HUMAN BODY TRACKING SYSTEM

Software Design Description Document

V1.1

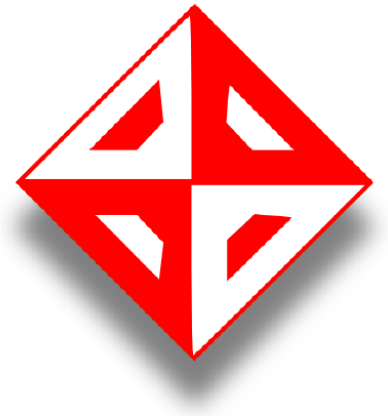
Mindless Rookies

Zehra Deniz Çelik

Burak Araz

Cem Aydın

Yalçın Savrun



Revision History

Date	Revision	Comment
03.01.2015	1.0	Created
01.03.2015	1.1	4.3 Design stakeholders and their concerns is updated. 4.8 Design rationale is updated. 5.2.2 Design Elements is updated. 5.3.2. Design Elements is updated. 5.3.2.1. Functional & Subordinates Attributes is updated. 5.4.2.1.3 Manager Class is updated. 5.8.1 Design concerns is updated. 5.10.2. Design Elements is updated. 5.11.2. Design Elements is updated. 6. Traceability Matrix is updated.

Preface

This document includes the Software Design Description (SDD) of the Human Body Tracking System. This document is prepared according to the “IEEE Standard for Information Technology – Systems Design – Software Design Descriptions – IEEE Std. 1016 – 2009”. Main purpose of this documentation is to provide a complete description of all the system design and views of the project.

The Human Body Tracking System aims to track human bodies in retailer environments in order to generate a heat map which shows usage weights of different tracks on a retailer environment. The target audience, who wants to make use of this system, can find all related design descriptions information in this document. It assists the software developer team, the stakeholders and the end users.

The first section of this document includes the conceptual model for SDD. The role and influences of this document are specified. The following sections include design description information content and design viewpoints of the system. Design viewpoints are indicated along with their related diagrams. They, as a whole, provide the details of the system modeling and architectural design concerns.

Content

1. Overview

1.1 Scope

This document describes the design description of the Human Body Tracking System. It provides detailed information about basic structure of the HBTS and how the implementation process will be carried out. Each design concern of the stakeholders are topic of at least one design view and these design views are described with corresponding design elements and modeled with related UML diagrams.

This document is prepared in IEEE 1016- 2009 standards.

1.2 Purpose

The purpose of this document is to provide a detailed description of the all design patterns of the HBTS. In The Software Design Description document, general description of design elements and their interactions, how the system will be structured, data & functional structure will be discussed in order to help producing test cases, and also satisfy requirements, design details indicated in the SRS document.

1.3 Intended Audience

Intended audience of this software design description document is all stakeholders which includes product managers, development team, and testers.

2. Definitions

TERM, ACRONYM, ABBREVIATION	DEFINITION
HBTS	Human Body Tracking System
Heat Map	Result of the system data in graphical representation. Data is measured with individual colors on the heat map.
Admin	The person who responsible for whole system
Store Manager	The person who is responsible for a branch store.
Brand Manager	The person who is responsible for a brand.
SDD	Software Design Document
OpenCV	Open Source Computer Vision. A library of real-

	time computer vision routines from Intel.
MVC	Model-View-Controller
Background Subtraction	It is a technique in the fields of image processing and computer vision.
Criteria	The attributes offered by the system to filter the heat map according to time intervals
UI	User Interface
Blob	Region of a digital image in which some properties are vary within described range of values or constant.
MOG	(Mixture of Gaussians) A method that works with probability density logic. It is also used for one of the background subtraction methods.
MySQL	It is an open-source relational database management system (RDBMS).
Use-Case Diagram	It is a graphic depiction of the interactions among the elements of a system.
Component Diagram	It depicts how components are wired together to form larger components and or software systems.
Deployment Diagram	It depicts a static view of the run-time configuration of processing nodes and the components that run on those nodes.
Class Diagram	A type of static structure diagram in UML that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes
ER Diagram	Entity Relationship Diagram (ERD) is a visual representation of different data using conventions that describe how these data are related to each other.
Sequence Diagram	A sequence diagram is an interaction diagram that shows how processes operate with one

	another and in what order.
State Transition Diagram	It describes the time-dependent behavior of a system.
Frame Grabber	It is an electronic device that captures individual, digital still frames from an analog video signal or a digital video stream.

3. Conceptual model for software design descriptions

In this section, conceptual model for SDD, which includes basic terms and concepts of SDD and the context in which SDD is prepared and used, will be explained. The conceptual model also gives information about stakeholders who will use SDD and how the SDD will be used.

3.1 Software design descriptions

This subsection includes information about main design subjects of the software product. Human Body Tracking System will be designed in an object-oriented fashion. OpenCV will be used during implementation of the body tracking part, because the product requires advance image processing and computer vision techniques and OpenCV is an open source computer vision and machine learning software library. Body tracking part will be implemented in C++ by using Eclipse IDE. Brands, stores, brand managers, store managers, blobs and their positions will be stored in MySQL database. For administration purposes of MySQL database, phpMyAdmin will be preferred. Tracked positions of a blob that are captured from a camera will be stored in database as JSON object and they will be used later for generating heat maps. Managers and admins will display heat maps from a web application. The web application will be implemented with ASP.NET MVC 4 framework that applies model-view-controller (MVC) pattern and C# programming language. Microsoft Visual Studio 2010 is chosen as an integrated development environment. In order to generate heat maps, a JavaScript library called Heatmap.js is planned to be used. C/C++ compiler and Intel TBB (Threading Building Blocks) of Intel Parallel Studio XE toolset will be frequently used during the project. They can be integrated with Eclipse IDE on Linux.

3.2 Software design descriptions within the life cycle

Software Design Descriptions document is created with a life cycle based on IEEE standards. It shows specifications of the various design situations which forms the document.

3.2.1 Influences on SDD preparation

The SDD document of Human Body Tracking system will be the primary key for software design. All design perspectives, functional, non-functional and interface requirements shape the outline of the architectural design. Moreover, demands of stakeholders may influence the content of SDD.

3.2.2 Influences on software life cycle products

Software process model used in this project is incremental development with plan-driven approach. By this process model, documentation which has to be changed in the process is reduced while taking feedbacks from customers and changes in their requirements are possible during development phase. After first iterations, a prototype will be shown to the company that the team runs the project together and also there will be a demo presentation in METU Computer Engineering Department. According to the feedbacks that will be given, final product will be implemented.

3.2.3 Design verification and design role in validation

This document verifies and validates that the requirements specified in SRS document matches the product design. All test cases and test plans will be prepared with this consistency motivation. All system components will be tested according to these test cases and checked whether the software product fulfills the requirements.

4. Design description information content

4.1 Introduction

The Software Design Description document of Human Body Tracking System analyzes how the system will be designed and developed. The scope of this section consists of these topics according to identification of the SDD, design concerns of the product and identified design stakeholders, related design viewpoints, design views, overlays and rationale.

4.2 SDD identification

The Human Body Tracking System will be completed with its all components and functionalities by the end of June 2015 after validation and verification tests. Resulting prototype after first increments is shown at 3rd week of January 2015. All rights of the product are reserved. Modification and distribution of Human Body Tracking System can only be done by Adoniss Yazılım who has the copyrights of the project. Scope, references, context and summary can be found under the section “Overview”. Moreover, under the section “Definitions” there can be found glossary of the document.

4.3 Design stakeholders and their concerns

Stakeholders of this project include a developer team, a mentor company that has the copyrights and managers of retail environments. Main concerns of the stakeholders are performance and efficiency of the system. Also, some challenges caused by capturing video images and processing them in an efficient way and time will be concerns that the stakeholders will face. For example, in an environment designed with multi cameras will create the need of merging those videos together and creating a solution from the merged image. Moreover, further statistical results may be needed for some cases, such as, categorizing tracked people by their sex, age etc. may be able to solve by more advanced algorithms and calculations. Flexibility of The Human Body Tracking System is will be able to answer such concerns if needed.

4.4 Design views

Design views of Software Design Description document helps design stakeholders about focusing on design specification from a specific point of view and focusing relevant requirements. Each specified design concern will be a separate topic of at least one design view in SDD document. Moreover, each design concern identified in the previous subsection is the topic of most of the design views in this document. Context, composition, logical, dependency, information, patterns use, interface interactions and state dynamics views will be explained in section 5 with their corresponding viewpoints. Some of these viewpoints will be clarified with their related UML diagrams.

4.5 Design viewpoints

In this document, context, composition, logical, dependency, information, patterns, interface, structure, interaction and state dynamics viewpoints will be specified with all aspects.

- ***Context Viewpoint:*** It describes interactions and all dependencies between the system and its environment such as users. It includes use cases and a context diagram showing the boundaries of the HBTS system.

- ***Composition Viewpoint:*** It describes how the design subjects are divided into its components and roles of those components in the system. These specifications can be used for estimating cost and scheduling concerns by the development team. Moreover, this section includes a deployment and a component diagram.

- ***Logical Viewpoint:*** It describes class structures, its objects and interactions between them. Also, it supports development and reuse of existing logical components. All aspects of class

dynamics, relationships between them and how they are implemented will be defined with class diagrams.

- ***Information Viewpoint:*** It describes how the system stores, manipulates, manages and distributes the data. It develops high-level view of static data structure and information flow. For instance, storing backdate maps, reviewing them, adding new data and users to the system will be clarified with the information viewpoint.

- ***Patterns Use Viewpoint:*** It describes design patterns and usage of design patterns which meet design ideas of the project.

- ***Interface Viewpoint:*** It describes the specifications of internal and external interfaces. Before the phase of detailed design, it gives detailed information to the development and tester team. How the system is used as a random user will also be described in this viewpoint

- ***Interaction Viewpoint:*** It describes the sequence of actions by defining how, when, where and at what level actions occur in the system. State dynamics views are preferred to be used in this project.

- ***State Dynamics Viewpoint:*** It describes dynamics of the system such as modes, states, transitions between those states and reactions of the system to different events. A state transition diagram will be used to clarify overall mechanism of the system.

4.6 Design elements

Design element means any element which has a role in design views. Design entities, design relationships, attributes and design constraints are all considered as one of those elements. All design elements are defined with its subcase under related topic defined in section 5 of the SDD document.

4.7 Design overlays

The project consists of two parts. Web Interface and Algorithm parts differ in design viewpoints and should not be confused with each other.

4.8 Design rationale

Human Body Tracking System aims to track human bodies in retail environments in order to generate a heat map which shows usage weights of different tracks on a retailer environment. In other words, system calculates places how much time their customers spend on different tracks and show this information on a heat map.

Design decisions of the system are made according to recommendations of “Adoniss Yazılım” who owns the end product. Also, similar design is preferred on the former products and this strengthened our motivation. Object oriented design with C++ programming language is preferred language to develop the Body Tracking part of the system. Moreover, in advance parts of image processing and machine learning applications OpenCV methods will be used in order to improve software maintainability, time-efficiency and providing faster development, lower cost of development and higher quality of software. Also, ASP.NET MVC 4 framework is used for implementing the web site part of the HBTS because of same concerns. The interface of the website should be understandable and user friendly. All design concerns will be handled in this direction.

