**BABEŞ-BOLYAI UNIVERSITY CLUJ-NAPOCA**

**FACULTY OF MATHEMATICS AND COMPUTER SCIENCE**

**SPECIALIZATION COMPUTER SCIENCE ENGLISH**

**DIPLOMA THESIS**

**WEB Application Development using .NET Technologies**

**Supervisor**

Assoc. Prof. Dr. Niculescu Virginia

**Author**

Laslo Marian

**2017**

**UNIVERSITATEA BABEŞ-BOLYAI CLUJ-NAPOCA**

**FACULTATEA DE MATEMATICǍ ŞI INFORMATICǍ**

**SPECIALIZAREA INFORMATICǍ ENGLEZǍ**

**LUCRARE DE LICENŢĂ**

**Dezvoltare Aplicatii WEB folosind Tehnologii .NET**

**Conducător ştiinţific**

Conf. Univ. Dr. Niculescu Virginia

**Absolvent**

Laslo Marian

**2017**

Abstract

This thesis focuses on making the recruiting process of an internship at an IT company easier for both the Human Resources team that does the recruiting and the potential intern that wants to enroll for the internship.

With the development of the IT companies in our world, junior developers (and not only) are more searched for each day. Most of the IT companies that want to recruit junior developers have at least an internship program a year. The goal of the internships is to get as many talented people into the company and make them interact with the people already working there.

The idea of the application was to create an application that makes the life easier for the companies that are recruiting by having an easy to use tool at their disposal while also keeping it simple for the potential intern who wants to join the internship program.

The result is a Web application that can satisfy both parties by providing a User Interface that is friendly looking and easy to use. The application provides a set of options for each user, depending on the role the user has been assigned to. The potential interns are notified automatically through emails when the managers change the status of their applications and when the managers schedule them for a technical test.

This work is the result of my own activity. I have neither given nor received unauthorized assistance on this work.

Babeș-Bolyai University Cluj-Napoca

Laslo Marian

Contents

[1. Introduction 6](#_Toc486229428)

[2. Web Applications 8](#_Toc486229429)

[2.1 What is a Web Application? 8](#_Toc486229430)

[2.2 Design Patterns 8](#_Toc486229431)

[2.3 The Model-View-Controller Pattern 10](#_Toc486229432)

[3. Web Applications using .NET 11](#_Toc486229433)

[3.1 What is .NET 11](#_Toc486229434)

[3.2 Advantages of using .NET in a Web Application 11](#_Toc486229435)

[3.3 Most common .NET Frameworks for developing web applications 12](#_Toc486229436)

[3.3.1 Web Forms 12](#_Toc486229437)

[3.3.2 ASP.NET MVC 12](#_Toc486229438)

[3.3.3 Web API 13](#_Toc486229439)

[3.4 ASP.NET MVC versus Web-API 13](#_Toc486229440)

[3.5 Conclusion 14](#_Toc486229441)

[4. Practical Application: InternRecruiter 15](#_Toc486229442)

[4.1 Development Process 15](#_Toc486229443)

[4.1.1 Software Requirements Specification 15](#_Toc486229444)

[4.1.2 Use Cases and Scenarios 16](#_Toc486229445)

[4.1.3 Analysis 17](#_Toc486229446)

[4.1.4 Design 18](#_Toc486229447)

[4.2 The Final Result 20](#_Toc486229448)

[4.2.1 Unregistered User 20](#_Toc486229449)

[4.2.2 Registered User – “General User” role 21](#_Toc486229450)

[4.2.3 Registered User – “Manager” role 22](#_Toc486229451)

[4.2.4 Registered User – “Administrator” role 23](#_Toc486229452)

[4.2.5 Unregistered OR Registered User – “Contact” Page 23](#_Toc486229453)

[24](#_Toc486229454)

[4.2.6 Sending Emails Automatically 24](#_Toc486229455)

[4.3 Summary 25](#_Toc486229456)

[5. What do we do next? 26](#_Toc486229457)

[5.1 Online Testing 26](#_Toc486229458)

[5.2 More than Internships 26](#_Toc486229459)

[5.3 Template Application 27](#_Toc486229460)

[6. Conclusion 28](#_Toc486229461)

[Bibliography 29](#_Toc486229462)

[Glossary 31](#_Toc486229463)

[List of Illustrations 33](#_Toc486229464)

# Introduction

**Why a web application?** The main reason is thatWeb applications provide usability for more users at the same time and with remote access. They also provide a better user experience in general and are significantly cheaper to develop than desktop applications.

From a user experience point of view, compared to the standard desktop application, which has to be installed on a computer in order to work, using a web application only requires an internet connection, a device that supports using a Web browser and access to the website that hosts the application in order to be used.

A good example of an application that has both desktop and Web support is Slack. In order to use Slack, you can either install the application on your desktop or, you can use the Web application for Slack and just access the website, login and start using it from the browser. The main advantage is that you don’t have to install anything else on the computer in order to use the application.

Another advantage of a web application is that it can be used on a large variety of devices. Nowadays, when smartphones and mobile devices in general are so common, it is worth mentioning that a web application can be accessed from any device that has a browser, this includes all the smartphones and tablets available on the market.

Furthermore, a web application can be used from multiple devices that use different architecture without having to develop the application for each one of them, which is the case for desktop applications. A good example would be to create a desktop application for both Mac OS and Windows; you would basically have to develop the application for both operating systems separately.

The purpose of this thesis is to help an IT company look for potential interns. The application should make the process of recruiting interns a lot easier for the company and for the members assigned to recruit the interns.

This thesis contains the following chapters:

**Chapter 1** – this chapter focuses on presenting the purpose and the main objective of this thesis and it also presents the advantages of Web applications.

**Chapter 2** – presents some general information about Web application development and about the MVC framework from .NET which is used in the application.

**Chapter 3** – contains information about Web applications using .NET. It also presents and compares the frameworks available in .NET for Web applications.

**Chapter 4** – this chapter presents a web application for internship applications which emphasizes the methodologies of web app development using .NET technologies; the chapter also presents how the application is structured and the interaction between the user and the application.

**Chapter 5** – presents some possible extensions of the application and how the already existing features can be improved.

**Chapter 6** – conclusions regarding the usage of the .NET technologies in Web applications based on the experience related to the developed application and also presenting some of the more difficult parts of developing the application and some final opinions.

# Web Applications

## 2.1 What is a Web Application?

A web application is an application that uses a web browser as a client [1]. The application’s complexity can vary from a simple merchandise selling application, to a complex document manager. Web applications are written in browser-supported languages like HTML, JavaScript and TypeScript because they depend on the browser to execute them [2].

There are two types of web applications, static and dynamic [2]. For example, a presentation page which only contains information about something is usually a static application because there is no need for server-side processing. On the other hand, even if the web application only contains a login or register functionality, it’s going to be a dynamic application because it requires server-side processing.

## 2.2 Design Patterns

Design Patterns are solutions to software design problems that you find in the application development of the real world. In 1994, four authors Ralph Johnson, John Vlissides, Richard Helm and Eric Gamma published a book called “Design Patterns: Elements of Reusable Object-Oriented Software” which explained the concept of design patterns in software development, these authors are known as Gang of Four (or GoF). [3]

The 23 Gang of Four design patterns are considered the foundation of all the other patterns and they are categorized into three groups, depending on the design problem they intend to solve: Creational, Structural and Behavioral. [3]

Some of the more commonly used design patterns in web applications are Singleton pattern, Adapter pattern, Inversion of Control, Dependency Injection and Façade pattern.

Singleton Pattern – This pattern is a creational type pattern and it is one of the simplest design patterns. The pattern makes sure that only a single instance of an object is created from a class, the object created can then be accessed without having to instantiate the object of the class. [3.1] Real life example: if we were to have the President of a certain country, let’s say Romania, a single instance of the “President of Romania” object would be enough, as there is only one President of Romania.

