

Movie World System

Taha Mohamed Alzein	(269055)
Muhammad Nadeem	(266704)
Fadi Atia Dasus	(266265)
Oskars Arajs	(266534)
Balkis Ibrahim	(260092)

Supervisors

Ole Ilsgaard Hougaard
Christian Flinker Sandbeck

ICT Engineering, VIA University College, Horsens

3rd Semester

Abstract

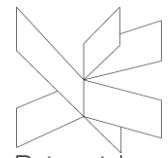
The purpose of this project is to develop a system by which customers are able to stream movies and book tickets for a specific cinema online.

Movie World System (MWS) is a desktop application created for streaming movies, viewing their description and relevant information including trailers, plus it provides the customer with a service to buy a ticket for movies.

MWS has to be implemented on the three tier Architecture basis and designed as a heterogeneous system. It has a user friendly Graphical Interface designed and developed using .Net Framework Form for the first tier (T1). The Second tier consist of business logic, web services and external Rest API (PayPal) and it is implemented in java (T2). DAO (Data access object) with model entities are presented in the data tier (T3). To store the entities "MSQL Azure" is used.

Table of contents

List of Figures	4
1. Introduction	5
2 User Stories and Requirements	6
2.1 User Stories	6
2.2 Functional Requirements.....	6
2.3 Non-Functional Requirements	7
3 Analysis	7
3.1 Scenarios	8
3.2 Use case diagram	8
3.3 Use case description	9
3.4 Activity diagram	10
3.5 Conceptual diagram	11
3.6 Database.....	12
3.7 Security	13
4 Design	18
4.1 Sequence Diagram.....	18
4.2 Class Diagram.....	20
4.3 GUI.....	21
4.3.1 Guest/Customer GUI	21
4.3.2 Admin GUI.....	26
4.4 Socket	27
4.5 Security	28
5 Implementation	32
6 Testing.....	35
7 Conclusion.....	37
8 References	38
9 Appendices.....	39



List of Figures

- Figure 1. Use case diagram***
- Figure 2 Watch Movie use case description***
- Figure 3. Activity diagram***
- Figure 4. Conceptual diagram***
- Figure 5. ER Diagram for MovieWorld***
- Figure 6. Risks assessment model***
- Figure 7. Class diagram***
- Figure 8. Main window for Guest/Customer.***
- Figure 9. Pop-up window.***
- Figure 10. Buy ticket button pressed***
- Figure 11. Ticket bought***
- Figure 12. Watch online button pressed***
- Figure 13. Subscribed for watching***
- Figure 14. Streaming movie for Customer***
- Figure 15. Add a movie***
- Figure 16. Remove a movie***
- Figure 17. Socket for Message***
- Figure 18. MD5Hash***
- Figure 19. GetCustomerByName***
- Figure 20. GetCustomerByName***
- Figure 21. Blob storage access policy***
- Figure 22. GetContainerSASToke***
- Figure 23. MovieDAO implementation***
- Figure 24. POST by PostMan***
- Figure 25. GET by PostMan***
- Figure 26. JUnit Test***

1. Introduction

Entertainment services are restricted in the developing countries due to a couple of reasons. For instance, paying online is one of the services that are not available to people in these countries. Paying with a credit card online is impossible, but it is possible to pay for a service via “PayPal”. In this way, even if people would like to buy a movie ticket for a cinema, they would not be able to do so, since a service like this is not available in these countries. Up till now some of these countries only provided On-Spot ticket, purchases.

[1] Moreover, home platforms such as “Netflix” [2] are not available in some country due to U.S. government restrictions on American companies [3]. Also the income level in these countries poses a challenge to Netflix since they provide an expensive service

A project that covers these bases would be very feasible. To provide customers with a simple environment to buy a ticket for a cinema or stream a movie at home.

The system should be heterogeneous so that server and web services are implemented in java and clients are designed in .Net Framework. The system is to be designed in a three tier architecture that consist of data tier, business logic tier and presentation tier. The relevant data is being stored in “MSQL Azure”.

The system deals with three actors: Admin, Guest and Customer. The Admins duties are to add, remove movies and read messages coming from Guests. The Guest can only purchase tickets, look at movies, see their description and subscribe to a movie streaming service. After subscription the Guest has become a Customer. A Customer is able to watch movies online after he has completed a subscription.

2 User Stories and Requirements

The purpose of this chapter is to draw the user stories which can help to create simplified requirements. The requirements are one of the major tasks in designing systems development (Alsaleh.S and Haroon.H., 2016). Requirements are a very critical phase for the functioning system. So the first step is to write the User stories and then requirements.

2.1 User Stories

As the name suggests, it describes how the User requires the software that is particularly helpful for the customer. Admin should be the one who needs the functioning software and customers can visit the software.

1. As an Admin I want to be able to add the movie to the system so that the Customer can see it.
2. As an Admin I want to be able to remove a movie from the system so that I can delete a movie.
3. As a Guest, I want to be able to select a movie so that I can see the movie details.
4. As a Guest, I want to be able to book a ticket for a certain movie, so that I can reserve a place for that movie.
5. As a Customer, I want to be able to stream the movie, so that I can watch the movie at home.
6. As a Guest, I want to be able to pay using an electronic payment method for my reservation or subscription so that I can have these services.
7. As an Admin I want to be able to read messages so that I can get Guest input.
8. As a Guest, I want to be able to send a message, so that the Admin can read it.

2.2 Functional Requirements

Based on the information from the user stories, the following functional requirements have been created and non-functional requirements are provided by the university,

1. The system must allow the Admin to be able to add a movie.
2. The system must allow the Admin to be able to remove a movie.

3. The system must allow the Guest to be able to see a list of movies.
4. The system must allow the Guest to be able to get movie information.
5. The system must allow the Guest to be able to book a cinema ticket for the selected movie.
6. The system must allow the Guest to be able to purchase a subscription electronically for streaming.
7. The system must allow the Customer to be able to get a ticket through his email.
8. The system must allow the Customer to be able to stream a movie.
9. The system must allow the Admin to read messages coming from Guests.
10. The system must allow the Guest to send messages to the Admin.

2.3 Non-Functional Requirements

1. The system must be implemented as a heterogeneous system (Server in Java, Client in .NET).
2. The system must be 3-tier architecture.
3. The system must use web services
4. The system must use a protocol for sockets and RMI
5. The system must include GUI for two clients (Customer and Admin)

3 Analysis

The analysis consists of scenarios, use case diagrams and use case descriptions. The **use case diagram** is the depiction of what a system can do for an Admin and customer. Use case diagram is based on the scenarios so these two are connected to each other. A **scenario** is a representation of what is going to happen when someone uses the system. Furthermore, **use case descriptions** have been made for each use case of the actors which participate in this flow.

3.1 Scenarios

Here only one scenario is presented and that is “add Movie” scenario of “Movie World” system.

Add Movie Scenario:

Admin chooses to add movie, Admin fills out the movie information (name, genre, description, duration, image URL, trailer URL, full movie URL) of the movie in the form, the system saves the information to database.

