Security System with Microsoft Kinect

Initial Design Report

-Erdinç Kaya
-İlhan Yoldaş Karabulut
-Alişan Yılmaz
-Utku Şahin

Sponsored by Simsoft

2011

Table of Contents

1. Introduction

This Initial Design report is a complete description on 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. This project 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 the mind. The main purpose of security cameras is to discourage the criminals. If a crime happens the security cameras can be used for an investigation. 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. 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 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 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

Our project is called Security system with kinect. As can be understand by it is name, we intend to build a program working with the device kinect that Microsoft developed. In order to make it more clearly we will give information about the system in 4 parts.

1.3.1 How Does Kinect Works?

Kinect is one of the newest devices that Microsoft developed. It is pretty similar to a video camera but can give information about all three dimensions of an object within the range. It returns the body positions with twenty [3] coordinates.

1.3.2 What Do We Intend To Do?

The kinect device will be placed to the desired area. Since every place has different security issues new movements will be implemented to the program by The Movement Processing System. Every movement will be categorized as their priority of suspicions. If any of these movements occur, the device will send warning information to both administrator and the owner of the product. These warning may be sent to the system itself, may be sent as an e-mail or text-message according to the importance of the situation. Even a message directly can be sent to the police if it seems necessary.

Our project has two basic parts. The first part will be the core of the system that analyze the movements and categorize them. The second part will be built on the first one.

1.3.3 The Movement Processing System

This is the core part of our project Movement Definition and recognition system (MDRS). With the help of the kinect we will categorize every movement and will be able to recognize what it was. After the hardware implemented we will be able to develop a movement database easily with these system.

1.3.4 The Security System

This is the part where the users manage the system. There are two interfaces. First one is for the owner of the product in order to register the administrators who will use the programs. The second interface for the administrators to use the products. Both interfaces have a video visualization, notification page and a last notification part. They will be able to see the situations occurred by clicking thumbnails. Instant notifications will make handling the situation easier. So, no one will be able to stand across a monitor watching everything.

1.4 User and Literature Survey

In this project there are two sides of the literature survey. One side is what kind of applications are made by using kinect. This side is critical for understanding abilities of kinect and what we can or cannot through using kinect. The other side is security systems and their limitations. By knowing security systems abilities we could know that how can we improve this systems.

When we look at the applications that are made by using kinect, we saw that they are mostly games and entertainment purposed applications [2]. However, when we examine these games they use skeletal information that Kinect provide, in run time. Kinect give us the ability of processing skeletal information simultaneously while getting this information. Literature survey shows us this is a new situation. Most of the research on processing skeletal information is based on comparing information sets which previously recorded [4]. In our project we will compare real time information that we obtain and previously recorded data. This ability makes us use Kinect based application as a security system.

There are lots of camera-based security systems that use different technologies like surveillance, motion triggered and infra-red cameras. Some of them can inform peoples whether there is a movement or not .However none of these system could identify what kind of a movement detected, so this systems cause a lots of unimportant warnings. For instance a mouse can cause a red alarm in a bank with motion triggered system. Our system solves this problem by identifying movement. In addition, as a part of our application it will be defined specific movements. Application will not use same data set for every place.

1.5 Definitions and Abbreviations

IDR: Initial Design Report.

SDK: Software Development Kit.

MDRS : Movement Definition and Recognation System.

HTR : Hierarchycal Translate and Rotation.

N/A : Not applicable.

1.6 References

Referenced documents for this IDR document is provided in Table 1. 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.

Table 1: References List

Reference Number	Title	Report Number or Version	Date	Publishing - Organization
1	**IEEE Recommended Practice for Software Requirements Specifications**	IEEE Std 830-1998 (Revision of IEEE Std 830-1993)	June 25, 1998	IEEE
2	**Microsoft Kinect Official Webpage, http://www.xbox.com/en-US/kinect.**	N/A	N/A	Microsoft

3	**Kinect SDK web page , http://research.microsoft.com/en-us/um/redmond/projects/kinectsdk/**	N/A	N/A	Microsoft
4	**Ontology-based Classification of 3D Virtual Human Motion Data**	N/A	N/A	Veysi İşler
5	**Motion Capture File Formats Explained**	N/A		**Department of Computer Science, university of Sheffield**

1.7. 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 includes the libraries and tools that will be used during software development. In section 8, time planning and scheduling issues will be demonstrated by a Gannt Chart. The document is ended with a conclusion.

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 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.

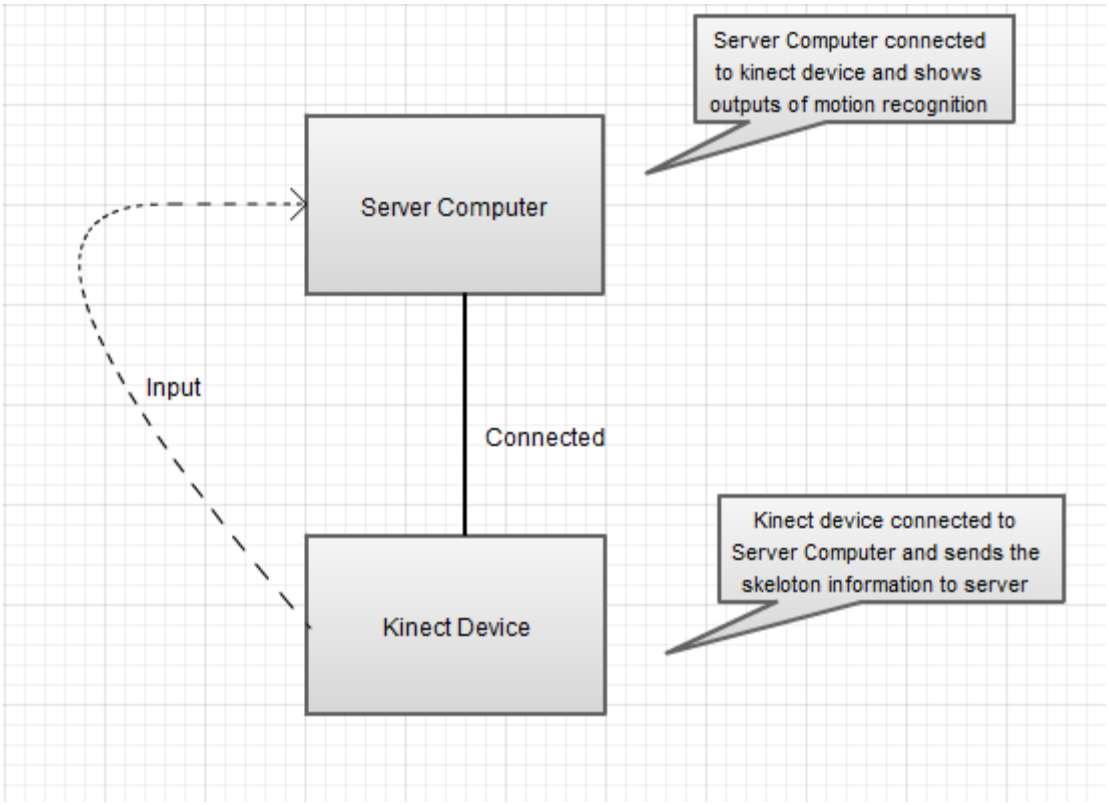

Figure 1 : Overall Design

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 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 has 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, 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 this initial design 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 have knowledge about 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 and processing movement files. Moreover the members are inexperienced on the movement recognition platform. 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 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 to express movements more efficiently. In addition, through designing the project, we also take the following features into consideration:

3.2.1 Reliability

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 guarantee that handle all problems comes from user, our problem solves this problem its own graphic user interface, because in the project's

graphic user interface user cannot enter many command, user just enters commands via buttons.