Adapter Pattern – This pattern is a structural type pattern which combines the capabilities of two independent interfaces by converting the interface of a class into another interface. [3.2]

Inversion of Control and Dependency Injection – These two patterns are said to be based on the “Hollywood Principle”, which says: “don’t call us, we’ll call you” which leaves the framework rather than the application do the invocation. They are used in order to remove dependencies from the code. [3.3] Inversion of Control is about calling the methods in a framework which will then call implementations provided by the application, instead of having the application. Dependency Injection is a form of Inversion of Control that has no object intervention and uses a framework component to pass parameters through constructors. [3.4]

Façade Pattern – This pattern is a structural type pattern and it is mostly used when there is a complex or hard to understand system that has a large number of interdependent classes. It hides the complexities of the system and presents and provides an interface for the client through which the client may access the system. [3.5]

On an Architectural level, the most common design patterns are Multilayer, Presentation-Abstraction-Control or PAC and MVC or Model-View-Controller on which we will focus later.

The **Multilayer** pattern splits the system into logical groupings of software components called layers. The layers can communicate with each other and they can also communicate with other applications. Each layer specializes in a certain kind of task; the tasks can then be divided even more by assigning them to sub-layers. The number of layers can vary from design to design, however the more common approach is to use the three layered model: [4]

* Data Layer: This layer takes care of the data access, both local and remote.
* Business Layer: System logic is split into different components and the main functionality of the application is also on this layer.
* Presentation Layer: This is the layer that takes care of the user interaction with the application and is also responsible for what the user actually sees. [4]

The **Presentation-Abstraction-Control** pattern is an architectural pattern designed for interactive systems. The pattern separates the concerns of an application into a hierarchy of cooperating components, each component being composed of a Presentation, an Abstraction and a Control. The PAC pattern’s purpose is to decompose the application into a hierarchy of abstraction. [4]

## 2.3 The Model-View-Controller Pattern

This is the most commonly used Design Pattern in Web applications and that is why we will focus more on it. MVC is a software architecture which separates the business logic of the application from the user interface. The application is separated into three parts: model, view and controller. A model represents knowledge and it could be a single object, or it could be a structure of multiple objects; it usually is a database of some kind [5]. The model takes care of the data and handles the basic behaviors of the application [6]. The view is a visual representation of the model, it handles the user interface and it does that by rendering the data from the model in a way that is convenient for the user interface. A view will have to know the semantics of the attributes of the model it represents. The controller represents the link between a user and the system, providing the user with input by arranging relevant views in appropriate places of the screen. It collects user input and handles calls to the model objects and view in order to execute fitting actions. [5]

The pattern is mainly used when the connection between the view and the rest of the application is not always available. The idea behind it is to make a clear separation between the objects that model our vision of the real world (model), display objects which are the elements we see on the screen (view) and the model objects should work without any reference to the view objects, model objects being completely ignorant of the UI. [7]

A really good and simple example of the MVC pattern being used would be a simple web page which contains HTML and CSS. The model is the HTML which is basically text given as information to the reader. The CSS is the view because it adds style to the information; this style can be switched with any other style by modifying the CSS, without having to alter the information. The browser can be seen as a controller. It mixes and renders the HTML and CSS into the final output, the web page. It also collects the input from the user and sends it to the JavaScript code needed for the page to function correctly. [5]

# Web Applications using .NET

## 3.1 What is .NET

ASP.NET comes from “Active Server Pages .NET” and is a powerful framework that was developed by Microsoft and was first released in 2002. Since then, it has built up a lot of popularity among web developers and software development companies who choose this framework because of how powerful it is in creating rich web applications and web services. The framework uses the Common Language Runtime, or CLR, which gives its users the chance to write ASP.NET code in any language supported by .NET. [8]

## 3.2 Advantages of using .NET in a Web Application

ASP.NET can be used for creating a diversity of web applications such as: web based applications, social networking websites, business and corporate websites and not only. [8]

Here are a few of its advantages in web development:

1. ASP.NET greatly reduces the amount of code needed for creating complex applications. This can improve the overall development speed and reduce its costs.[8]
2. The ASP.NET framework supports multiple languages [8], giving flexibility by allowing its users to choose a programming language that suits their interests best.
3. ASP.NET provides strong security and reliability because of its built-in Windows authentication system and allows custom configuration for every application. [8]
4. It drastically boosts overall performance by using native optimization, just-in-time compilation and smart caching technologies. [8]
5. It is constantly updated by Microsoft in order to keep up with the most recent technologies and requirements [8]
6. The .NET framework is a very mature framework which is constantly maintained and upgraded by Microsoft. Since the latest version, .NET Core, it is also open source, which gives a lot of flexibility and balance. [9]
7. Visual Studio is a great bonus. It is considered by a large group of developers to be the best IDE in the world, but of course, this is subjective. Visual Studio has advanced code generation and completion capabilities [9]
8. Visual Studio’s code generation and completion features can be improved by using a tool called ReSharper which is available for some of its supported languages.
9. C# - a powerful programming language. Its syntax is very similar to Java but with some more functionality. Just like Java, C# is truly an object oriented language, the difference being that its target is the enterprise level development. [9]
10. Since .NET has been around for a lot of years, it has managed to build a generous user base all over the world. [9]

## 3.3 Most common .NET Frameworks for developing web applications

The options presented below take care of the business part of the application (back-end frameworks).

### 3.3.1 Web Forms

Web Forms populates the Web server with ASPX pages. The users have to request the pages and navigate through them. A Web Forms application can be turned into a hybrid, with focus on the SPA (or Single Page Application) model since ASPX provides support for Ajax features. Because direct HTML and JavaScript exposure was dangerous for both applications and developers back when Web Forms was introduced, Microsoft’s Web Forms kept HTML away while also proved to have great results. [10]

### 3.3.2 ASP.NET MVC

MVC populates the Web server with controller classes and (Razor) views. This framework provides access to the pages through more descriptive URLs that refer to actions specified in the controller instead of pages. Compared to other frameworks, the difference for the users of the application is small but there is quite a difference (both good and bad) for developers. MVC can be turned into a hybrid framework focused on the SPA model by adding Ajax features. [10]

### 3.3.3 Web API

Web-API is an Application Programming Interface (API) which receives and handles HTTP calls. It has architecture close to the ASP.NET MVC architecture since it also uses actions, model binding, controllers and routing. Web-API populates the Web server only with HTML and JavaScript files. Even if a SPA application will work significantly faster than a multiple page application, the SPA is more challenging to develop because it usually works best with more advanced front-end frameworks like AngularJS, AngularJS 2, React, EmberJS and so on. [10]

## 3.4 ASP.NET MVC versus Web-API

The major differences between the two are:

* A Web API layer is created from a number of controller classes but it is derived from a different base class ApiController. The HTTP GET methods represented on a controller can return raw data without having to explicitly pass the data from an ActionController container, as in ASP.NET MVC. [10]
* In the case of Web API, content negotiation is invisible to the developer most of the time. The data is returned as a JSON by default, however, the JSON formatter can be replaced with a general-purpose of type-specific custom XML formatter. The formatters are registered in the configuration of the application when it is created. [10]
* ASP.NET MVC has some of its aspects bound to Microsoft’s Internet Information Services (IIS) therefor the IIS is the best Web server choice for an MVC application. In the case of Web API, hosting can be done outside of IIS because it abstracts the hosting environment. [10]
* Web API aims at being main part of any Web application because of its capacity to provide a HTTP-reachable back-end, the tricky part being that it still needs an HTML-rendering engine. This engine can be server-side and be taken from ASP.NET MVC or even ASP.NET Web Forms but it can also be client-side and have the shape of a single page application. [10]

## 3.5 Conclusion

In conclusion, whichever ASP.NET technology or framework you choose, you can build a strong Web application with any of them. If you’re not so familiar with JavaScript technologies, Web Forms or MVC would be the best solution because of how easy it is to configure and use them out of the box. If you’re more familiar with JavaScript or you want to have a separate server-side or client-side HTML rendering engine while also using its ability to provide HTTP-reachable back-end, Web API would be the best solution.