3.2 Use case diagram

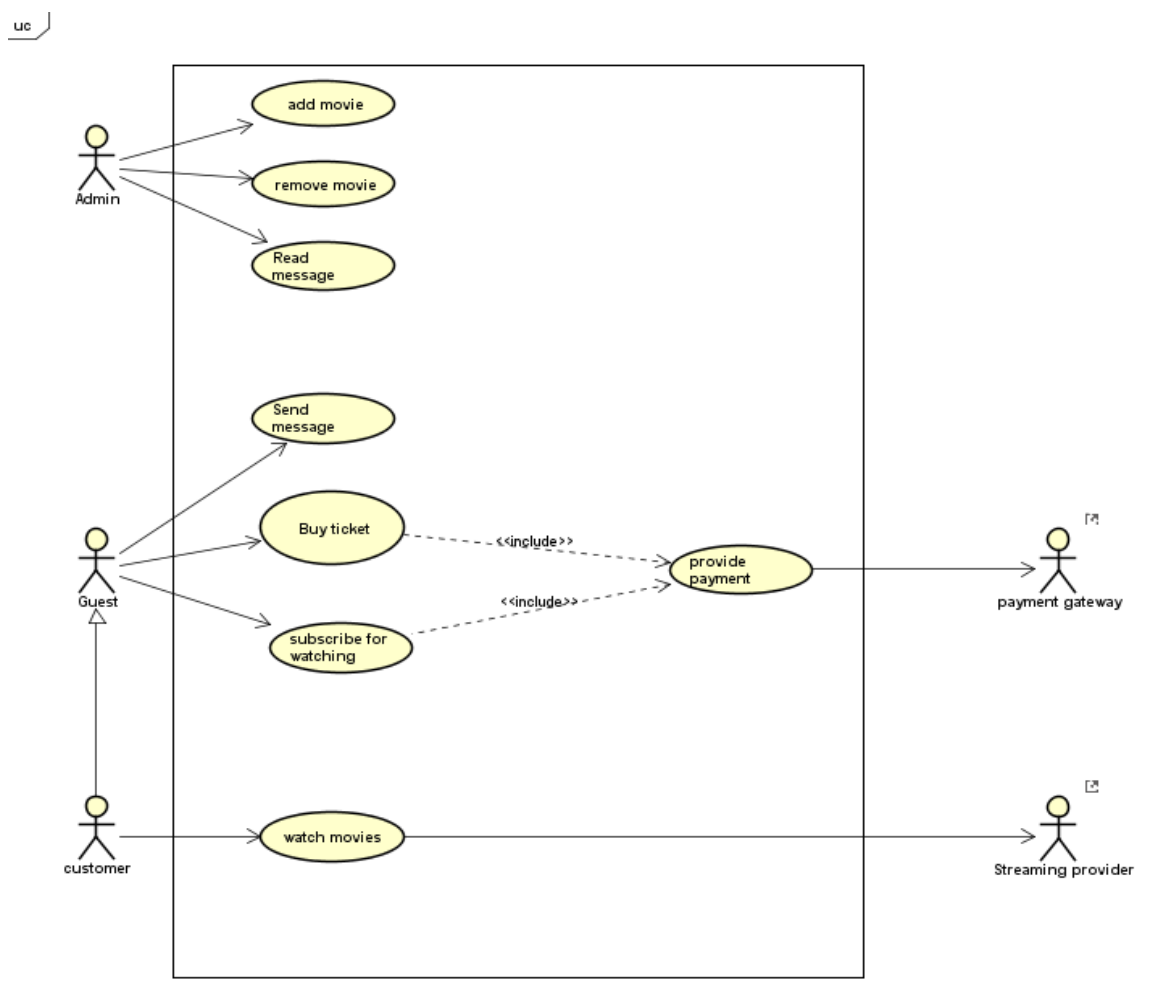


Figure 1. Use case diagram

In the Figure 1, shown above, the use cases were created by following the requirements. There are 3 actors: "Admin, customer, guest" are involved in the process. each of them has different use case based on his need, the diagram is also showing the 2 external systems that we use to provide online payment and movie streaming.

3.3 Use case description

Based on the use case diagram shown above, several use case descriptions were made for this subchapter. Only one-use case description will be shown as an example while the rest can be seen in **Appendix 2**.

ITEM	VALUE
UseCase	watch movies
Summary	
Actor	customer Streaming provider
Precondition	At least one movie is available. customer has a valid subscription.
Postcondition	customer is able to watch the movies.
Base Sequence	1- The system displays list of movie. 2- the customer chooses one movie. 3- the system display movie information. 4- the customer chooses to play movie. 5- an external system provides a streaming.
Branch Sequence	
Exception Sequence	1- the customer can cancel at any time
Sub UseCase	
Note	

Figure 2 Watch Movie use case description

In **Figure 2** “Watch Movie” use case is shown. The description seen above has been made as a guideline during the implementation of the system. Also, an exception sequence can be seen which means that the postcondition of the main success scenario cannot be reached.

3.4 Activity diagram

The activity diagram describes the flow between the action of “Subscribing to watch a movie” and what consequences some interactions will be produced if executed by the actor.

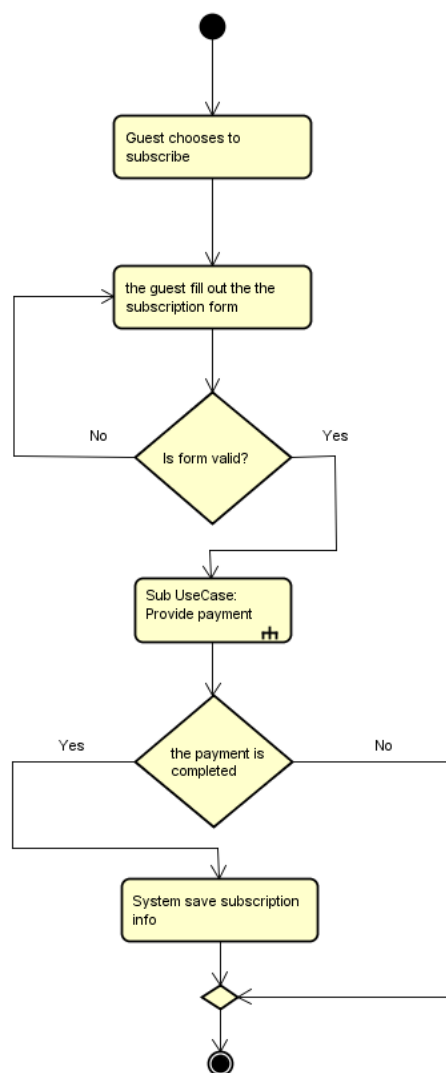


Figure 3. Activity diagram

In the example above, it shows the steps taken by the Guest to subscribe for watching movies.

In this example, there is a sub usecase involved called “Provide payment”

*For more examples of activity diagrams, see **Appendix 4**.*

3.5 Conceptual diagram

The conceptual diagram depicts the relationships between components. As shown below the three-tier architecture system visualizes how the tiers are interacting with each other. Starting from the third tier, which contains data access object together with model Entities which present tables in the database.

Business Logic Tier (Tier 2) consists of many services that provide the logic of the system, side by side with External API namely (PayPal). All business logic operations are implemented in this tier. Web Services are also presented in this tier. Presentation tier (Tier 1) occupies the top level, which consists of the client that consumes the web services.

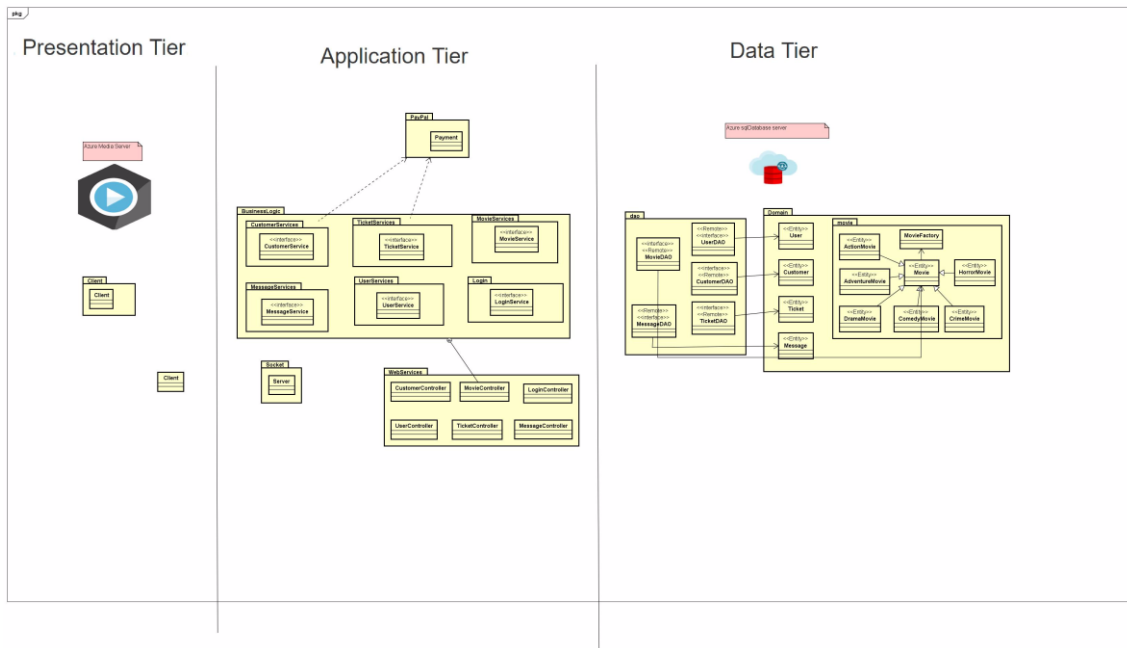


Figure 4. Conceptual diagram

3.6 Database

For this semester project, we were given the opportunity to choose our database on our own. We chose to use a cloud service database named "MSSQL server". We chose this cloud service for the multitude of different services it provides.

For this project, we are using three services from this provider. The first one would be "Azure SQL Database" for storing our clients and applications information.

The second one is "Media services" that we are using to stream videos related to our project. And the third one is "Blob storage" that is used to store movie files.

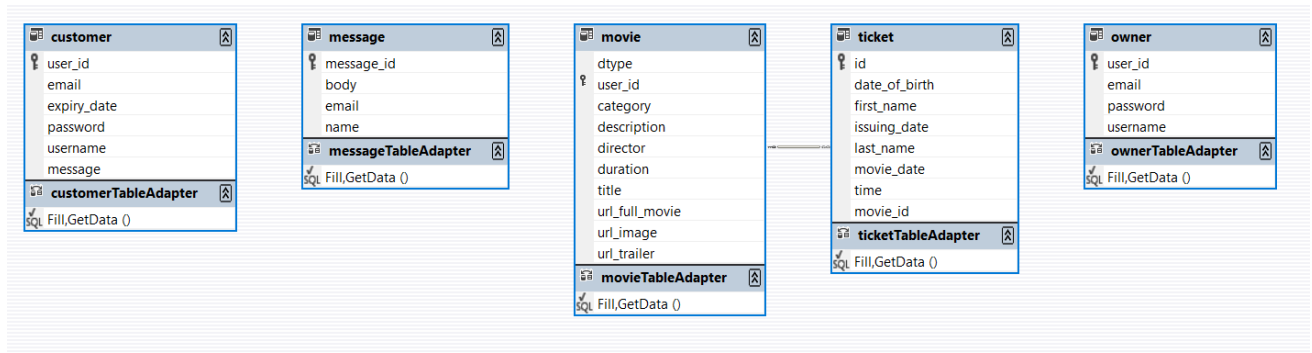


Figure 5. ER Diagram for MovieWorld

The figure above is an ER diagram for our database model. For this project, we only needed 5 tables and in each table, we have a single primary key.