3.2.2 Usability

Since this project intends to simplify the user's life, and a standard user does not need to have any information about the program, we are trying to create a user-friendly interface. 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 a notification. 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 protect user's life.

3.2.3. Portability

This project has no portability, because one of the main constraints is place and distance, if we add portability this project, we destroy everything. Hence portability is not an option for this project.

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.

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. Because 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.

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.

4.1.1 Administrator Table

Administrator table keeps the information about administrators.This table 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
CellPhone	String	10 characters	Alphanumeric characters

Figure 2: 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 from the movement table. 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	

Figure 3: Movement Table

4.1.3 Tree Table

Tree table keeps the information about the tree structure of movements. This table includes 2 attributes namely, ParentMovement, ChildMovement.

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

Figure 4: Tree Table

4.1.4 Log Table

Log table keeps the information about logs. This table 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	

Figure 5: Log Table

4.1.5 Notification Table

Notification table keeps the information about notifiations. 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	

Figure 6: Notification Table

4.2 Data Dictionary

4.2.1 Entities

4.2.1.1 Admin

As it is stated before, 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.
- **CellPhone :** This attribute keeps the cell phone number for each admin. Its type is string.

4.2.1.2 Movement

As it is stated before, 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 before, 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 before, 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 before, 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.

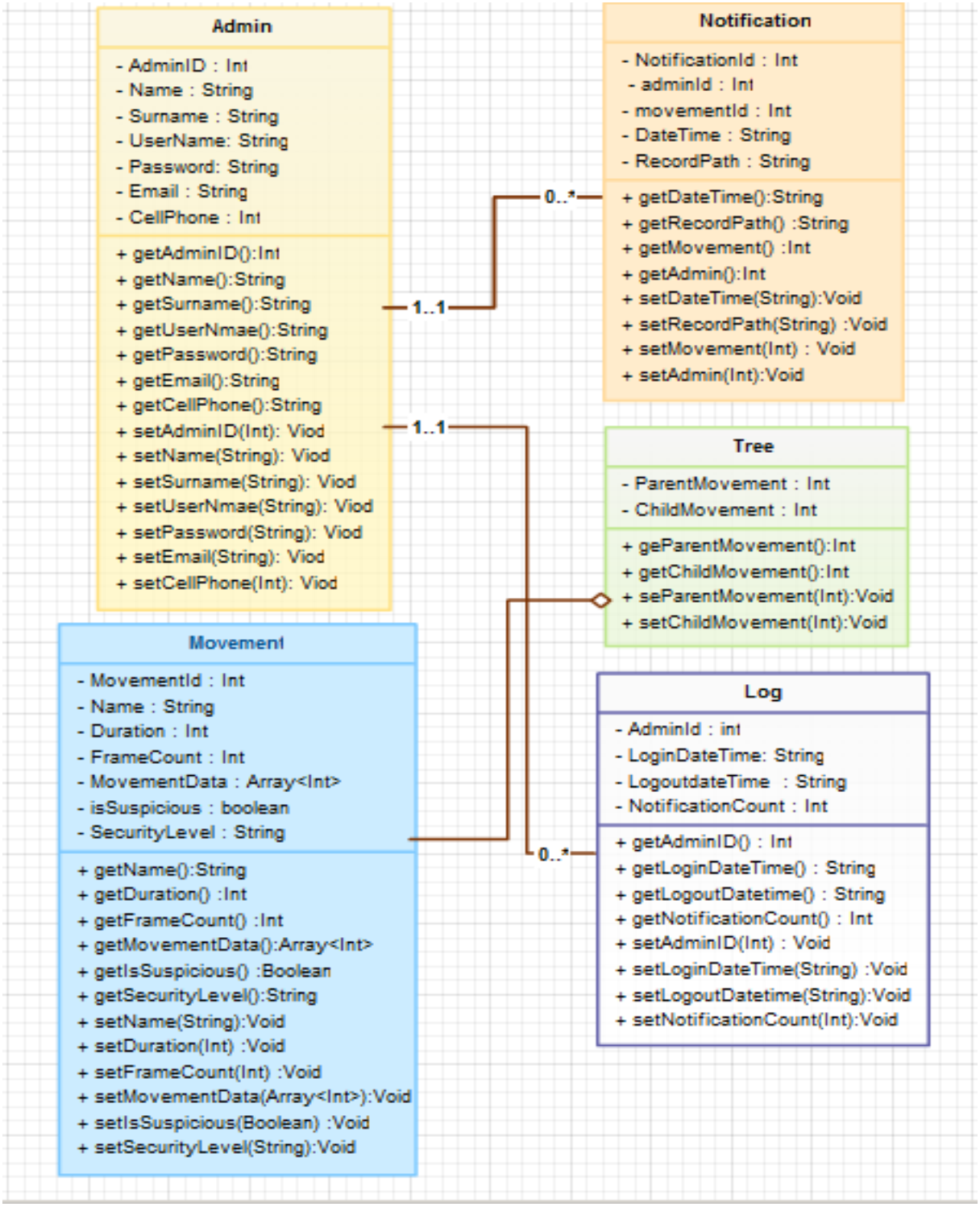

Figure 7 : Class Diagram.

