**Graduate Directed Project – II**

**Office Hours**

**Contributors:**

**Ashok Atkuri**

**Revanth Vallamsetty**

**Varshitha Guntakandla**

**Swathi Dasari**

|  |  |  |
| --- | --- | --- |
| **S.No** | **Table of Contents** | **Page No** |
| **I** | **Introduction brief: Introduce client, Introduce project** |  |
| 1 | Problem statement | 3 |
| 2 | Proposed solution | 3 |
| **II** | **Requirements Artifacts** |  |
| 1 | Requirements Gathering | 4 |
| 2 | Functional requirements and use-case diagrams | 4 |
| 3 | Nonfunctional requirements | 10 |
| 4 | Specification requirements | 11 |
| **III** | **Analysis & Design Artifact** |  |
| 1 | Software styles/architectures | 13 |
| 2 | Analysis model (data base diagram) | 14 |
| 3 | Design model of the system//functionalities of the components | 15 |
| 4 | Data model//SQL queries for database | 16 |
| **IV** | **Implementation and Testing Artifacts** |  |
| 1 | Testing methodologies and evaluations | 18 |
| 2 | Technologies and tools used | 28 |
| 3 | Screen shots | 29 |
| **V** | **Deployment Artifacts for Users and Clients** |  |
| 1 | User’s guide for the users and client | 32 |
| 2 | Installation guide | 41 |
| 3 | Execution guide | 42 |
| 4 | Project itself | 42 |
| **VI** | **Project Management Artifacts** |  |
| 1 | Risk list, pros and cons | 43 |
| 2 | Summary of requirements that have been completed | 43 |
| 3. | Summary of requirements that have not been completed | 44 |
| 4 | Conclusion | 44 |

1. **Introduction**

**Project Title:** Office Hours

**Project Client:** Dr. Patrick Immel (pimmel@nwmissouri.edu)

**Project Start Date:** June 11, 2018 **Project End Date:** Dec 14, 2018

**Project Goal:** Office hours helps the instructor to schedule and update office hours easily and eliminates the tedious task done using paper and helps the instructor to display important messages and alerts. It also provides a good interface for faculty and students to communicate and helps a student to schedule appointments with the instructor and these appointments will be synchronized with google calendar.

**1.Problem statement**

**Prepared by: Swathi Dasari Date: 11/30/2018**

Scheduling and updating day-to-day schedule is a tedious process for a university professor. The main problem arises when the schedule of the advisor changes, they notify the changes by sticking the printed paper on the door which needs a lot of effort and in emergency, they may not even be able to update the schedule

Another problem occurs while scheduling an appointment by the students, who need to contact in an emergency, may or may not have professor’s personal number or email. They need to drop a mail to book an appointment with the professor which may not be seen by other students who may also want to see them at the same time. A traditional method like collecting the papers from the students makes the instructor work even more difficult.

**2. Proposed solution**

**Prepared by: Swathi Dasari Date: 11/30/2018**

The system should be designed to offer details like available office hours, the teaching schedule of the advisor, sending messages if the advisor is out of town, and to schedule an appointment by the student should be displayed on their office doors. Professors should be able to view the messages, scheduled appointments and also update their office hours.

Instead of maintaining the records manually, the student should able to capture a photo while dropping off an important paper at faculty's office. Faculty should able to grade students based on the photos and timestamp.

In addition to this security of the data being used is a concern and would require higher security measures.

**II. Requirement Artifacts**

**1. Requirements gathering**

**Prepared by: All Date: 11/30/2018**

We gathered the functional and non-functional requirements for the application from our clients. Any requirements which describes the behavior of the system i.e what the system should do is considered as a functional requirements . some examples of functional requirements are as follows:

* Authorization
* Authentication levels
* User functionalities
* Admin functionalities

Any requirements which describes how the system perform and what its limitations are is considered as a non- functionalities requirements. Some examples of non - functional requirements are as follows:

* Security
* Availability
* Confidentiality
* Integrity
* Reliability
* Efficiency

**2. Functional requirements and use-­‐case diagrams**

**Prepared by: Varshitha Guntakandla , Swathi Dasari Date:11/30/2018**

**User Functional Requirements:**

The system MUST display their availability status indicator all the time.

The system MUST display professors schedule up to date.

The system MUST provide an option to book an appointment at a particular time and date

The system MUST show the slot as unavailable once an appointment is made at that time

The system MUST provide an option to leave a message

The system MAY have an option for a video call with professor

The system MAY have an option for a voice call with professor

The system MUST save a photo including the timestamp that students take it with their paper while submissions

The system SHOULD allow the student to retake a picture if it isn't clear

**Admin Functional Requirements:**

The system MUST display user, the welcome page or splash screen once a user visits the application.

The system MUST allow the user to sign up

The system MUST require users SId, username, password while sign up.

The system MUST provide security by password protection ie. ( by using his username and password )

The system MUST allow existing users to log in with username and password.

The system MUST display the home page to the admin, once logged in.

The system MUST allow the admin to logout from the application.

The system MUST display a message when logout is done successfully.

The system MUST display the login page when the logout is done successfully.

The system MUST allow admin to create the schedule

The system MUST allow admin to make changes in already existing schedule

The system MUST save the weeks schedule of professor with time and day.

The system MUST notify admin after an appointment is made.

The system SHOULD provide with the number of appointments made that day or week

The system MUST show admin messages.

The system MUST provide a clear difference between read and unread messages.