Also, data storage is an important concern due to necessity of storing large amount of data. Thus a database design and usage will have an importance in this project.

4.9 Design Languages

Design viewpoint specifications of the system will be demonstrated with using Unified Modeling Language (UML).

5. Design viewpoints

5.1 Introduction

Design viewpoints determine the conventions for a system including the architectural models, languages and notations. They are used in the realization part of the design descriptions and constraints of the Human Body Tracking System. Several design viewpoints are defined in the following subsections. UML is used as the design language.

5.2 Context viewpoint

Context Viewpoint describes the abstract model of the system and the relationships, dependencies and interactions between the system and its environment.

5.2.1 Design Concerns

As shown in Figure 1, Human Body Tracking System consists of many components and it is must to work these systems in harmony to provide 90% accuracy. The flow is camera to database, after the data starts to come, firstly Body Tracking System interprets the data and sends write request to server. After writing the data to database, Web User Interface become ready to answer Users request's through Server. The following subsection explains the user's abilities in detail by using use case diagrams.

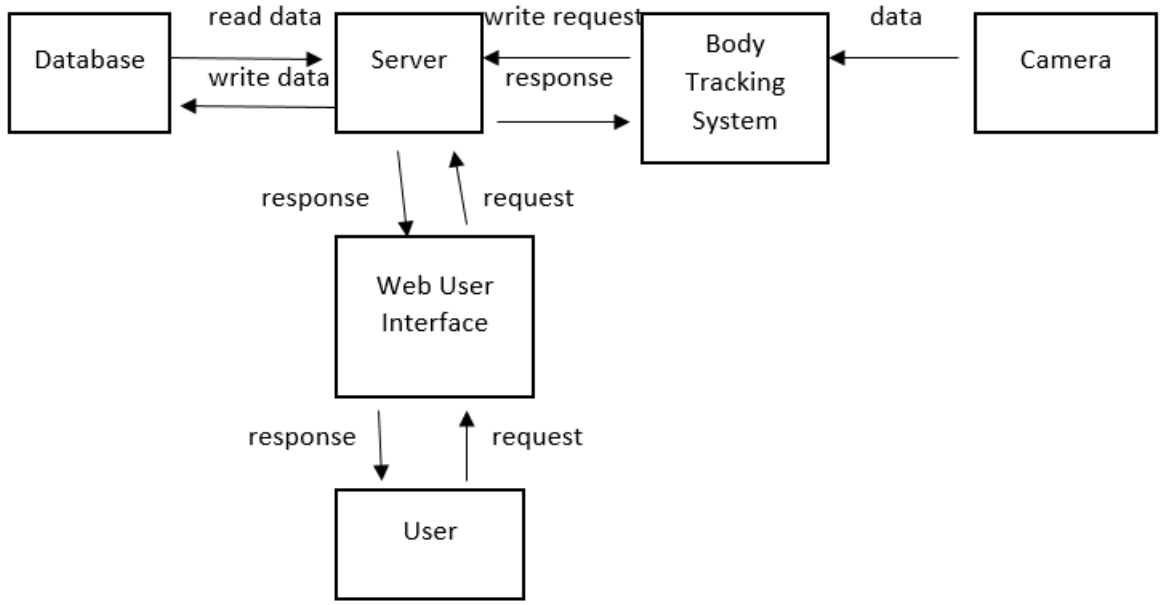


Figure 1 – Context Diagram

5.2.2 Design Elements

This system has 3 major actors who are Manager, Shop Manager and Admin. Each one of them has different functionalities on the system. So, the users id and password are checked when the user logins to the system to set the functionalities.

Figure 2 shows that Manager who is the manager of more than one store, can login to the system and choose one of the store from the drop down list. Also, after choosing one of them, he/she can display the Heat Map of the chosen store. Manager gets the same screen with the Shop Manager after choosing store.

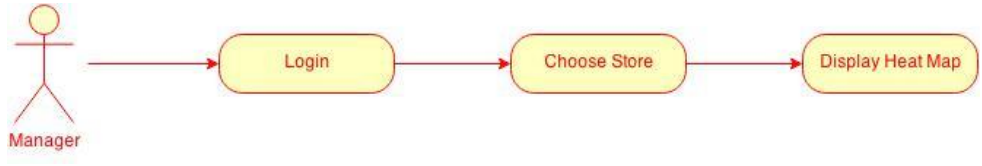


Figure 2 – Use Case Diagram of Manager

Figure 3 shows that Shop Manager who is the manager of just one store, can login and display the Heat Map of the store. By the way, he or she can visualize the customer's behaviors with the Heat Map and can change the sales strategy by changing the order of the shelf in the store.

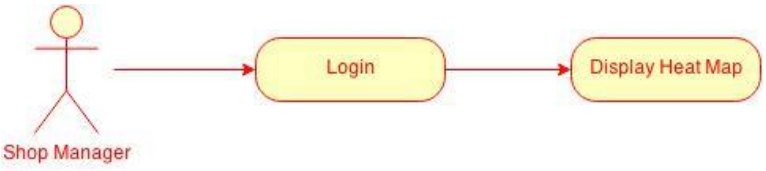


Figure 3 - Use Case Diagram of Store Manager

Figure 4 shows that Admin who is the developer of system or the contact person, has lots of functionality. These functionalities are adding, deleting editing brand and also adding, deleting, editing Manager after logging in to the system. These functionalities help Admin to maintain the system. To provide consistency, if the admin wants to add, edit or delete Brand Manager, he/she needs to add, edit or delete Brand. Brand Manager is the subset of Brand and Brand Manager is meaningless without the Brand.

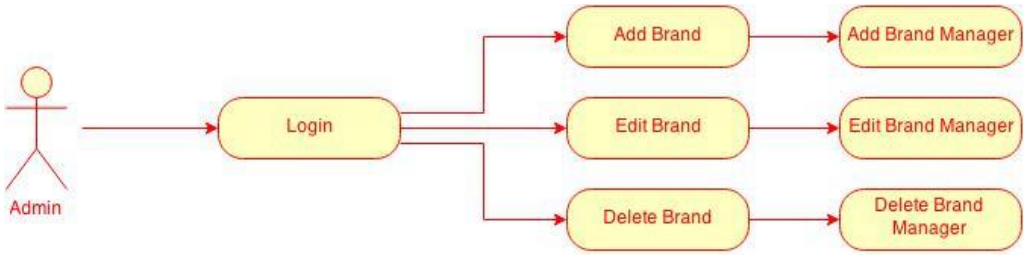


Figure 4 - Use Case Diagram of Admin

Moreover, Admin can add, delete and edit store information about the chosen brand as shown in Figure 5. If Admin wants to add, delete or edit one of the store, first he/she selects one of the brand which has this store and moreover, admin can add, edit or delete store manager after choosing one of the store.

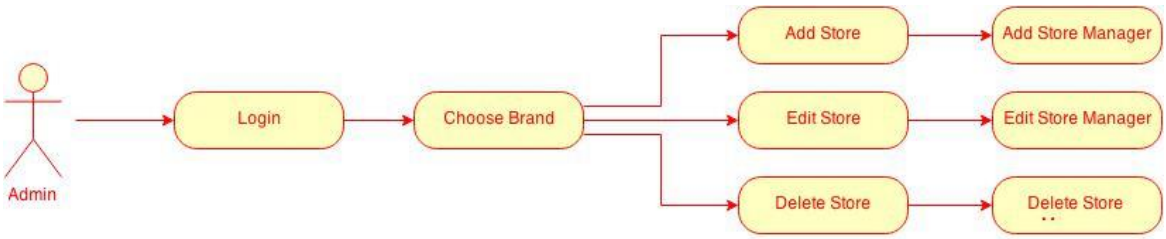


Figure 5 - Use Case Diagram of Admin

Also, Admin has right to display Heat Map of the store after choosing brand and store as shown in Figure 6. Admin needs to display the heat map, because if unexpected actions occur in the system, admin can interfere to the system and fix the erroneous part.

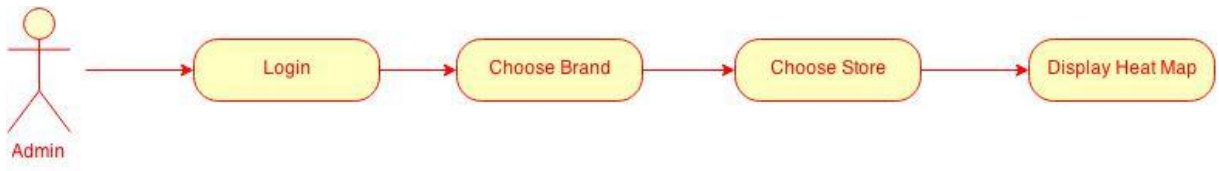


Figure 6 - Use Case Diagram of Admin

Figure 7 shows that all users can logout from the system.

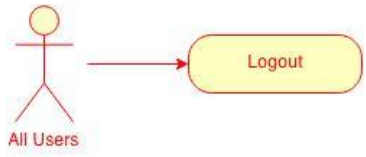


Figure 7 - Use Case Diagram of All Users

5.2.3 Example Languages

Use case diagrams showing user interactions with the system and a context diagram showing the system boundary have given in the previous sections by using UML modeling Language.

5.3 Composition viewpoint

Composition viewpoint is essential to clarify the relationship between the components.

5.3.1. Design Concerns

The project consists of two main parts. Therefore relationships between those parts and it's components are very important for estimating cost, staffing and schedule for the development effort.

In this section the physical and software components of the system will be described in detail. The connections and interactions between those components will be explained in order to provide better understanding of the project. User environment and external environment of the system is going to be explained.

5.3.2. Design Elements

In section 5.2, the system's interaction with the users is shown in a Context Diagram. Detailed explanations of the components will be given in this section.

As it was stated, the system mainly consists of two parts: Web and Algorithm parts. Abstract definitions of the components were also given in the section 5.2.

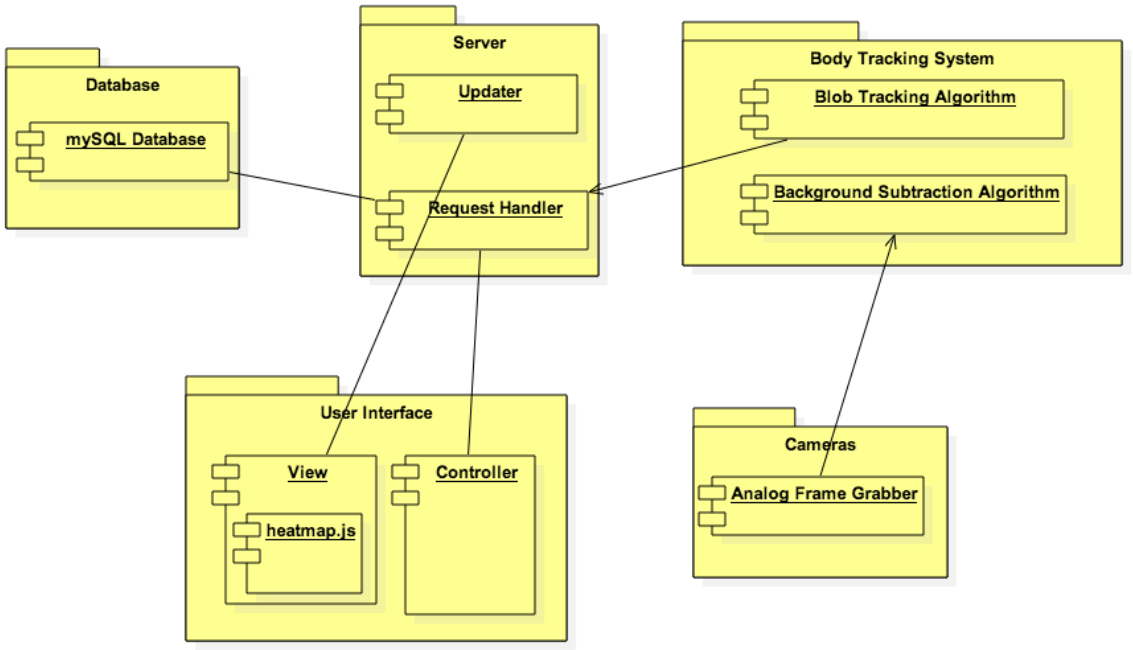


Figure 8 - Component Diagram

The components of the system are shown on the above component diagram. First cameras must be installed at the retail stores that will use the product. Analog cameras will be used for the system. An analog frame grabber component will capture the video and then send it to the Body Tracking System for further processing. Body Tracking System includes algorithms that are used in the system. Background Subtraction Algorithm and Blob Tracking Algorithm components will process the video and produce some meaningful data. This data will be written to a database through a server. Server is also making the connection between database and the user interface.

