|  |
| --- |
| **UniVERsity of central missouri** |
| **Chatbot for Appointment Management System** |
|  |
| **Under the guidance of Dr. SAM RAMANUJAN** |
| **(05/02/2017)** |

**By,**

**Nishanth Pavurapu**

**(700648212)**

# INDEX

1. ABSTRACT 2

2. SYSTEM REQUIREMENTS 2

3. SYSTEM ANALYSIS 7

4. SYSTEM DESIGN 9

5. IMPLEMENTATION 17

6*.* testing 21

7. OUTPUT SCREENS 25

8. Conclusion 27

9. Future ENHANCEMENTS 28

10. rEFERENCES 29

# 1. **ABSTRACT**

To create an intelligent and interactive chatbot which interacts with the user and completes actions for his requests through conversations. This project has two parts; one is the appointment management system which allows the users is schedule appointments with other users registered with the system. Users here can be of any type students/professors etc., but should possess a valid email address.

Secondly the vital part of the project is to create an intelligent and Natural Language understandable chatbot, and thereby integrating it with the appointment management system. This project allows users to schedule an appointment, cancel their appointment and view his appointments. The aim of this project is to develop a web based application which manages the overall enrolment system and flexible communication of students with professors regarding enrolments.

# **2. SYSTEM REQUIREMENTS**

## 2.1 Functional Requirements:

1. User can login in two modes as a AMS (Appointment Management System) user and as a admin.
2. User with valid credentials are only allowed Authentic user only can login into system. Validates user by searching in database.
3. If user can successfully have logged in he/she can able to do the following functions.

* User can schedule an appointment with another existing user of AMS.
* User can cancel his previous appointment.
* User can re-schedule his previous appointment which was scheduled earlier.
* User can view his appointments for a given selected date.
* User can view and edit his profile information.

1. On successful user login, user can invoke his personal chatbot, which can perform the below operations with natural language conversation.

* Chatbot can schedule the user’s appointment with another existing user of AMS.
* Chatbot can cancel a user appointment which was scheduled.
* Chatbot can re-schedule the user’s previous appointment to another slot which was scheduled earlier.
* Chatbot can show the list of appointments scheduled for the user on a given selected date.

1. At any point of time user can go to home by clicking on the application logo located at top left corner.
2. At any point of time user can be able to logout by clicking the logout button, which is made available the menu bar located at top right corner.

## 2.2 Non-Functional Requirements:

* The website and the chatbot assistant should support interruptions for regular laptop/desktop device tasks. (performance)
* All the services required for website to function needs to available 24 X 7. (Reliability/ availability)
* System is reliable if network connections are maintained and access the scenario until client kills his connection. (Reliability).
* The cost of product maintenance and development should be minimum. As the system uses all free tools to develop, hence production cost is low. (Cost)

## 2.3 Assumptions

* Users must have an internet connection, which provides the connectivity between all the services of system/website.
* Users will be required a valid login credentials to access the features of the system.

## 2.4 Software Requirements

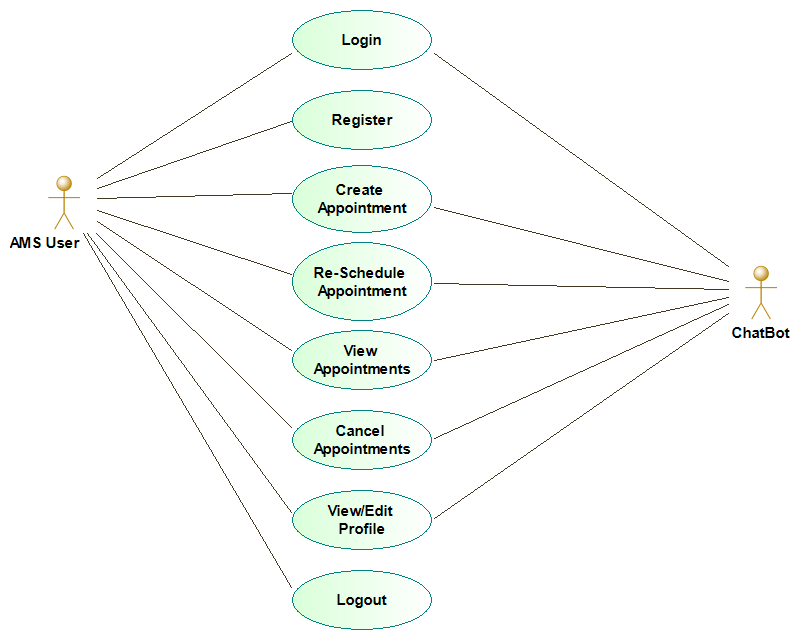
* Operating System : Windows
* Languages : C-Sharp (C#)
* Web Technologies : ASP.NET, BootStrap, JQuery
* Application Server : Microsoft Azure, IIS
* Scripting : HTML, CSS, JavaScript
* Development framework : .NET Framework, LUIS.ai
* Database : Microsoft SQL Server
* IDE : Visual Studio

## 2.5 Hardware Requirements

* Hardware : Minimum of Pentium 4 processor
* Hard Disk : 1GB or above
* RAM : 256MB (minimum)
* Internal Memory : 256MB or more

## 2.6 Detailed Software Requirements

*2.6.1 Use case model:*



*2.6.2 Actors:*

|  |  |
| --- | --- |
| **Actor Name** | **Description** |
| Standard User(AMS) | An authorized AMS user can be able to login/logout/create/re-schedule/view/cancel the appointment. They allowed to view/edit their profile. |
| Chatbot | Chatbot can be able to login/create/re-schedule/view/cancel the logged in user’s appointment. |

# 3. SYSTEM ANALYSIS

## 3.1 Existing System:

In the current system, we have many other appointment management systems for handling appointments between users of an organization. We have both desktop and web applications for this use case. But we don’t have any system combined with a virtual assistant which can do the transaction based on user conversation in natural language.