3.7 Security

The security and integrity of any application should be taken in consideration while the project idea is still being analyzed. Without putting any thought in securing ones application the person might or most likely will reach a point when the application fails or because of consumer feedback, people will stop using it.

Since this application is dealing with user sensitive information, we should assume that at points there will be threats regarding the safety of our customer data. One way we protect our users is by keeping our sensitive data in our secure cloud database.

If the attackers gain access to our user sensitive information it would breach our systems security standards and would put at risk all of our clients.

There are many types of cybersecurity threats, but the most common ones are:

- Denial-of-service (DoS) and distributed denial-of-service (DDoS) attacks.
- Man-in-the-middle (MitM) attack.
- Phishing and spear phishing attacks.
- Drive-by attack.

- Password attack.
- SQL injection attack.

Denial-of-service attacks are the ones carried out by people who intend to make a service or machine (computer) unavailable for intended users. These attacks flood the target system with unneeded requests making the service temporarily or indefinitely unavailable for legitimate requests.

Man-in-the-middle attacks are carried out when two parties are communicating with each other through a seemingly secure connection, but when in-fact they are being transmitted through a third party. This is specifically a threat while the two intended parties are using an unsecure WiFi connections where the attacker is in range to intercept the messages being sent and relay them to the intended destination. With this kind of an attack the two parties think they are messaging each other, while in fact their messages are being relayed or possibly even altered.

Phishing and spear phishing attacks are attempts to steal sensitive information from unaware users. Attackers that use “Phishing attacks” disguise themselves as trusted service provider in an attempt to steal credit card details, passwords or other user relevant information. Usually attackers send out mass emails to people using the same site in order to trick them into opening a provided link that will transfer them to their desired service. When clicked, the link will transfer them to a different webpage. Although the webpage looks and acts legitimate, but in fact serves the attackers needs. The users enter the information needed (Passwords, usernames, credit card details and etc.) and sends it to the site. While the user thinks its information is being treated securely, it never made it to the desired destination rather to the attacker that scammed the users.

Password attacks or password cracking is the process of getting a hold of a user’s password by trying different possible combination. There are different methods on how

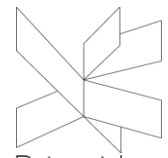
to get a user's password. Some of them include: Dictionary attack, Brute force, Rainbow table, Phishing and etc.

The Brute force method means simply guessing passwords until you find the correct one. This might take minutes and it might take weeks. Due to the reason passwords are getting longer and more complex "Brute force" gets less useful because it takes more and more time to actually "crack" the password, people are switching to rather the "Dictionary" method to make it more simple to use.

The Dictionary method requires is a text file containing words people might use as passwords. The longer this file is, the higher the chance that one of them might be the correct one. Although this works in cases when users choose simple passwords, word combination or simply substituting the letter "l" for a 1. But in a perfect scenario where users choose safe passwords by combining capital letters with small letters and adding multiple digits throughout the password, this method starts to get less and less useful due to the fact that much more possibilities occur and needs to be added to the text file.

SQL injection attacks are targeting the user's database in order to get valued information like passwords, credit card details editing the syntax that either stores or retrieves values from the database.

A common way of doing this would be retrieving information by means of the Logon function. When the user is asked for a username and password to log in, a attacker might pass a sql query to the username tab and retrieve wanted information from there.



Threat event	Threat source	Threat source Characteristics			Relevance	Likelihood of attack initiation	Vulnerabilities and predisposing conditions
		Capability	Intent	Targeting			
Gaining unauthorized access	hacker	Low	High	Users Admins	Possible	Moderate	Weak passwords, Authenticity problems
Obtaining user information	Hacker	Moderate	High	Users	Possible	High	Weak security related to database
Denial of Service	Hacker	High	High	Server	Possible	High	

Severity	Likelihood attack succeeds	Level of impact	Risk
High	Low	High	Moderate
High	Moderate	High	High
High	High	High	High

Figure 6. Risks assessment model

STRIDE model

STRIDE is a threat model developed by Microsoft to identify computer security threats in six different categories. Thus creating the mnemonic STRIDE

Spoofing is a when a person or program masks their identity and impersonates as a different person or service. In this project we are sending emails to clients so they could receive their tickets.

If someone decided to impersonate our system, they would have to send a really similar email as to the one our application is sending.

Tempering is when information is being manipulated in order to gain advantage over someone or to change privileges.

In this project the things of interest for tempering would be the URL links to movies we provide for streaming or/and the re-direct URL is to the payment gateway. If an attacker would be able to change the URLs the re-directed site might contain harmful scripts or malware.

Repudiation is when someone challenges the authenticity of a certain thing, like a key card. Or as it might be in our project when it's developed, the issue of copyright might be problem after this sort of an attack.

Since the system is providing movies for streaming, the cinema that would own this application would have to keep the copyright privileges safe.

Information disclosure is when an attacker is able to see information otherwise denied to them. This would be personal pictures or data files that would be kept private.

Denial of service is a major risk for this application since its main purpose is to stream video. These kind of attacks are usually carried out by activists that want to limit the accessibility of a site or network.

In many cases these kinds of attacks aren't meant to steal or profit, but rather to limit the accessibility of a site or service, and to lower the credibility and trust of a company or service in the eyes of the consumer.

Elevation of privilege is when a person receives more privileges in a system where he in fact should not have the privileges. This may result in a simple user receiving admin privileges and is able now to add, edit or copy information they otherwise would not get access to.

In this application the graphical user interfaces are separate. Meaning that the Customer side is separated from the Administrator side.

4 Design

4.1 Sequence Diagram

To get a better understanding on how the system behaves while certain commands are being given a sequence diagram is created. The sequences diagram will provide a visual representation of what steps the system will take and which methods will be executed to fulfil a certain task.

The following figure will present an example of a sequence diagram. Subscription diagram has been selected for this example.

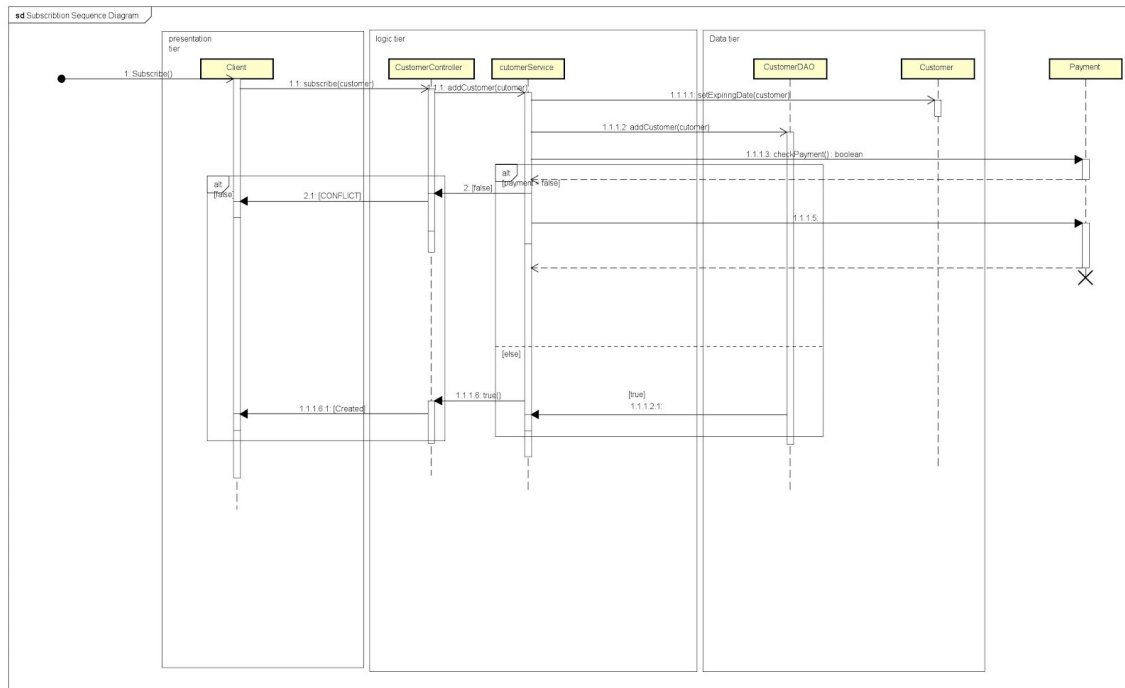


Figure 6. Sequence diagram

This sequence diagram illustrates the function of adding a customer to the database. Starting from client side where customer clicked the button of (Subscribe), the action will be taken to the server, where the business logic will be applied.

