Login Page

There are username and password sections that will direct 3 different users to different pages with a simple login button. Moreover, there will be a product logo which is not made yet.

6.3.2 Administrator Page

As mentioned before, the administrator page has 4 main parts. The live video steaming part is a stable part and won't change and will keep displaying the current video captured by Kinect. The notification history part has two functions. First one is the previous notifications can be found using the arrows in that part. The second one is if one click on the notifications, the information (Admin Name, Security Level, Date, Time and Duration) will be displayed at the right middle side of this page under last notification part. The last notification works similar as the notification history except date, time and duration information will be displayed all the time. The watch video button displays the video of the last notification selected with a pop up video editor.

6.3.3 Owner Page

The owner page is almost same as the administrator page. The only difference is "add new administrator" button. This button directs the owner to the new administrator page.

6.3.4 New Administrator Page

In this page the new administrators can be added to the data base. The name, surname, user Id, password, email and cell phone information will be taken. There is no optional information, every field should be filled. The insert button will add the new administrator to the database. All administrators will be displayed at the list box at this page. Clicking on the name of the administrator will display the information at the fields used to add new administrators. Using update button information can be changed. Delete button will simply erase the selected administrator.

6.3.5 Movement Defining Page

In this page there are only 2 data required name of the movement and the security level of it. After pressing the insert button the movement will be added to the database. All the movements will be presented at the list box at the right side of the page. Update Movement button is going to expand or decrease the ranges of the movements defined before by recording movement again. It is in order to optimize the movement saved to the database.

7. Detailed Design

7.1 InputHandler

7.1.1 Classification

InputHandler is a component with sub-components in it. It can be classified as a class.

7.1.2 Definition

InputHandler is a component which is designed for controlling the inputs of the system and creates a connection with the Processor component . It handles the inputs which are coming from the Kinect SDK or user. It has two components, namely, UserInputHandler and KinectInputHandler. InputHandler component has a relationship

only with Processor.

7.1.3 Responsibilities

The main role of the InputHandler component is getting data from the Kinect SDK and the user. Since this application is composed of two main parts, responsibilities of the sub-components of the InputHandler are changing respect to its duty. UserInputHandler always listens to user commands. When an user command comes it handles. Since user commands comes from user interfaces, UserInputHandler always keeps yourself alive. Another responsibility is handling Kinect inputs. This duty belongs to KinectInputHandler. When Kinect SDK dispatches an input, KinectInputHandler grabs and process it as needs to be.

7.1.4 Constraints

The InputHandler component has some constraints related to coordinates which comes from Kinect SDK or user interfaces. Since InputHandler component gathers the information from the Kinect SDK or user interfaces and sends them all to the Processor component. These inputs must be prepared inputs, that means processor component handles not all inputs only specific inputs so, they have to pass some tests to be processed. Another constraint is user constraint. This part is the most simplest constraint, because user constraints comes from user interface elements, therefore we do not need to prepare input. UserInputHandler just sends user input to the Process component.

7.1.5 Composition

As mentioned in the Definition part, InputHandler component has two sub-components which are KinectInputHandler and UserInputHandler. This two sub-components are designed for handling, preparing the inputs for Process component in

order to use more efficiently. KinectInputHandler is active all the time, it works only the Kinect SDK sends input which is composed of coordinates of the human body, otherwise it waits for the input. Furthermore Kinect SDK sends input when the frame changes, hence KinectInputHandler gets input for every frame, and KinectInputHandler process them all. When data is gathered from the user interface, user interface sub-component sends the data to the UserInputHandler sub-component of the processor. UserInputHandler is also active as long as application runs. UserInputHandler gets input when user gives command that means, user push any buttons of the user interface.

7.1.6 Interactions

InputHandler component has an interaction only with the Processor. When InputHandler gets input from the Kinect SDK or user interface, before sending these inputs to the Processor, in order to process more efficiently InputHandler prepares inputs. Since Processor is the unit which makes processes to these data in order to get results, there should be a component which is feeding the Processor and it is InputHandler.

7.1.7 Resources

InputHandler has two main resources. They are Kinect SDK which is the resource of the KinectInputHandler sub-component and the user interface of the application which is the resource of the UserInputHandler sub-component. Our application uses two input resources and they are the Kinect SDK and the user interface of the application. Since the InputHandler is the component which is dealing with the , the inputs coming from the Kinect SDK and the user interface of the application is handled by this component.