## 3.2 Proposed system:

The proposed system is a web based appointment management application used by the registered user of AMS. The main aim of this project is to create a web based application to create, view and re-schedule, cancel appointments with the other registered user on AMS and along with a natural language understandable chatbot which can.

## 3.3 Design and Implementation Constraints:

All modules are being developed and implemented thoroughly based on preliminary requirements gathered from brain storming. The application is designed in such a way that the user will be able to easily interact with the screen and also with the natural text conversation.

***3.5.1 User documentation***

We are planning to provide a user guide manual for basic understanding of the system, which help the user in keep going with the application and not get blocked. Online help will be also provided if the system is made available in the real-time environment. We are going to explain each and every step clearly about our product so that any user can easily understand it.

## 3.4 Module description:

**Module 1: Standard user login using Website.**

If the AMS user login using the website below are the functions which they can perform.

* User can create/re-schedule the appointments with other users of AMS using the navigation menu. Length of one single appointment is fixed (30 Minutes) for this release.
* Edit/View Profile using the navigation menu.
* View the list of scheduled appointments.
* Cancel the scheduled appointments.

**Module 2: User login using Chatbot**

If the AMS user login using the chatbot below are the functions which they can perform with the help of bot.

* Chatbot allows user to login in to AMS and appointment transactions
* Chatbots helps user in creating/re-scheduling the appointments for the user. Chatbot also helps the user in canceling the scheduled appointment.
* Chatbot also lists all the scheduled appointments for a day.

**Module 3: AMS Web API Service**

This module of the application acts as a bridge between the AMS website and the Chatbot. Bot uses this API to query and manipulate the AMS data. All the communication is done using JSON objects. Below are the functions this module provides.

* Creating/re-scheduling an appointment using GET request.
* Cancelling an appointment using GET request.
* Viewing scheduled appointments based on a particular date/ with a specific user for a given day.

**Module 4: AMS BOT Service**

This module of the application is responsible for receiving and sending communication to/from user using the Bot console. Below are the functions performed with this module.

* Communicate with the user using the Bot Console.
* Process the conversation with AMS Web API and LUIS.ai endpoint.

# 4. SYSTEM DESIGN

## 4.1 Class Diagram and Entity Diagram:

Class diagrams classify the class structure of a system, together with the properties and methods of each class. Also portrayed are the various relationships that can occur between classes, such as an inheritance (IS-A) relationship. The Class diagram is one of the most broadly used diagrams from the UML specification.

## Classes and their Classification:

* **Boundaries:**

Boundary classes handle the communication between actors and the system's internal components. There might be user interfaces, system interfaces or device interfaces (for example). They are identified by each actor–use-case pair on the system's use-case diagram, with one and only one boundary class existing for each pair.

Boundary classes - sender and receiver.

* **Entities**

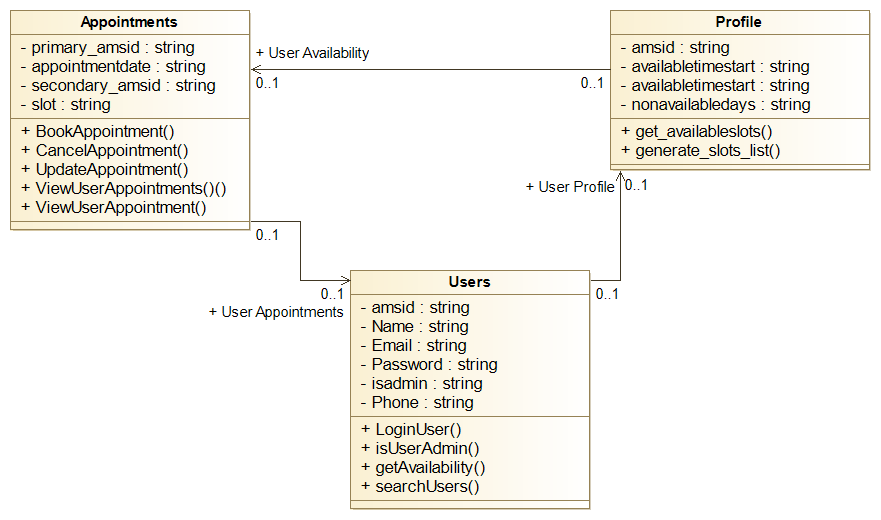
Entity classes model the information handled by the system, and sometimes the behavior associated with the information. They should not be identified as database tables or other data-stores.

Entity class - encrypts and decrypt

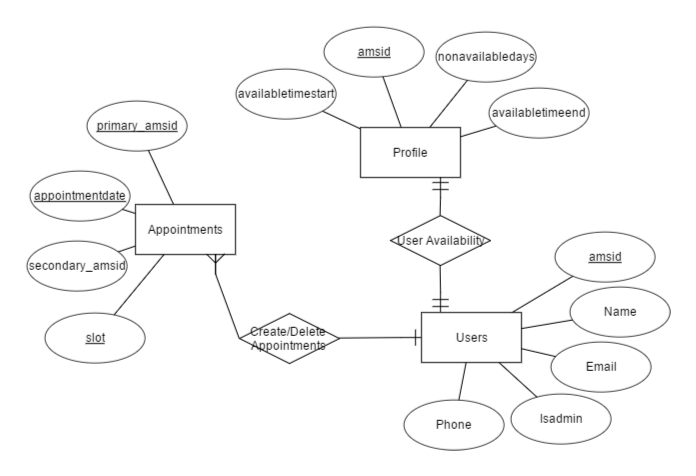
* **Controls**

Control classes handle the flow of control for a use-case and can therefore be coordinating representation classes. These do not do everything in the use case, but co-ordinate with other classes that can do the work for them.