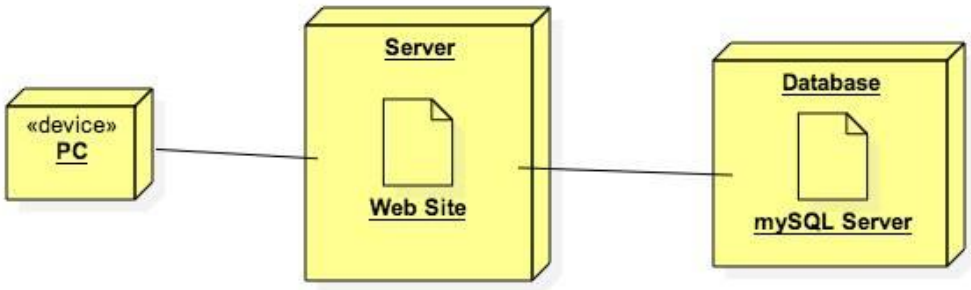


Figure 9 - Deployment Diagram

The deployment diagram above also shows the deployment logic of the web interface. The database will store the information about the blob objects that was tracked by the algorithm. The web-site which is running on a server will have a connection with the database to use the related data. Users of the system will use the web-site through their PCs by connecting to the server.

5.3.2.1. Functional & Subordinates Attributes

Cameras

Functional Attribute: Cameras can be considered as the data collectors of the system. They must be installed in the system environment. The cameras in the system are analog cameras.

Subordinates Attribute: Camera entity consists of a Analog Frame Grabber. Analog Frame Grabber component is accepting and processing video signals. It also contains an analog-to-digital converter in itself. After this conversion the video is sent to the Body Tracking System to be processed.

Body Tracking System

Functional Attribute: Body Tracking System entity is the part of the system where image processing is implemented on the captured video. All the algorithms are applied in this part of the system.

Subordinates Attribute: Body Tracking System consists two components. First one is Background Subtraction Algorithm component. In this component Mixture of Gaussian algorithm is implemented on the video to detect the moving objects in the video. The second part is Blob Tracking Algorithm component. The result of the Background Subtraction Algorithm will be processed in Blob Tracking Algorithm component to retrieve meaningful information about the moving objects on the video. The results of this component will be sent to the database.

Server

Functional Attribute: Server entity is making the connection between other entities of the system. It is responsible for the interaction between multiple components. It will take the data that come from Body Tracking System entity and write that to the database. Also the interaction between database and the web user interface will be done via this entity.

Subordinates Attribute: Server entity consists of two components. Request Handler component is the main component which handles the interaction between Body Tracking System, Database and the User Interface. Updater component is for the User Interface as it is giving the User Interface the ability to check changes in the system. Updater component will notify the User Interface.

Database

Functional Attribute: Database stores the information that comes from the Body Tracking System entity. It also includes information about brands, stores, brand managers and store managers. **Subordinates Attribute:** Database used in the system is a MySQL database.

User Interface

Functional Attribute: User Interface entity is the front end of the application and it is the part where users will interact with the system via a web interface.

Subordinates Attribute: As it was already stated in the previous parts of this report, MVC is the design pattern for the web interface. Therefore User Interface entity consists of View and Controller components. Controller component is connected to the Server via Request Handler component. View component is the part where users can display the system. View component also contains heatmap.js component which is a JavaScript library responsible for drawing the heat map with the related data.

5.4 Logical Viewpoint

The aim of this section is to give detailed information about classes, objects and relationships between them.

5.4.1 Design Concerns

The logical viewpoint is employed to show the development and reuse of abstractions and their implementations. This means, object oriented programming simplifies to maintain and modify existing code while new objects are created. Since identifying object classes is often difficult part of object oriented design, during the implementation phase of the project there can be some changes in object identification.

5.4.2 Design Elements

We have two class diagrams. One is for the web side and the other one is for the algorithm side.

5.4.2.1 Design Elements of Web

Web part of this project has 5 classes named as Store, Brand, Admin, Manager and User, and one interface named as Human Body Tracking System as shown in Figure 10.

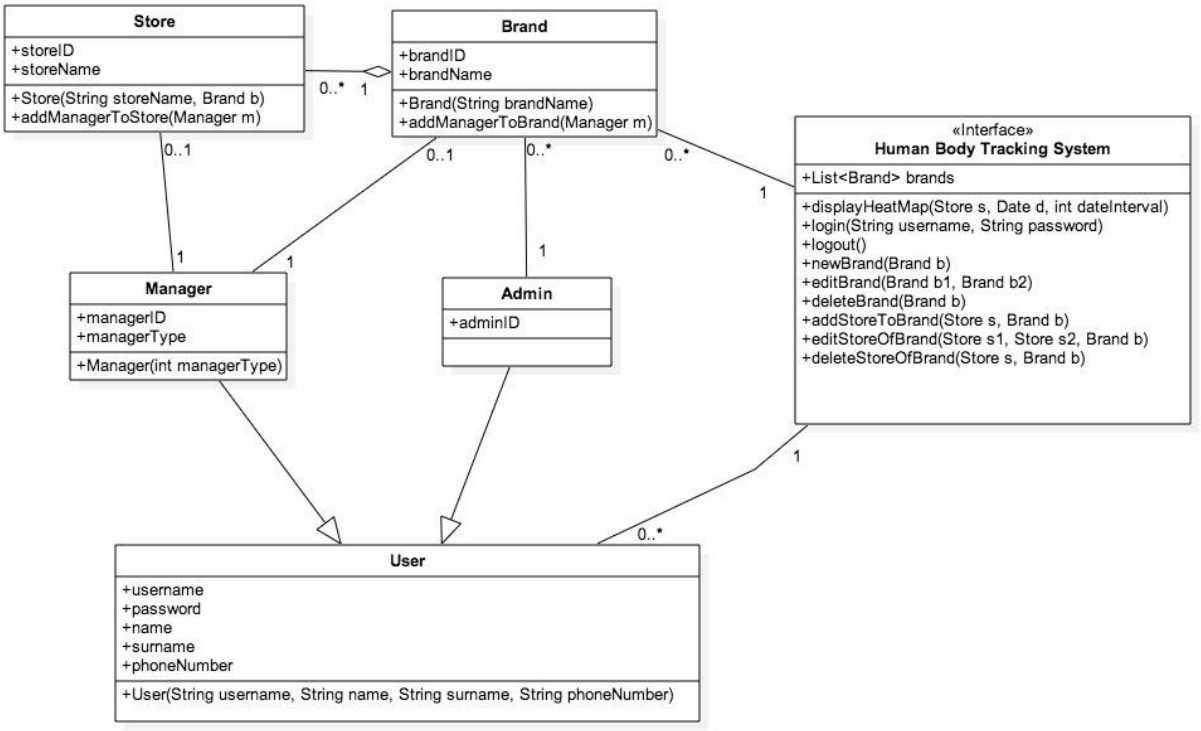


Figure 10 - Class Diagram of Web Side

5.4.2.1.1 User Class

This class has generic entities and each user in the system needs to enter these information before signing up to the system. Design attributes and functions are below table with their names, types, visibilities and definitions.

Name	**Type/Return value type**	**Visibility**	**Definition**
Username	string	Public	This variable keeps the user name of a user.
Password	string	Public	This variable keeps the password of a user.
Name	string	Public	This variable keeps the name of a user.
Surname	string	Public	This variable keeps the surname of a user.
phoneNumber	string	Public	This variable keeps the phone number of a

			user.
User()	User	public	This function is the constructor of a User Class.

5.4.2.1.2 Admin Class

This system may have more than one admin and it is necessary to hold their id to separate each one. Also, Admin has aggregation relationship with the User. Design attributes and functions are below table with their names, types, visibilities and definitions.

Name	**Type/Return value type**	**Visibility**	**Definition**
adminID	int	public	This variable keeps unique id of a specific admin user.

5.4.2.1.3 Manager Class

This system has two type managers, one of them store manager and the other one is manager. That's why it is necessary to hold these users with their own id and type information. Also, Manager has aggregation relationship with the User. Design attributes and functions are below table with their names, types, visibilities and definitions.

Name	**Type/Return value type**	**Visibility**	**Definition**
managerID	int	Public	This variable keeps the id of a manager.
managerType	enum	Public	This variable keeps the type of a manager.
Manager()	Manager	Public	This function is the constructor of a User Class.

5.4.2.1.4 Store Class

This system is expected to load a lot of stores. All stores have its own information and has one to many relationships with the Manager and also has a weak entity relationship with the Brand. Design attributes and functions are below table with their names, types, visibilities and definitions.

Name	**Type/Return value type**	**Visibility**	**Definition**
storeID	int	Public	This variable keeps the id of a store.
storeName	string	Public	This variable keeps the name of a store.
Store()	Store	Public	This function is the constructor of a Store Class.
addManagerToStore()	Manager	Public	This function adds Manager to the Store Class.

5.4.2.1.5 Brand Store

This system is expected to load a lot of brands. All brands have its own information and have one to many relationship with Manager, Admin Class and Human Body Tracking System Interface. Also, it has a weak entity relationship with the Store. Design attributes and functions are below table with their names, types, visibilities and definitions.

Name	**Type/Return value type**	**Visibility**	**Definition**
brandID	int	Public	This variable keeps the id of a brand.
brandName	string	Public	This variable keeps the name of a brand.
Brand()	Brand	Public	This function is the constructor of a Brand Class.
addManagerToBrand	Manager	Public	This function adds

			Manager to the Brand Class.

5.4.2.1.6 Human Body Tracking System Interface

The system is expected to get all functionalities to the interface. Also, it has one to many relationship with the Brand and User. Design attributes and functions are below table with their names, types, visibilities and definitions.