7.1.8 Processing

The InputHandler component handles the inputs gathered from the Kinect SDK and the user interface of the application. In the KinectInputHandler sub-component gathers data from the Kinect SDK in real-time. While doing this, KinectInputHandler sub-component uses Kinect SDK' s functions and libraries in order to connect the Kinect' s camera and get frames and coordinates of the human body which generated by Kinect SDK as input.

UserInputHandler sub-component is designed to be gather data from the user interface of the application. The inputs of the user interface is gathered from the keyboard or mouse. Since the user interface of the application is designed with C#, capture functions are also captured with C#'s classes.

7.1.9 Interface/Exports

The set of services:

- Gathers data from Kinect SDK
- Gathers data from user interface of the application
- Sends the data to the Processor component.

7.2 ProcessManager

7.2.1 Classification

ProccesManager is a component of the overall system. This component is used for defining and recognizing movements.

7.2.2 Definition

In the requirement analisis report it is stated that our system needs to define movements according to data which kinect device provides and should recognize movememts which are previously defined. ProcessManager is the main component of our system so this component sends the processed data to related components of the system.

7.2.3 Responsibilities

As stated in the definition part of this component, ProcessManeger is responsible from defining and recognizing human movements. In order to deal with this task this component should process kinect data at run time. This processing includes two main executions. First one is dividing kinect data to appropriate portions by means of frames that contains all of the data which defines a movement. The second one is comparing these portions with the previously defined movement information.

7.2.4 Constraints

Processmaneger has some constraints about timing since this component should divide the frames of the kinect data in portions and compare these portions with previously defined movements at the same time. Applying comparing algorithms to movements is another constraint since this comparison must be accurate and fast.

7.2.5 Composition

ProcessManager consist of two sub components, these components are defined in MovementDefinition class and MovementRecognition class. MovementDefinition class is responsible from defining movements and turning the kinect data to form that can be saved in the HTR file format. The properties and purpose of the HTR files explained in Libraries and tools part of this document in detail. MovementRecogniton

component is doing main task recognizing movements. This component divides the kinect data frames to portions and compares these portions with previously defined movements.

7.2.6 Uses/Interactions

Since the ProcessManager is the main component of the system it has interactions with all of the other components of the system. It uses Inputmanager as the input source, gets both kinect data and user inputs by using this component. DatabaseManager connects the ProcessManager to database. From the DatabaseManager ProcessManager gets previously defined motion information and user information which registered to system. ProcessManager uses OutputManager in order to inform users about the recognized motion and security level of the motion.