**Class Diagram**

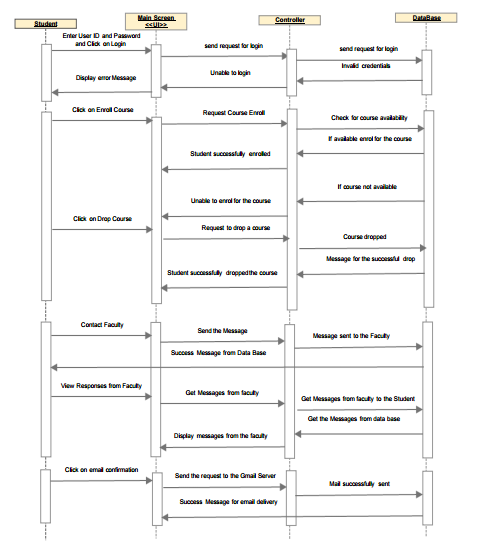


**Entity Diagram**

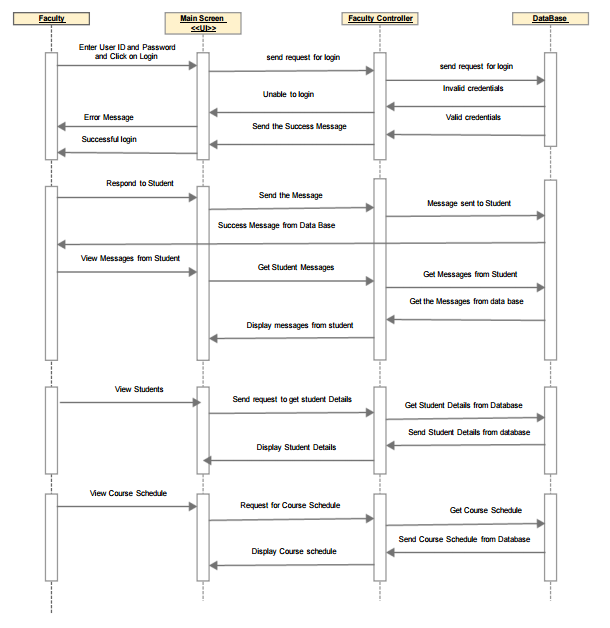


**4.2 Sequence Diagram:**

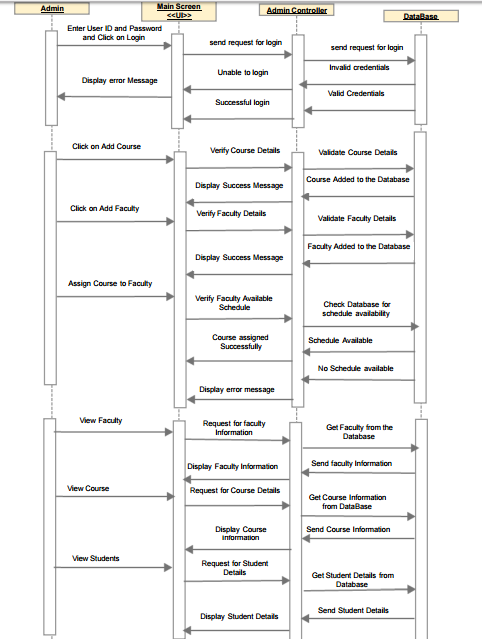
Sequence diagrams document the interactions between classes to achieve a result, such as a use case. The Sequence diagram lists objects horizontally, and time vertically, and models these messages over time.



**Sequence Diagram for Faculty:**

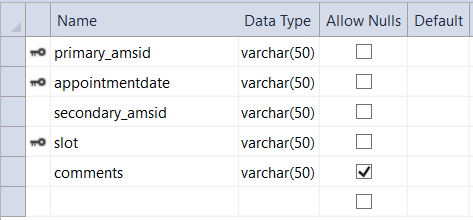


**Sequence Diagram for Admin:**



**Tables Structure**

**Appointments**



CREATE TABLE [dbo].[Appointments] (

[primary\_amsid] VARCHAR (50) NOT NULL,

[appointmentdate] VARCHAR (50) NOT NULL,

[secondary\_amsid] VARCHAR (50) NOT NULL,

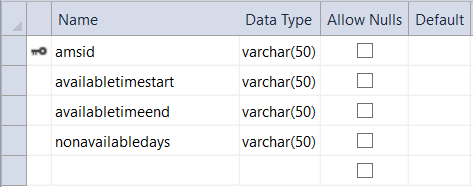
[slot] VARCHAR (50) NOT NULL,

[comments] VARCHAR (50) NULL,

CONSTRAINT [PK\_Appointments] PRIMARY KEY CLUSTERED ([primary\_amsid] ASC, [appointmentdate] ASC, [slot] ASC)

);

**Profile**



CREATE TABLE [dbo].[profile] (

[amsid] VARCHAR (50) NOT NULL,

[availabletimestart] VARCHAR (50) NOT NULL,

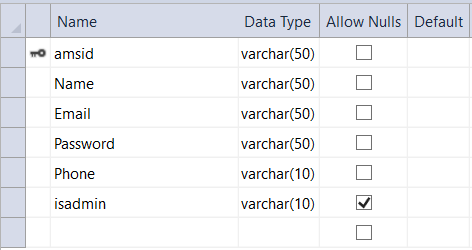
[availabletimeend] VARCHAR (50) NOT NULL,

[nonavailabledays] VARCHAR (50) NOT NULL,

PRIMARY KEY CLUSTERED ([amsid] ASC)

);

**Users**



CREATE TABLE [dbo].[Users] (

[amsid] VARCHAR (50) NOT NULL,

[Name] VARCHAR (50) NOT NULL,

[Email] VARCHAR (50) NOT NULL,

[Password] VARCHAR (50) NOT NULL,

[Phone] VARCHAR (10) NOT NULL,

[isadmin] VARCHAR (10) NULL,

PRIMARY KEY CLUSTERED ([amsid] ASC)