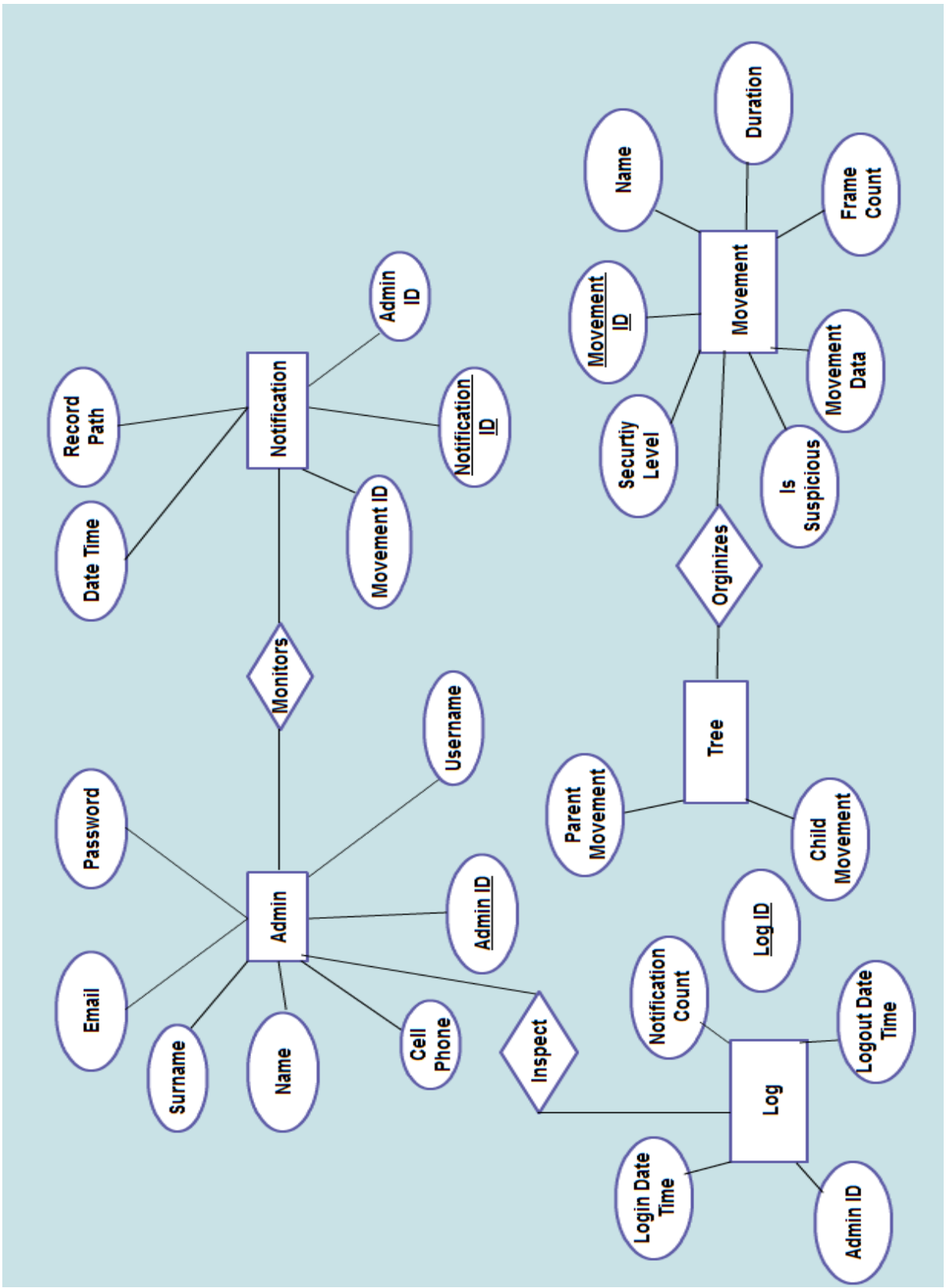

Figure 8 : ER Diagram .

5. System Architecture

5.1. Architectural Design

The system consists of 4 component 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 responsible from 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.

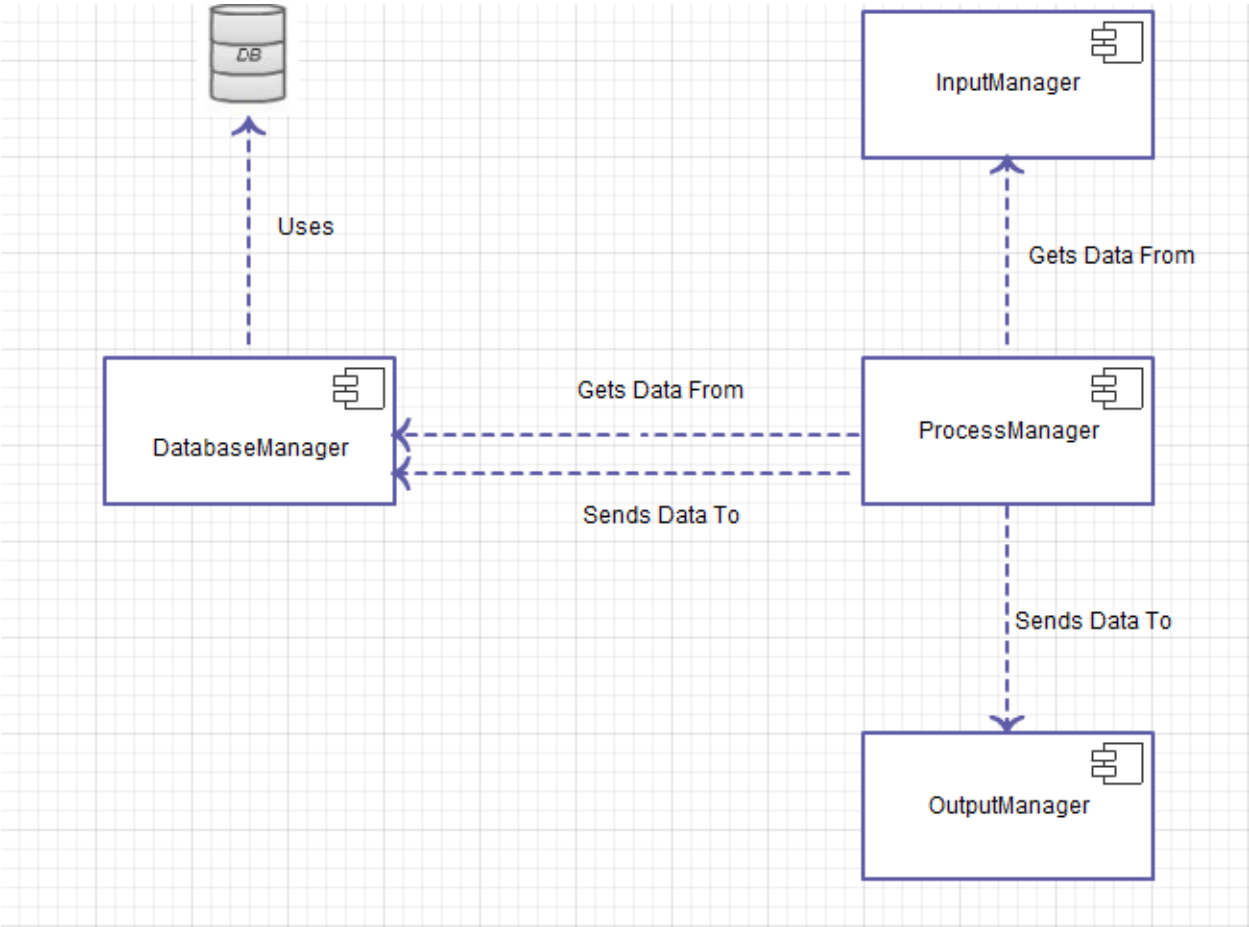

Figure 9 : Component Diagram.

5.2. Description of components

In this section decomposition of the subsystems in the architectural design is explained in detail. The interaction between the components can be seen as a sequence diagram in the following figure.