7.2.7 Resources

Resources of the ProcessManager is memory, CPU and disk . ProcessManager reads HTR files from the disk and stores them in the memory. All processes use these HTR files by reading from memory in the execution of the program. ProcessManager uses CPU for every execution of the system. A sample HTR file illustrated in figure 17.

```
#Comment line ignore any data following # character
#Hierarchical Translation and Rotation (.htr) file
[Header]       #Header keywords are followed by a single value
FileType htr         #Single word string
DataType HTRS        #Translation followed by rotation and scale data
FileVersion 1        #integer
NumSegments 18       #integer
NumFrames 2               #integer
DataFrameRate 60          #integer, data frame rate in this file
EulerRotationOrder ZYX
CalibrationUnits mm
RotationUnits Degrees
GlobalAxisofGravity Y
BoneLengthAxis Y
ScaleFactor 1.000000
[SegmentNames&Hierarchy]
#CHILD   PARENT
Head     Neck
Neck     Chest
Chest    Hips
LeftShoulder  LeftCollar
RightShoulder RightCollar
LeftElbow     LeftShoulder
RightElbow    RightShoulder
LeftWrist     LeftElbow
RightWrist    RightElbow
Hips     GLOBAL
LeftHip  Hips
RightHipHips
LeftKneeLeftHip
RightKnee     RightHip
LeftAnkle     LeftKnee
RightAnkle    RightKnee
LeftCollar    Chest
RightCollar   Chest
[BasePosition]
#SegmentName Tx, Ty, Tz, Rx, Ry, Rz, BoneLength
Head     0.000003    141.966248  0.000002    -37.745777  -8.179454   6.203664    80.000046
Neck     0.000000    379.566772  0.000000    32.855431   -5.194619   1.823337    141.966263
Chest    0.000000    94.891693   0.000000    0.000002    12.877405   -0.000012   379.566803
LeftShoulder  0.000003    154.023666  0.000005    171.957962  -79.977570  -157.094910 215.219986
RightShoulder -0.000001   146.408707  -0.000001   20.020029   60.865143   -8.207462   240.139038
LeftElbow     0.000007    215.219971  -0.000011   -24.162848  54.442085   -17.491318  336.440125
RightElbow    -0.000021   240.139038  0.000007    -21.925606  -43.617706  23.102211   322.101532
LeftWrist     -0.000014   336.440094  0.000002    173.178864  -0.531422   156.628021  100.348160
RightWrist    0.000005    322.101501  0.000026    1.868945    1.747480    14.125610   109.739998
Hips     377.886597  1077.530640 704.474976  -179.011169 36.455761   -178.006714 94.891701
LeftHip  95.336761   5.238981    -8.336124   -0.485735   -9.415434   -176.117065 525.916443
RightHip-96.119095   -4.691877   -4.857316   2.457280    -1.423929   -179.125565 528.302429
LeftKnee 0.000004    525.916443  0.000001    -10.646533  -14.541370  -1.855900   514.711975
RightKnee     -0.000002   528.302429  0.000004    -12.768950  2.118825    -3.063609   521.764099
LeftAnkle     0.000003    514.711975  0.000009    97.965164   -0.000001   0.000000    253.446518
RightAnkle    0.000001    521.764099  -0.000002   99.242271   0.000002    0.000000    256.283173
LeftCollar    19.999998   366.483276  -15.126845  169.660507  -4.312743   90.901894   154.023682
RightCollar   -19.999996  366.483215  -15.126822  171.369568  4.171492    -91.645760  146.408707
#Beginning of Data. Separated by tabs
[Head]
#Fr      Tx     Ty     Tz     Rx     Ry     Rz     SF
1  -0.000004   11.246399   -0.000001   -8.720660   -3.164685   6.637906    1.000000
2  -0.000005   11.559982   -0.000003   -8.967685   -3.106098   6.445398    0.999999
[Neck]
#Fr      Tx     Ty     Tz     Rx     Ry     Rz     SF
1  0.000000    -12.073059  0.000010    0.396232    -0.246686   -4.863737   1.079219
2  -0.000002   -12.097473  -0.000004   0.425361    -0.269309   -4.636292   1.081428
```

Figure 17: Sample HTR file.

7.2.8 Processing

Our system has two main processes these are movement definition and movement recognition. Since ProcessManager is the main part of the system the subcomponent of this component handles these processes.

In the movement definition mode after user enters the duration, name and security level of the movement by using input manager, MovementDefinition subcomponent starts getting kinect skeleton data from kinect by using InputManager. Although HTR files keeps movement information in hierarchal way, Kinect gives us information of all body positions independent of the other body positions. After getting the kinect skeleton data along the duration which user defines, MovementDefinition subcomponent turns this data to HTR file format.

In the movement definition mode all the movement information which previously stored in HTR file format are read from the disk to memory. In this mode InputManager sends the Kinect data to MovementDefinition subcomponent continuously. While getting Kinect input MovementDefiniiton subcomponent devides this kinect data to intervals of frames which contains the movements. After that the portions of the frames are compared with defined movement by comparing the fields of the HTR files. If a portion matches with defined movements OutputManager component is informed about this recognition. Portioning and recognizing processes are done simultaneously along the program execution.

7.2.9 Interface/Exports

ProcessManeger is gathers information from InputManger . Processes this data by using HTR files. It sends the recognition information to OutputManger. You can find all the information about components which used as service by ProcessManager from section 5. System Architecture. Detailed information is mentioned in section 8.3.

7.3 Database Manager

7.3.1 Classification

The DatabaseManager is a component of overall system.Database manager is a class that is used for executing database queries and providing process manager with datasets.

7.3.2 Definition

As it can be understood from the name, Database manager component is actually a database handler unit. DatabaseManager class is used to access database which holds all of the information about data objects of the project. This data objects are admin, movement, tree, log and notification.

7.3.3 Responsibilities

The responsibility of this component is handling the data transfer between the database and the program. It controls all database operations and provides ProcessManager with some functions that used for executing queries on the database.

7.3.4 Constraints

This component actually plays a major role since its job is provide data for the Process Manager, which is the component making the actual job. There are also contraints referring to database while creating the tables such as ‘not null’ properties. Since the job is querying, input and output values are limited with their own types.