Name	**Type/Return value type**	**Visibility**	**Definition**
brands	List<Brand>	Public	This variable holds the brands.
displayHeatMap(Store s, Date d, int dateinterval)	Void	Public	This function shows heat map the store with the date and date interval information.
login(String username, String password)	Void	Public	This function logins the user to the system with the username and password information.
logout()	Void	Public	This function logouts the user from the system
newBrand(Brand b)	Void	Public	This function adds new Brand to the system.
editBrand(Brand b1, Brand b2)	Void	Public	This function edits the brand in the system.
deleteBrand(Brand b)	Void	Public	This function deletes the brand.

addStoreToBrand(Store s, Brand B)	Void	Public	This function adds store to a brand.
editStoreOfBrand(Store s1,Store s2, Brand b)	Void	Public	This function edits the store information of a brand.
deleteStoreOfBrand(Store s, Brand b)	Void	Public	This function deletes the stores of a brand.

5.4.2.2 Design Elements of Algorithm Side

This project has 3 classes in algorithm side name as CV_EXPORTS_W, CV_WRAP and Blob as shown in Figure 11.

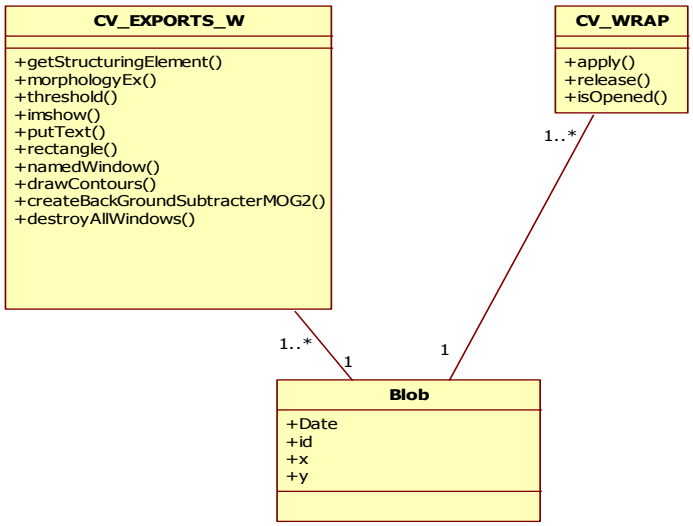


Figure 11 - Class Diagram of Algorithm side

5.4.2.2.1 CV_EXPORTS_W Class

The functions which belong to CV_EXPORTS_W class provide lots of facilities to the project in terms of applying and creating back ground subtraction, deleting noises and blobbing the objects in the frame. Design attributes and functions are below table with their names, types, visibilities and definitions.

Name	**Type/Return value type**	**Visibility**	**Definition**
getStructuringElement()	Mat	Public	This function

			creates the kernel to apply a frame.
morphologyEx()	OutputArray	Public	This function applies kernel to a frame.
threshold()	OutputArray	Public	This function applies threshold value to a frame.
imshow()	void	Public	This function visualizes result frame to a user.
putText()	void	Public	This function puts a frame number to a screen.
rectangle()	void	Public	This function rectangles to detected object.
namedWindow()	void	Public	This function creates window with a specific name.
drawContours()	void	Public	This function draws contours to detected objects.
createBackGroundSubtracterMOG2()	Ptr<BackGroundSubtractorMOG2>	Public	This function creates back ground subtractor.
destroyAllWindows()	void	Public	This function destroys all

			windows created by namedWindow( ).

5.4.2.2.2 CV_WRAP

The functions which belong to CV_WRAP provide some methods. Design attributes and functions are below table with their names, types, visibilities and definitions.

Name	**Type/Return value type**	**Visibility**	**Definition**
apply()	void	Public	This function applies MOG2 function to a frame.
release()	void	Public	This function releases the captured video.
isOpened()	bool	Public	This function checks whether the video is opened.

5.4.2.2.3 Blob Class

The fields in this class are created for each frame by using the other class functions and they put into Blob vectors. After the blobbed objects exits the frame, the vector is sent to the database as JSON object. Design attributes and functions are below table with their names, types, visibilities and definitions.

Name	**Type/Return value type**	**Visibility**	**Definition**
Date	String	Public	This variable keeps the date values.
id	int	Public	This variable keeps the id of each blobbed objects.

x	int	Public	This variable keeps the x coordinate of blob.
y	int	Public	This variable keeps the y coordinate of blob.

5.5 Dependency viewpoint

It was explained in detail at the Composition viewpoint.

5.6 Information viewpoint

The purpose of this section is to give information about persistent data content of the system. This viewpoint develops a high-level view of static data structure.

5.6.1 Design Concerns

In this project, persistent data storage is very important due to necessity of keeping brands, their stores, managers and people's tracked positions in the stores. Most time consuming part of the database design of HBTS is 'Blob' table because a blob may have so many positions in a store hence this table would grow so much in time. Due to this problem, positions of a blob will kept as JSON object in corresponding database field. With the help of JSON data, every tracked person in a store will occupy only one database record.

5.6.2 Design Elements

There are six data entities; 'User', 'Brand Manager', 'Store Manager', 'Brand', 'Store' and 'Blob'. Below entity-relationship diagram shows these entities, their attributes and the relationships between these entities.

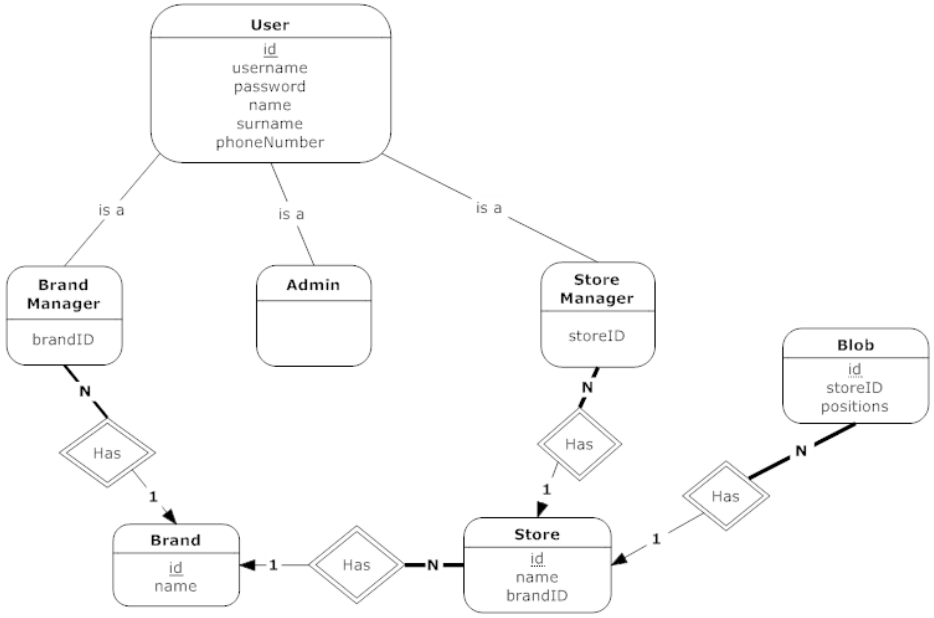


Figure 12 - Entity Relationship Diagram

'User' entity represents all types of users who are able to use the software product. 'Brand Manager', 'Admin' and 'Store Manager' entities are sub types of this entity and all attributes of 'User' super type are actually attributes of its sub types.

- id: This attribute is unique identification number for the user. It is the primary key of corresponding table.
- username: The e-mail address that is used for registering.
- password: It represents password of a user. It is encrypted in the database.
- name: It represents name of a user.
- surname: It represents surname of a user.
- phoneNumber: It represents phone number of a user.

'Brand Manager' entity stands for a manager that is responsible for a brand.

- brandID: It represents the unique identification number of the brand which brand manager is responsible for. It is the foreign key in the corresponding table.

'Store Manager' entity stands for a manager that is responsible for a store.

- storeID: It represents the unique identification number of the store which store manager is responsible for. It is the foreign key in the corresponding table.

'Brand' entity represents a brand registered in the system.

- id: This attribute is unique identification number for a brand. It is primary key of corresponding table.
- name: It represents the name of a brand.

'Store' entity represents a store registered in the system.

- id: This attribute is unique identification number for a store. They form primary key of relationship between 'Brand' and 'Store' with 'id' in 'Brand'.
- name: It represents the name of a brand.
- brandID: It represents the unique identification number of the brand which store is belong to. It is the foreign key in the corresponding table.

'Blob' entity stands for a person in a store that is tracked by the system.

- id: This attribute is unique identification number for a blob. They form primary key of relationship between 'Brand', 'Store' and 'Blob' with 'id' in 'Store' and 'id' in 'Brand'.
- storeID: It represents the unique identification number of the store which blob is tracked in. It is the foreign key in the corresponding table.
- positions: It is the text field of JSON object that holds tracked positions of a blob.

There are one-to-many relationships between 'Brand' & 'Brand Manager', 'Store' & 'Store Manager', 'Blob' & 'Store' and 'Brand' & 'Store'. In addition, 'Store Manager and 'Blob' are weak entities of 'Store' and 'Store' and 'Brand Manager' are weak entities of 'Brand'. This means, instances of these entities do not make sense on their own.

5.7 Pattern use viewpoint

In this project, model-view-controller (MVC) is used as architectural pattern for implementing web side. Although MVC is developed originally for desktop applications, it has been widely used as web application architecture. It separates presentation and interaction from the system data.

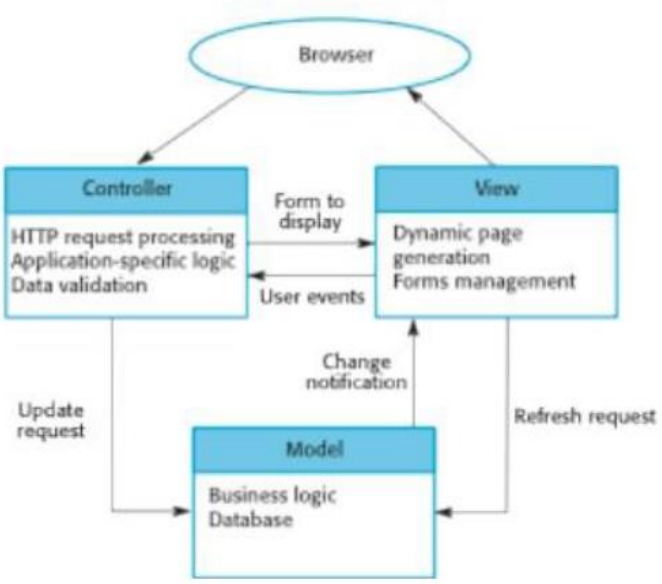


Figure 13 - Architecture of web based application that uses MVC pattern

- Model component manages the system data and associated operations on that data. It handles the state of the application.
- View component defines and manages how the data is presented to the user. It renders model.
- Controller component manages user interaction and passes these interactions to the View and the Model.

It allows the data to change independently of its representation or vice versa. Moreover, because there are multiple ways to view and

interact with data in this project, MVC architecture is convenient. Figure 13 shows architecture of web-based application prepared using MVC architecture. In this project, model contains objects representing users, brands, stores and blobs. The views are the templates for rendering login page, heat map page etc. Controller receives user requests from view and translates them into actions which the model should take. Then it selects relevant view to show the response.

5.8 Interface viewpoint

This section includes detailed information about external and internal interfaces of the software product.