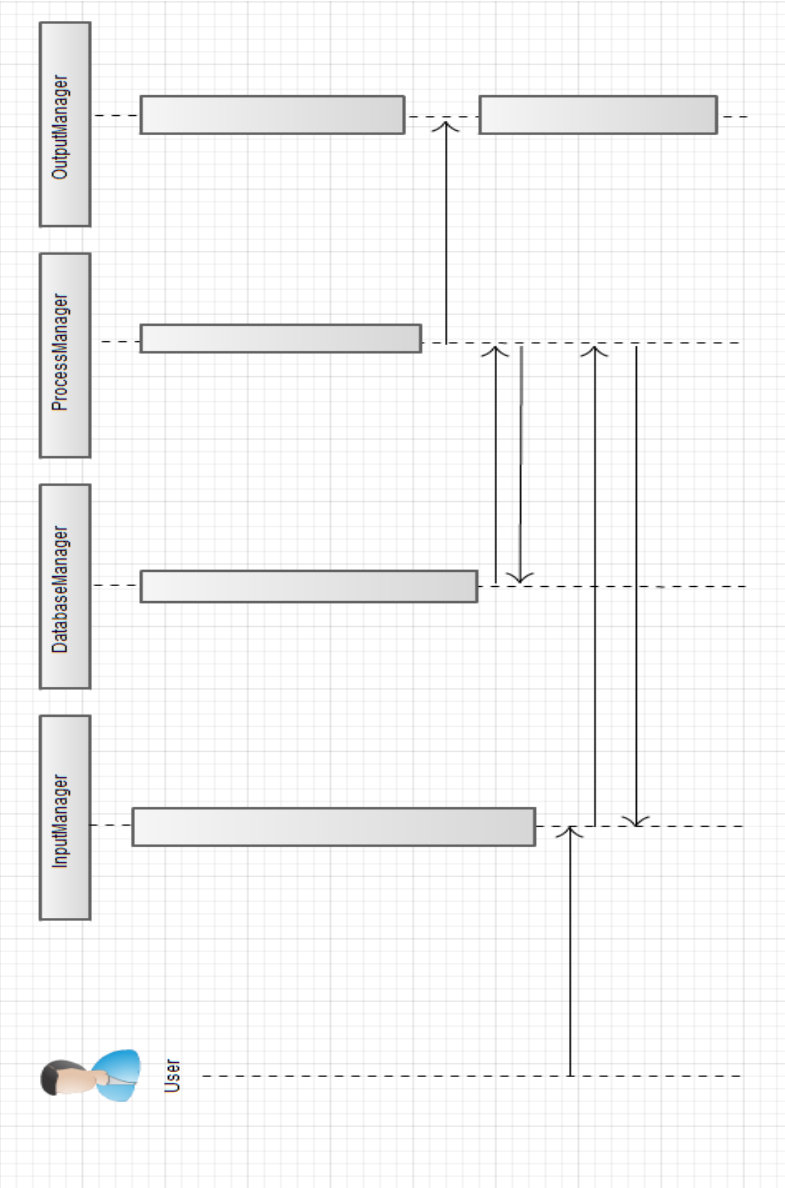

Figure 10 : State Diagram.

5.2.1. Database Manager

5.2.1.1 Processing Narrative for Database Manager

DatabaseManager is used for communication with database .Database is 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, because database is a sub-component of the databaseManager , is not an input interface of DatabaseManager. Since ProcessManager sends data to DatabaseManager in order to query to database ProcessManager is input interface of DatabaseManager . Database manager sends the results of the queries to the ProcessManager, ProcessManager is output interface of the Database Manager.

5.2.1.3. DatabaseManager Processing Detail

- ProcessManager request for data.
- DatabaseManager uses this request as input.
- DatabaseManager queries this input 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 for data to DatabaseManager and after DatabaseManager query this request to database and sends resulting data to ProcessManager.

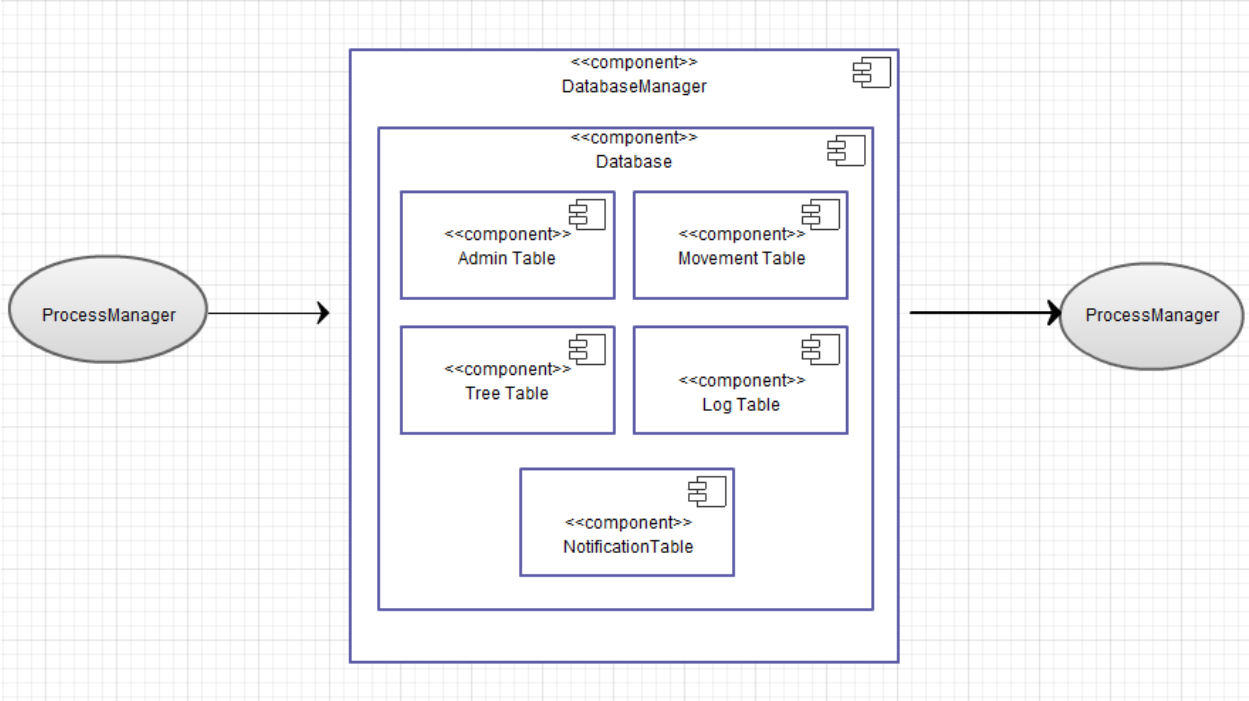

Figure 11 : DatabaseManager Component

5.2.2. InputManager

5.2.2.1 Processing Narrative for InputManager

InputManager component is responsible from getting input from kinect device and the 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 manage the data coming from kinect.

5.2.1.2 InputManager Interface Description

Input interface of the InputManager is UserInterfaceHandler sub-component of the ProcessManager because when an input needs to be get from user, UserInterfaceHandler notify the InputManager to change current state to suitable state which gets needed information from user.

Output interface of the InputManager is UserInterfaceHandler and KinectInputHandler . InputManager sends the data which user gives , to

UserInterfaceHandler and gathered information from kinect device to KinectInputHandler .

5.2.2.3. InputManager Processing Detail

When InputManager in User interaction mode;

- UserInterfaceHandler sends request for user input.
- InputManager gets user inputs.
- InputManager sends data to UserInterfaceHandler sub-component of ProcessManager.

When InputManager is in motion detection mode;

- Data is gathered via kinect device.
- Data is sended to KinectInputHandler sub-component 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 UserInterfaceHandler and KinectInputHandler sub-components of ProcessManager.