7.3.5 Composition

As it is explained and illustrated in Section 5.2.1, this component has one major subcomponent, the database and the database consists of mainly five tables Administrator Table, Movement Table, Tree Table, Log Table and Notification Table. Database subcomponent is responsible for keeping the information about the objects and the relations between these objects. These tables are the minor, but the most essential sub-components for our project.

7.3.6 Interactions

This component is not reached by the user directly. It will work on background and will be used by ProcessManager. There is no other component that DatabaseManager component is interacted.

7.3.7 Resources

There are some resources that DatabaseManager component uses. To begin with, the database itself is a resource for this project. Moreover, in order to keep database, MySQL is used (Detailed informaton about MySQL can be found in Section 8.3.. In order to integrate MySQL with the project, mysql connector is used, and it is another resource that this component uses. Since the database is permanent, some non-volatile storage to keep the tables is necessary.

7.3.8 Processing

In processing, since this component is responsible from connecting to the database, and making queries. Connecting to the database will be achieved by means of MySQL libraries for C#. A code sample to connect to the database is shown in Figure 18:

```
string MyConString = "SERVER=" + Host + ";" +
                     "DATABASE=" + Database + ";" +
                     "UID=" + User + ";" +
                     "PASSWORD=" + Pass + ";";

MySqlConnection connection = new MySqlConnection(MyConString);
```

Figure 18 : Database connection.

Where

- “Host” is a host name, the location of the MySQL database server, for example “localhost”.
- “User” is the username
- “Pass” is the password.
- “Database” is the name of the MySQL database reached from host

After it successfully connected to the database, queries can be executed basically as shown in Figure 19;

```
MySqlCommand command = connection.CreateCommand();
MySqlDataReader Reader;
command.CommandText = "select * from mycustomers";
connection.Open();
Reader = command.ExecuteReader();

while (Reader.Read()){
        string thisrow = "";

        for (int i= 0;i<Reader.FieldCount;i++)
                thisrow+=Reader.GetValue(i).ToString() + ",";

        listBox1.Items.Add(thisrow);
}

connection.Close();
```

Figure 19: Query execution.

7.3.9 Exports

The set of services:

- Stores data objects
- Stores relationships between data objects
- Provides the connection between other components and the database
- Returns information on demand

7.4 OutputHandler Component

7.4.1 Classification

This is a component of the overall system. OutputManager used for showing the processed data to users.

7.4.2 Definition

OutputManager component is the component that handles outputs, namely the user interface and enriched video displays and notifications which are composed of video frames. It acts as a bridge between the processor and the graphical user interface. User can interact with the graphical user interface via OutputManger component.

7.4.3 Responsibilities

This component is actually responsible from delivering the output to graphical user interfaces and displaying the outputs. This component takes input from ProcessManager.

7.4.4 Constraints

Since this component simply shows the outputs of the system to users this component has no constraint.

7.4.5 Composition

OutputHandler component has only one sub-component which is UserInterface component. UserInterface displays the output on the application' s screen. Since UserInterface sub-component takes it's input from OutputHandler, OutputHandler is resource of UserInterface sub-component. User interface sub-component displays two different output which are video display, and notifications. Video displays come from directly Kinect SDK, and notifications come from ProcessManager.

7.4.6 Uses/Interactions

As mentioned the responsibilities part, this component is responsible for showing outputs of the system to users. OutputManager is gets the data to show the user from ProcessManager so this component is frequently interacts with ProcessManager . Since OutputManager shows the outputs to user via user interfaces this component uses user interfaces.

7.4.7 Resources

Resource of OutputManager is disk. OutputManager shows the user related data and motion related data to users. User related data are the administrator information and this data are kept in database and motion related data are videos of the motion this videos are also kept in disk, hence main resource of the OutputManager is disk.

7.4.8 Processing

It can be said that OutputManager is managed by ProcessManager. According to current state of the ProcessManager Outputmanager decides which information should be shown to users. In the motion definition mode of the ProcessManager after it defines the movement sends the information about defined data is sent to OutputManager this data contains duration, type and security level of the motion. In the movement recognition mode of the Processmanager , for every detected movement date, security level and campured video information is send to OutputManager. After getting this

information, OutputManager shows this information to users in both motion definition and recognition modes.