The first step in streaming a movie is subscribing to the service. When the subscribe button has been clicked, addCustomer method will be invoked from customerController, where the business logic will be executed.

So the first method in the chain will be setExpireDate(), in order to set an end subscription date. Then the system will check the payment status and If there is a customer with the same username in the database. Depending on the result of the previous conditions. If payment is approved and there is no customer with the same username in the database, the system will create a new customer, and a created status will be returned to the client. Otherwise, a conflict will occur.

4.2 Class Diagram

Starting from the Presentation Tier, where the actual client is interacting with the system to perform a certain operation. In this tier all graphical interfaces have been implemented to provide two types of interfaces. First, the Admin GUI which provide the main functionalities (add movie, remove movie). Second, the Guest GUI which provides the main services (streaming and buying a ticket online).

Presentation Tier communicates with the second Tier, where the system controls application functionality by performing detailed operations.

In the business Tier, five controllers will provide HTTP verbs in order to expose web services. The controllers coordinate with the services that add the logic needed for each function.

Here is where the “heavy lifting” of the application is done. It uses data that it has retrieved from the database or been sent by the user via the user interface.

Services apply a layer of abstraction to make the system open to use external systems in the future. At the time of documenting, there is one API (PayPal) that will play a vital role in manage payment methodology and provide the logic needed for that purpose. The services pull the data from the third tier, where data access object packages provide CRUD operations for persisting and retrieving data from the actual database which is located on the cloud.

Model package is the collection of objects that will be mapped to the database using Hibernate ORM framework.

Positive and negative sides of using three Tier architecture

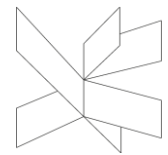
Advantages:

Three Tiers architecture is designed for CRUD applications where the database is the center of the application design which fit perfectly in the manner of this project.

Disadvantages:

There is ambiguity about where application, abstraction and domain level abstraction should go, and in practice, the application logic gets mixed with domain logic.

Each layer is coupled to the layer below it.



Project Report of Group 7: Movie World System

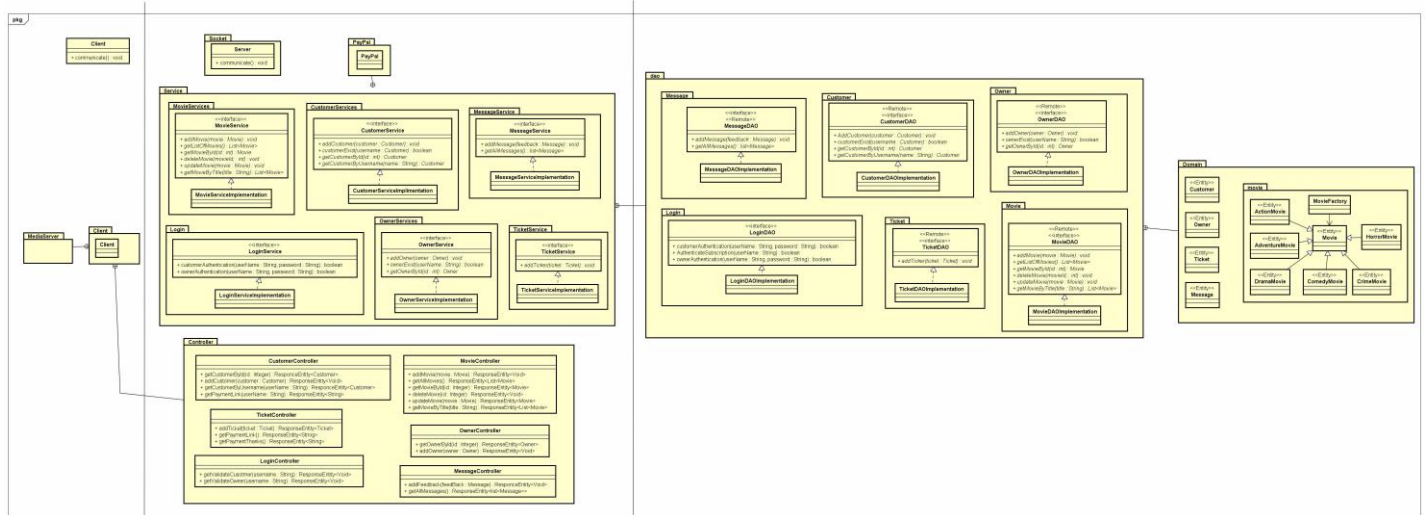


Figure 7. Class diagram

4.3 GUI

We created the GUI by following the WireFrame graphical design. See appendix 4 for more information

4.3.1 Guest/Customer GUI

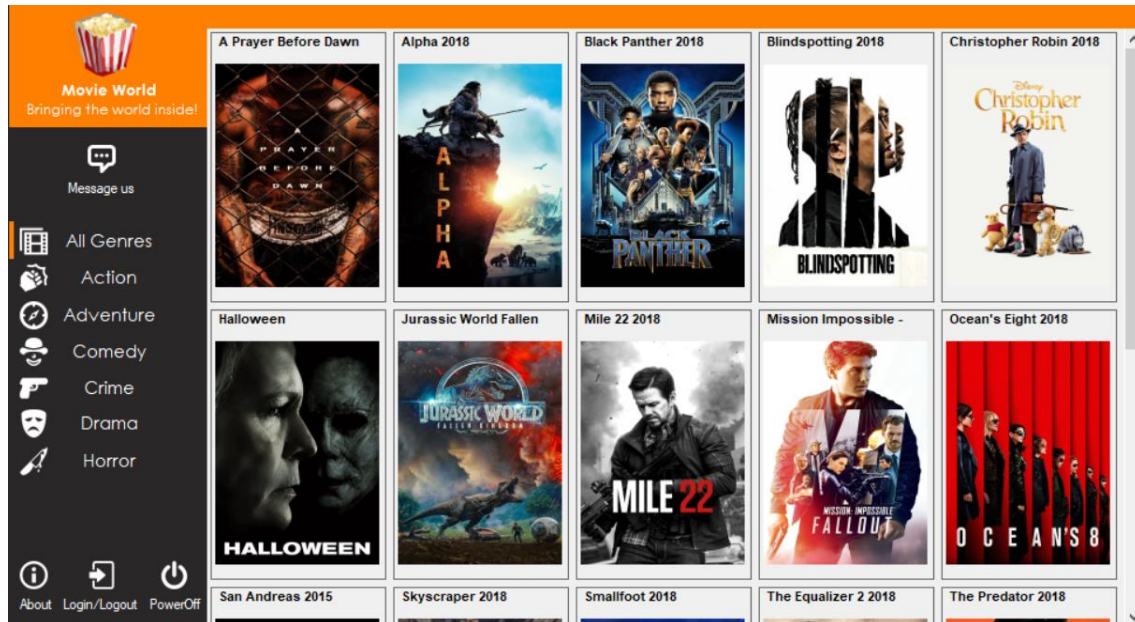
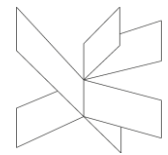


Figure 8. Main window for Guest/Customer.

This is the main window that is being presented to the Guest/Customer.



Figure 9. Pop-up window.

This is a pop-up window that appears over the main window when a movie is selected. In this window the Guest can see a short description about the movie. There are 3 buttons, each with its own function.

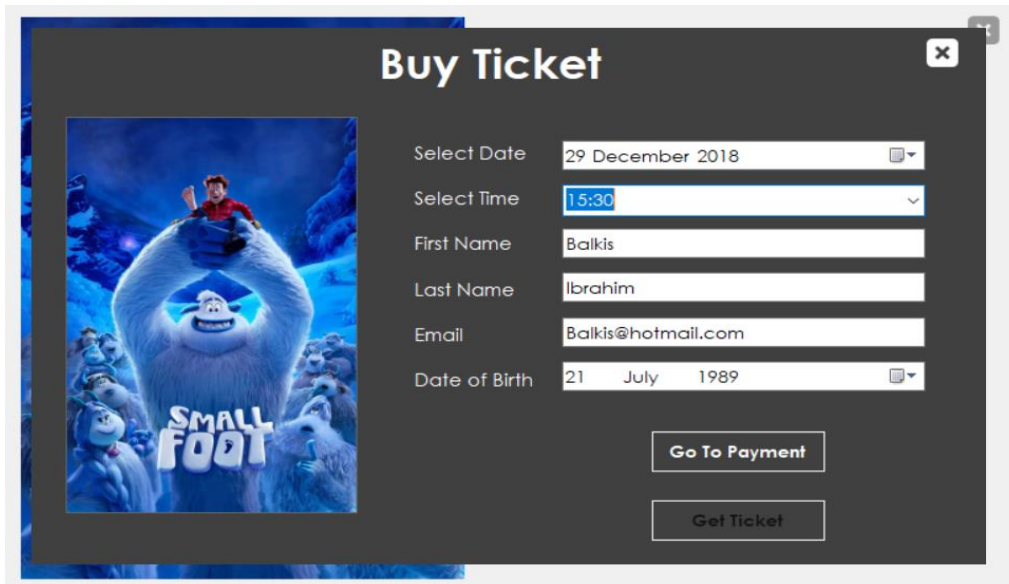


Figure 10. Buy ticket button pressed

After the button is pressed, the pop-up window will change to this window where the Guest/Customer can buy a ticket for a movie.