# Practical Application: InternRecruiter

The purpose of this thesis is to help an IT company look for potential interns. The application should make the process of recruiting interns a lot easier for the company and for the members assigned to recruit the interns.

## 4.1 Development Process

### 4.1.1 Software Requirements Specification

Specification of the software requirements is the first step when designing software applications. These specification requirements can be functionalities supported by the application, or constraints and particular cases that the application must offer a support for. There are only two types of requirements: functional, which define how the system cooperates with the environment, and nonfunctional, which define the usability and overall performance of the system. They may also include use cases which describe the software’s interactions with the user but the software requirements don’t offer any design suggestions or possible solutions to business or technology problems. [11]

The SRS is many times considered to be the parent document because every successive project management document like statements of work, software architecture specifications, design specifications, documentation plans and testing and validation plans are all related to the SRS. A well designed SRS should achieve four major goals: provide feedback to the customer, decompose the problem into component parts, serve as input to the design specifications and serve as the parent document for testing and validation strategies applied to the requirements for verification. [11]

The requirements of this application written in explicit language would look close to this:

“Create a web application which will allow a user (potential intern) to register and submit an application for different types of internships at a company. The application must support three different roles: General User (potential intern), Manager (the company’s Human Resources employees) and an Administrator (the general administrator of the application). The General User can submit an application to the available types of internships; the Manager can read, accept or reject applications submitted by the General User. The Manager should also be able to notify when the General User’s application has been accepted/rejected; the Administrator is the one that gives roles to other users (can assign a Manager role to a General user role for example) and he can also edit or delete users.

The application must run without lag or problems of any type. If something is loading in the background, the user must be notified through the UI of the application. The application must also be easily extendable.”

We can identify the functional and nonfunctional requirements from the above specification:

*Functional requirements:*

* Submit an application as a General User
* Accept/reject submitted applications as a Manager
* Schedule users to a technical test if their application was accepted
* Change the role of a user (or any other data regarding the user) as an Administrator
* Register to the application as an Unregistered User
* Access the Home/About/Contact pages as any type of User

##### *Nonfunctional requirements:*

* The application must be easily extendable

### 4.1.2 Use Cases and Scenarios

A scenario is a scene which portrays an interaction with a proposed system or user. It is also a tool used to describe a certain use of a proposed system by capturing the system viewed from the outside, e.g., a user. A use case is a task which needs to be performed by the actor with the help of the system, the actor is a user of a system belonging to a particular role and it can be a person or it can be an external system. It must be mentioned that the actor is a role not an individual and the actor must be beneficiary of the use case. [12]

The next step is to identify the actors and use cases matching the functional requirements, by analyzing the above functional requirements. See illustration 4.1.

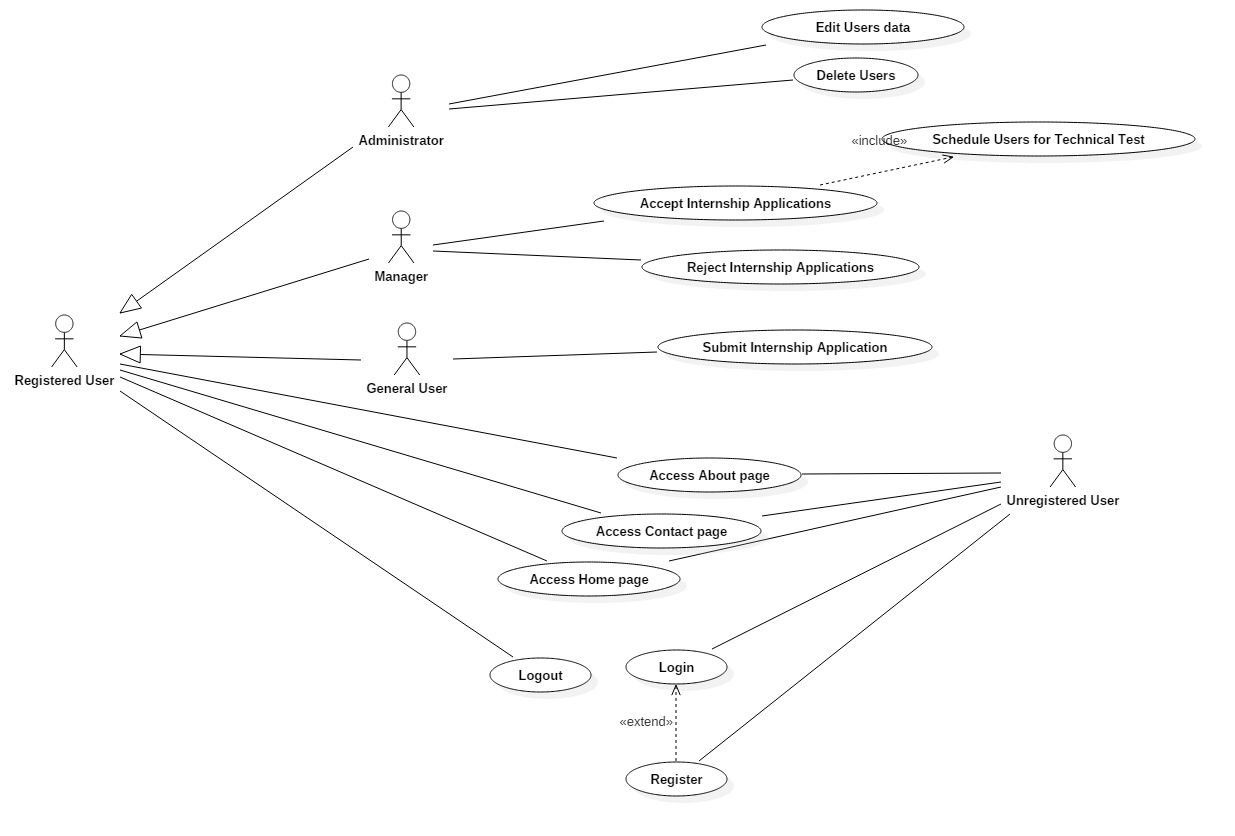


Illustration - Use Case diagram

### 4.1.3 Analysis

The requirements specification is further refined in the Analysis stage of development by making sure that the requirements are correctly defined and complete and that they are consistent and not ambiguous. In this phase, the developers have to formalize the requirements specification and examine in more detail the exceptional cases. They also clarify, validate and correct the requirements if needed. [13]

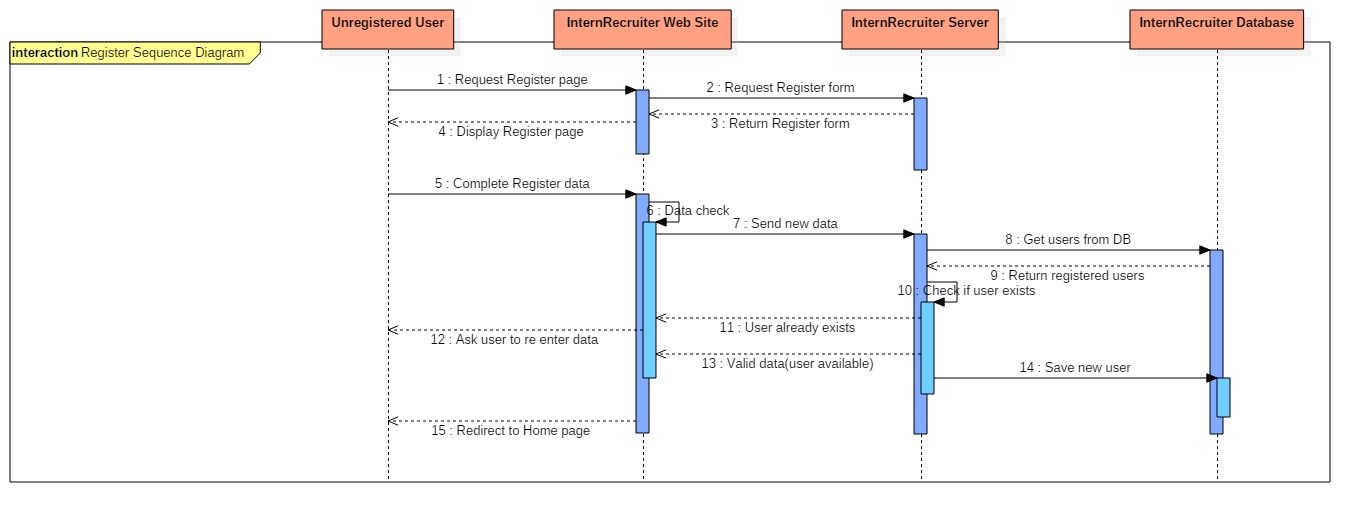
 In order to connect the users to objects, a Sequence diagram can be used. The diagram shows how the objects interact with each other over a period of time which makes the order of interactions easier to understand while also showing a lifespan of each object. [13]