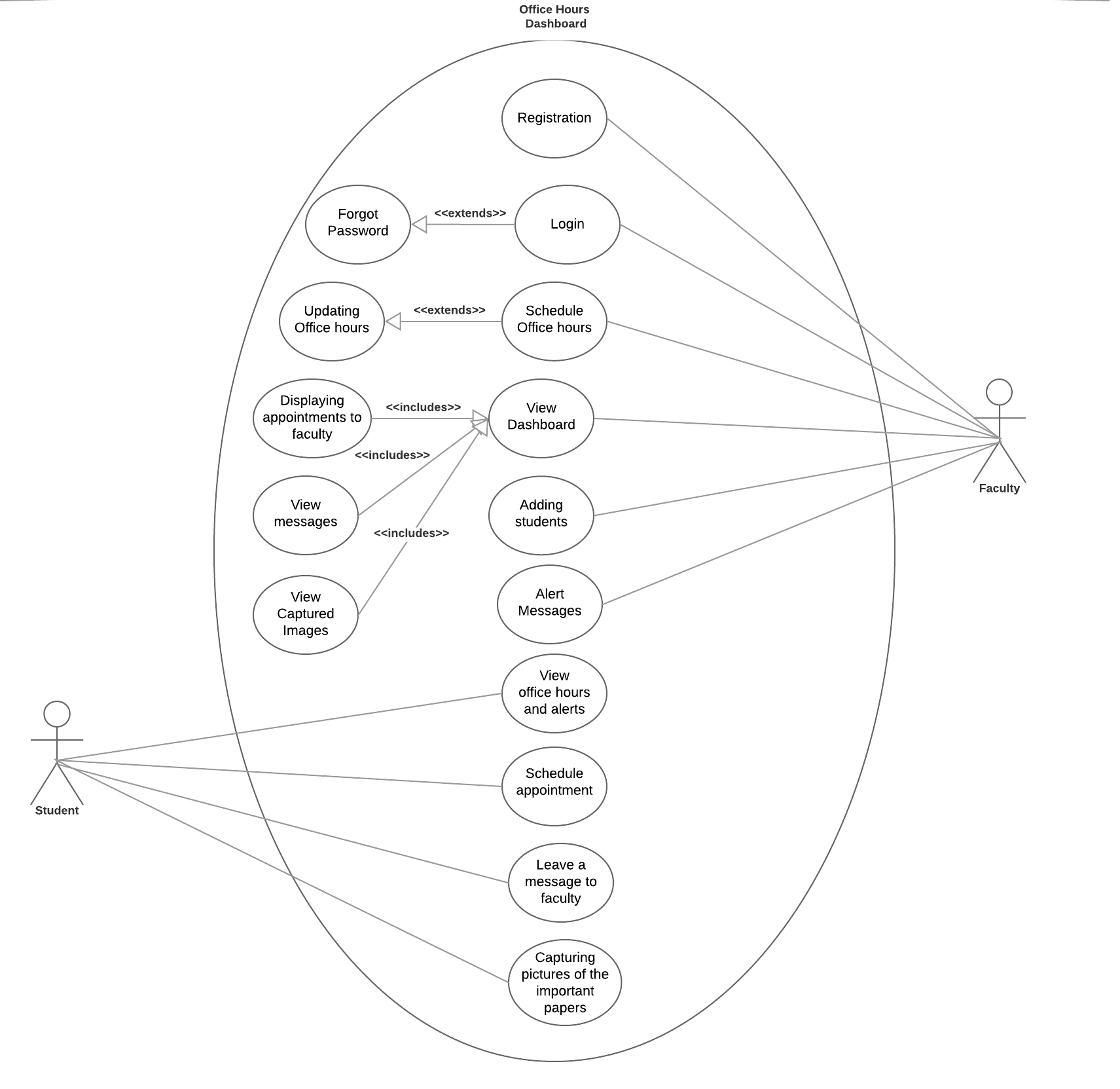
The system MUST provide all the pictures taken for submissions in order of time.

The system SHOULD provide a basic or previous template for notes.

The system MAY provide suggestions while typing SID.

The system SHOULD let admin add students into the database.

**Use Case Diagram**

****

**Use Case scenario 1:** Faculty Registration

Story: Faculty should create an account to access the system.

Primary Actor: Faculty

Level: User goal

Scope: System level

Normal sequences:

1. Once the faculty visit the website, he/she can able to see the welcome home page with faculty registration .
2. The faculty was able to create a new account.
3. The application opens registration form that includes Username, email address, password.
4. The faculty is authenticated based on username and a new account is created for if authentication is a success.

**Use case name 2:** Login

Story: Faculty provides username and password to login. The application authorizes and navigates to the faculty dashboard on success.

Primary Actor: Faculty

Level: User goal

Scope: System

Normal sequence:

1. Faculty should enter his/her unique user id.
2. Faculty should enter his/her password.
3. On submit, the application should navigate to dashboard if login is a success.
4. The system will display the home page once logged in.

**Use case name 3:** Schedule Office hours

Story: Faculty should be able to schedule his office hours into the system.

Primary Actor: Faculty

Level: User goal

Scope: System

Normal sequence:

1. After login faculty selects schedule office hours that redirects to the schedule page.
2. In the schedule, page faculty enters his/her office hours and submits.
3. submitted schedule should be saved to the database and reflected for the student.

**Use case name 4:** Updating Office hours

Story: Faculty should be able to update office hours if there is already an entry.

Primary Actor: Faculty

Level: User goal

Scope: System

Normal sequence:

1. After login faculty should be able to see scheduled office hours if he/she already created a schedule.
2. Editing of the existing schedule should be enabled for the faculty by providing some templates.
3. Faculty saves the updated changes and these should be persisted in the database.
4. The updated schedule should be reflected for the student.

**Use case name 5:** View messages

Story: faculty should be able to view messages left by his/her students.

Primary Actor:Faculty

Level: User goal

Scope: System

Normal sequence:

1. After login faculty should be able to click on view messages.
2. The faculty is navigated to view messages page.
3. List of messages should be displayed to faculty and unread messages should have a different color.
4. Messages read by faculty should be archived.

**Use case name 6:** View office hours and alerts

Story: Student should be able to view office hours and any alerts.

Primary Actor: Student

Level: User goal

Scope: System

Normal sequence:

1. A student should be able to see office hours of the faculty.
2. Students should also be able to see the alert messages posted by faculty.
3. A student should be able to do this without login.

**Use case name 7:** synchronization of schedule appointment to google calendar

Story: Student should be able to schedule an appointment and synchronized to the Google calendar of the advisor.

Primary Actor: Student

Level: User goal

Scope: System

Normal sequence:

1. A student should be able to schedule an appointment by providing his/her sID.
2. The system should check for conflicts and notify the student if any.
3. A mail should be sent to the instructor.
4. A student should schedule their appointment by providing their name, email ID and SID. After scheduling, the appointment details of the student is synchronized to the Google calendar.

**Use case name 8:** Leave a message to faculty

Story: Student should be able to leave a message to the instructor in the form of text or video messages

Primary Actor: Student

Level: User goal

Scope: System

Normal sequence:

1. The student should be able to leave a message to the instructor by providing his/her sid.
2. Instead of sending the textual message students should able to send video messages to the instructor with his/her 919# number.
3. The message should display on faculty dashboard in messages section.

**Use case name 9:** Capturing pictures of the important papers

Story: Student should be able to send the pictures of their important papers to the instructor

Primary Actor: Student

Level: User goal

Scope: System

Normal sequence:

1. A traditional method like collecting the papers from the students and maintaining the records makes the instructor work even more difficult instead of maintaining the records manually the student should able to capture a photo of the paper and send it to the instructor by providing his/ her SID.
2. The system provides the countdown before taking the picture.
3. The system provides all the pictures taken for submission in the order of the time.
4. This message should be displayed on a faulty dashboard in the message section.

**Use case name 10:** Faculty views scheduled appointments

Story: All the scheduled appointments by the students should be displayed on the calendar of the instructor with time.

Primary Actor: Instructor

Level: User goal

Scope: System

Normal sequence :

1. Whenever student books for an appointment and if it confirmed it should reflect on the instructor calendar with the time.
2. The faculty should able to see the number of appointments scheduled in a day or week.

**Use case name 11:** Alert Messages

Story: faculty should able to notify the students with messages

Primary Actor: Faculty

Level: User goal

Scope: System

Normal sequence:

If the faculty has important message , out of station or sick etc. , he/she should able to notify the student immediately on the notification board.

**Use case name 12:** Logout

Story: Faculty clicks on Logout and his/her session is terminated.

Primary Actor: Faculty

Level: User goal

Scope: System

Normal sequence:

1. When Faculty login the user session starts.
2. When faculty clicks on logout the user session terminates and navigates back to login.
3. On logout click if there are any unsaved changes user should be prompted before ending the session.

**3. Non functional requirements**

**Prepared by: Varshitha Guntakandla Date: 11/30/2018**

**Interface:**

* The system MUST be simple to operate and should not lead users to confusion.
* The system MUST provide appropriate feedback messages to the users.
* The system MUST provide good error messages and inform the user where the error occurred.

**Performance:**

* The system MUST be stable and should not crash under any circumstances.
* The system SHOULD NOT take a long time for processing the user requests.
* The system SHOULD give fast response to the user request.
* The system needs to establish the video call connection without any lag.