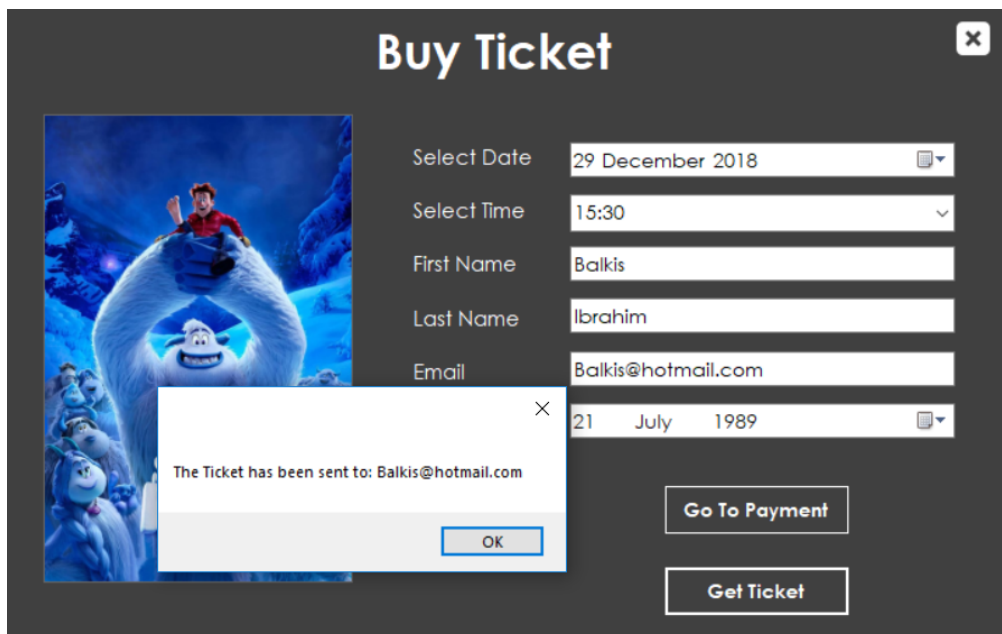
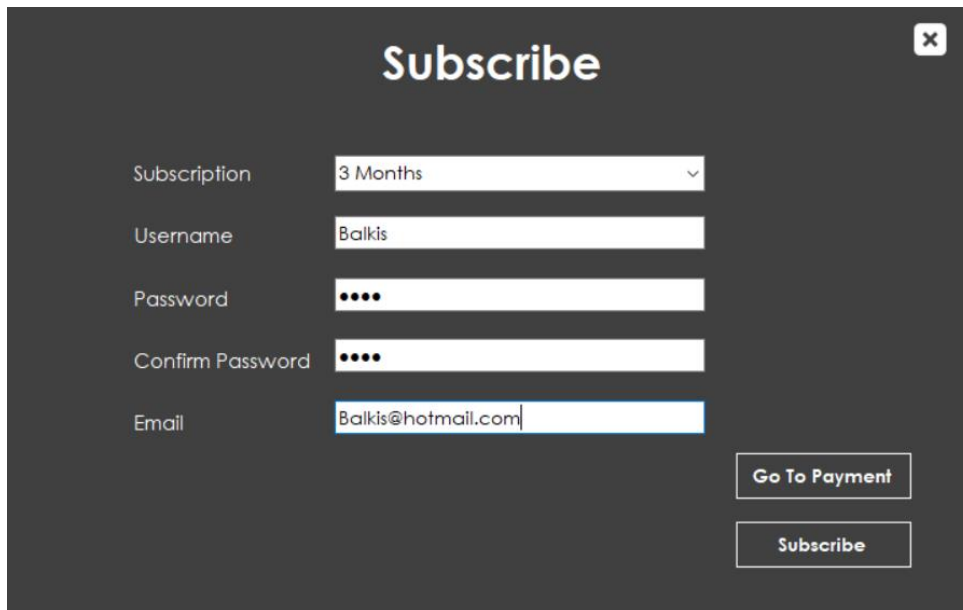


Figure 11. Ticket bought

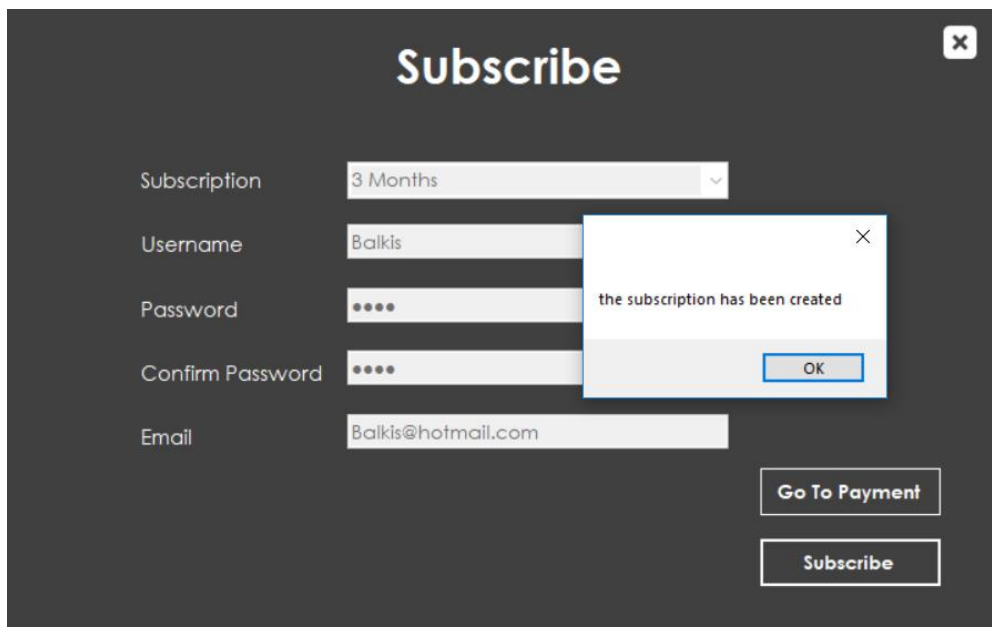
After the fields are filled out and the payment has been made, a message will pop-up informing the Guest that the ticket has been sent to their email.



The screenshot shows a dark-themed 'Subscribe' window. It contains five input fields: 'Subscription' (a dropdown menu set to '3 Months'), 'Username' (containing 'Balkis'), 'Password' (masked with four dots), 'Confirm Password' (masked with four dots), and 'Email' (containing 'Balkis@hotmail.com'). At the bottom right, there are two buttons: 'Go To Payment' and 'Subscribe'.

Figure 12. Watch online button pressed

After the guest presses the button for “Watch online”, this window will appear with multiple fields that have to be filled out in order to finalize the payment.



This screenshot shows the same 'Subscribe' form as Figure 12, but with a success message pop-up overlaid in the center. The pop-up is a small white box with a close button (X) in the top right corner. It contains the text 'the subscription has been created' and an 'OK' button at the bottom. The background form fields and buttons are still visible but slightly dimmed.

Figure 13. Subscribed for watching

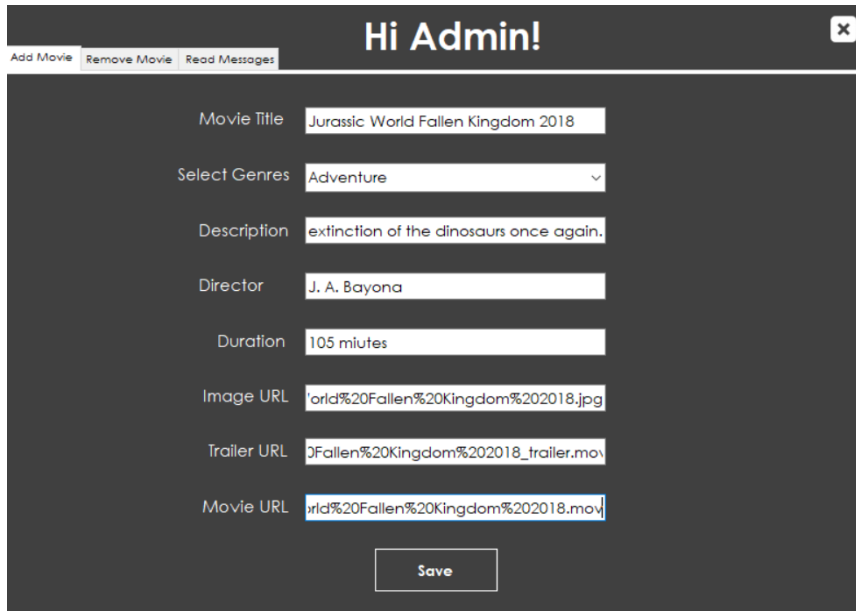
After all of the fields are filled out and the payment has been made, a message will pop-up warning the Guest that the subscription has been made. After the subscription has been made the Guest now counts as a Customer in this application



Figure 14. Streaming movie for Customer

After subscribing to watch movies online, the Customer can come back to the application any time to watch movies online, but only while the subscription is valid.