Illustration 2 - Sequence Diagram for the Registration Process

The Sequence diagram in Illustration 4.2 shows the Registration process.

### 4.1.4 Design

Module diagrams are a good way to illustrate the structure of the application. We do this by showing the dependencies between different models from the application.

The interaction of the models can be seen in Illustration 4.3.

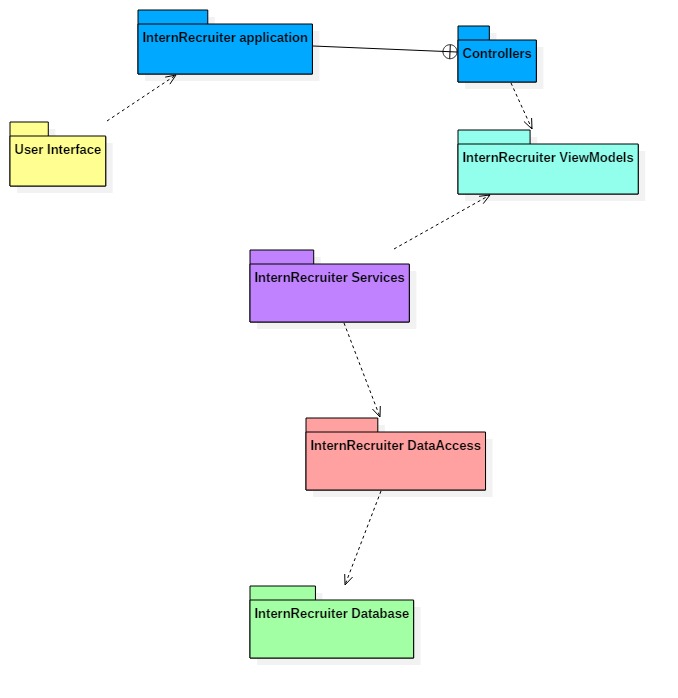


Illustration - Module diagram of the application

The application uses Entity Framework with the Database First approach. What this means is that we first create a table in the database, for example, a table that will hold the users. The User class is then generated from the table through Entity Framework Database First, by transforming the table properties into fields; when the database schema changes, the classes must also be updated. This is the DataAccess module which connects to the Database module.

The Services module encapsulates the business logic and makes the work of the Controllers easier. The Controllers have to use this layer in order to get the data from the Database through the DataAccess module.

The Controllers take the data from the Database through the Services module and the DataAccess module and then they pass it to the User Interface or Views.

ViewModels represent how the data is going to be displayed on the Views (User Interface). They are, as their name say, models for the views and they only represent the data that must be also shown on the Views.

## 4.2 The Final Result

When the application is started, it will launch the Home page on the chosen Browser. The Home page presents the application’s goal and it also gives some details about how to get started, either as an unregistered user or a newly registered user.

### 4.2.1 Unregistered User

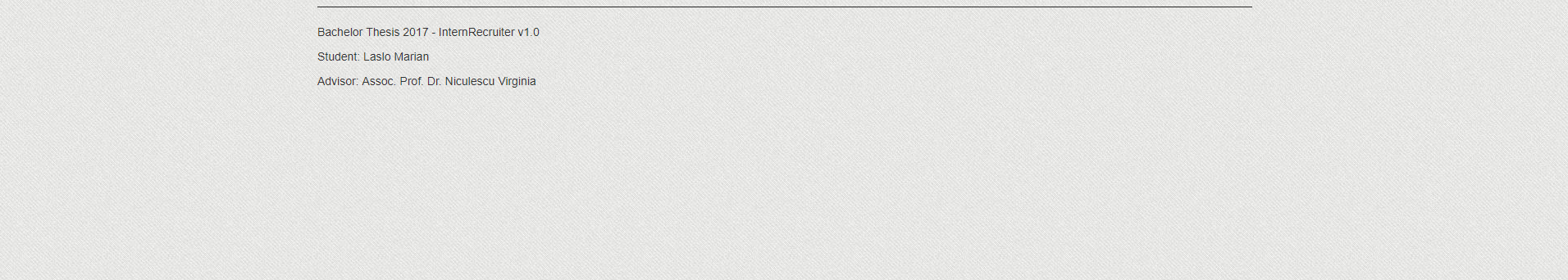
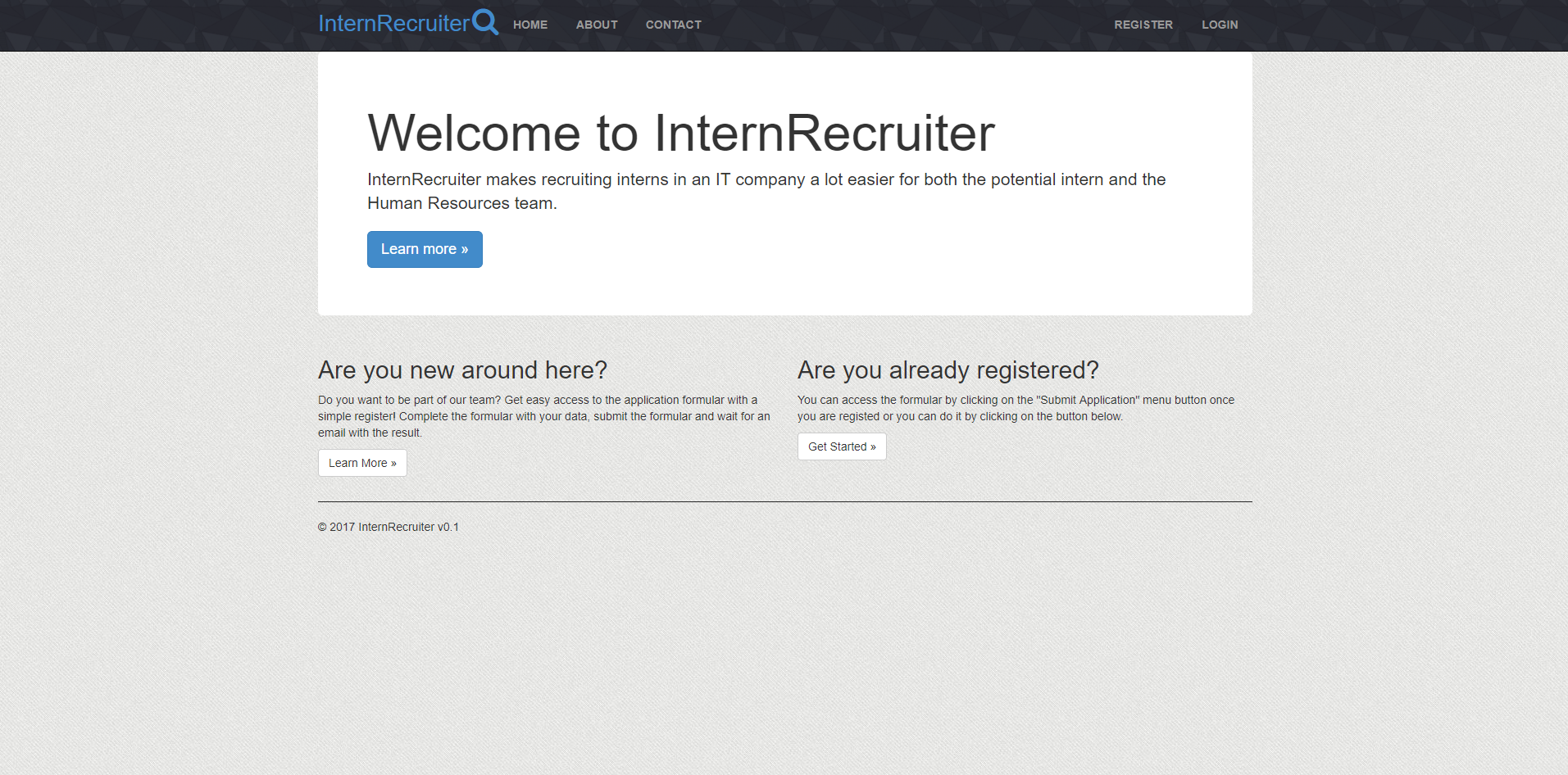
**** The application also has a navigation menu on the top of the page which may be different depending on the user role and if the user is registered or not.