7.2.9 Interface/Exports

The set of services:

- Gathers data from Processor Component
- Sends it to Head-Up Display and displays the output

8. Libraries and Tools

8.1 Microsoft Kinect SDK

The SDK includes Windows 7 compatible PC drivers for Kinect device. It provides Kinect capabilities to developers to build applications with C++, C#, or Visual Basic by using Microsoft Visual Studio 2010 and includes following features:

- **Raw sensor streams:** Access to low-level streams from the depth sensor, color camera sensor, and four-element microphone array.
- **Skeletal tracking:** The capability to track the skeleton image of one or two people moving within the Kinect field of view for gesture-driven applications.
- **Advanced audio capabilities:** Audio processing capabilities include sophisticated acoustic noise suppression and echo cancellation, beam formation to identify the current sound source, and integration with the Windows speech recognition API.
- **Sample code and Documentation**

8.2 Microsoft Visual Studio 2010

Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop console and graphical user interface applications along with Windows Forms applications, web sites, web applications, and web services in both native code together with managed code for all platforms supported by Microsoft Windows, Windows Mobile, Windows CE, .NET Framework, .NET Compact Framework and Microsoft Silverlight.

Visual Studio includes a code editor supporting IntelliSense as well as code refactoring. The integrated debugger works both as a source-level debugger and a machine-level debugger. Other built-in tools include a forms designer for building GUI applications, web designer, class designer, and database schema designer. It accepts plug-ins that enhance the functionality at almost every level—including adding support for source-control systems (like Subversion and Visual SourceSafe) and adding new toolsets like editors and visual designers for domain-specific languages or toolsets for other aspects of the software development lifecycle (like the Team Foundation Server client: Team Explorer).

Visual Studio supports different programming languages by means of language services, which allow the code editor and debugger to support (to varying degrees) nearly any programming language, provided a language-specific service exists. Built-in languages include C/C++ (via Visual C++), VB.NET (via Visual Basic .NET), C#(via Visual C#), and F# (as of Visual Studio 2010[3]). Support for other languages such as M, Python, and Ruby among others is available via language services installed separately.

It also supports XML/XSLT, HTML/XHTML, JavaScript and CSS. Individual language-specific versions of Visual Studio also exist which provide more limited language services to the user: Microsoft Visual Basic, Visual J#, Visual C#, and Visual C++.

8.3 HTR Files

The Motion Analysis HTR (Hierarchical Translation-Rotation) format was developed as the native skeleton format for the Motion Analysis skeleton generating software. It was created as an alternative to the only existing other format at the time (the BVH format) because of deficiencies in the BVH specification. The Acclaim file format was not yet entered in to the public domain and wasn't available for them to use.

A rarely used variant of the HTR format, the GTR format (Global Translation-Rotation) is identical to the HTR format except the hierarchy information is skipped. It is almost never used but it will be noted where the GTR would be different than the HTR.

The HTR file format is an excellent all around format. It contains everything necessary for a motion capture file including lots of flexibility in data types and ordering and it has a complete basis pose specification where the starting point for both rotations and translations are given. HTR file format keeps skeleton information in hiararchycal way. This hearachy is shown in figure 20.