**Security:**

* The system MUST allow only the authorized personnel to edit the content like adding a schedule, alert messages etc...
* The system MUST not show appointments, messages or other related data of a user to others.
* The system must secure the password and login details of the user.
* The system MUST NOT be able to open any other websites or applications expect the current one.

**Integrity:**

* The System should organize with the google calendars.
* The data maintained by the system should be without corruption and the data should be accurate.
* The system data should be updated regularly and it should be authenticated.
* The System should show the message as sent by the user.

**Compatibility:**

* The system must be compatible with all tablet devices.

**4. Specification requirements**

**Prepared by: Ashok Atkuri Date: 11/30/2018**

**Platforms:**

* Windows/Linux

**Technologies:**

* Front- End: HTML, CSS, JavaScript, Bootstrap, .cshtml(Razor Pages)
* Database: SQLServer
* Client-Side Processing: ASP.NET MVC
* Language: C#
* Server: IIS

**Additional Libraries:**

* Microsoft.Graph
* System.Net.Http.Extensions
* System.Net.Http.Primitives
* System.Web.Optimization
* WebGrease
* Microsoft.Owin
* Microsoft.Owin.Security
* Antlr3.Runtime
* Newtonsoft.Json
* System.IdentityModel.Tokens.Jwt
* Microsoft.IdentityModel.Protocol.Extensions
* Microsoft.IdentityModel.Logging
* Microsoft.IdentityModel.Tokens
* Microsoft.IdentityModel.Protocols.OpenIdConnect
* Microsoft.IdentityModel.Protocols
* System.Net.Http
* System.Web.Helpers
* System.Web.WebPages
* System.Web.Mvc
* EntityFramework

**Hosting Strategy:**

Windows/Linux servers

Heroku

**Windows/Linux Server Requirements:**

RAM: 4 GB

Storage: 50 GB

Processor Type: Multi core

Number of Processor: 4

**Database Servers:**

RAM: 4 GB

Storage: 50 GB

Processor Type: Multi core

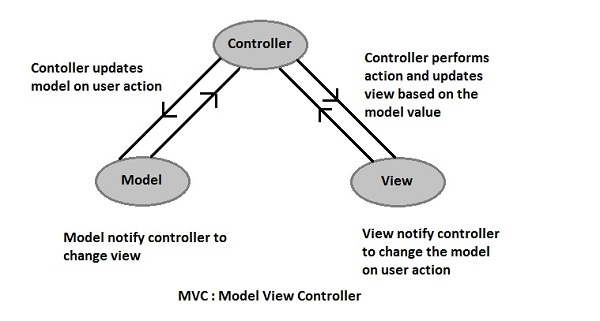
Number of Processor: 4

**III. Analysis & Design Artifact**

**1. Software styles/architectures**

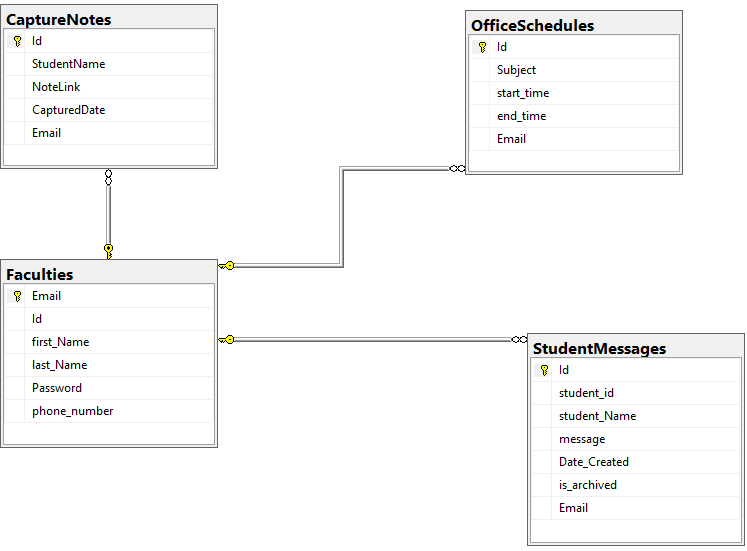
**Prepared by: Revanth Vallamsetty Date: 11/30/2018**

We have implemented the project in MVC architecture. It consists of below   
 Model: deal with the database or any data-related functionality.  
 View: Everything the user will see. Basically the pages that we’re going to send to the client.  
 Controller: The logic of our site, and the glue between models and views. Here we call our models to get the data, then we put that data on our views to be sent to the users.



**2. Analysis model (database diagram)**

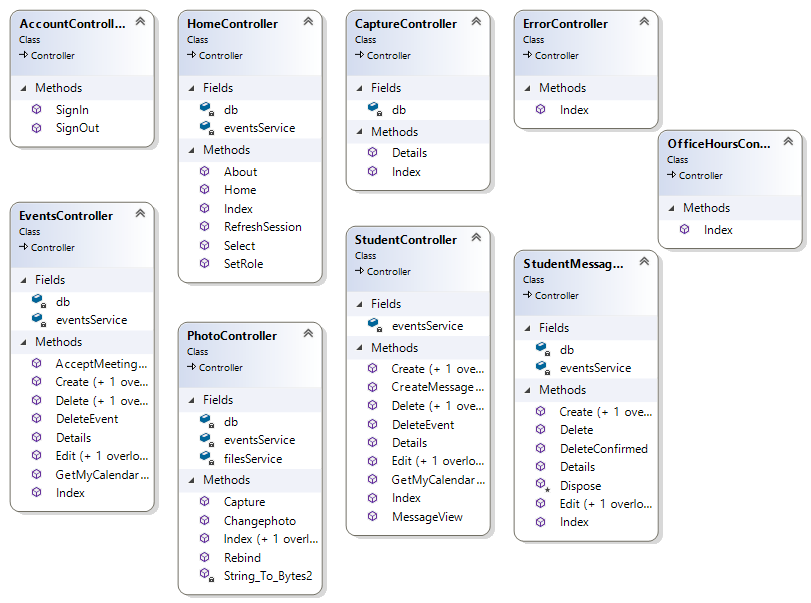
**Prepared by: Revanth Vallamsetty Date: 11/30/2018**