Illustration 4 - Home page for unregistered user; navigation menu differs

### 4.2.2 Registered User – “General User” role

Every newly registered user has the “General User” role by default, which means that the user can submit applications for the internship and can also change the password if he wishes to. Every user, registered or unregistered, can access the “Home”, “About” and “Contact” pages, no matter what role the user is assigned to.

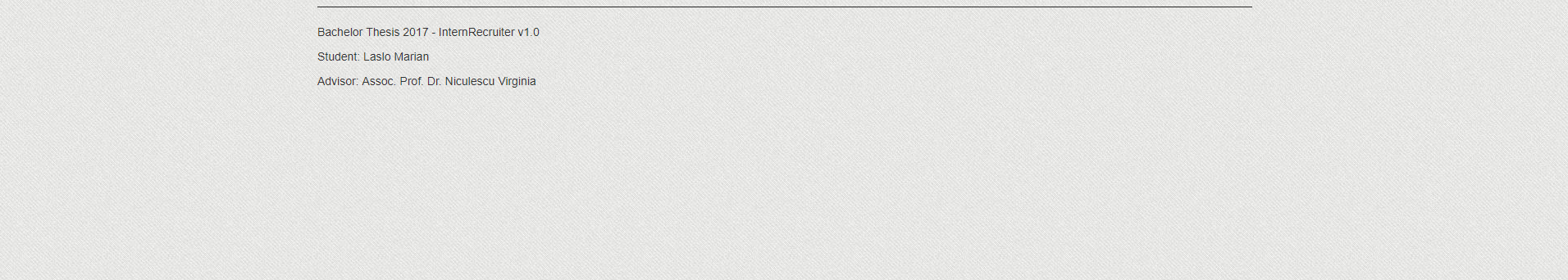
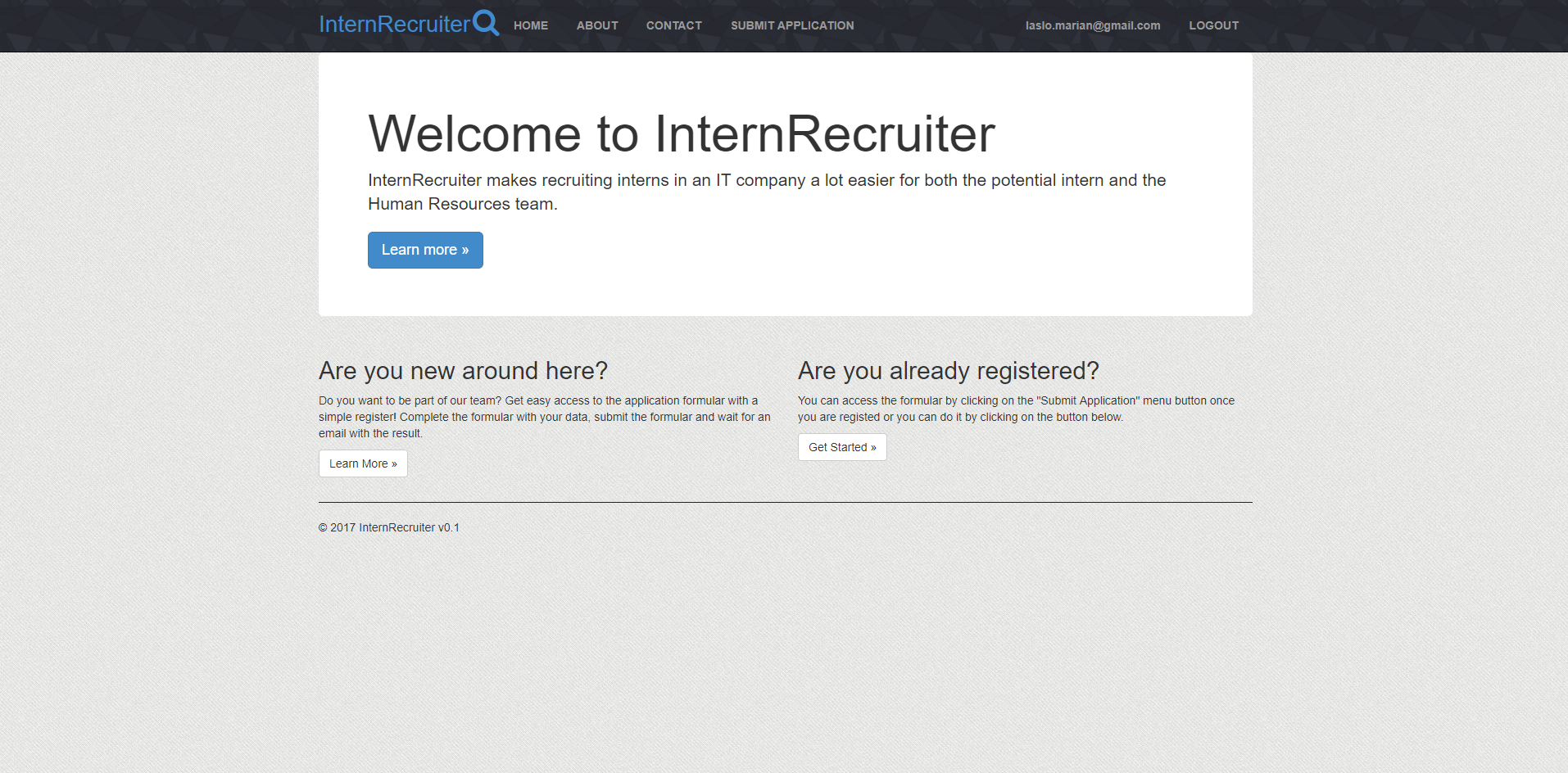
****If the user logs in with an account assigned to the “General User” role, the “Submit Application” button shows up on the navigation menu. The “Submit Application” page will allow the user to submit a form-type application which will contain the personal data of the user and some questions regarding his previous experience in the domain, if he has had any.

Illustration 5 - Home page for General User

### 4.2.3 Registered User – “Manager” role

If the user logs in with an account assigned to the “Manager” role, the “Submit Application” button doesn’t show up anymore, instead, the “Manage Applications” button will show up. This page will allow a “Manager” to accept or reject internship applications submitted by the users in the “General User” role. They can also schedule the users for a technical test if they internship application has been accepted.

The “Manage Applications” page presents each application registered to the application. The page also shows what the status of each application is, the schedule date and if it has been scheduled or not and it provides the option to view the application of each user in order to be accepted or scheduled by clicking on the “Details” button.