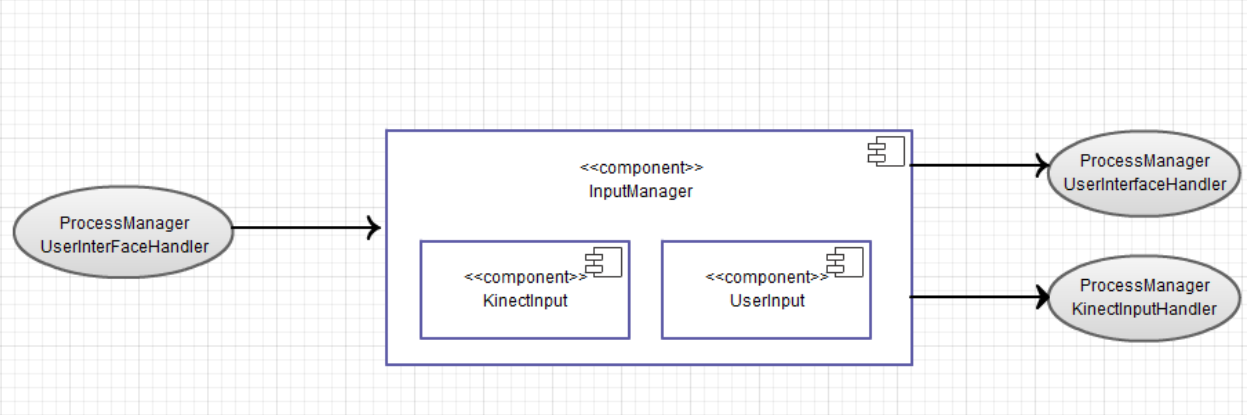

Figure 12 : InputManager component.

5.2.3. ProcessManager

5.2.3.1 Processing Narrative for ProcessManager

ProcessManager is responsible from 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 relataed 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 interce of ProcessManager is DatabaseManager, kinectInput sub-component of InputManager and UserInput sub-component of InputManager. Those components send data to KinectInputHandler and UserInterfaceHandler sub-components of ProcessManager.

Output interface of ProcessManager is InputManager , DatabaseManager and mainly OutputManager . DatabaseManager and InputManager waits request for data from Input manager so these are 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 in User interaction mode;

- ProcessManager demands for data from UserInput sub-componet of InputManager.
- After getting data sends requests to DatabaseManager.
- Gets data sets related to requests from DatabaseManager .
- Sends data sets to OutputManager.

When ProcessManager is in motion detection 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.

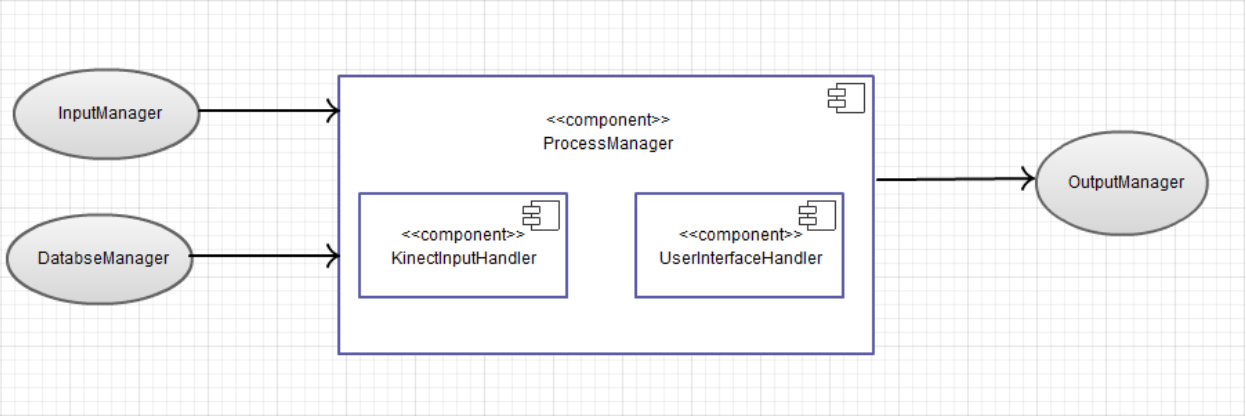

Figure 13: ProcessManager component.

5.2.4. OutputManager

5.2.4.1 Processing Narrative for OutputManager

The OutputManager is responsible from showing processed data in user interfaces. When the program is in Motion detection state KinectInputhandler sub-component of ProcessManager sends the output of motion detection to the OutputManager and when current state is user interaction UserInterfaceHandler outputs the data to OutputManager.

5.2.4.2 OutputManager Interface Description

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

5.2.4.3. OutputManager Processing Detail

When ProcessManager in User interaction mode;

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

When ProcessManager is in motion detection mode;

- KinectInputHandler 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.

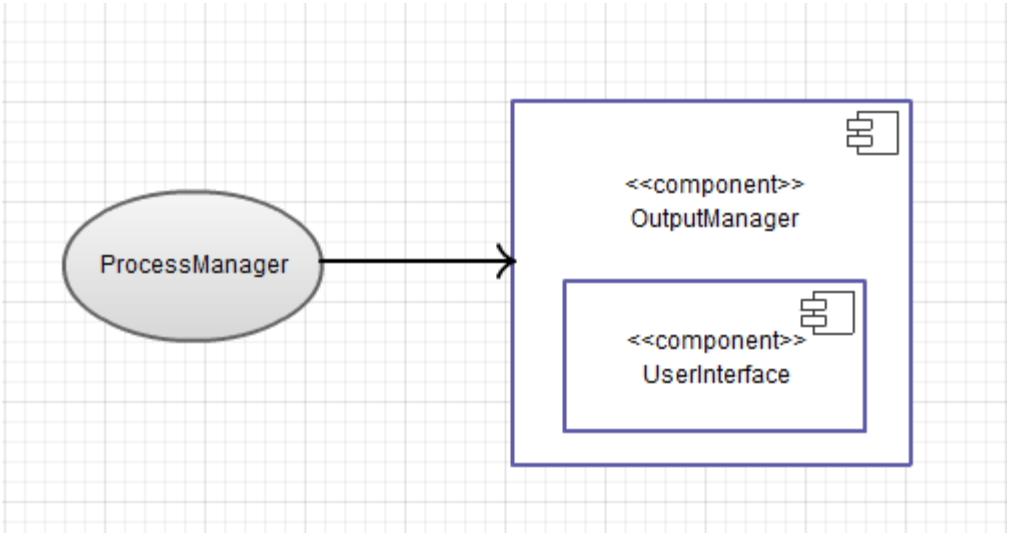

Figure 14: OutputManager component.

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 KinectInputHandler and UserInterfaceHandler sub-components of the ProcessManager .

Finally, we decided to have a 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 are gather information from different sources users and kinect device.