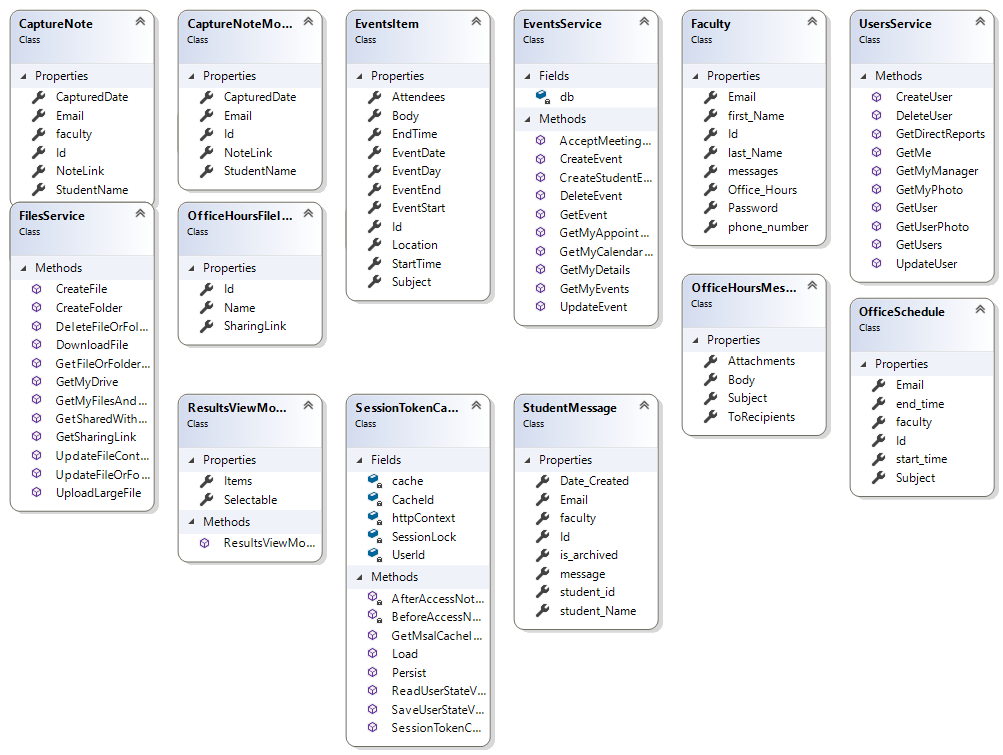
**3. Design model of the system**

**Prepared by: Revanth Vallamsetty Date: 11/30/2018**

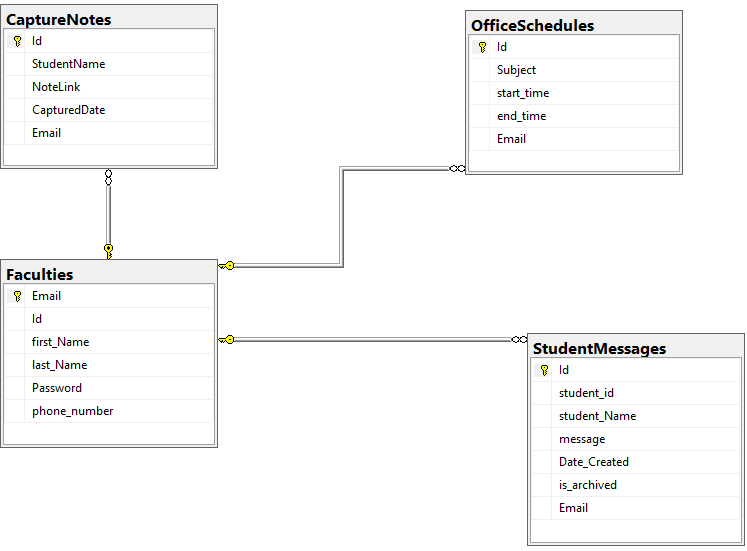
Controllers:



Services and Models:



**4. Data model**  
**Prepared by: Revanth Vallamsetty Date: 11/30/2018**



**IV. Implementation and Testing Artifacts**

By writing the detailed test plan, we will get an idea of what all the functionalities need to be tested and how to test each functionality. This test plan will be helpful to detect and correct the errors and to deliver a perfectly working application. We need to get an idea of how to test the functionalities using the mentioned tools.

**1. Testing methodologies and evaluations**

**Prepared by: Ashok Atkuri , Swathi Dasari Date: 11/30/2018**

**To be tested**

|  |  |
| --- | --- |
| **Feature description** | **List of functional requirements** |
| Faculty Registration | FR 14, 15, 16, 17, 18, 19, 20, 21, 22, 23 |
| Schedule & Updating Office hours | FR 1, 24, 25, 26, 27, 35, 36 |
| Faculty View messages | FR 31, 32, 33 |
| Student view office hours and alerts | FR 2 |
| Synchronization | FR 3, 4, 6, 7, 8, 28, 29, 30 |
| Leave a message to faculty | FR 9, 10, 11, 28 |
| Capturing pictures of important papers | FR 5, 12, 38 |
| Update Database | FR 38 |

**Not to be tested**

|  |  |  |
| --- | --- | --- |
| **Feature description** | **List of functional requirements** | **Rationale** |
| Difference between read and unread messages | FR 33 | Requires a manual verification. Cannot test automatically |
| Slot must be unavailable once the appointment is made at that time | FR 8 | Requires a manual verification. Cannot test automatically |

**Test Case 1: Faculty Registration**

**Scope:** Test various input fields on registration

**Action:** Submit registration request

**Verification Steps:** Verify the user has been registered with given details

**Scenario 1:** Test leaving a required field bank on the new user registration

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement** | **Test Case Description** | **Test Type** | **Completed** |
| FR1.1 | Check whether the faculty can sign with Microsoft | Manual | Yes |
| FR1.2 | Do not store registration details in the database if there is any missing information | Manual | yes |
| FR1.3 | Do not process the registration if the mail entered in incorrect | Manual | yes |
| FR1.4 | Retrieve and display the program details in database | Manual | yes |
| FR1.5 | Do not display the program details if there is any erroneous information or improper data | Manual | yes |
| FR1.6 | System alerts for users for filling mandatory fields | Manual | yes |
| FR1.7 | System should not alert users if all the mandatory fields are filled | Manual | yes |

### 

### **Test Case 2: Login**

**Scope:** Test the Microsoft login functionality

**Action:** verifying whether the Microsoft login accessible with proper account credentials

**Verification Steps:** Verify whether the student and faculty is able to login with Microsoft or not

**Scenario 2:** Test the Microsoft login credentials with proper account details.

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement** | **Test Case Description** | **Test Type** | **Completed** |
| FR2.1 | Check the Microsoft login functionality | Manual | yes |
| FR2.2 | Check the Microsoft login is accessible with proper account credentials | Manual | yes |
| FR2.3 | Check whether the student is able to login | Manual | yes |
| FR2.4 | Check whether the faculty is able to login | Manual | yes |

### **Test Case 3: Schedule & Updating Office hours**

**Scope :** Test the schedule and updating office hours of the faculty

**Action:** verifying schedule of the faculty and update manually or from outlook calendar

**Verification Steps :** Verify whether the schedule is updated from outlook calendar

**Scenario 3:** Test the schedule and update with outlook calendar

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement** | **Test Case Description** | **Person Responsible** | **Completed** |
| FR3.1 | Accessing the schedule page based on the faculty credentials | Manual | Yes |
| FR3.2 | Check whether the schedule is updated manually | Manual | Yes |
| FR3.3 | Check whether the schedule is updated with proper time and date format | Manual | Yes |
| FR3.4 | Check the credentials of outlook calendar | Manual | Yes |
| FR3.5 | Synchronizing the outlook calendar with schedule page | Manual | Yes |
| FR3.6 | Check whether the schedule is updated with outlook calendar | Manual | Yes |
| FR3.7 | Check whether the schedule is updated with proper time and date format from the outlook calendar | Manual | Yes |

### **Test Case 4:** Faculty View messages

**Scope:** Test messages sent from the students and test whether the faculty can view those messages

**Action:** verifying the messages from the students and verifying the view of the messages of the faculty

**Verification Steps:** Verify whether the faculty able to view the messages

**Scenario 1:** Test whether the faculty s able to view the messages from the students

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement** | **Test Case Description** | **Test Type** | **Completed** |
| FR4.1 | Check the credentials of the student whether that student is under that faculty or not | Manual | Yes |
| FR4.2 | Check the whether its displaying the name and SID of the student is displaying or not | Manual | Yes |
| FR4.3 | Check the correct format of the message | Manual | Yes |
| FR4.4 | Check the timestamp of that message | Manual | Yes |
| FR4.5 | The messages sent by the students are updated to the database | Manual | Yes |