4.3.2 Admin GUI



Hi Admin!

Add Movie Remove Movie Read Messages

Movie Title: Jurassic World Fallen Kingdom 2018

Select Genres: Adventure

Description: extinction of the dinosaurs once again.

Director: J. A. Bayona

Duration: 105 miutes

Image URL: orld%20Fallen%20Kingdom%202018.jpg

Trailer URL: DFallen%20Kingdom%202018_trailer.mov

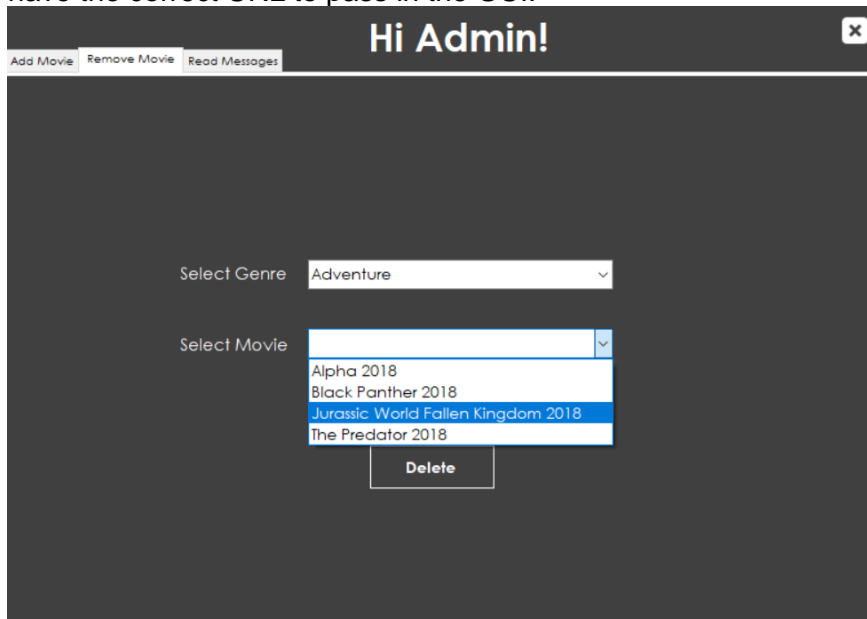
Movie URL: rld%20Fallen%20Kingdom%202018.mov

Save

Figure 15. Add a movie

From this window the Administrator can add a movie to the application by filling out all of the relevant fields.

But the Administrator has to add the movie file manually to the database in order to have the correct URL to pass in the GUI.



Hi Admin!

Add Movie Remove Movie Read Messages

Select Genre: Adventure

Select Movie: Alpha 2018, Black Panther 2018, Jurassic World Fallen Kingdom 2018, The Predator 2018

Delete

Figure 16. Remove a movie

From this part of the interface the Admin is able to remove any movie he/she chooses.

In this document, only one example will be shown regarding the GUI, for the rest see **Appendix 3 - Admin Guide.**

4.4 Socket

One of the requirements for this project is to design and implement protocols for sockets. For this purpose socket is implemented on sending welcome messages. TCP/IP is used for communication between first tier and second tier but only for sending greeting message.

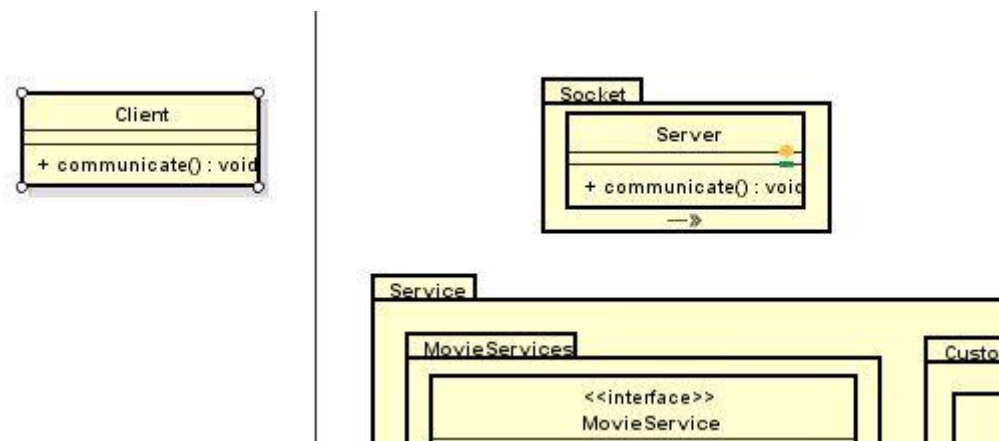


Figure 17. Socket for Message

SMTP (simple mail transfer protocol)

The SMTP protocol is being used to send the Customers their respective Tickets via their email. The sender has to be a gmail account but the receiver can be any email host provider.

4.5 Security

Password protection

Passwords related to the system are being stored in a database in hashed form. This is done to increase security in the system by providing a string of characters that represent user/ admin passwords in hash form in our database. When a Customer or admin is being registered, they choose a password and after they have chosen a password, the password will be sent to a hashing function called MD5.

```
public static string MD5Hash(string text)
{
    MD5 md5 = new MD5CryptoServiceProvider();

    //compute hash from the bytes of text
    md5.ComputeHash(ASCIIEncoding.ASCII.GetBytes(text));

    //get hash result after compute it
    byte[] result = md5.Hash;

    StringBuilder strBuilder = new StringBuilder();
    for (int i = 0; i < result.Length; i++)
    {
        //change it into 2 hexadecimal digits
        //for each byte
        strBuilder.Append(result[i].ToString("x2"));
    }

    return strBuilder.ToString();
}
```

Figure 18. MD5Hash

In this method, the password is being encrypted with a hash function called MD5. It will save the password in a hashed form in our database.

When a Customer or Administrator will try to log in, their plain text password will be converted into the hashed version and compared to the hashed string in our database.

If the comparison returns true, it will log in the Customer or Admin.

If, in a situation where our database is broken into and the passwords are compromised, the attacker still won't be able to use the hashed passwords to log in, since the hash function only works one way. Meaning if the attacker used the exact password hashing function and tries to pass the hashed string to the function it will return a new string that's created by hashing the already hashed string.

In this project we are using MS Azure as our database. Azure database is a secure cloud service for storing data in a data safe environment. And for this projects needs we assume MS Azure to be more then safe of a storage space.

```
@Override
public Customer getCustomerByUsername(String username) {
    Customer customer = (Customer) entityManager
        .createNativeQuery("select * from Customer as a WHERE a.username = ?0", Customer.class)
        .setParameter(0, username).getSingleResult();
    return customer;
}
```

Figure 19. GetCustomerByName

The above code snippet is our solution to some SQL injections. With this solution, we are preventing any possible attacks by using parameterized statement which will skip user input, if it's not valid and it will prevent any dangerous query injections the user might try to pass in the system.

This was the only native query used in the system, because the main responsibility was taken by the Java Persistence API on top of Hibernate.

Moreover, validation has been taken into consideration from both client and server side. To minimize invalid inputs from user.



```

@Entity
@Table(name = "Customer")
public class Customer implements Serializable {
    /**
     *
     */
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "user_id")
    private Integer id;

    @Column(name = "username", nullable = false, length = 20)
    private String username;

    @Column(name = "expiryDate")
    private String expiryDate;

    @Column(name = "password", length = 200, nullable =
false)
    private String password;

    @Column(name = "email", length = 2000)
    private String email;
}

```

Figure 20. GetCustomerByName

Secure Socket Layer

SSL is the standard for encrypting the communication link between the client and the server. This way the communication link is being secured and reduces the possibility of man in the middle attacks and secures the data that is being transmitted. To get a SSL certificate a company must complete a form stating their information and so on, to get verified as a trusted service provider.

All client without a valid certificate will not be able to reach the server or sharing data with it. For our project we are using "Self-signed certificate" this means that we distribute personally the certificate to our clients. This is done for the sake of simplicity, and because our application, for now, is used in between small number of clients. And its not shared on the World Wide Web.

Meaning all the connection links between the server and the client has been published using https rather than http.

Azure Shared access signature

Our movie files are stored in “Blob storage” located in Azure. In case someone copied the URL of a movie and added it to a web browser, the web browser would not be able to reach the destination.