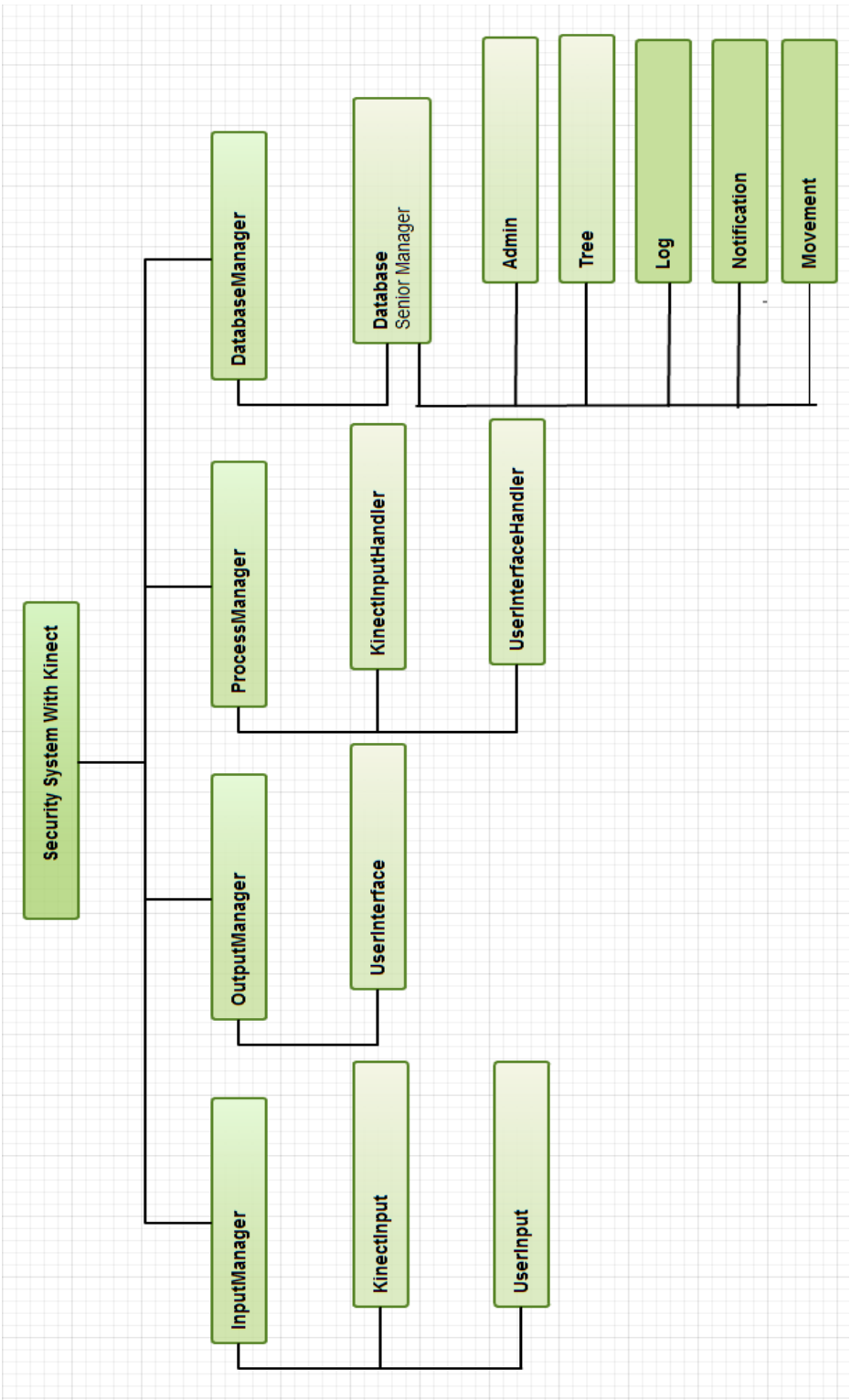

Figure 15: Aggregation Hierarchy Diagram.

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.

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.

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.

6.1.4 New Administrator Page

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

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.

6.2 Screen Images

Figure 16 : Login Page.

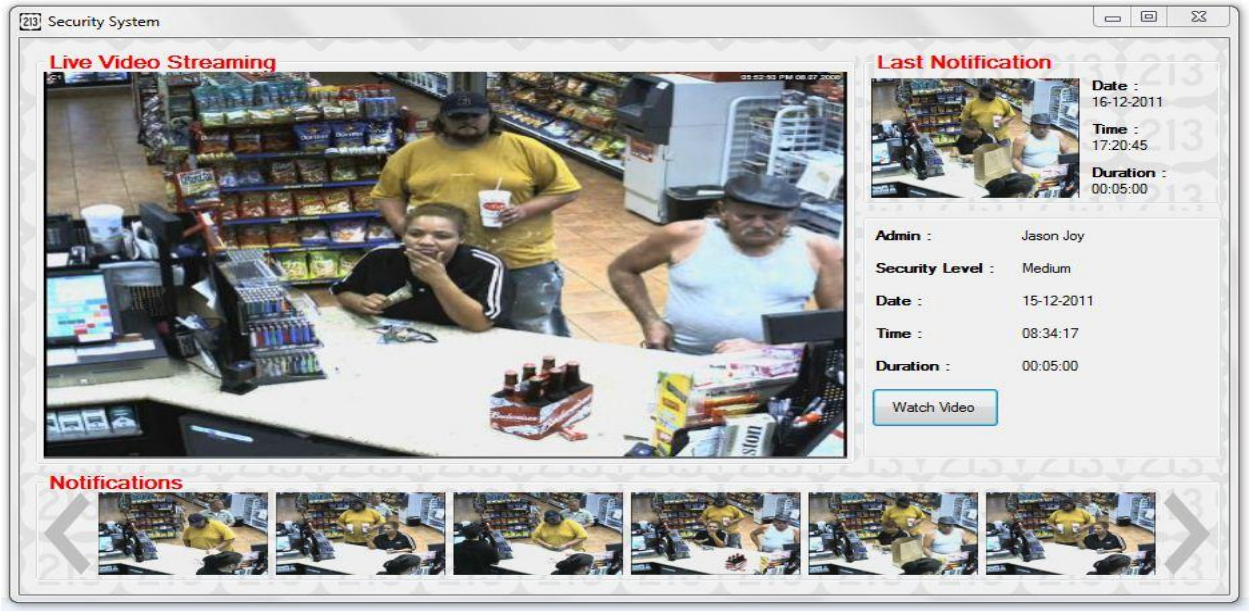

Figure 17 : Administrator Page.