### **Test Case 5:** Student view office hours and alerts

**Scope:** Test whether the students are able to view the office hours and alerts from the faculty

**Action:** verifying the office hours of the faculty and alerts from the faculty

**Verification Steps:** Verify whether the student able to check their office hours

**Scenario 1:** Test the office hours of the faculty and alerts from faculty

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement** | **Test Case Description** | **Test Type** | **Completed** |
| FR5.1 | Check whether the office hours are from the particular faculty | Manual | Yes |
| FR5.2 | Student should able to view office hours with proper date and time | Manual | yes |
| FR5.3 | Student able to check the alerts messages from faculty | Manual | yes |

### **Test Case 6:** Synchronization

**Scope:** Test whether the outlook calendar is synchronizing with the system

**Action:** verifying the synchronization of outlook calendar with proper credentials

**Verification Steps:** Verify whether the credentials are correct

**Scenario 1:** Test the credentials of the outlook calendar and synchronization of outlook calendar with that system.

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement** | **Test Case Description** | **Test Type** | **Completed** |
| FR6.1 | Check whether the faculty is sign in outlook or not | Manual | yes |
| FR 6.2 | Synchronize the outlook calendar with the application | Manual | yes |
| FR 6.3 | Check whether the details updated from outlook calendar are displayed in application | Manual | yes |

### 

### **Test Case 7: Leave a message to faculty**

**Scope:** Test whether the student can leave a message the faculty

**Action:** verifying whether he/she can able to send message to the faculty

**Verification Steps:** Verify the credentials of the student and check the steps to send messages to the faculty and check whether they are stored in database

**Scenario 1:** Test whether the student able to send the message to the faculty.

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement** | **Test Case Description** | **Test Type** | **Completed** |
| FR7.1 | Check the credentials of the student whether he/she belongs to particular faculty | Manual | yes |
| FR7.2 | Student should able to send message to faculty | Manual | yes |
| FR7.3 | When message is sent , student should get a pop saying “message sent” | Manual | yes |

### **Test Case 8: Capturing pictures of important papers**

**Scope:** Test whether student can able to capture the pictures of important papers

**Action:** verifying whether the camera is allowing to capture the pictures

**Verification Steps:** Verify the accessibility of the camera and see whether the student is able to capture the pictures of important papers and stored in the database.

**Scenario 1:** Test whether the student able to access the camera or not and test whether the pictures are stored in database or not.

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement** | **Test Case Description** | **Test Type** | **Completed** |
| FR8.1 | Web Camera must allow the students to capture the important papers | Manual | Yes |
| FR8.2 | The captured pictures are stored in the database | Manual | yes |
| FR8.3 | The stored pictures are able to see by the faculty | Manual | yes |

### **Test Case 9: Database**

**Scope:** Test all the information is stored in database

**Action:** verifying whether the information is stored correctly in proper table or not

**Verification Steps:** Verify whether stores data are retrieved or not

**Scenario 1:** Test whether the information are stored, retrieved and updated with proper credentials.

|  |  |  |  |
| --- | --- | --- | --- |
| **Requirement** | **Test Case Description** | **Test Type** | **Completed** |
| FR9.1 | Check whether the credentials are stored in the database or not | Manual | yes |
| FR9.2 | The messages sent by the students are stored in database | Manual | yes |
| FR9.3 | In faculty view the messages are retrieved. | Manual | yes |

**2. Technologies and tools used**

**Prepared by: Ashok Atkuri Date: 11/30/2018**

· Unit testing (NUnit)