);

# 5. IMPLEMENTATION

## 5.1 Selected Software:

* Languages : C#
* Web Technologies : ASP.NET, JQuery, WCF
* Application Server : IIS Express
* Scripting : HTML, CSS, Bootstrap, Java Script
* Development framework : .Net Framework, Microsoft Bot Framework,

LUIS.ai

* Database : Microsoft SQL Server
* IDE : Visual Studio

## 5.2 Creating Projects

* **Creating a ASP Web project**
  + Open Visual Studio IDE
  + Go to the File menu, then New, and select Website.
  + New Project wizard window opens, select “ASP.NET Web Forms Site”.
  + Give a project name and click OK.
* **Creating a WCF project**
  + Open Visual Studio IDE
  + Go to the File menu, then New, and select Website.
  + From the templates gallery, select WCF Service
  + Give a name for the project and click OK.
* **Creating a Bot Application**
  + Open Visual Studio IDE
  + Go to the File menu, then New, and select Project.
  + From the templates gallery, select Bot Application
  + Give a name for the project and click OK.

Save the entire project as a solution and give a solution name. Set the bot project as a startup project and use the solution to build the project and run the project.

**5.3 Running your application**

Before running the application, we must build the solution, use build menu from the menu and select “build solution”. Use the Debug from the overhead menu and Select “start debugging”.

**5.4 .Net Framework**

*5.4.1. Overview*

ASP.NET is a unified Web development model that includes the services necessary for you to build enterprise-class Web applications with a minimum of coding. ASP.NET is part of the .NET Framework, and when coding ASP.NET applications you have access to classes in the .NET Framework. You can code your applications in any language compatible with the common language runtime (CLR), including Microsoft Visual Basic and C#. These languages enable you to develop ASP.NET applications that benefit from the common language runtime, type safety, inheritance, and so on.

.NET is based on standards and a unified programming model. Web standards are open and non-proprietary, they are built to be first-class constructs and they drive interoperability and integration of new existing systems. .NET uses open architecture and it provides a robust, open platform for developing the next generation of Web Application, XML Web services and Windows Applications. .NET is also multi language because it works in the language of your choice.

.NET will allow you to create programs that transcend device boundaries and harness the connectivity of the Internet in your applications. Furthermore, it is viable for all your applications, as you no longer need to think about two separate infrastructures—one for Web applications and another for internal or desktop applications.

The .NET Framework is a technology that supports building and running the next generation of application and XML web services. It is used to fulfill the following objectives.

* To provide a consistent object-oriented programming environment whether object code is stored and executed locally, executed locally but Internet-distributed, or executed remotely.
* To provide a code-execution environment that minimizes software deployment and versioning conflicts.
* To provide a code-execution environment that promotes safe execution of code, including code created by an unknown or semi-trusted third party.
* To provide a code-execution environment that eliminates the performance problems of scripted or interpreted environments.
* To make the developer experience consistent across widely varying types of applications, such as Windows-based applications and Web-based applications.
* To build all communication on industry standards to ensure that code based on the .NET Framework can integrate with any other code.

*5.4.1. Main Characteristics of .NET*

* **Cross-platform**

.NET provides key functionality to implement the app features you need and reuse this code regardless of the platform target.

* **Open Source**

.NET is one of the many projects under the stewardship of the .NET foundation.

* **Flexible Deployment**

We can deploy in two main ways: framework-dependent deployment or self-contained deployment.

* **Modular**

It is modular because it is released through NuGet in smaller assembly packages. Rather than one large assembly that contains most of the core functionality.

# 6. TESTING

## Software Testing

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and code generation. It is the process of executing a program or system with the intent of finding errors.

## 6.1 TESTING OBJECTIVES

* To ensure that during operation the system will perform as per specification.
* To make sure that system meets the user requirements during operation.
* To make sure that during the operation, incorrect input, processing and output will be detected.
* To see that when correct inputs are fed to the system the outputs are correct.
* To verify that the controls incorporated in the same system as intended.
* Testing is a process of executing a program with the intent of finding an error.
* A good test case is one that has a high probability of finding a yet undiscovered error.

The software developed has been tested successfully using the following testing strategies and any errors that are encountered are corrected and again the part of the program or the procedure or function is put to testing until all the errors are removed. A successful test is one that uncovers a yet undiscovered error.

## 6.2 TEST CASE DESIGN:

## White box testing

White box testing is a testing case design method that uses the control structure of the procedure design to derive test cases. All independents path in a module are exercised at least once, all logical decisions are exercised at once, execute all loops at boundaries and within their operational bounds exercise internal data structure to ensure their validity. Here the customer is given three chances to enter a valid choice out of the given menu. After which the control exits the current menu.

This type of testing can be commenced at an earlier stage. One need not to wait for the GUI available. Testing is more through, with the possibility of covering most paths. White box testing is like the work of a mechanic who examines the engine to see why the car is not moving. This type of testing is applicable to the following levels of software testing.

## Unit Testing

Unit testing is essentially for the verification of the code produced during the coding phase and the goal is test the internal logic of the module/program. In the Generic code project, the unit testing is done during coding phase of data entry forms whether the functions are working properly or not. In this phase all the drivers are tested they are rightly connected or not.

## Integration Testing

All the tested modules are combined into sub systems, which are then tested. The goal is to see if the modules are properly integrated, and the emphasis being on the testing interfaces between the modules. In the generic code integration testing is done mainly on table creation module and insertion module.

## System Testing

This testing is a series of different tests whose primary is to fully exercise the computer-based system. This involves:

* Implementing the system in a simulated production environment and testing it.
* Introducing errors and testing for error handling.