5.8.1 Design concerns

Interface Viewpoint is used to indicate the relationship between libraries, tools and interfaces. Figure 14 shows the main components that appear in functional flow of HBTS. Body tracking system continuously operates on frames that come from cameras in the field. At this point, Body Tracking System frequently uses OpenCV library for background subtraction and tracking the blobs. Then, blobs and their positions (as JSON) that come from Body Tracking System are written to the MySQL database in order to be later displayed in user interface of HBTS web application. This user interface is formed by the 'view' component of the MVC framework. When users request to display heat map, Heatmap JS library generates heat maps to the user interface from corresponding positions of blobs. Below functions and objects of Heatmap JS library will be integrated into the 'view' component and frequently used during heat map generation.

- *h337* : It is the name of the global object registered by heatmap.js. It will be used to create heatmap instances.
- *heatmapInstance* : It is the object returned by h337.create(). A heatmap instance has its own internal data store and renderer where data can be manipulated. As a result the heatmap gets updated.
- *h337.create(configObject)* : It is used for creating heatmap instances. A heatmap can be customized with the configObject. configObject includes features like background color, gradient, radius of points, opacity and blur.
- *heatmapInstance.setData(object)* : It initializes a heatmap instance with a dataset. "min", "max", and "data" properties are required for the object. "min" and "max" determine the density of the point. "data" includes array of points (with x and y

coordinates in pixel space). setData removes all previously existing points from the heatmap instance and re-initializes the data store.

In addition, for CRUD operations of brands, stores and managers, web application reads/writes related data from/to MySQL database.

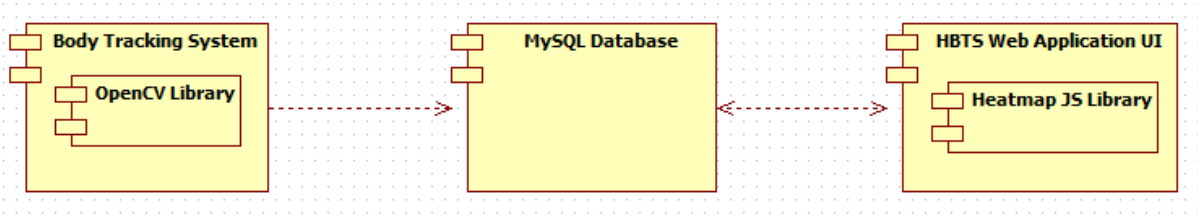


Figure 14 - Component Diagram

5.8.2 Design elements

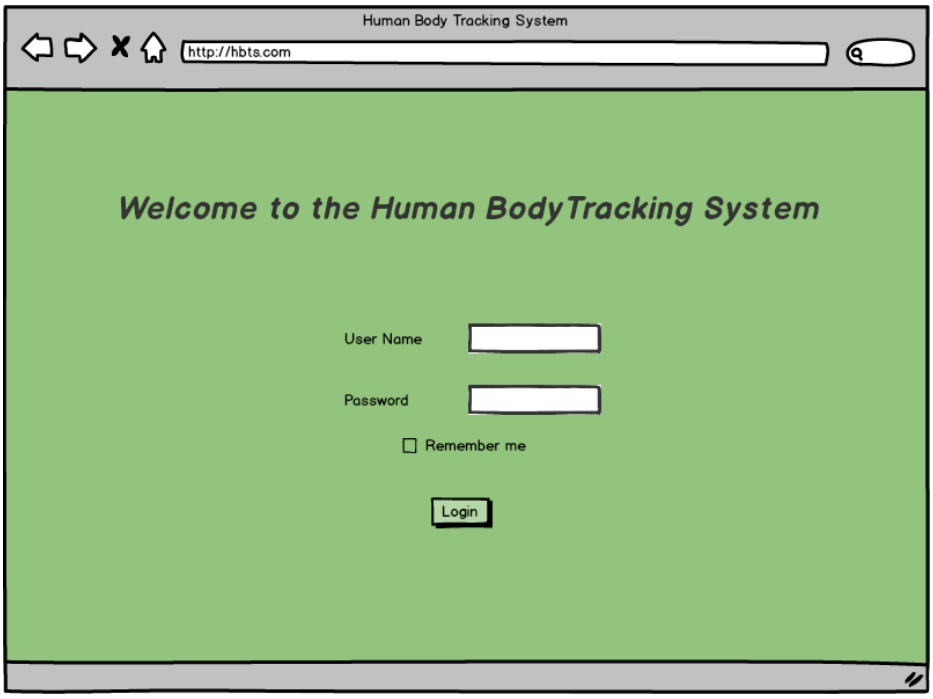


Figure 15 – Login Page

Figure 15 shows login page of the system and all kinds of users must login first so that they can use the functionalities of the system. User name should be entered as e-mail address that is registered to the database. If the 'Remember me' option is checked by the user, username and password information are remembered by the browser for the next time. After user clicks the 'Login' button, he/she is directed to the home page if entered username and password pair exist in the database.

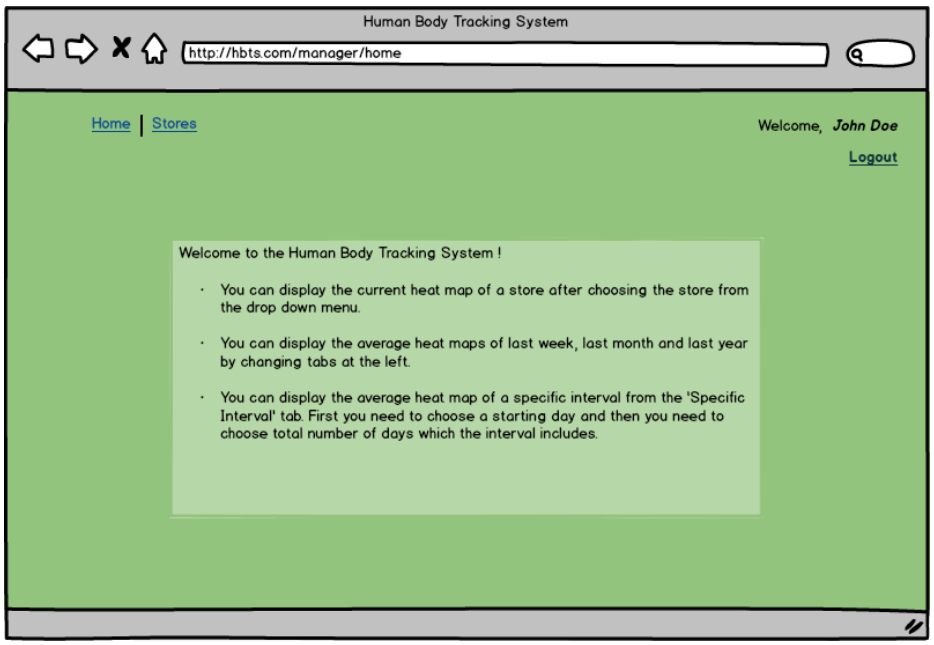


Figure 16 – Home page of Manager

Above figure shows the home page of brand manager and it includes some instructions about usage of the system. At the top right corner of the page, there is a welcome message with the name and surname of the manager. Manager can logout from the system by clicking 'Logout' that is under the welcome message. These are applicable for all pages of web user interface. When 'Home' is chosen from the navigation bar, user is directed to this page. The other option 'Stores' from the navigation bar provides to display heat maps of stores that available to the brand manager.

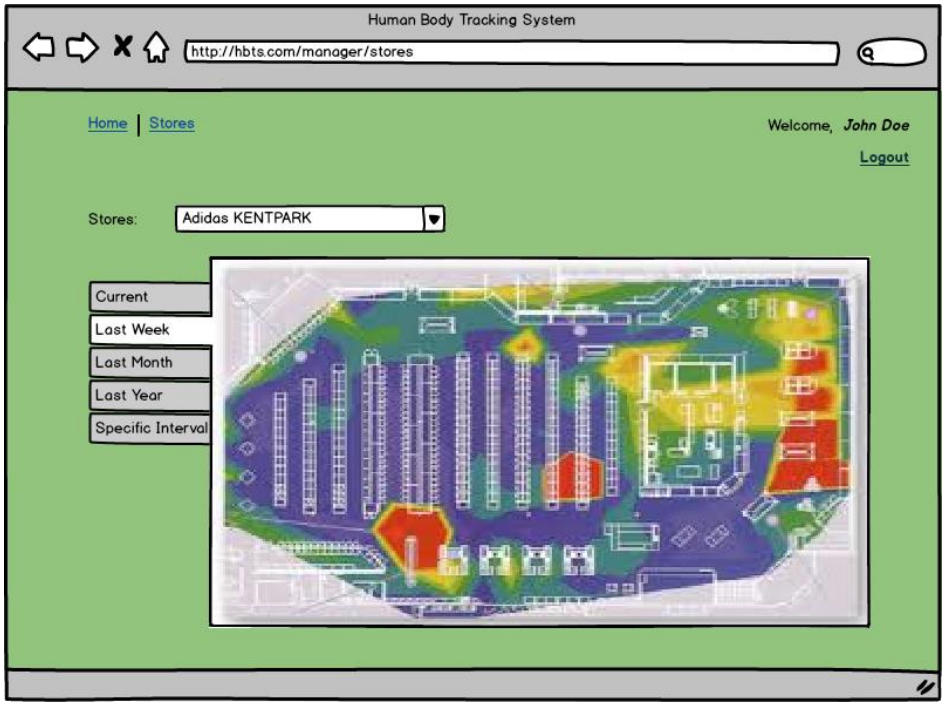


Figure 17 – Stores page of Brand Manager when 'Last Week' tab is active

First of all, brand manager must choose a store from drop down list in order to display available heat maps of that store. At the moment he/she chose the store, current heat map is displayed in the middle of the page. User can display different heat maps corresponding to different time intervals by clicking tabs at the left. As mentioned before, current heat map of the store is displayed by default. Figure 17 shows the heat map when 'Last Week' tab is active. While 'Current', 'Last Week', 'Last Month' and 'Last Year' tabs directly open heat map of that time interval, user should enter two information first in 'Specific Interval'. Below figure shows corresponding page when 'Specific Interval' tab is active.

Figure 18 - Stores page of Store Manager when 'Specific Interval' tab is active

User should first choose the beginning date of the interval from the date time picker and then enter the total number of days that the interval includes. After these two operations, when the 'Display' button is clicked, average heat map that encompasses whole interval appears. For example, heat map in the Figure 18 shows average heat map of an interval that begins with 15.09.2014 and ends with 30/09/2014.

Figure 19 - Heat map page of Store Manager

If a store manager logins to the system, a home page which is almost same with page at Figure 16 welcomes the user. There is 'Heat Map' option in the navigation bar instead of 'Stores' and also home page instructions are prepared according to capabilities of this user type. When the user clicks the 'Heat Map', same tabs of manager are appeared on the left. The only difference is that there is not any drop down list for selection of store.

Home page of admin includes some statistical data like total number of brands and stores. Navigation bar contains 'Home', 'Brands', 'Stores', 'Heat Map' and 'New' options. Brands page of admin is displayed at Figure 20. All brands in the system are listed in a table and pagination can be used to navigate between pages of table. The table includes the name of the brand, the number of stores in the system that the brand has and name of the manager of the brand. Admin can delete or can update a brand by clicking corresponding buttons in the last column.