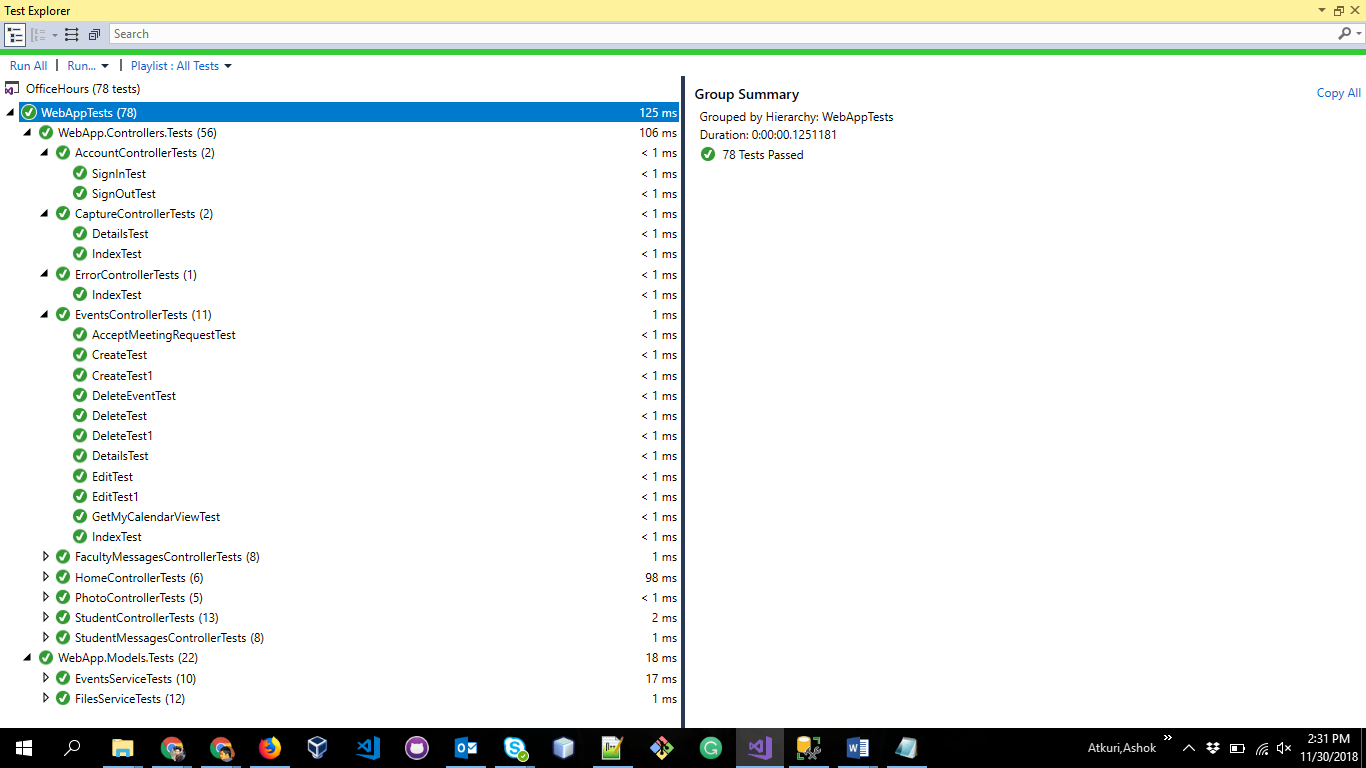
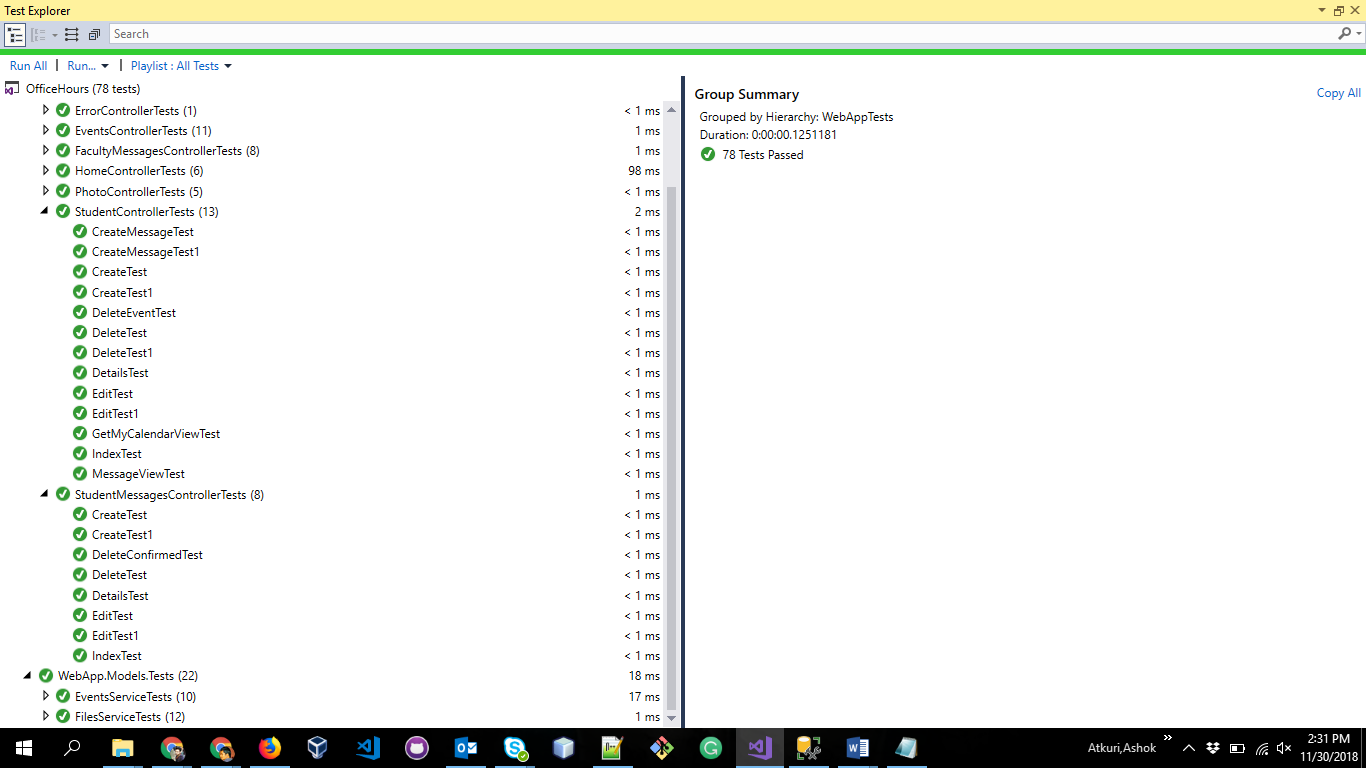
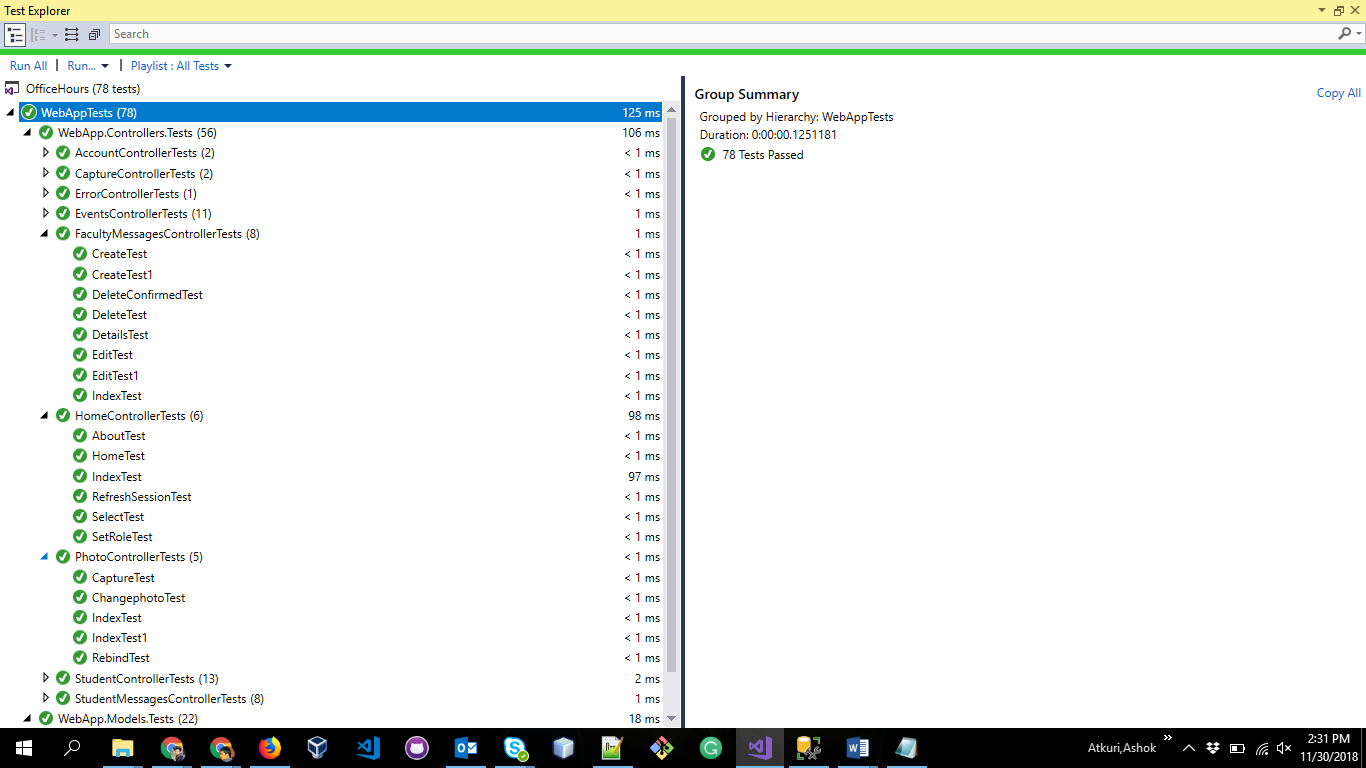
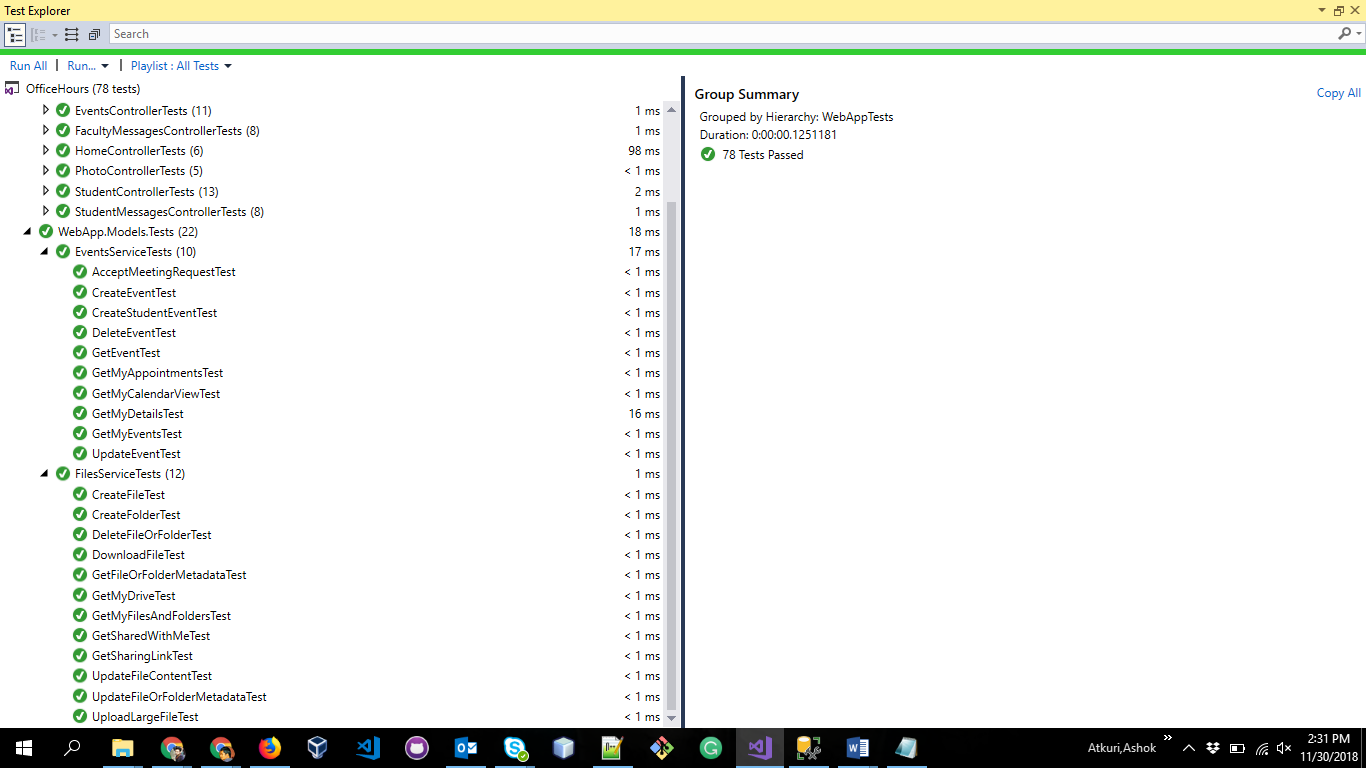
· UI testing (Manual)