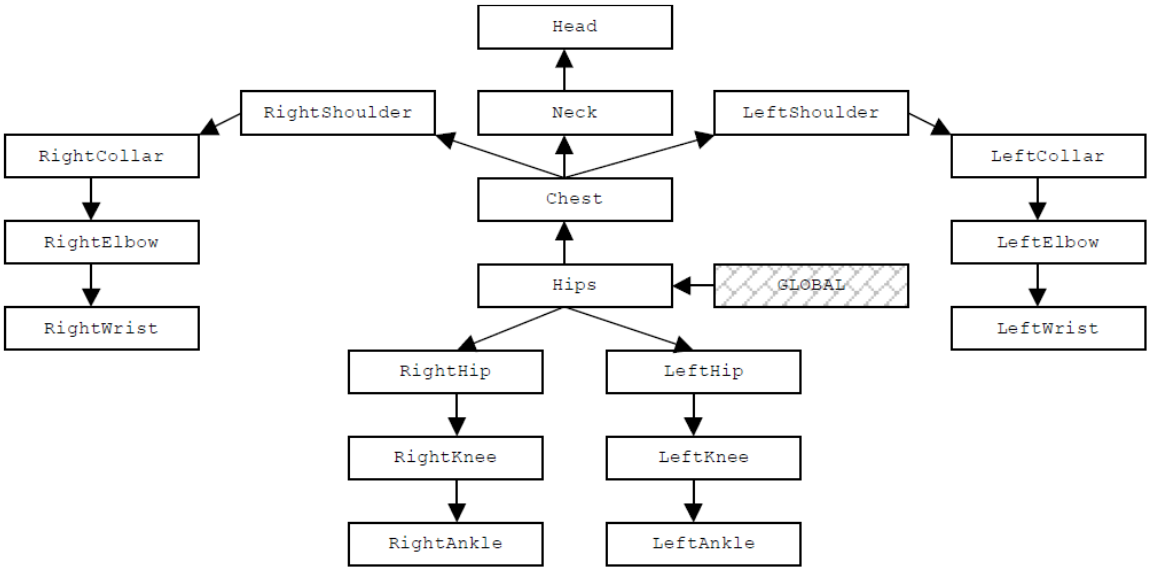

Figure 20 : HTR Hiearchy Diagram.

The HTR file contains multiple sections: header, segment names & hierarchy, base postion and motion sections. They appear in this order and are denoted by keywords in square brackets "[]". Comment lines can appear in the file and are denoted by a hash mark "#" which can appear at any point in a line (not just at the beginning of a line) the rest of the line is a comment line.

The header section contains global parameter information and starts with the keyword **"[HEADER]"** on a line by itself. Keywords for parameters in this section appear as the first word of any line, the value for the parameter immediately follows the

keyword. When parsing the file simply store the parameter values for later when the motion data is composed into a transformation matrix. The following parameters can appear in this section:

- **FileType** - this describes the type of file this is. The value will be either "htr" or "gtr". If this parameter isn't present you should assume the file is an htr file.
- **DataType -** this describes the order of transformation composition of the translation, rotation and scale values. This value is almost always "HTRS" but it's certainly possible for it to be another order.
- **FileVersion** - this describes the file version of the HTR format. This document is for file version 1.
- **NumSegments -** this gives the count of the number of segments found in the file.
- **NumFrames -** this gives the number of frames (samples) in the file
- **DataFrameRate** - this gives the sample rate of the data in the file. This number is expressed as the number of samples per second.
- **EulerRotationOrder** - this specifies the order in which the X,Y and Z rotation values are to be composited. This value does frequently vary from one HTR file to the next so you should be ready to handle different rotation orders.
- **CalibrationUnits**- this is essentially the translation units for the file. Metric values are common for HTR files and can sometimes be expressed as a fraction of a meter. For example "1/10.000 m" would be decimeters. Millimeters, on the other hand, will be indicated with the common abbreviation "mm".
- **RotationUnits** - this gives the rotation units for the file. This is almost always expressed in "Degrees".
- **GlobalAxisofGravity** - the HTR file allows some flexibility in the spefication of the world coordinate system. This value specifies the global up axis of the data. The positive Y axis is the default, the positive Z axis is a common alternative.
- **BoneLengthAxis** - this is axis along which each segment is aligned. If this isn't present you can assume the default is the Y axis.
- **ScaleFactor -** this is a global scale applied to the motion data.

The next section is the segment names and hierarchy section. This is denoted by the keyword **"[SegmentNames&Hierarchy]".** The first word on each line is the name of a segment and the second word is the parent of that segment. If that segment does not have a parent (i.e. is the root object) then the second word is the keyword **"GLOBAL".** This means that you shouldn't ever have a segment in an HTR file that has the name "global", this would likely confuse most HTR file parsers.

The next section is the base position section and is denoted by the keyword **"[BasePosition].** Each line begins with a segment name followed by the translation, rotation and segment length. The translation and rotation values appear in the order X, Y and then Z. It's convenient to create a matrix that is the composition of the base position rotations. The base rotation is used repeatedly so doing the composition now saves time later.

The motion section isn't denoted by the "motion" keyword. Instead, the data for each segment appears in turn and each section begins with the name of the segment in square brackets. Each line of motion data for a segment starts with the sample number followed by the translation values, then the rotation values and then a scale factor which gets applied to the length of the segment. The translation and rotation values appear in the order X, Y and then Z.