## Black Box Testing

Black Box Testing attempts to find errors in following areas or categories, incorrect or missing functions, interface error, errors in data structures, performance error and initialization and termination error. Here all the input data must match the data type to become a valid entry.

The following are the different tests at various levels:

System testing, Integration testing, Acceptance testing.

Tests are done from a user’s point of view and will help in exposing discrepancies in the specifications. Tester need not know programming languages or how the software has been implemented. Tests can be conducted by a body independent from the developers, allowing for an objective perspective and the avoidance of developer-bias. Test cases can be designed as soon as the specifications are complete.

## Validation Testing

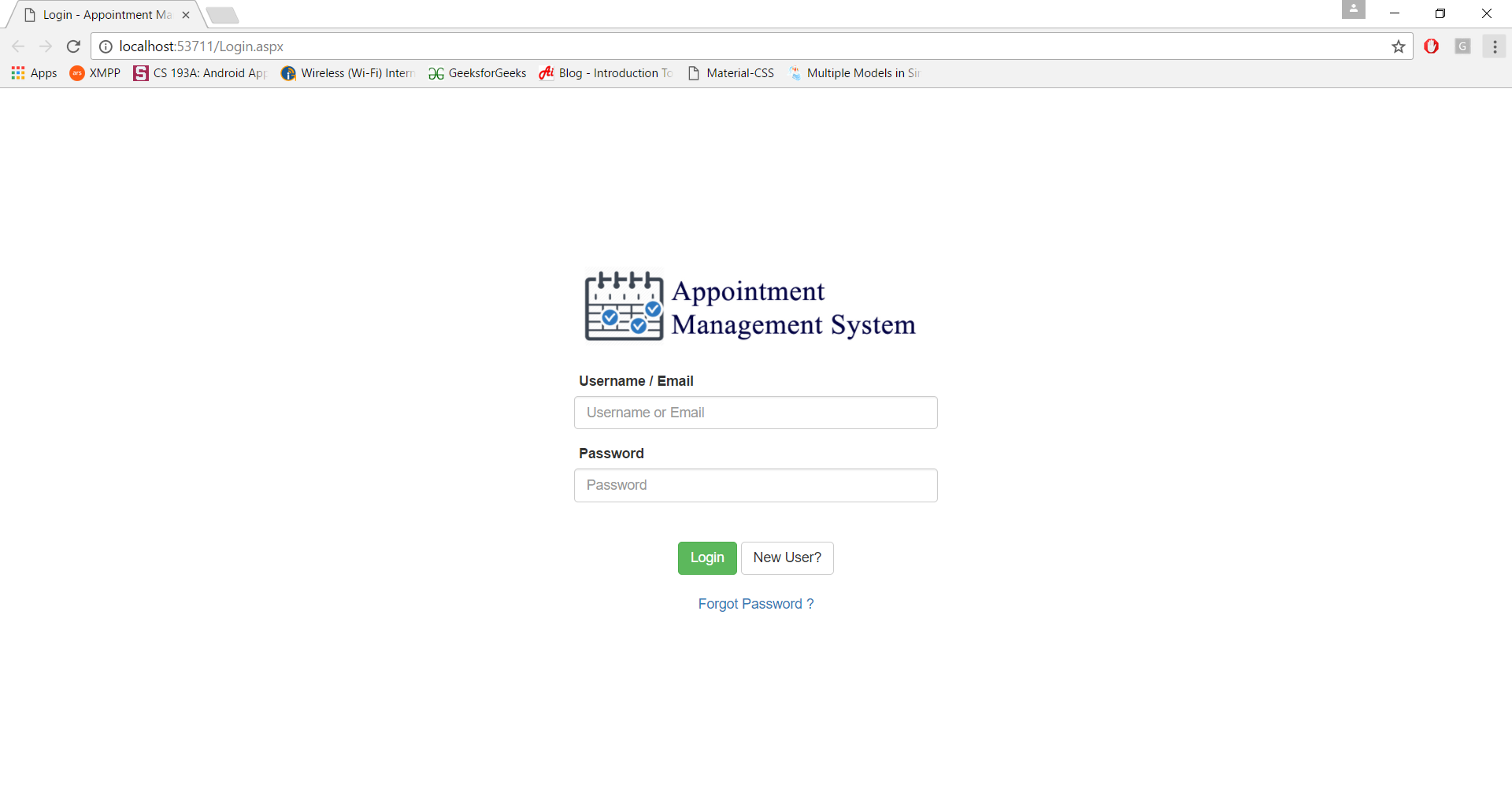
This testing concentrates on confirming that the software is error-free in all respects. All the specified validations are verified and the software is subjected to hard-core testing. It also aims at determining the degree of deviation that exists in the software designed from the specification; they are listed out and are corrected.