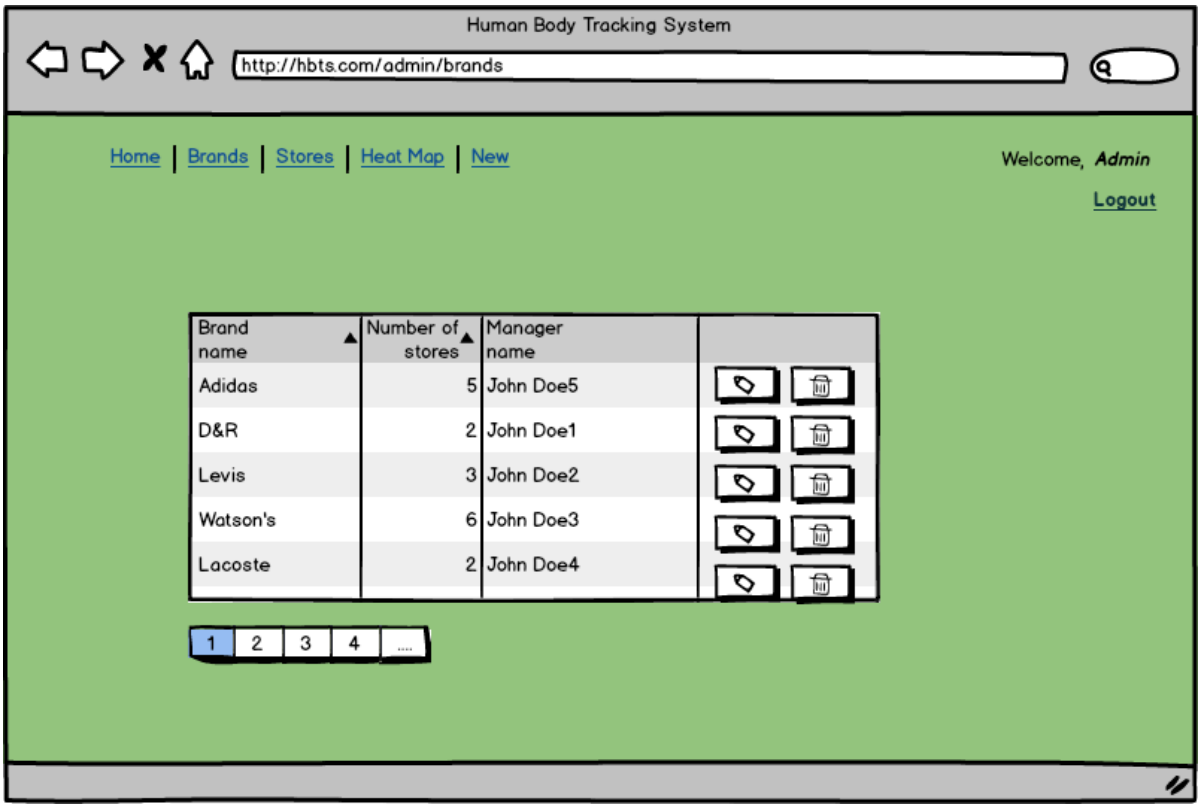


Figure 20 - The brand page of admin

In addition, admin can click items in the number of stores and manager name columns. If an item from manager name column is clicked, detailed information about that manager is displayed like in Figure 21. This figure shows the case that 'John Doe5' is clicked from the Figure 20.

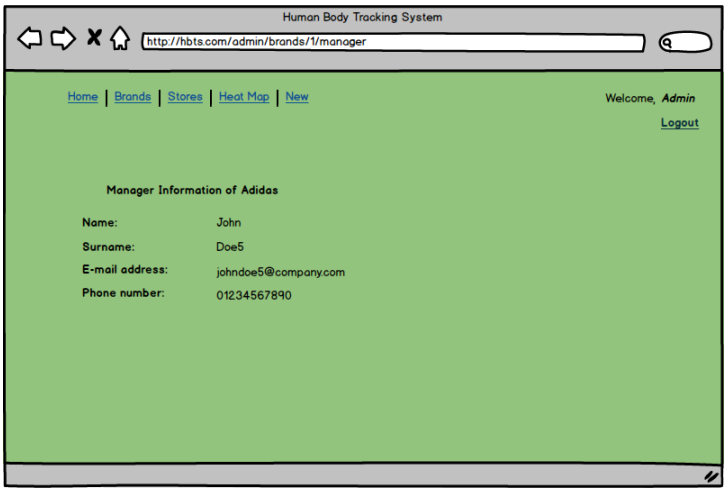


Figure 21 - Manager details page of admin

If an item from number of stores is clicked, new page is appeared containing a table of stores. For example, if '5' is clicked from the first row in Figure 6, below page is opened. It includes all stores of selected brand and manager names of these stores. Admin can delete or edit a store from options in the last column of table. In addition detailed information about store manager can be displayed by clicking the name of store manager like Figure 21. In addition, admin can display, delete or edit stores after clicking 'Stores' tab at the main menu.

'Stores' tab displays a table like in Figure 22 but this table list all stores in the system not only stores of one brand.

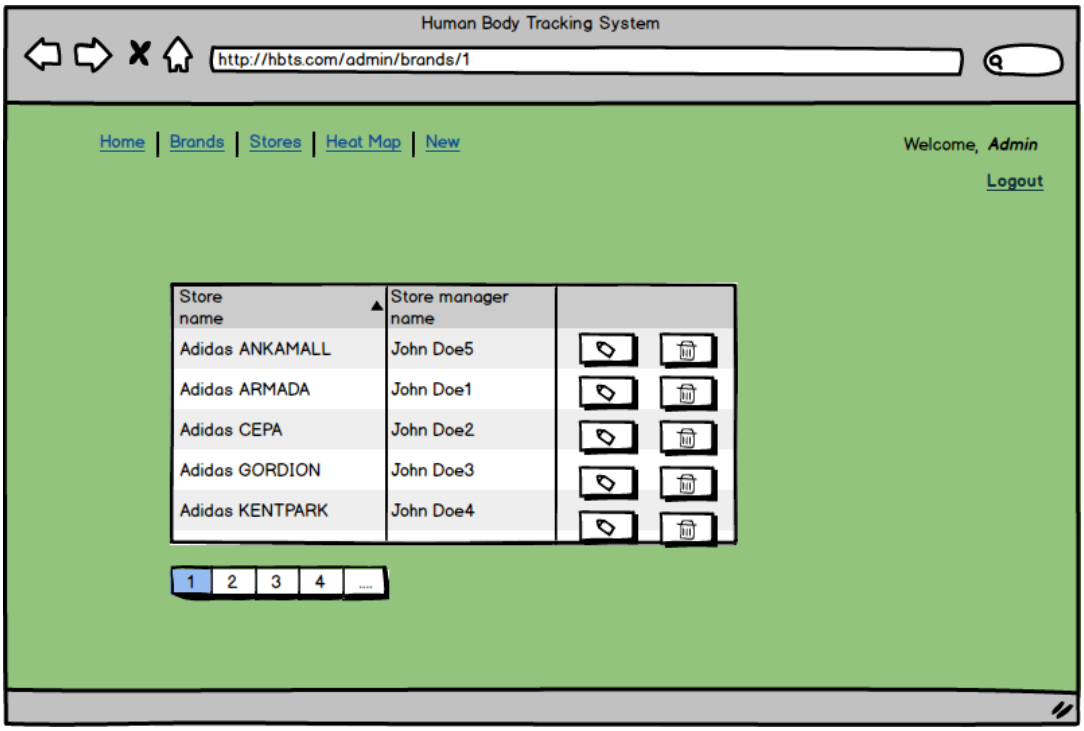


Figure 22 - Stores page of admin

'New' option in the navigation bar can be expanded like in the Figure 23. Admin can add new brand and a new store from options in the figure. Because pages of these options seem like each other, only new brand page will be explained in this document.

Figure 23 - New option in the navigation bar

When Admin clicks 'Brand' from these options, below page is displayed. There is one text box for the name of new brand. The other text boxes are for information about manager that is responsible for this new brand. Name, surname, e-mail address and phone number of the manager must be entered to text boxes respectively. After admin clicks the 'Add' button, system generates password for the new user and information is saved to the database. This newly generated password is send to the user via e-mail.

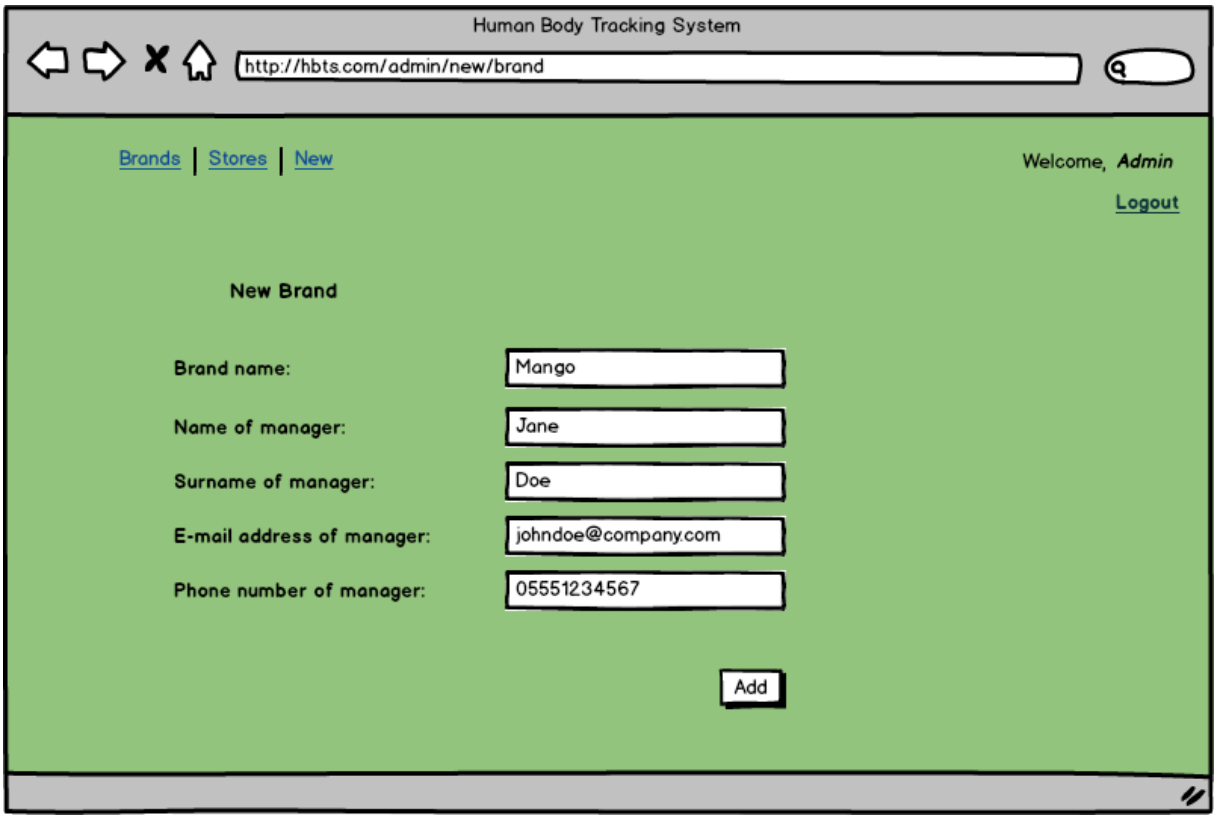


Figure 24 - New brand page of admin

Admin can display heat maps of all stores in the system from 'Heat Map' option in the navigation bar. Figure 25 shows current design of developer team for this page. First drop down list is for selection of brand. After this selection, admin choose a store from the second drop down list that contain narrowed stores list. Then he/she should choose the time interval for displaying heat map from the interval tabs. Design in Figure 25 may change through time.