8.4 MySQL

MySQL is a relational database management system that runs as a server providing multiuser access to a number of databases. MySQL is written in C and C++. Its SQL parser is written in yacc. MySQL works on many different system platforms, including Linux, Mac OS X, and Microsoft Windows. Many programming languages with language-specific APIs include libraries for accessing MySQL databases. These include MySQL Connector/Net for integration with Microsoft's Visual Studio (languages such as C# and VB are most commonly used) and the ODBC driver for Java. In addition, an ODBC interface called MyODBC allows additional programming languages that support

the ODBC interface to communicate with a MySQL database, such as ASP or ColdFusion. The HTSQL - URL based query method also ships with a MySQL adapter, allowing direct interaction between a MySQL database and any web client via structured URLs. The MySQL server and official libraries are mostly implemented in ANSI C/ANSI C++.

9. Time Planning

9.1 Term 1 Gannt Chart

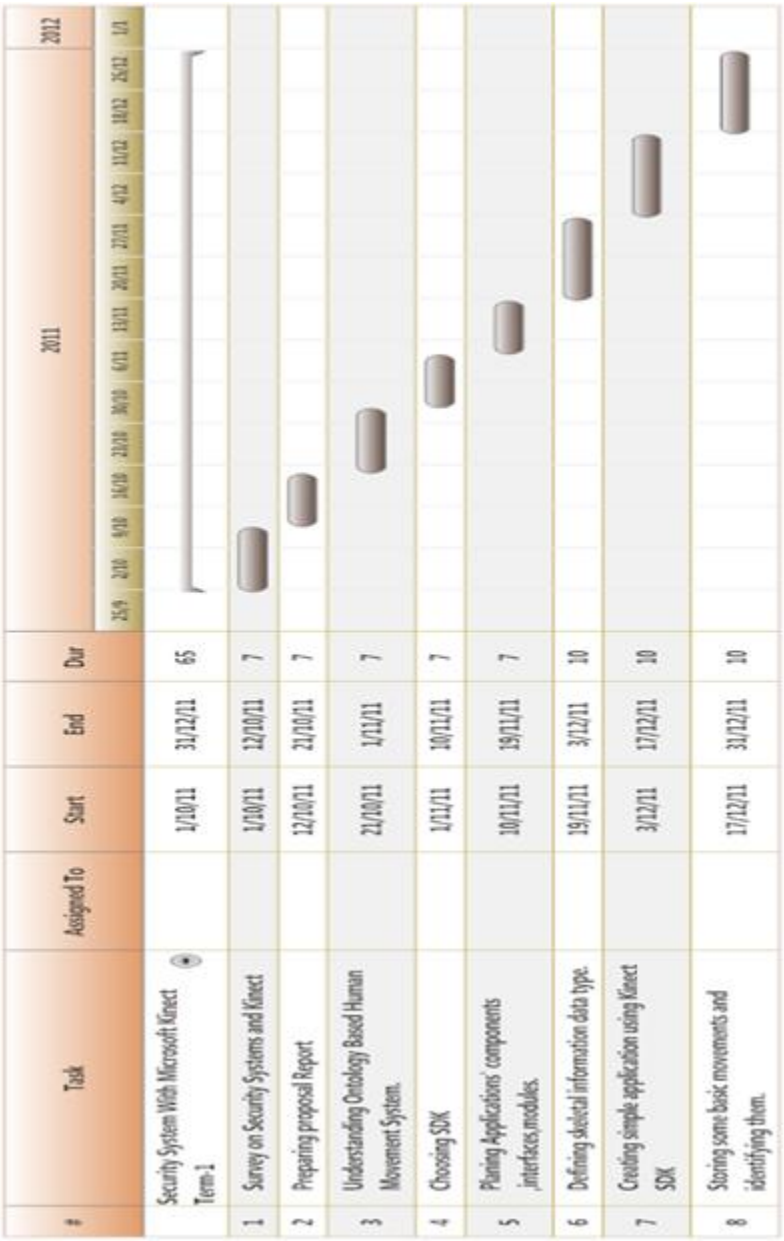

Figure 21: Term 1 Gannt Chart.

9.2 Term 2 Gannt Chart

Figure 22: Term 2 Gannt Chart.

10. Conclusion

This document describes the design levels of the Security System with Microsoft Kinect project conducted by 213. It starts with a definition of the real world problem and explains the solution that the project proposes. These explanations include basically design considerations, data design, system architecture, libraries, tools and time planning. Furthermore, class diagrams with data flow diagrams and design of the user interfaces are showed in the document in detail. Consequently, this document is prepared to conduct better design approaches to Security System with Microsoft Kinect project at implementation.