**6.3 Test Cases**

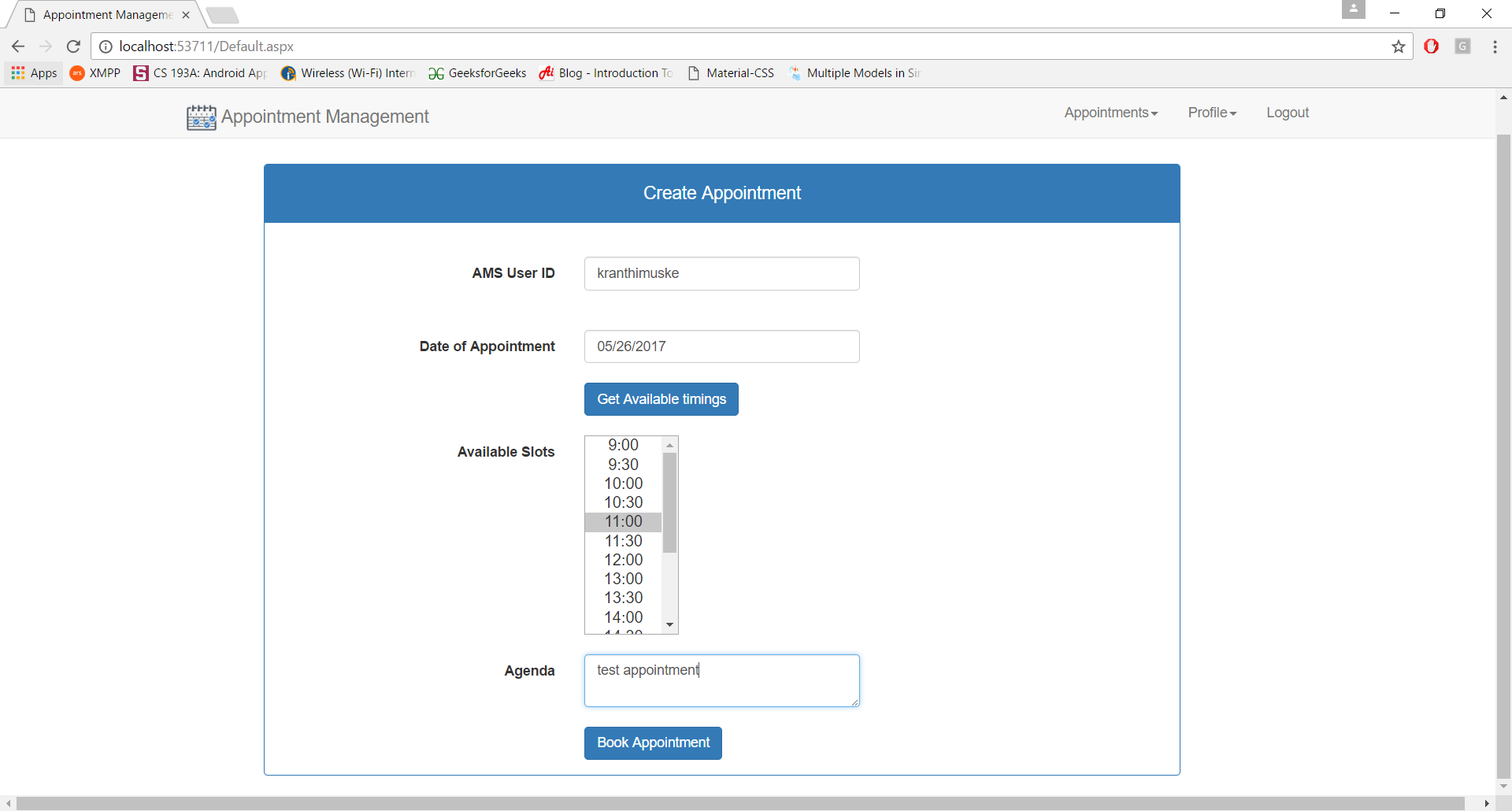
|  |  |  |
| --- | --- | --- |
| **Test case 1: TC1** | | **Priority (H, L):** High |
| **Test Objective:** User login | | |
| **Test Description:**  User should be able to login to the AMS systems using valid credentials using Website or Chatbot | | |
| **Requirements Verified:** Yes | | |
| **Test Environment:**  Application must be hosted to localhost (IIS Server). | | |
| **Test Setup/Pre-Conditions:** | | |
| Actions | Expected Results | |
| The user should be provide username and password. | User can successfully login to the system. | |
| **Pass: Yes Conditions pass: Yes**  **Fail**: No | | |
| **Problems / Issues:** NIL | | |
| **Notes**: Successfully Executed | | |