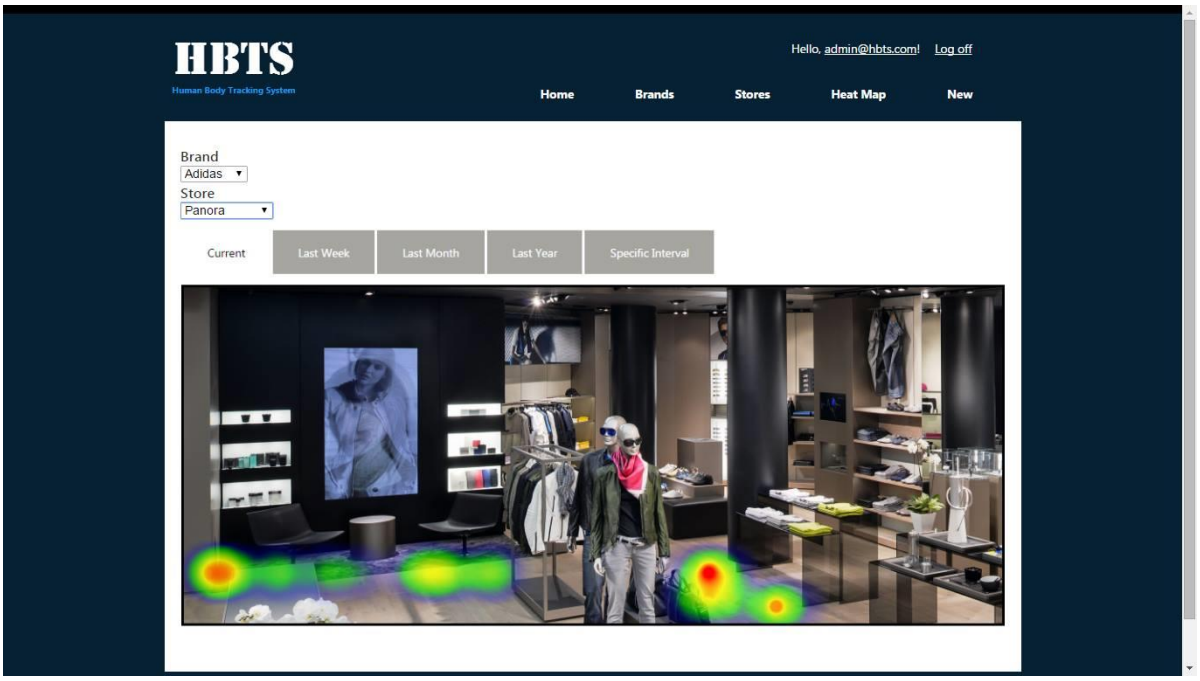


Figure 25 - Current Design of Web Application

System will give appropriate errors and warnings according to user interactions. When a text box is left empty, a red error message *'This field cannot be left empty*' will be appeared at right of the text box. When admin is adding a new brand or store, if duplicate value occurs, an error message '*There is already a brand with this name!*' will be appeared on the screen. In addition, while a brand or a store is deleting, a warning message '*Are you sure about deleting this brand*?' will be appeared.

5.9 Structure viewpoint

It was explained in detail at the Composition viewpoint.

5.10 Interaction viewpoint

This section will provide information about the interactions between the entities of the system. Sequence diagram will be used to explain the functionality of the system.

5.10.1. Design Concerns

The flow of the operations with a user in the system is shown in this viewpoint. Responsibilities of entities and their interactions with the user is going to illustrated in this section to describe the functionality of the system.

5.10.2. Design Elements

In this viewpoint, two kinds of interaction will be illustrated. First one is the interaction between the entities of the system while processing the video and second one is the interaction between the entities when user is requesting to display information from the web-interface.

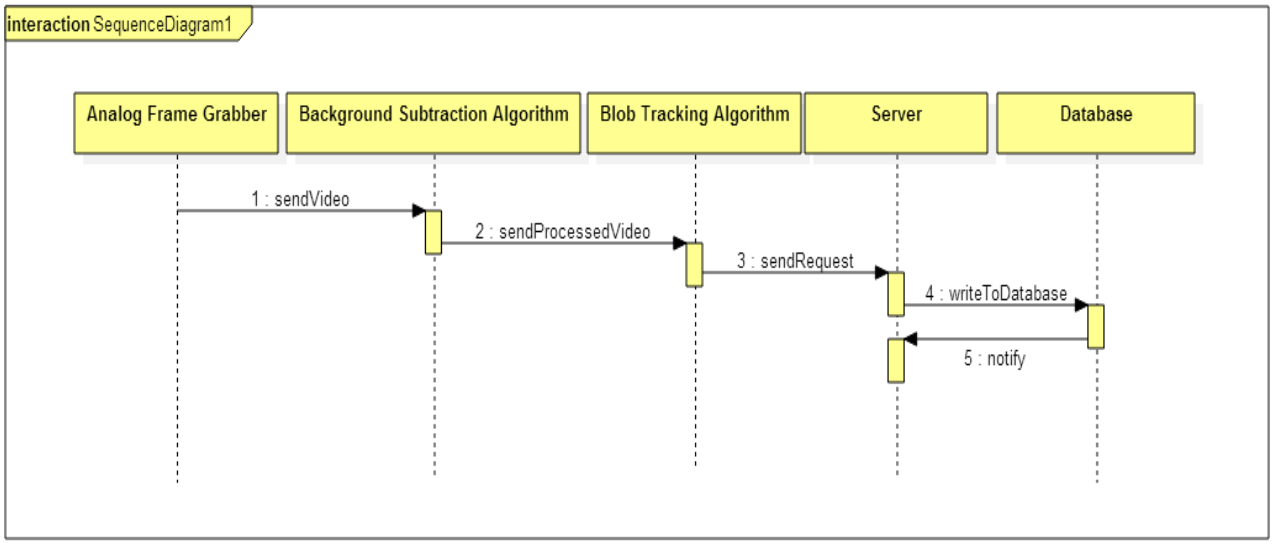


Figure 26 - Sequence Diagram 1

In the above Sequence Diagram, the interactions between Camera, Body Tracking System, Server and Database entities are shown. The flow of this operations starts with the Analog Frame Grabber sending the video to Body Tracking System to be processed. In Body Tracking System entity, first Background Subtraction algorithm is implemented on the video and its results are sent to Blob Tracking component. After Blob Tracking algorithm finishes its job and produce the data needed for the system, it sends a request to the server to write this data to the database. Server delivers this data and writes it to the database. At last database notifies the server that the operation was successful.

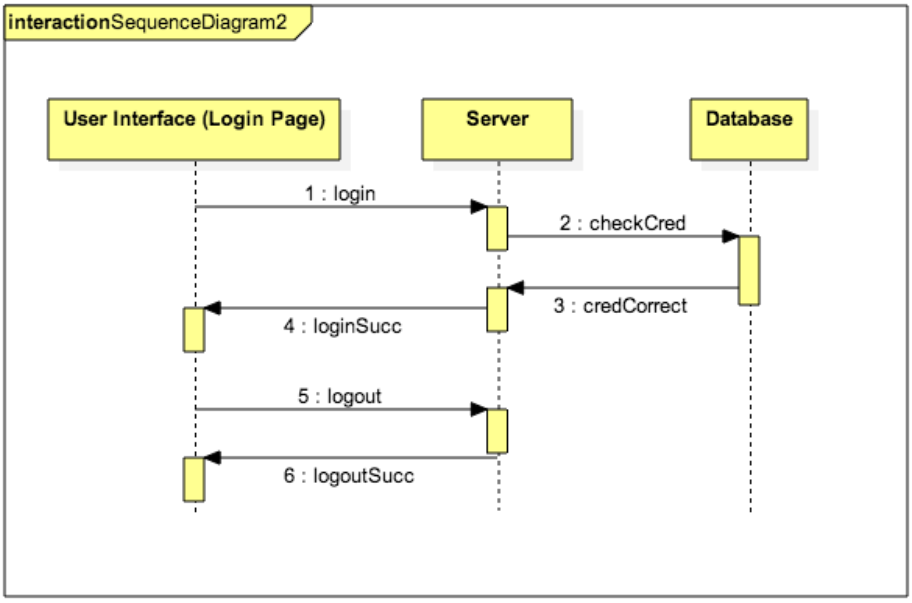


Figure 27 - Sequence Diagram 2

Sequence Diagram 2 is showing the interaction between the users and the system for the login and logout use cases. When a user wants to login to the system, a request is sent from the User Interface to the Server. Server connects to the database and checks the credentials of the user that wants to log in with a query. If the response from the database is affirmative then the server notifies the User Interface that the log in operation is successful. Logout operation does not need any database connection and just notifying the server that a user of the system is logging out is enough for the interactions between components.

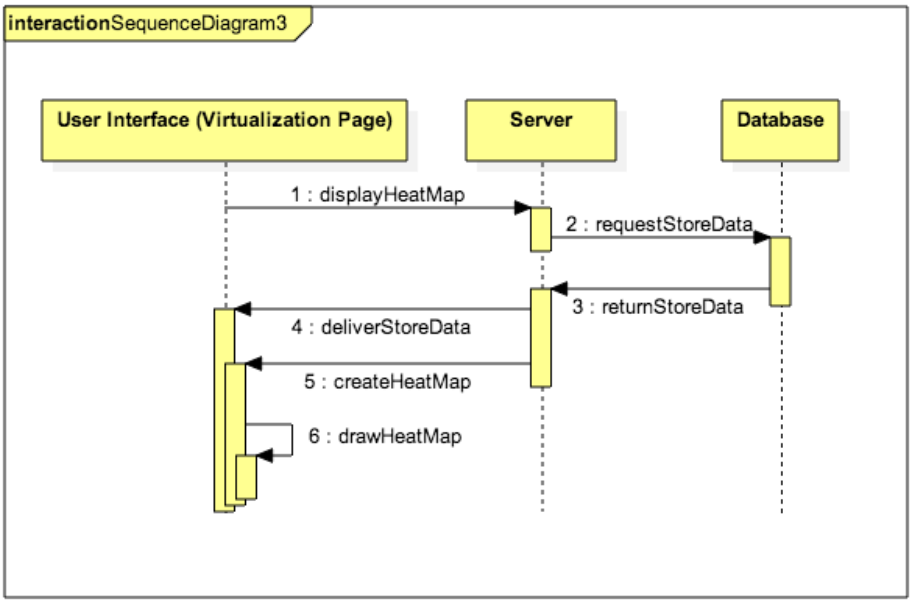


Figure 28 - Sequence Diagram 3

Above Figure 28 is showing the interactions when a user wants to display the heat map of a store. When display button is pressed in the user interface, a message is sent to the server to retrieve the

data from the database. The server transmits that message the database and database responds with the appropriate data. The server delivers the data to the user interface and also delivers a message which acts as a trigger for starting to create the heat map with the data to make it meaningful for the users. heatmap.js library in the User Interface draws the heat map as the last step of the interaction.