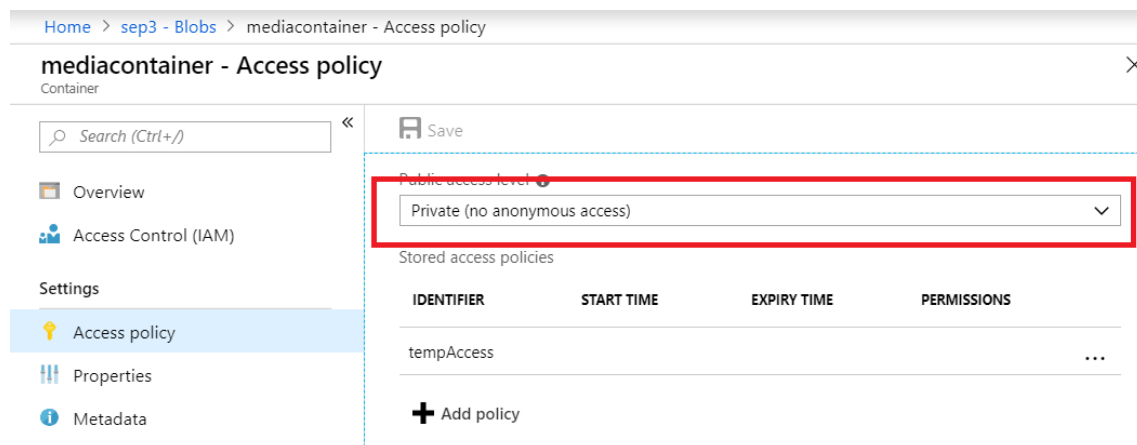


Figure 21. Blob storage access policy

In the figure above, it shows the access level policy for blob storage. Which is set to private, so no one can access it without an access token.

5 Implementation

```
private static string GetContainerSASToken(string xURL)
{
    string containerName = "mediacontainer";
    string azureStorageSharedKey = "gGzDkZqtx3T++aAeF7pvT3XqN3F2ZqMPsg9ic0EHnURBUqpogrspFzd/3BiDRZ4Vfmg";
    string canonicalPathToResource = string.Empty;
    string storageConnectionString = "DefaultEndpointsProtocol=https;AccountName=sep3;AccountKey=gGzDkZqtx3T++aAeF7pvT3XqN3F2ZqMPsg9ic0EHnURBUqpogrspFzd/3BiDRZ4Vfmg";
    string[] tokens = storageConnectionString.Split(';');
    string storageAccountName = tokens[1].Remove(0, 12);
    string permissions = "r";
    DateTime startTime = DateTime.UtcNow;
    DateTime expiryTime = startTime.AddHours(867240);
    string policyIdentifier = "tempAccess";
    string signature = string.Empty;

    //Parse the connection string and return a reference to the storage account.
    CloudStorageAccount storageAccount = CloudStorageAccount.Parse(storageConnectionString);
    //Create the blob client object.
    CloudBlobClient blobClient = storageAccount.CreateCloudBlobClient();
    //Get a reference to a container to use for the sample code, and create it if it does not exist.
    CloudBlobContainer container = null;

    container = blobClient.GetContainerReference(containerName);
    canonicalPathToResource = ("/" + storageAccountName.Trim() + "/" + containerName).Trim();
    //Get a reference to a blob within the container.
    byte[] keyForSigning = System.Convert.FromBase64String(azureStorageSharedKey.Trim());

    string sStartTime = startTime.ToUniversalTime().ToString("yyyy-MM-ddTHH:mm:ssZ");
    string sExpiryTime = expiryTime.ToUniversalTime().ToString("yyyy-MM-ddTHH:mm:ssZ");

    string stringtosign = permissions + "\n" +
        sStartTime + "\n" +
        sExpiryTime + "\n" +
        canonicalPathToResource + "\n" +
        policyIdentifier;

    using (var hmac = new HMACSHA256(keyForSigning))
    {
        signature = System.Convert.ToBase64String(
            hmac.ComputeHash(Encoding.UTF8.GetBytes(stringtosign))
        );
    }

    BlobContainerPermissions bcPermissions = new BlobContainerPermissions();
    bcPermissions.SharedAccessPolicies.Add(policyIdentifier, new SharedAccessBlobPolicy { });
    container.SetPermissionsAsync(bcPermissions);

    string sharedAccessSignature = string.Format("st={0}&se={1}&sr=c&sp=r&sig={2}&si={3}",
        Uri.EscapeDataString(sStartTime),
        Uri.EscapeDataString(sExpiryTime),
        Uri.EscapeDataString(signature),
        Uri.EscapeDataString(policyIdentifier));
    string url = xURL + "?" + string.Format(sharedAccessSignature);

    return url;
}
```

Figure 22. GetContainerSASToken

This function generates a token using a primary key and adds this token to the URL. When the token is added it makes the URL accessible only to people that are using “Movie World “system.

The URLs filled in the “add movie” form, are passed to a method called “GetContainerSASToke”, in order to create a new version of this URL with the additional access token, that's been included in the new URL before being stored in the database.

PayPal gateway is playing a vital role in the system, it provides the logic needed to performed payments, by directing the customer to the PayPal main payment page. Moreover, it validates the status of the payment.

Object-relational mapping (ORM) tool has been used in the system for providing a framework in order to map domain models to a relational database located in Azure.

Furthermore, Java persistence application has been used as a specification to provide Hibernate with the guideline methods in order for accessing, persisting, and managing data between java objects.

It's worth mentioning that spring gives a bunch of configuration out of the box. For instance, managing transactions using JTA can be so important when dealing with update where an object will be managed and persisted to the Database.

```
@Transactional
@Repository
public class MovieDAOImplementation implements MovieDAO {

    @PersistenceContext
    @Autowired
    public EntityManager entityManager;

    @Override
    public void addMovie(Movie movie) {
        entityManager.persist(movie);
    }

    @Override
    public void updateMovie(Movie movie) {
        Movie selectedMovie = getMovieById(movie.getId());
        selectedMovie.setTitle(movie.getTitle());
        selectedMovie.setDirector(movie.getDirector());
        selectedMovie.setDescription(movie.getDescription());
        selectedMovie.setCategory(movie.getCategory());
        selectedMovie.setDuration(movie.getDuration());
        selectedMovie.setUrlFullMovie(movie.getUrlFullMovie());
        selectedMovie.setUrlImage(movie.getUrlImage());
        selectedMovie.setUrlTrailer(movie.getUrlTrailer());
        entityManager.flush();
    }
}
```

Figure 23. MovieDAO implementation

The figure above illustrate how movie will be updated in the database from the DAO package using entity manager.

@Transactional provides us with a transaction management tool for persisting data in the database, and it has been useful in this function since we are updating the movie by editing all of its fields. The operation will succeed if, and only if, all of the updated fields have been saved in the database. In case any errors causes that the data to not be saved, @Transactional will make automatic “RollBack” and keep the original version of the data.

@Repository annotates classes at the persistence layer, which will act as a database repository making CRUD operations possible from this class.

@Persistence context is a set of entities such that for any persistent identity there is a unique entity instance. Within a **persistence context**, entities are managed. The EntityManager controls their lifecycle, and they **can** access datastore resources. Update movie method will take a movie object as an argument. The method will request the movie record from the database, using entity manager instance that has been auto wired from the container. Then update all off the columns in that record based on the passed movie object. And finalized the operation by using the flush() method from the entity manager.

6 Testing

Restful web services used in the system, has been tested using PostMan to perform manual test to ensure that everything is working as it was intended.

In the figure below, several HTTP verbs have been tested.