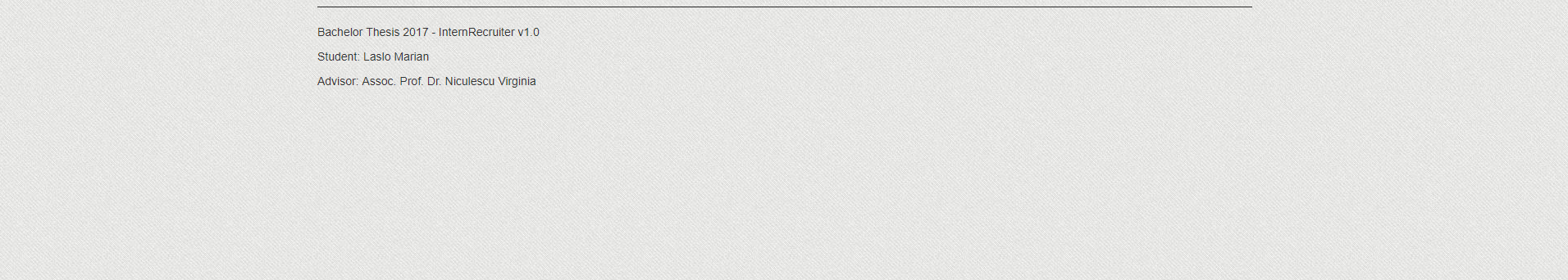
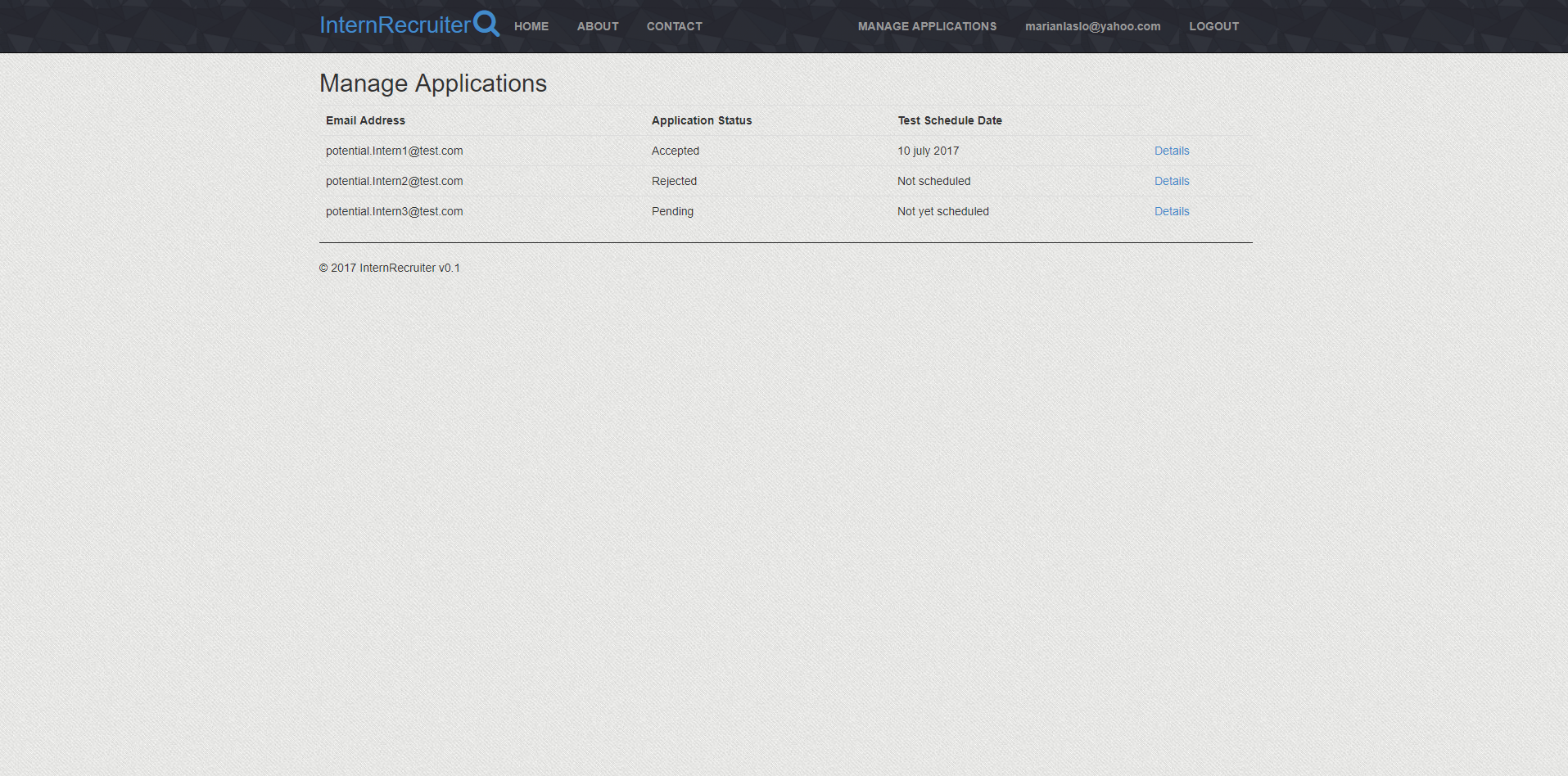
****The “Manager” role is given by the “Administrator” of the application by accessing the “Manage Users” page. The “Administrator” should know which users are from the Human Resources team and assign them the “Manager” role after they create their account.