**3.Screenshots**

**Prepared by: Ashok Atkuri Date: 11/30/2018**

**Unit Testing:**

**Screenshots(Contollers):**

**  
  
  
Models:  
**

**V. Deployment Artifacts for Users and Clients**

The project application provides details like schedule, booking an appointment by the student and sending message if the advisor is out of station. Dropping an email or booking appointment and sending messages directly, and easy access of the appointments for the advisor wherever he is. Providing security for the data throughout the process.

This app was created using ASP.NET MVC by Section 2 of CSIS 44692, Graduate Directed Project, at Northwest Missouri State University during the Fall Semester of 2018. It was created under the direction of Dr. Aziz Fellah , afellah@nwmissouri.edu.

**1. User’s guide for the users and client:**

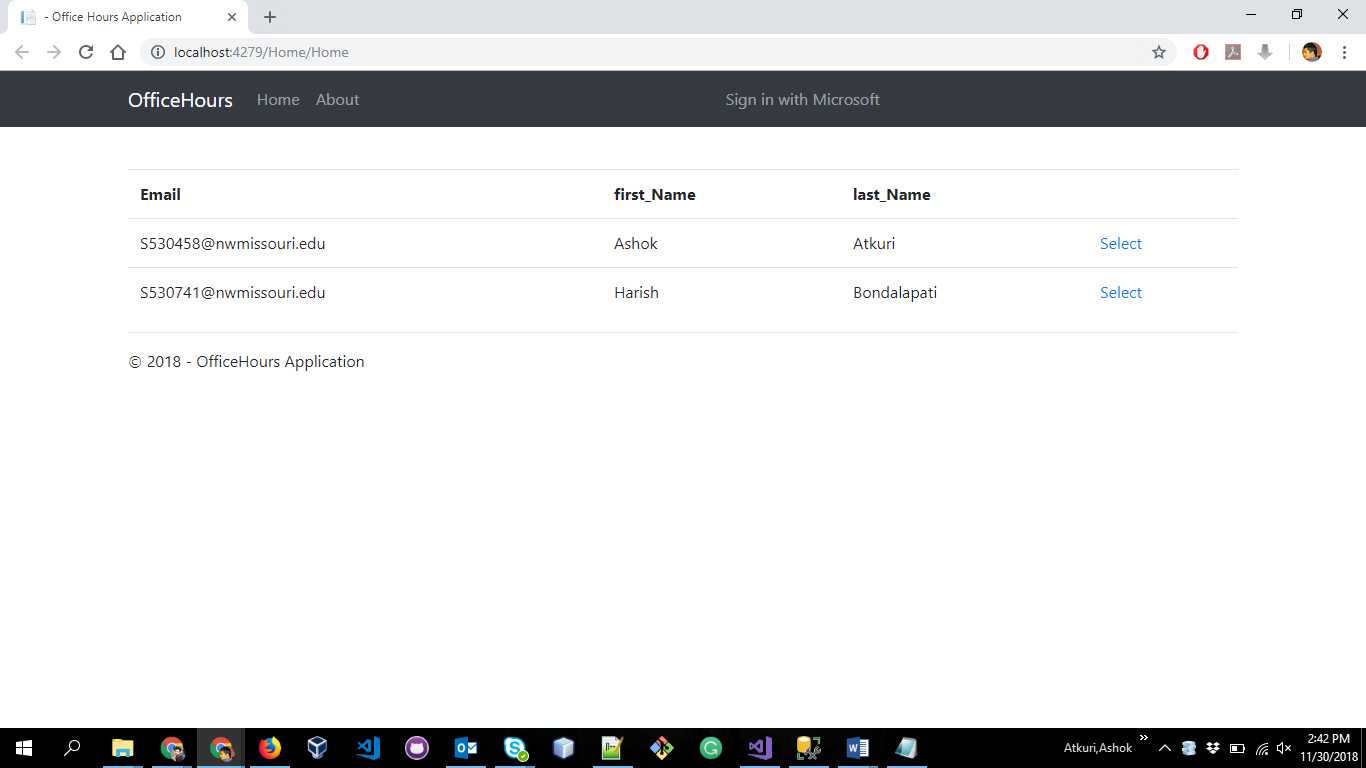
**Prepared by: Revanth Vallamsetty, Swathi Dasari Date: 11/30/2018**

* The user needs to select the faculty that he wants to interact. Then the user can view the office hours of the faculty.
* Once the user clicks on the signin, the system redirects the user to Microsoft organisation page where he needs to provide his credentials.
* If the user is student, the system redirects to student home page where he can view the office hours of the faculty.
* Then the user can schedule the appointment, send/view messages, capture the notes/submit the photo of the paper that he wants.
* If the user is a faculty, then he is provided with options to choose i.e., student view or faculty view.
* If the user clicks on the student view he can get the view of the student and can perform operations specified in the above steps.
* If the user chooses faculty view then he can view the appointments made by students,Messages sent by the students and the papers submitted by them.
* The user can change the availability status in his view.
* The user needs to logout once he is done using the application.