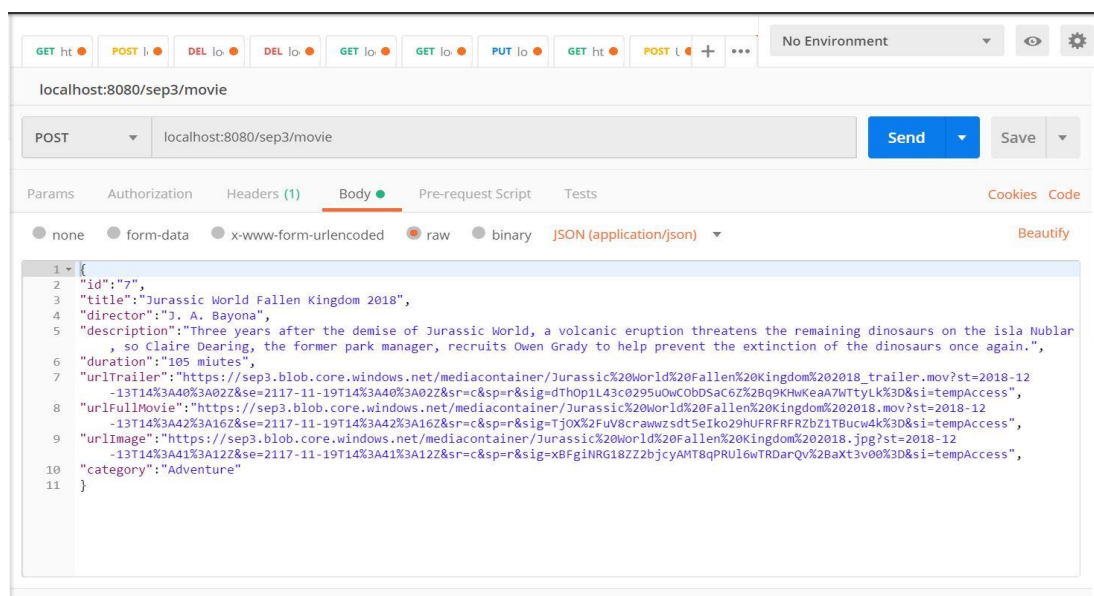


Figure 25. POST by PostMan

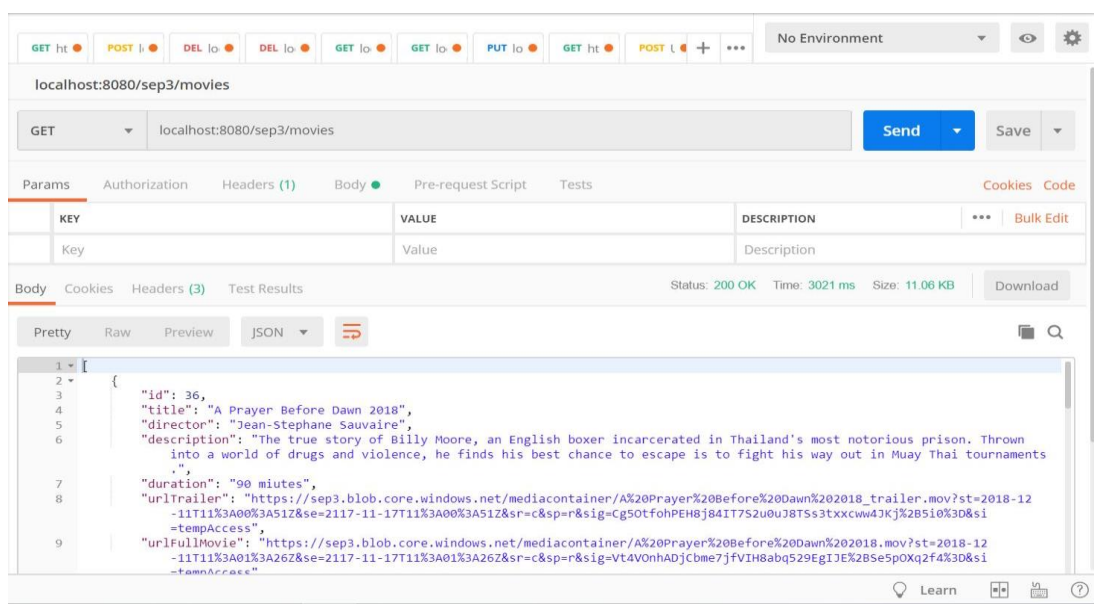
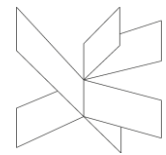


Figure 26. GET by PostMan



Project Report of Group 7: Movie World System

The screenshot displays an IDE with a JUnit test class and its execution output. The test class, `Sep3JpaApplicationTests`, is located at `package sep.via.dk.sep3JPA;` and includes imports for `org.junit.Assert`, `org.junit.runner.RunWith`, `org.springframework.test.annotation.Rollback`, `org.springframework.test.web.servlet.MockMvc`, and `org.springframework.test.web.servlet.MvcResult`. The test method `addMovie()` is annotated with `@Test` and `@Rollback`. It uses `MockMvc` to perform a POST request to `/sep3/movie/` with a JSON body containing movie details. The test asserts that the response status is `CREATED` and that the response header `Location` is `http://localhost/movie/1`.

The execution output shows the test passing successfully. The console output includes the following log messages:

```
118-12-19 03:45:34.856 INFO 21696 --- main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [
name: default
...
118-12-19 03:45:34.157 INFO 21696 --- main] org.hibernate.Version : HHH000412: Hibernate Core (5.3.7.Final)
118-12-19 03:45:34.161 INFO 21696 --- main] org.hibernate.cfg.Environment : HHH000206: hibernate.properties not found
118-12-19 03:45:34.393 INFO 21696 --- main] o.hibernate.annotations.common.Version : HCANN000001: Hibernate Commons Annotations (5.0.1.Final)
118-12-19 03:45:34.629 INFO 21696 --- main] org.hibernate.dialect.Dialect : HHH000400: Using dialect: org.hibernate.dialect.MySQL5InnoDBDialect
118-12-19 03:45:36.382 INFO 21696 --- main] j.LocalContainerEntityManagerFactoryBean : Initialized 3PA EntityManagerFactory for persistence unit 'default'
118-12-19 03:45:37.823 INFO 21696 --- main] o.s.s.concurrent.ThreadPoolTaskExecutor : Initializing ExecutorService 'applicationTaskExecutor'
118-12-19 03:45:37.987 WARN 21696 --- main] org.springframework.web.servlet.mvc.annotation.AnnotationMethodMapping : spring.jpa.open-in-view is enabled by default. Therefore, class-level cacheability will always be true.
118-12-19 03:45:39.579 INFO 21696 --- main] o.s.t.s.a.s.SpringServletTestContext : Initializing Spring 'testDispatcherServlet'
118-12-19 03:45:39.579 INFO 21696 --- main] o.s.t.s.a.s.SpringServletTestContext : Initializing Servlet ''
118-12-19 03:45:39.606 INFO 21696 --- main] o.s.s.a.e.web.EndpointLinksResolver : Exposing 2 endpoint(s) beneath base path '/actuator'
118-12-19 03:45:39.663 INFO 21696 --- main] o.s.t.s.a.s.SpringServletTestContext : Completed initialization in 82 ms
118-12-19 03:45:39.729 INFO 21696 --- main] s.via.dk.sep3JPA.Sep3JpaApplicationTests : Started Sep3JpaApplicationTests in 9.981 seconds (JVM
```

Figure 27. JUnit Test

JUNIT test has been performed on mock data, as shown above.

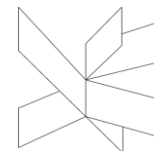
7 Conclusion

Movie World System is a desktop application developed for customers to see movie trailers, descriptions and streaming movies. The customers are also be able to purchase tickets for a cinema. The Project was finished successfully, meeting the starting requirements.

There are some areas of the project that could be “upgraded” to add some extra features.

Future Work:

- Seat Reservation has not been implemented.
- System should be developed for web and android application.
- Search the movie by genre or actor should be developed as well.



8 References

Cinema in Syria [1]

<http://www.cinema-alzahra.com/english/home-page/>

Online streaming (Netflix) [2]

<https://www.netflix.com/dk-en/>

[3]

<https://help.netflix.com/en/node/14164>

Guidelines Development and Summary, IT-SDJ2X-S18 Session Material, [Last accessed 05/04/2018] via the link: <https://studienet.via.dk/Class/IT-SDJ2X-S18/Session%20Material/SDJ2-S18-19GuidelinesDeploymentAndSummary.pdf>

Ken Schwaber, Jeff Sutherland, 2011; The Scrum Guide, [Last accessed 10/04/2018] via the link: https://studienet.via.dk/Class/IT-SWE1X-S18/Session%20Material/Scrum_Guide.pdf

Project report, 2017 (Appendix 3) VIA Engineering Guidelines [Last accessed 10/04/2018] via the link: [https://studienet.via.dk/projects/Engineering_project_methodology/General/Guidelines/2017%20Project%20Report%20\(Appendix%203\)%20-%20VIA%20Engineering%20Guidelines.pdf](https://studienet.via.dk/projects/Engineering_project_methodology/General/Guidelines/2017%20Project%20Report%20(Appendix%203)%20-%20VIA%20Engineering%20Guidelines.pdf)

Project description, 2017 (Appendix 1) VIA Engineering Guidelines [Last accessed 27/02/2018] via the link: [https://studienet.via.dk/projects/Engineering_project_methodology/General/Guidelines/2017%20Project%20Description%20\(Appendix%201\)%20-%20VIA%20Engineering%20Guidelines.pdf](https://studienet.via.dk/projects/Engineering_project_methodology/General/Guidelines/2017%20Project%20Description%20(Appendix%201)%20-%20VIA%20Engineering%20Guidelines.pdf)

Process report, 2017 (Appendix 2) VIA Engineering Guidelines, [Last accessed 05/04/2018] via the link: [https://studienet.via.dk/projects/Engineering_project_methodology/General/Guidelines/2017%20Process%20Report%20\(Appendix%202\)%20-%20VIA%20Engineering%20Guidelines.pdf](https://studienet.via.dk/projects/Engineering_project_methodology/General/Guidelines/2017%20Process%20Report%20(Appendix%202)%20-%20VIA%20Engineering%20Guidelines.pdf)

Rohit Joshi, 2015; Java Design Patterns, [Last accessed 14/03/2018] via link; <http://enos.itcollege.ee/~jpoial/java/naited/Java-Design-Patterns.pdf>

Thomas Connolly, Carolyn Begg, 2015; Database Systems A Practical Approach to Design, Implementation and Management, [Last accessed 18/05/2018] via the link;

<http://people.stfx.ca/x2011/x2011asx/5th%20Year/Database/Database%20Management%20Textbook.pdf>

Videos:

Programming Knowledge Learning, 2014, July, Java Eclipse GUI Tutorial 1 Creating First GUI Project in Eclipse, [Video file]. Retrieved from <https://www.youtube.com/watch?v=r8Qiz9Bn1Aq>

9 Appendices

Appendix 1 Project Description
Appendix 2 Use case description and activity diagram
Appendix 3 User GUIDE
Appendix 4 Diagrams
Appendix 5 Scenarios
Appendix 6 Contract