Illustration 4.2.3 Manage Applications page for the Manager role

### 4.2.4 Registered User – “Administrator” role

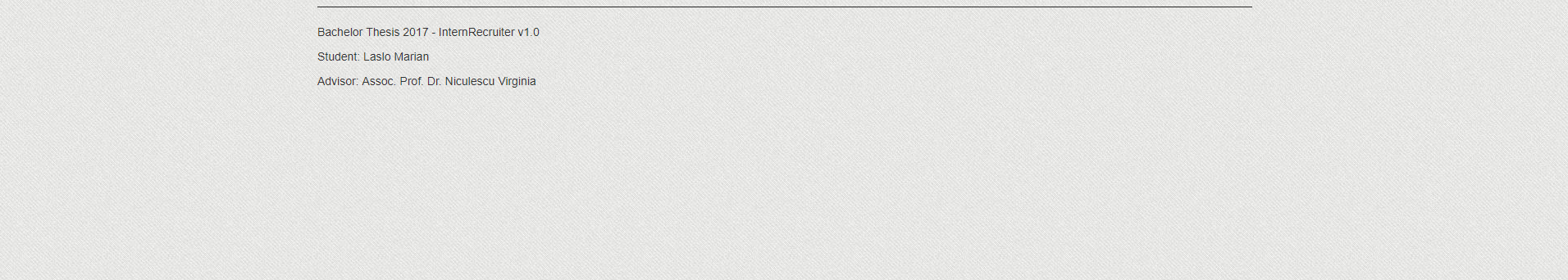
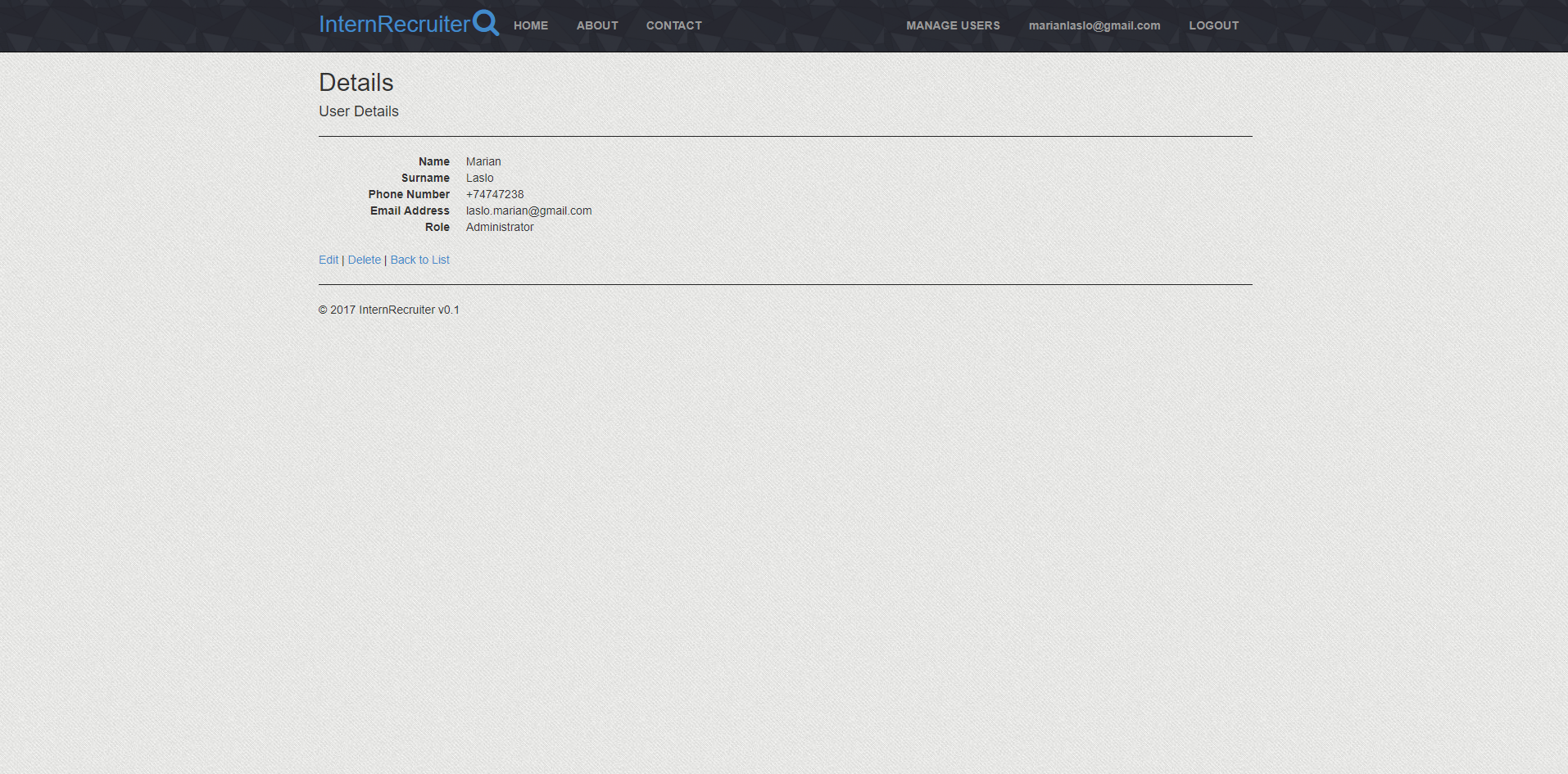
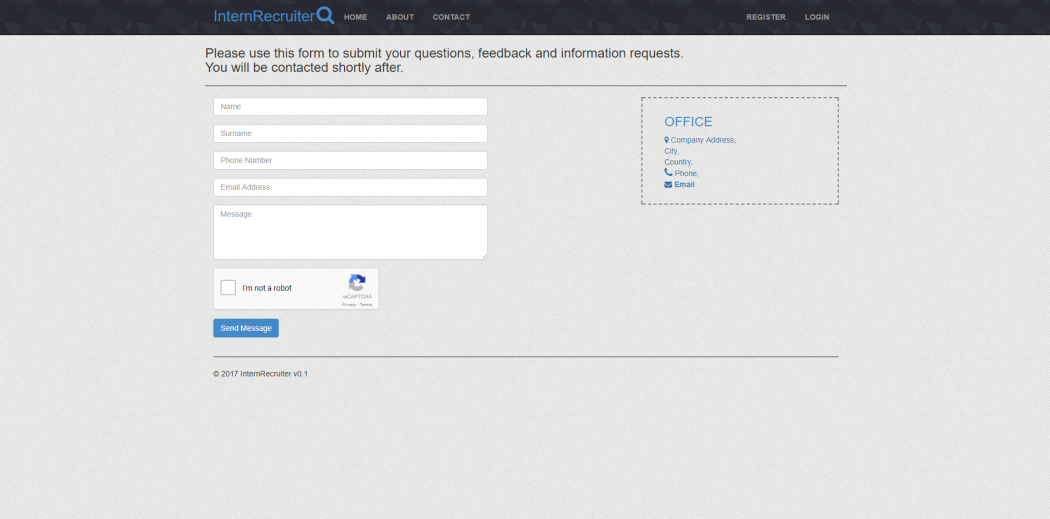
****The application will have only one user assigned to the “Administrator”. If the user logs in with the account assigned to this role, the “Manage Users” button will be displayed. This page will allow the “Administrator” to delete users created users or edit their data.

Illustration 6 - Manage Users page for Administrator when user is selected

### 4.2.5 Unregistered OR Registered User – “Contact” Page

After an internship application is submitted, the “General User” that submitted it will be notified through an email if the application was accepted or rejected. If the application is accepted, the user will also receive a different email containing the schedule information. The schedule date may be changed if the user doesn’t agree with it by responding to the initial email or by accessing the contact page of the application.

The Contact page offers the possibility for both registered and unregistered users to contact the company. This can be done directly through the “Contact Form” from the page which sends an email containing your name, surname, phone number, email and message to the company email address. In order to prevent mail spam and abuse to the company, the form uses the Google reCAPTCHA service. Google reCAPTCHA is a FREE service provided by Google in order to protect your application against spam, bots and abuse. Not only it protects from bots and abuse but at the same time it also lets the valid users pass through without trouble. [14]

### 

Illustration 7 - Contact Page available for all users

### 4.2.6 Sending Emails Automatically

The application uses .NET’s Simple Mail Transfer Protocol (SMTP) Class which allows applications to send emails using a FREE SMTP Server. Whenever an internship application is set to the “Rejected” or “Accepted” status, the users whom have had their application set to this status receive an email to notify them about the application’s situation.

The application sends the mails by combining the SMTP Server and .NET’s SmtpClient Class. Mails are also sent automatically to the users when a “Manager” schedules a date for the technical test.

## 4.3 Summary

The starting point was the requirements specification which has then been transformed into a list of more informal requirements. These requirements were then processed even further for a better understanding of the desired application, and finally, we developed the application into a solution that can be easily extended.

# What do we do next?

The application may look complete but there is always room for improvement. Even if the application meets all the requirements, it is worth thinking about how it can be further optimized and expanded.

The application is currently developed to look for interns in a certain company. A potentially difficult task would be to transform the application into a template-like application. This means that the application could be adapted for any company that is looking for potential interns without having to the structure and core of the application.

## 5.1 Online Testing

The application could be developed even further by allowing the users to take an online technical test on the application after their application has been accepted. The test would have a given number of questions and problems and it would also have a timer that would start running once you’ve started the test.

We could do this by generating a test for each user after their application has been set to “Approved” and we could send the link to the generated test through the email. The testing page would be basically a single page that would change its content when the user goes to the next set of questions and when there are no more questions, the test would be completed. If the timer reaches “00:00” before the user completes all the questions, the user would get no points for the questions he didn’t answer for. The user could also have the possibility to navigate forward and backward through the sets of questions.

The results of each test would be held on a page “Manage Tests” on which the “Manager” would be presented with each user who has completed a test, the score to the test and the completion time. The users could be ordered based on the score and completion time so that the managers could see which user got the most points in the shortest times.

## 5.2 More than Internships

Another possibility of expanding would be to allow users to apply for employment interviews, not only internships. Of course, there would still be a need of interaction between the company and the user who is applying for such an interview, so a face to face interview would still be necessary but only the final interview would be face to face.