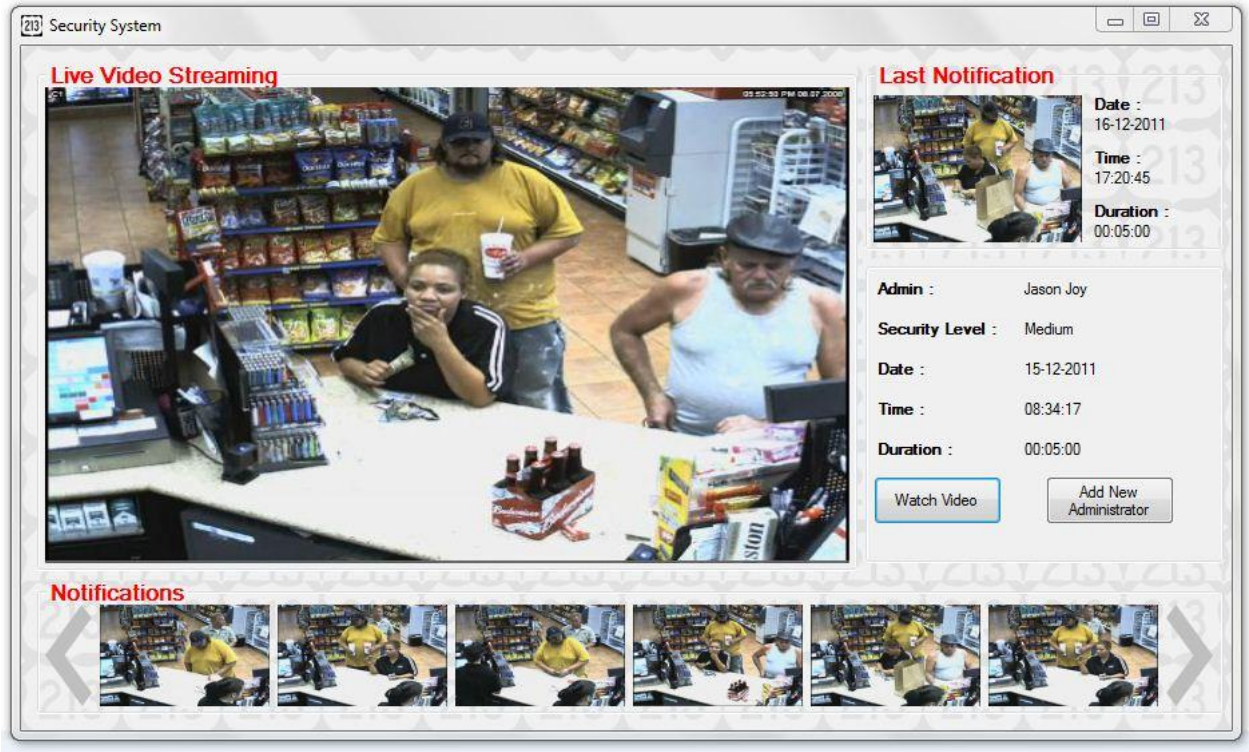

Figure 18: Owner Page.

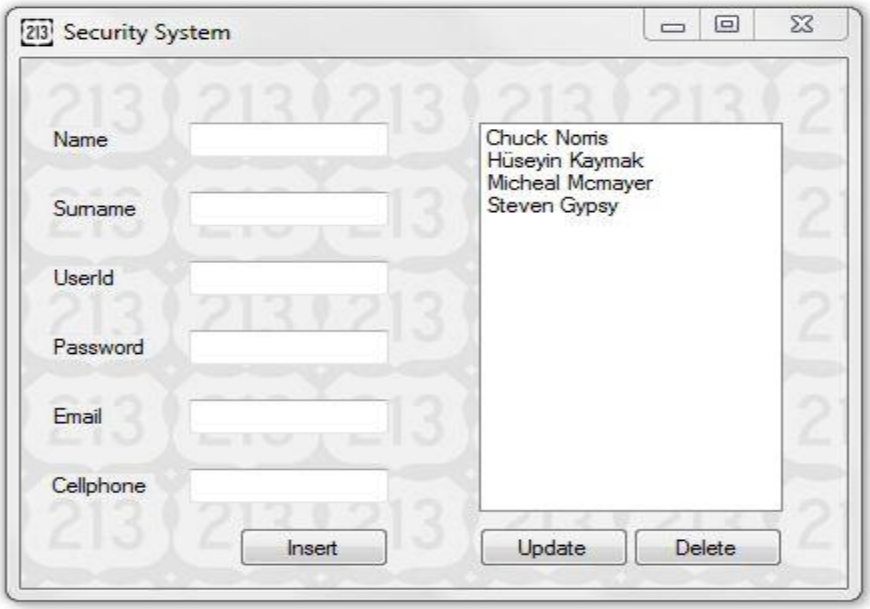

Figure 19 : New Administrator Page.

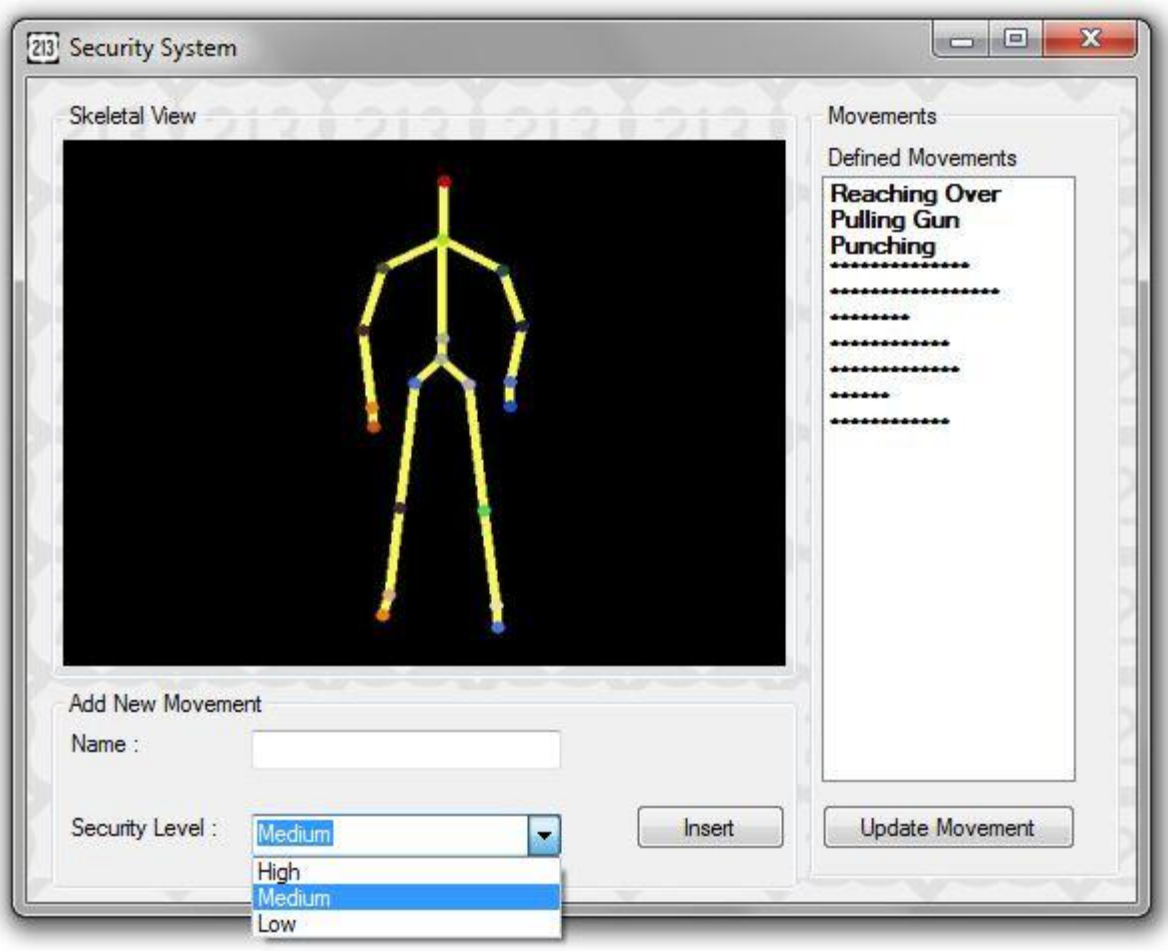

Figure 20 : Movement Defining Page.

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 to 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. 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 are

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. Libraries and Tools

7.1 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**

7.2 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++.

7.3 HTR Files

The HTR format (Hierarchical Translation Rotation) has been developed as a native format for the skeleton of the Motion Analysis software. It has been created as an alternative to the BVH format to make up for its main drawbacks. It also has a complete base pose specification, which consists of indicating the starting point for both rotations and translations. The stored data is grouped by segments; all motions from the first segment are read, then those from the next segment, and so on.

The HTR format contains four sections: Header, Segment Names & Hierarchy, Base Position, and the motion data section.