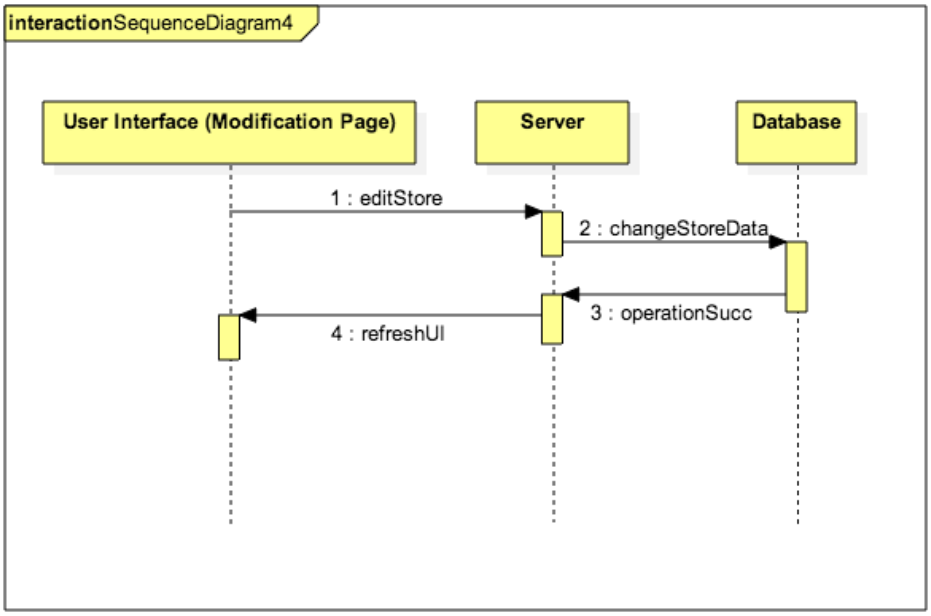


Figure 29 - Sequence Diagram 4

Figure 29 is representing the interactions when the admins of the system is making a modification of the data of the system. editStore message between User Interface and Server is representing all add, edit, delete operations that an admin is able to do. Then the server is transmitting a message to the database to make the change that is wanted by the user in the data. After the modification in the database is made, database sends an operation successful message to the server and the server notifies the user interface to refresh the related information.

5.11 State dynamics viewpoint

In this section, system behavior and states of the system are given by the help of state transition diagram.

5.11.1 Design concerns

State dynamics viewpoint inform designers, developers and testers about dynamic view of the system which includes states, transitions and reactions to events.

As it was stated before, the system mainly consists of two parts. The algorithm part of the system does not contain any dynamic or reactive events. The static behavior of the Algorithm part is explained in detail in section 5.3 and 5.10 therefore a state diagram will not be given here.

The Web Interface consists user interaction with the system, therefore the reaction of the system varies due to the users' requests and a state transition diagram of the Web Interface will be given in this section to illustrate the behavior of the system according to those requests.

5.11.2. Design Elements

In order to use the system, users must first login. After login operation and the comparison of the users' credentials, pages will be displayed to the users according to their user type. All user information stored has also an attribute about which user type that user is. The checking of which user type does the user belong is done in the back end of the system. The decision box (check userType) in the State Transition Diagram represents this. Admins of the system are responsible for site maintenance and their capabilities are different from other users of the system therefore they will be directed to another page where they can do add, edit and delete operations in the system.

Other users of the system, who are managers of retail stores, can use the system to display the heat maps of their stores. If they are brand managers, they are able to display all the stores of that brand.

Users can go back to the initial state of the system by logging out anytime they want.

All these operations can be seen in the below State Transition Diagram as states.

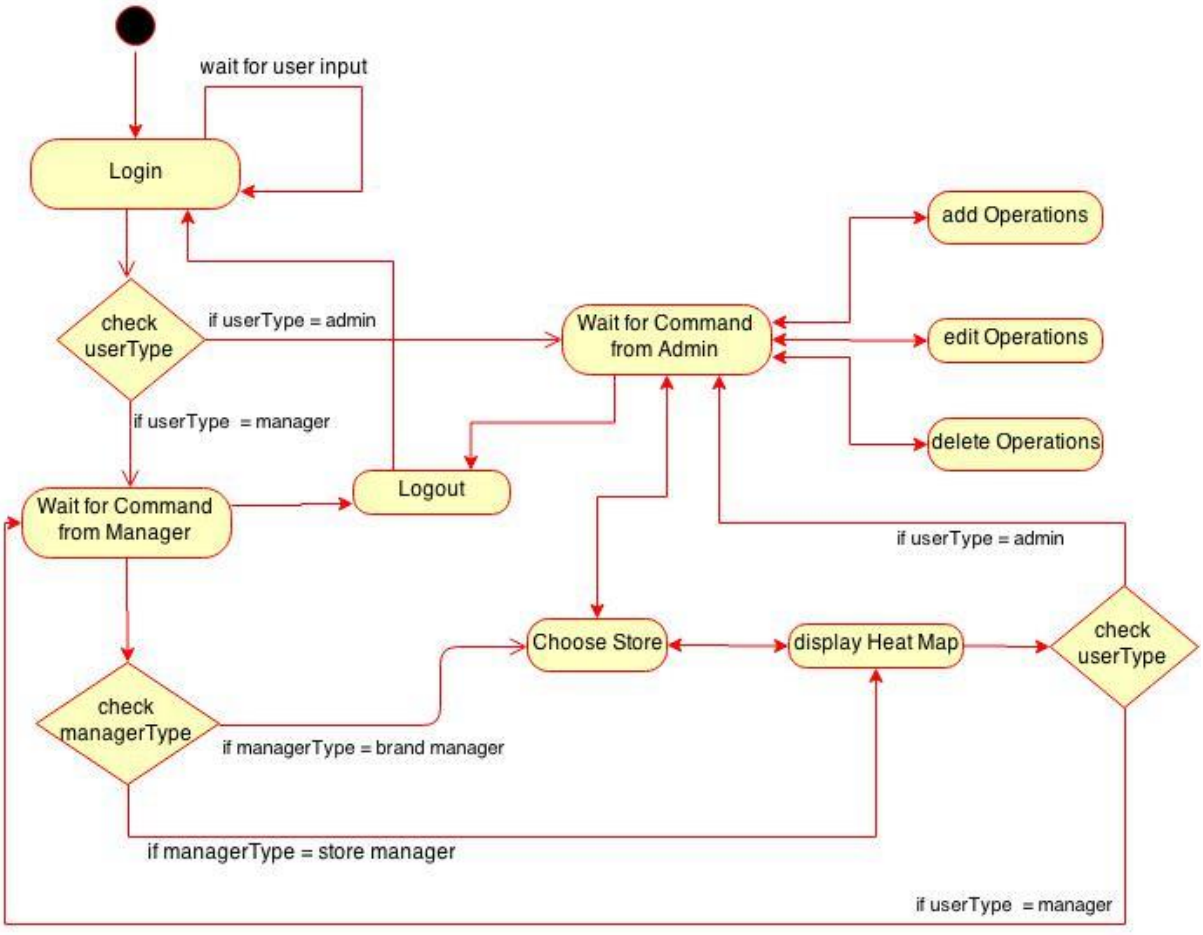


Figure 30 - State Transition Diagram

5.12 Algorithm Viewpoints

Algorithms are the crucial part of our project because the project needs to work with approximately 90% accuracy. This means that the algorithms must be efficient and effective. The steps of Human Tracking System in Retail Environment's algorithm are listed below;

1- Take data from the camera via frame grabber which is an electronic device that captures individual, digital frames from an analog video signal or digital video stream.

2- After frames starts to come, background subtraction will applied to the frame for detecting background and foreground scene. The name of this operation is Mixture of Gaussians which corresponds to MOG2 function in the OpenCV. Figure 31 shows the frame after applying back ground subtraction.

Figure 31 – Background Subtraction

3- After detecting moving objects, noises are needed to be removed. The name of this deleting noise operation is Opening and Closing which corresponds to MORPH_OPEN and MORPH_CLOSE functions in the OpenCV. Figure 32 shows the frame with noise.

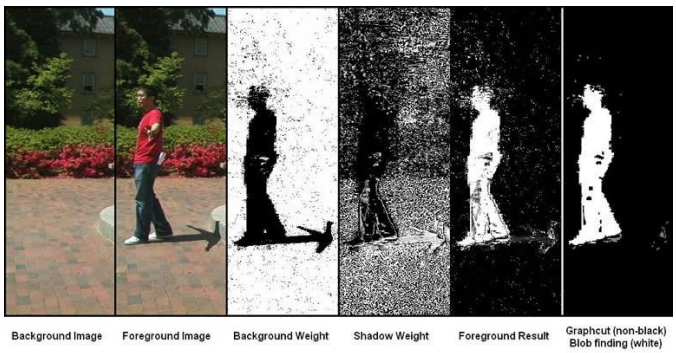


Figure 32 – Noisy Frame

4- After cleaning the binary image, white and moving objects are defined as an object class. In terminology, the name of this operation is “Blob”. After blobbing objects with rectangle shapes, the bottom left point of rectangles is taken as a reference point of objects and this point is going to be shown in the heat map. Figure 33 shows the frame after blobbing.

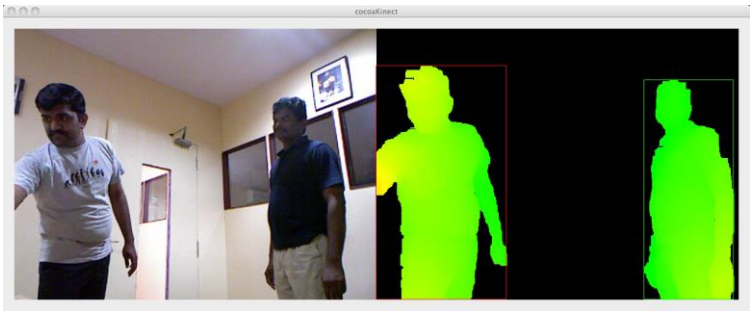


Figure 33 - Blob

5- After making Blob, the objects will be tracked. Such algorithms can be used for these operations like MeanShift, Kalman and CamShift.

6. Traceability Matrix

Use Case Design Elements	UC-1	UC-2	UC-3	UC-4	UC-5	UC-6	UC-7	UC-8	UC-9	UC-10	UC-11
Figure 2	X		X								
Figure 3	X		X								
Figure 4	X					X	X	X			
Figure 5	X		X							X	X
Figure 6	X		X	X	X						
Figure 7		X									
Figure 10	X	X	X	X	X	X	X	X	X	X	X
Figure 15	X										
Figure 16		X									
Figure 17		X	X	X	X						
Figure 18		X	X	X	X						

Figure 20		**X**		**X**		**X**	**X**	**X**			
Figure 22		**X**	**X**		**X**					**X**	**X**
Figure 24		**X**				**X**					
Figure 27	**X**	**X**									
Figure 28			**X**	**X**	**X**						
Figure 29						**X**	**X**	**X**	**X**	**X**	**X**
Figure 30	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**	**X**

7. Conclusion

This Software Design Description Document includes the system architecture and implementation details of the Human Body Tracking System project. In this document, basics of data design, modules and design viewpoints of the system are described respectively. The software tools, frameworks and libraries that will be used while designing and developing the system are also identified. The design issues are specified along with the related design viewpoints.