## 5.3 Template Application

A more complex feature would be to make the application behave like a template. This template would allow different companies to use the same “Portal” application in order to recruit interns. The application would have the same features but the pages that are company related would be different for each company.

This could be done by getting the text for the “About”, “Contact”, “Home” pages and the internship application questions from files. A company could login with a company username and create a company environment. The files containing the company data and desired questions would be added to the environment and it could use the application to recruit interns. Another company can use the same application but on a different environment, with different files and have the same features and functionalities.

# Conclusion

Considering the research presented in this thesis, we can tell how easy building a Web application using .NET technologies is. You can choose between three main frameworks: MVC, Web-API and Web Forms and no Matter which technology, you can build a solid Web application with any of them but. Depending on what the application should do and what technologies you are familiar with, one may favor the other, but in the end, all of them are good options when developing such an application.

It is obvious that the demand for junior developers and for interns on the current IT market is high. The need of developers is still on the rise and companies invest a lot in recruiting the best from when they are younger so that they can teach them their ways.

On the other side, there are more and more potential interns each year. The world of computer has attracted more and more people over the years and now, with the IT market still on the rise, the number of developers is raising fast, and even so, it is still not enough to satisfy the market.

We see more and more companies offering paid internships to students and a lot of other benefits that come with the internship, like giving the opportunity to stay in the company after finishing on either a part-time or full-time program.

Looking back at where we were when the IT market started developing in our country, and where we are now, the difference is enormous. The world of computers has attracted a lot of people in the last few years and we can even say that we depend on computers and on the Internet.

# Bibliography

[1] Daniel Nations, October 17, 2016, “Improve Your Understanding of Web Applications”, <https://www.lifewire.com/what-is-a-web-application-3486637>

[2] Amos Ndegwa, May 31, 2016, “What is a Web Application?”, <https://www.maxcdn.com/one/visual-glossary/web-application/>

[3] Shailendra Chauhan, July 14, 2013, “Gang of Four (GOF) Design Patterns in .Net”, <http://www.dotnettricks.com/learn/designpatterns/gang-of-four-gof-design-patterns-in-net>

[3.1] Shailendra Chauhan, June 8, 2013, “Singleton Design Pattern”, <http://www.dotnettricks.com/learn/designpatterns/singleton-design-pattern-dotnet>

[3.2] Shailendra Chauhan, July 8, 2013, “Adapter Design Pattern”,

<http://www.dotnettricks.com/learn/designpatterns/adapter-design-pattern-dotnet>

[3.3] Simone Busoli, July 24, 2007, “Inversion of Control and Dependency Injection with Castle Windsor Container - Part I”, <http://dotnetslackers.com/articles/designpatterns/InversionOfControlAndDependencyInjectionWithCastleWindsorContainerPart1.aspx>

[3.4] Garrett Hall, July 1, 2011, “Inversion of Control vs Dependency Injection”,

<https://stackoverflow.com/questions/6550700/inversion-of-control-vs-dependency-injection>

[3.5] Shailendra Chauhan, July 14, 2013, “Façade Design Pattern” ,

<http://www.dotnettricks.com/learn/designpatterns/facade-design-pattern-dotnet>

[4] Connor Patrick, University of Cape Town, May 15, 2014, “Service Frameworks and Architectural Design Patterns in Web Development”, <https://people.cs.uct.ac.za/~cpatrick/Honours/files/LiteratureReviewPTRCON001.pdf>

[5] Jeff Atwood, May 5, 2008, “Understanding model-view-controller”, <https://blog.codinghorror.com/understanding-model-view-controller/>

[6] Dave Jarvis, January 9, 2013, “What is MVC?”, <https://softwareengineering.stackexchange.com/questions/127624/what-is-mvc-really>

[7] Martin Fowler, July 18, 2006, “GUI Architectures”, <https://martinfowler.com/eaaDev/uiArchs.html>

[8] Alex Seryj, Editor in Chief at QArea, August 8, 2015, “Advantages of using Asp.NET in Web Development”, <https://qarea.com/articles/advantages-using-aspnet-web-development>

[9] Matt West, Quora, January 7, 2015, “Advantages of using Asp.NET over other frameworks”, <https://www.quora.com/What-is-the-advantage-of-using-ASP-NET-over-other-web-application-frameworks>

[10] Leonardo Esposito, Code Magazine, November/December 2013, “Planning Web Solutions today”, [http://www.codemag.com/Article/1312051](%20http://www.codemag.com/Article/1312051)

[11] Donn Le Vie Jr., August 29, 2010, “Writing Software Requirements Specifications (SRS)”, <https://techwhirl.com/writing-software-requirements-specifications/>

[12] William Y. Arms, Cornell University, 2014, “Scenarios and Use Cases”, <https://www.cs.cornell.edu/courses/cs5150/2014fa/slides/D2-use-cases.pdf>

[13] Bernd Bruegge, Allen H. Dutoit, Object-Oriented Software Engineering Using UML, Patterns, and Java (2009, third edition), <http://dbmanagement.info/Books/MIX/POO_Software_Engineering_Using_UML_Patterns_and_Java_3rd_Edition.pdf>

[14] Google reCAPTCHA presentation website, <https://www.google.com/recaptcha/intro/android.html>

# Glossary

**.NET** – Software framework developed by Microsoft which runs mainly on Windows

**Intern** – a trainee who works, sometimes without pay, in order to gain work experience

**UI** – User Interface

**Slack** – more complex instant messaging software that also supports groups

**Mac OS** – The Operating System used on Apple’s Macintosh devices.

**HTML** – HyperText Markup Language, standard markup for creating web pages.

**JavaScript** – A programming language that runs on the browser and makes web pages interactive.

**TypeScript** – It is defined as “JavaScript that scales”. It is a typed superset of JavaScript which compiles to plain JavaScript and is mostly used in the newer versions of Angular (2+)

**Browser-supported languages** – Programming languages that run on browser

**Server-side processing** - The processing is done on the server

**CSS** – Stands for Cascading Style Sheets and is used to stylize web pages

**CLR** – Common Language Runtime, virtual machine component of the .NET framework

**IDE** – Integrated Development Environment, software application which supplies software developers with facilities

**Just-in-time compilation** - Compilation is done during run time rather than before the execution

**Razor** – A viewing engine used by ASP.NET

**Back-end** – The business logic of the application

**API** - a set of tools or libraries that assist developers when building software applications

**HTTP** – HyperText Transfer Protocol, defines how messages are transmitted over the World Wide Web

**HTTP request** – An HTTP client sends an HTTP request to a server which has a specific format

**IIS** – Internet Information Services, web server created by Microsoft

**PAC** – Presentation-Abstraction-Design, an architectural pattern used in software development

**MVC** – Model View Controller, another architectural pattern commonly used in software development

**SMTP** – Simple Mail Transfer Protocol

# List of Illustrations

[Illustration 1 - Use Case diagram 17](#_Toc486219776)

[Illustration 2 - Sequence Diagram for the Registration Process 18](file:///C:\Users\Toshin\Desktop\Bachelor-Thesis-Laslo-Marian.docx#_Toc486219777)

[Illustration 3 - Module diagram of the application 19](#_Toc486219778)

[Illustration 4 - Home page for unregistered user; navigation menu differs 20](file:///C:\Users\Toshin\Desktop\Bachelor-Thesis-Laslo-Marian.docx#_Toc486219779)

[Illustration 5 - Home page for General User 21](file:///C:\Users\Toshin\Desktop\Bachelor-Thesis-Laslo-Marian.docx#_Toc486219780)

[Illustration 6 - Manage Users page for Administrator when user is selected 23](file:///C:\Users\Toshin\Desktop\Bachelor-Thesis-Laslo-Marian.docx#_Toc486219781)

[Illustration 7 - Contact Page available for all users 24](file:///C:\Users\Toshin\Desktop\Bachelor-Thesis-Laslo-Marian.docx#_Toc486219782)