The header section contains global parameter information:

- file type
- data type
- file version
- number of segments
- number of frames
- data frame rate
- Euler rotation order
- calibration units
- rotation units
- global axis of gravity
- bone length axis
- scale factor

7.3 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++.

8. Time Planning

8.1 Term 1 Gannt Chart

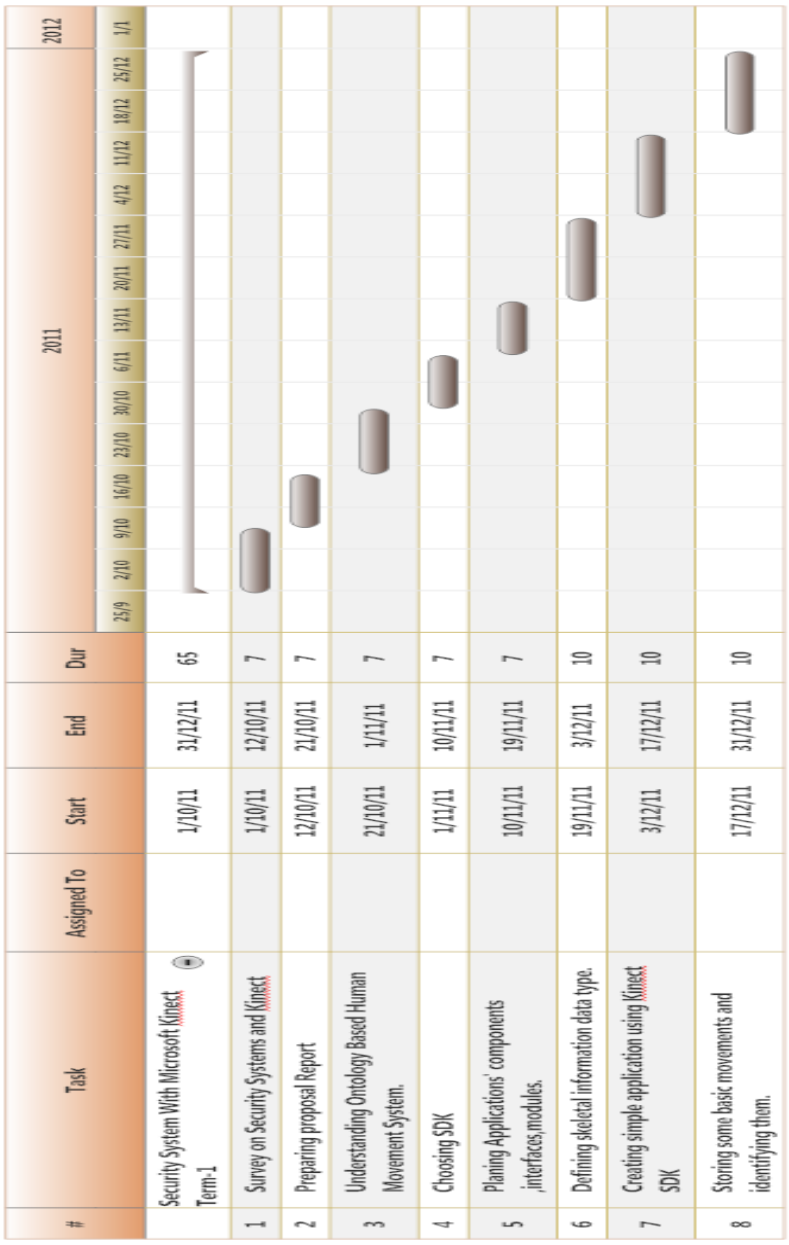

Figure 21: Term 1 Gannt Chart.

8.2 Term 2 Gannt Chart

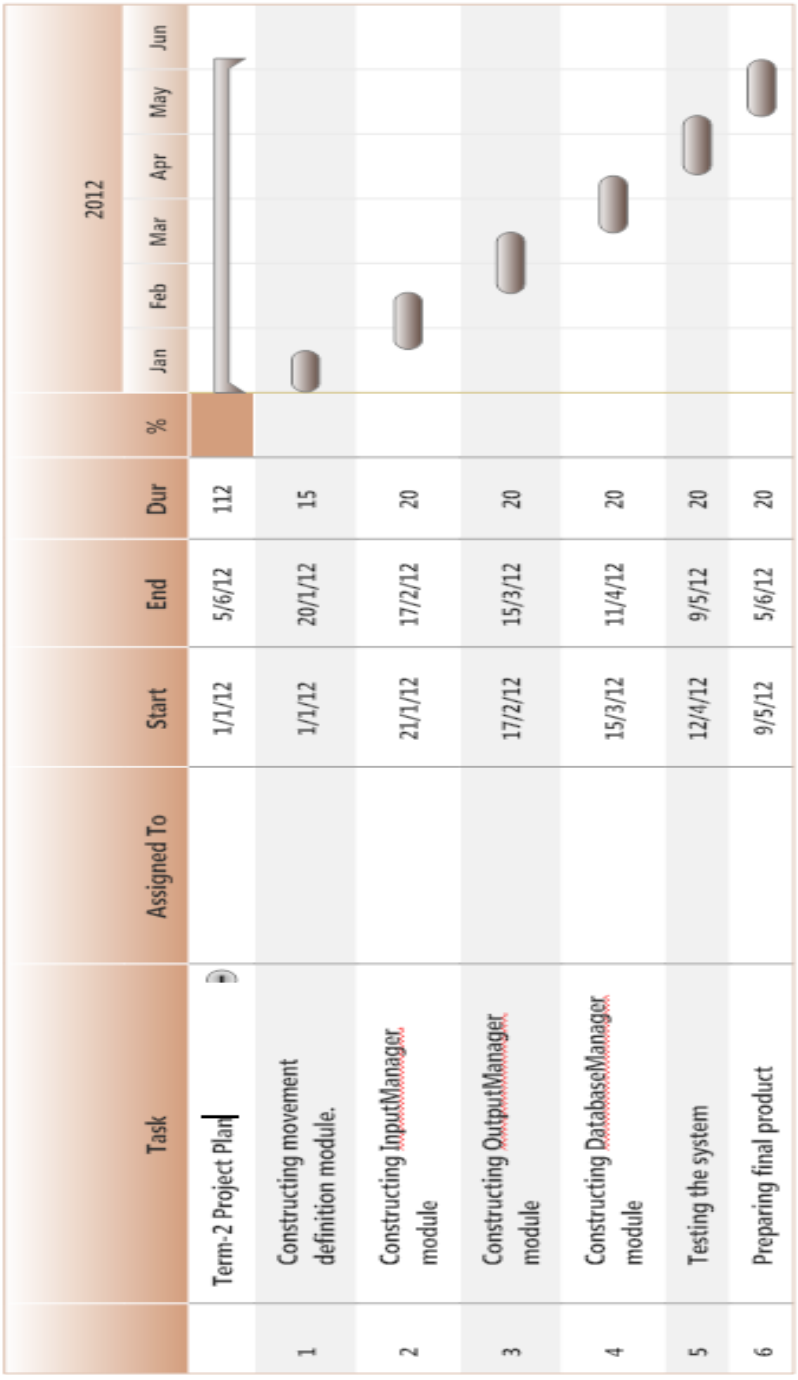

Figure 22: Term 1 Gannt Chart.

9. 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. This explanations includes basically design considerations, data design, system architecture, libraries,tools and time planing. 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.