**IVAN ZDRAVKOV**

**GRADUATION THESIS REVIEW**

**DESIGN AND IMPLEMENTATION OF A SYSTEM FOR ONLINE PAYMENTS AND ACCESS TO SERVICES**

Following the exact structure of the graduation thesis, a simple review of the chapters and points made in the thesis will be presented below. A URL containing the original thesis in Bulgarian ([Дипломна работа - Иван Здравков.docx](https://bitbucket.org/ivanzdravkovdiplomproject/diplom-project/src/a65674b14aee1b2be7742553484562b447e7fca8/%D0%94%D0%B8%D0%BF%D0%BB%D0%BE%D0%BC%D0%BD%D0%B0%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%20-%20%D0%98%D0%B2%D0%B0%D0%BD%20%D0%97%D0%B4%D1%80%D0%B0%D0%B2%D0%BA%D0%BE%D0%B2.docx?at=develop" \o "Дипломна работа - Иван Здравков.docx)) and the source code can be found at chapter 8 of the following review.

**Chapter 1 - Introduction**

* 1. **Formulation of the problem** – The task is to design and implement a web-based system that governs the process of payment and access control for a network of sports facilities. A client of the facilities uses PayPal to pay his monthly subscription or a predefined number of practices, and is issued an RFID card that can be used to verify his legitimacy to access the facilities. The system should also keep track of his payment and attendance histories.
  2. **Architecture** – Client-server MVC (Model View Controller) application using external services (**PayPal API** for payment processing, **Yahoo! Finance** **Info** for the currency exchange Rates, etc). As an extra to the classical 3-layer model are employed a Data Layer and an ORM models.
  3. **Back End Technologies**
* ASP.NET MVC 5.0
* ASP.NET Web Api 2.0
* ASP.NET Identity 1.0
* ASP.NET Razor View Engine
* Entity Framework 6.1.0
* MSSQL
* C# 5.0
  1. **Front End Technologies**
* HTML 5
* CSS 3
* JavaScript
* jQuery
* jQuery Validate
* Modernizr
* Bootstrap
* Knockout
  1. **Development Tools**
* Visual Studio 2013
* SQL Server 2012
* Fiddler
* Google Chrome F12 Developer Tools

**Chapter 2 – Designing the elements of the user interface**

2.1. **Multi-language support through .resx files** – the application is implemented in English and Bulgarian languages. The .resx files are Key-Value files that hold word names in the Key field and the translated word in Value field. With the use of a different file, you can employ multiple language support for an unlimited number of languages.

2.2. **Razor View Engine** – the Engine helps build HTML as .cshtml templates. This allows us to build basic HTML pages in the back end using the computing power of the server as well as C#.

2.3. **Designing the \_Layout page** – the \_Layout page is the master page of the application that holds the basic UI elements such as the headers, footers, menus, etc. All other HTML content is injected into this page.

2.3.1. **Navigation** – a review of the navigation HTML

2.3.2. **Body** - a review of the body HTML

2.3.3. **Footer** - a review of the footer HTML

2.4.  **CSS styles** - a review of the CSS styles used in the application

2.5. **Designing the register page** - a review of the Register page of the application

2.6. **Designing the login page** - a review of the Login page of the application

2.7. **Designing the Subscription page of the application** - the web page a logged in user uses to pay his practices or monthly subscriptions via PayPal.

2.8. **Designing the User Management page** – the web page the staff of the training facilities uses to bind users to RFID cards.

2.9. **Designing the Visits and Payments pages** – the web pages the user can use to see queries of his payments made as well as the visits he made with extra information about these.

**Chapter 3 – Designing the Relational Data Base**

3.1. **Relational Model** – Basic Information about the relational model and why it was chosen for this assignment.

3.2. **Identity Tables** – The required tables the ASP.NET Identity needs to properly build the authorization and authentication of the system.

3.3. **Other Tables** – All the other tables that hold additional information for the users, facilities, contact info, addresses, RFID data, payment and attendance histories, etc.

3.4. **Entity-Relationship Diagram** – A full diagram of the ER Model

**Chapter 4 – Designing and implementation of the Web Services**

4.1. **BaseController and EntityHelpers classes** – a review of the 2 basic classes that hold most of the shared functionality between controllers. The BaseController holds a DBContext property, that is used to access the DatanBase, powered by Entity Framework. Custom logic is built upon the BaseController to automatically write in the Date and who was an entry in the data base created and/or modified by.

4.2. **Payment Controller** – here the conversion of currencies is done using the Yahoo! Finance Api. The PayPalCheckoutInfo class is also used here to build a model to post to the PayPal Api to create a PayPal request and transfer the user to the respectful PayPal web page where he can enter his credentials and pay for the service. A listener is implemented to verify the transaction is successful, award the user the goods he paid for and redirect him to the web site.

4.3. **Subscription Controller**- the payment controller is the back end implementation of the Subscription web page we discussed earlier. Here the HTML for the webpage is build.

4.4. **Log Controller** – this is the controller used to show the client information about his previous payments and visits.

**Chapter 5 – Software Review**

5.1. **InternalDB Project** – an SQL project that takes care of initializing the Data Base, Creating the required tables and inserting the minimal data required for the system to function.

5.2. **InternalDAL Project** – the Data Access Layer of the application. This project holds the Entity Framework .edmx Model used to connect with the Data Base

5.3. **Resources Project** – this project holds the .resx files that provide the multi-language support of the application

5.4. **WebSite Project** – the ASP.NET MVC Server.

5.5. **RFID Reader Project** – the project builds a console application that acts as an RFID listener. It uses the MasterCom.dll library the RFID Reader’s manufacturer has provided and periodically checks a Port the reader is connected to for incoming readings. When a card is inserted in the reader, the data from the card is posted to the web server that verifies if the user has access to the facilities. If he does a 200 OK Response is returned and the user is granted physical access, and if he does not a 401 Unauthorized Response is returned and the user is rejected access.

5.6. **Mifare1k** – The application that is used to write data to the RFID cards. When a user is registered into the system, a staff member issues a unique card to them he can then use to access the facilities.

**Chapter 6 – User Manual**

6.1. **Client Manual** – A user manual for the client to get to know the system, Including Facebook, Twitter and G+ authentication.

6.2. **Staff manual** – A user manual for the facility staff to get to know the system and how to issue and revoke RFID cards to users.

**Chapter 7 – Conclusion and applicability**

Using developed frameworks and APIs can drastically decrease the development time. The web system presented in this thesis is a working and finished software product, ready to be shipped.

**Chapter 8 – Additions**

8.1. **Source Code** – The thesis was created using the Atlassian Bitbucket GIT Source Control System. The full source code and the original thesis in Bulgarian can be found on the following URL <https://bitbucket.org/ivanzdravkovdiplomproject>

8.2. **Literature**

1. <http://msdn.microsoft.com/en-us/library/ff926074.aspx>
2. <http://www.w3schools.com/aspnet/mvc_intro.asp>
3. <http://www.asp.net/mvc>
4. <http://www.asp.net/web-api>
5. <http://msdn.microsoft.com/en-us/library/bb510741.aspx>
6. <http://www.asp.net/entity-framework>
7. <http://www.beabigrockstar.com/guides/aspnet-oauth>
8. <https://developer.paypal.com/>
9. <http://www.stronglink-rfid.com/download/SL500-User-Manual.pdf>