# 7. OUTPUT SCREENS

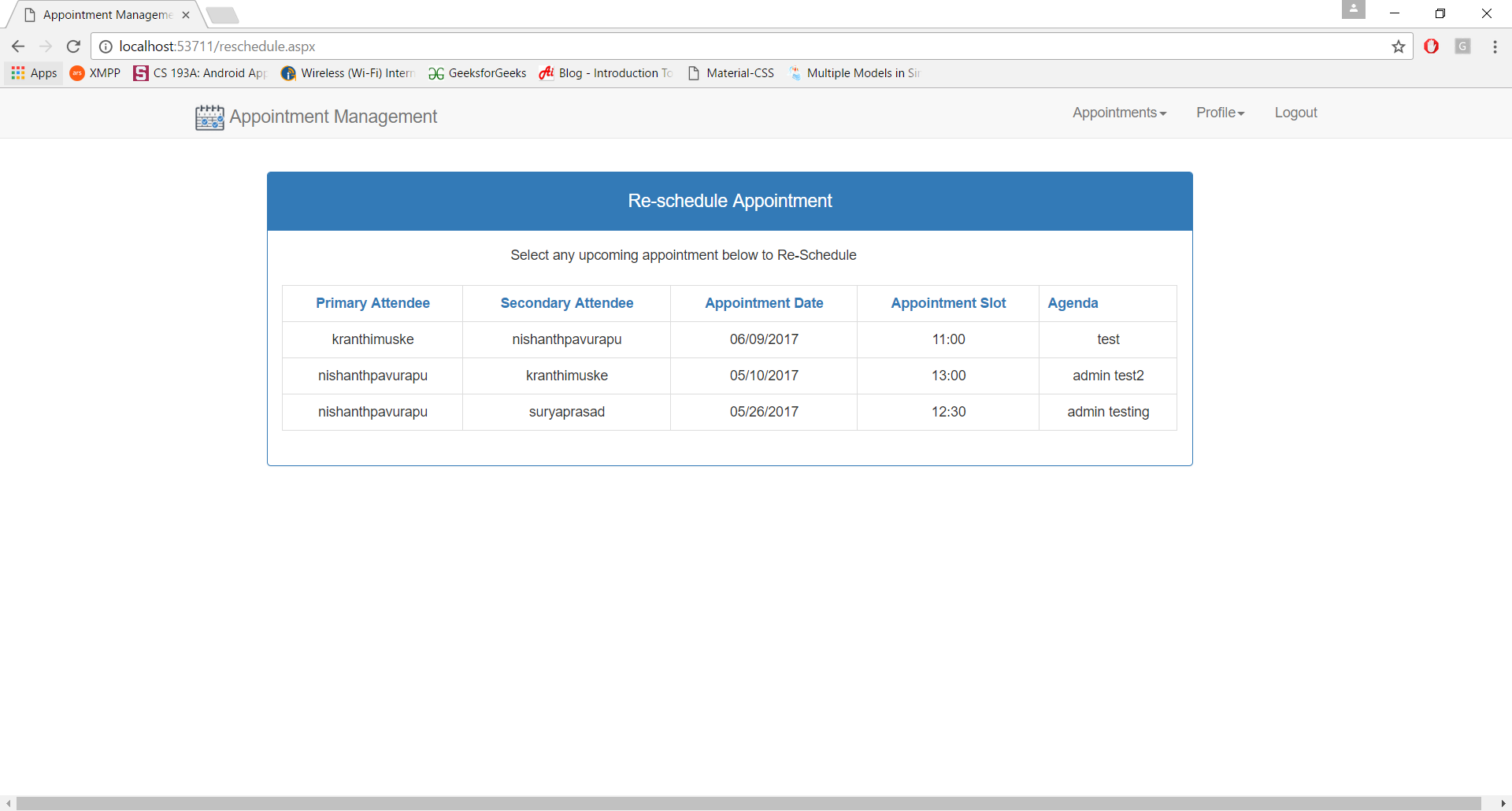
## 7.1 Login Screen

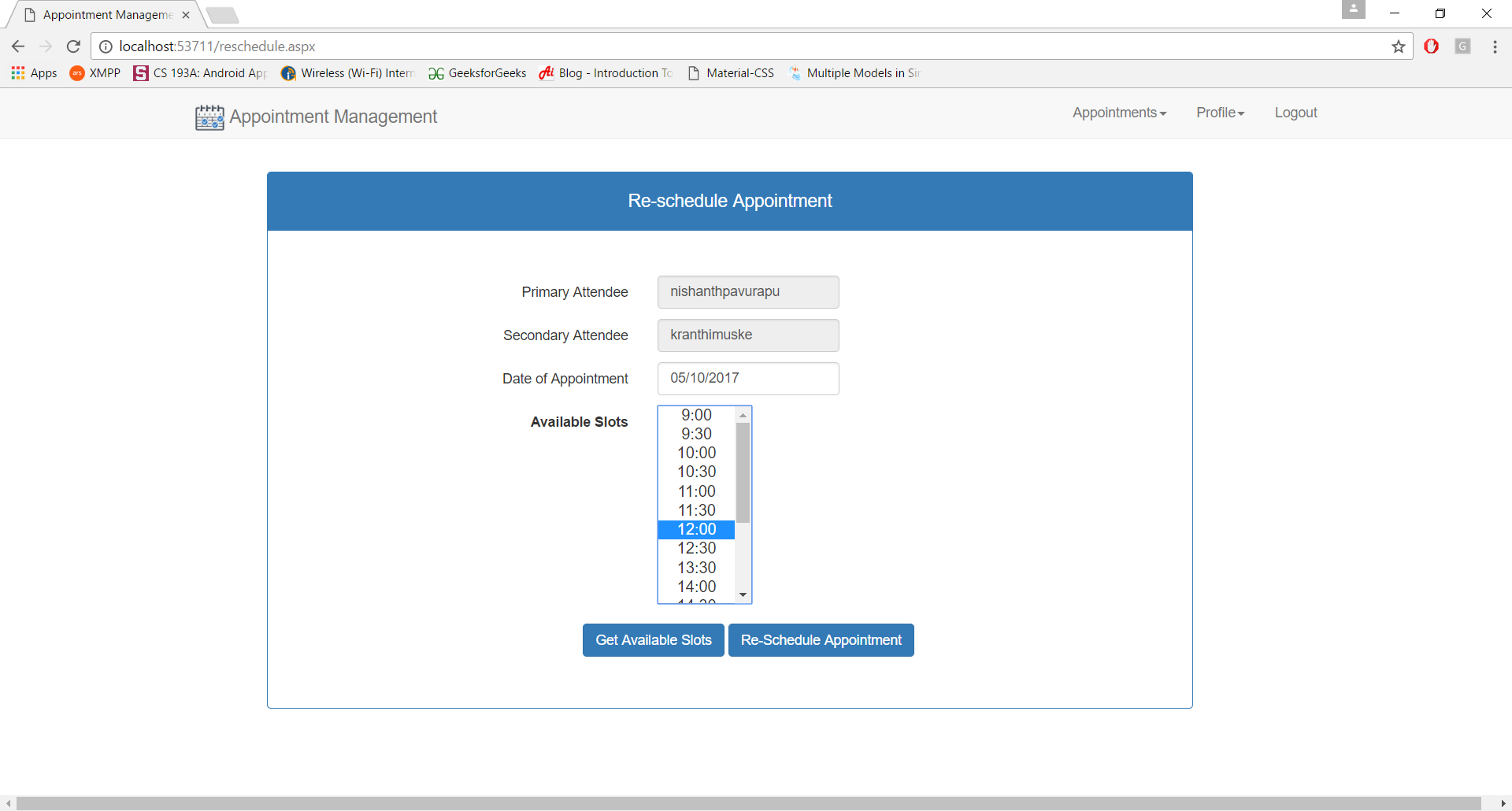


## 7.2 Create Appointment Screen

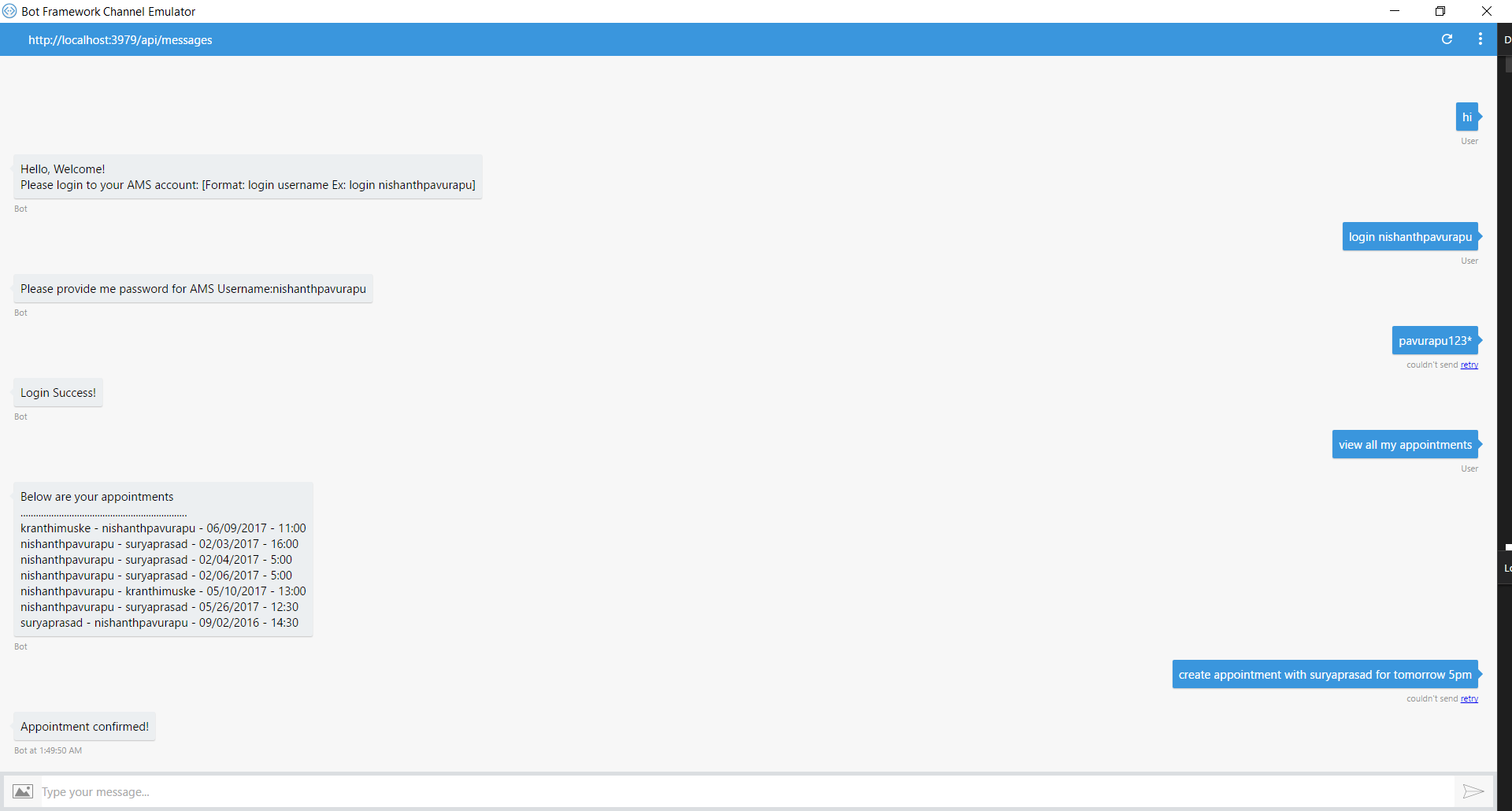


## 7.3 Re-schedule Appointment Screen





## 7.6 Chatbot (Bot Emulator)



# 8. CONCLUSION

With the developed and deployed application, any valid registered user of AMS (Appointment Management System) can login and create/re-schedule/cancel/view his appointments with the other users of AMS system. User can access AMS system either by using website or the chatbot (Bot4AMS). Chatbot can be added to Slack/Skype/Facebook or any other channels to directly access the AMS system to do them operations. It will be easy for the user to use the system as the chatbot understands the natural language of the user and performs the operations based on his/her request.

# 9. FUTURE ENHANCEMENTS

As we all know the requirements for a project keeps on changing as the time goes by to cope up with the latest technologies, and scalability issues and other new feature requirements, which is a continuous discovery process. It is not conceivable to develop a system that satisfies every requirement of the user. New user and new feature requirements comes as more the system is used. Below are some of the future enhancements that could be done to this system/project are:

* In future, we can more functionality to the application providing scheduling using monthly/weekly/day calendar something like an outlook.
* In future, we could include the mobile SMS notifications on every appointment transaction like email which we do now.
* We can increase the understanding capabilities of the chatbot by making it identify more entities and intents to the model.
* Handling more date and time patterns from the user’s conversations to the bot.

# 10. REFERENCES

1. SOFTWARE ENGINEERING - Roger.S.Pressman
2. <https://msdn.microsoft.com/en-us/library/ee958158.aspx>
3. <https://docs.botframework.com/en-us/csharp/builder/sdkreference/>
4. <https://docs.microsoft.com/en-us/azure/cognitive-services/luis/home>
5. <https://www.w3schools.com/js/>
6. <http://stackoverflow.com/>
7. <http://getbootstrap.com/components/>
8. <https://www.codeproject.com/Articles/680100/Overview-of-the-NET-Framework.>
9. <https://msdn.microsoft.com/en-us/library/zw4w595w(v=vs.110).aspx.>
10. [http://softwaretestingfundamentals.com/black-box-testing/.](http://softwaretestingfundamentals.com/black-box-testing/)