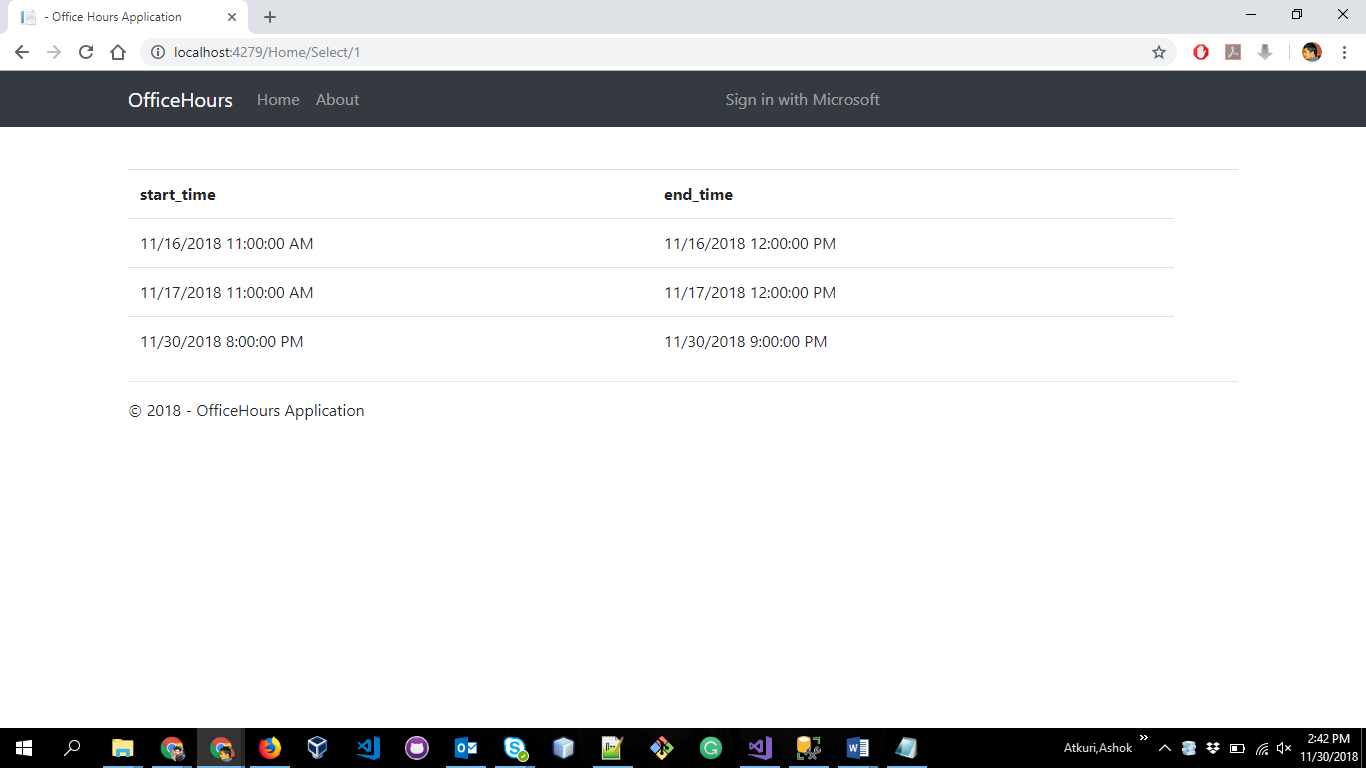
**Screenshots:**

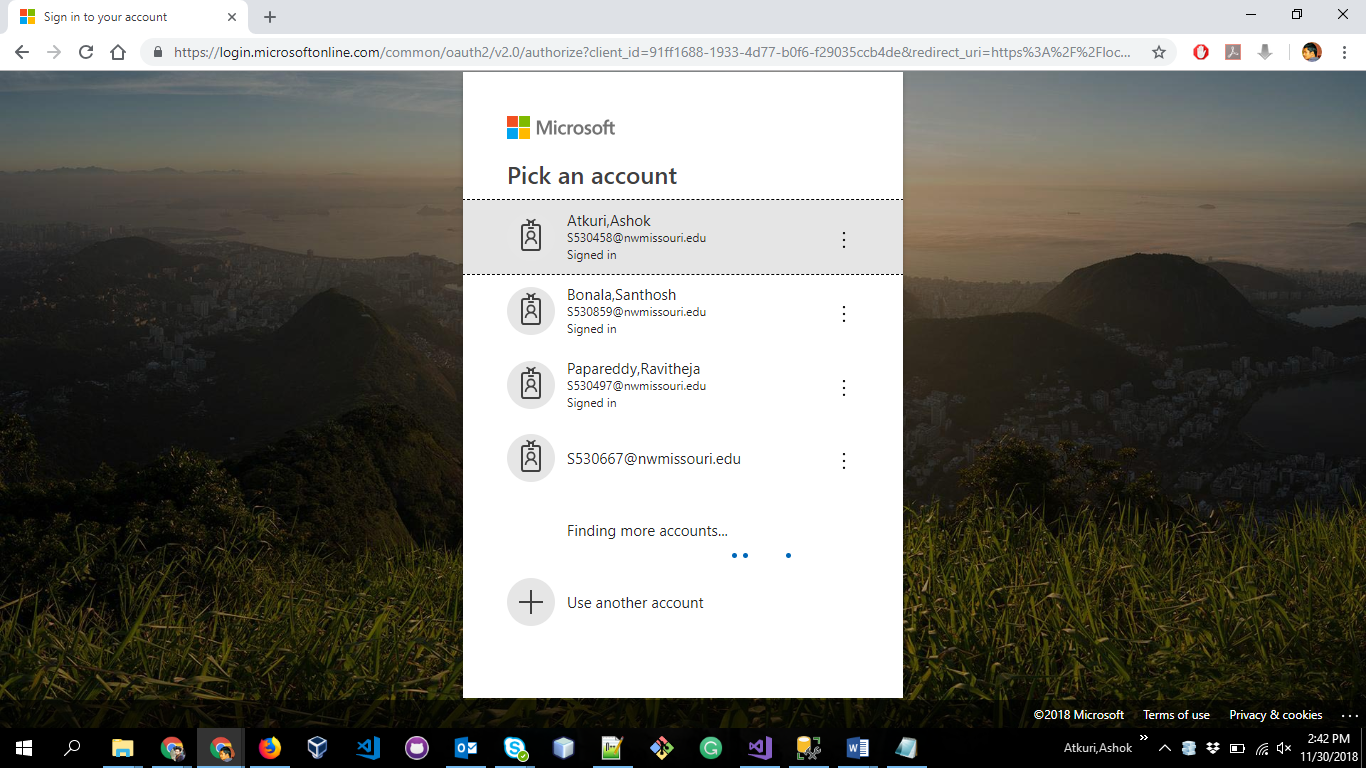
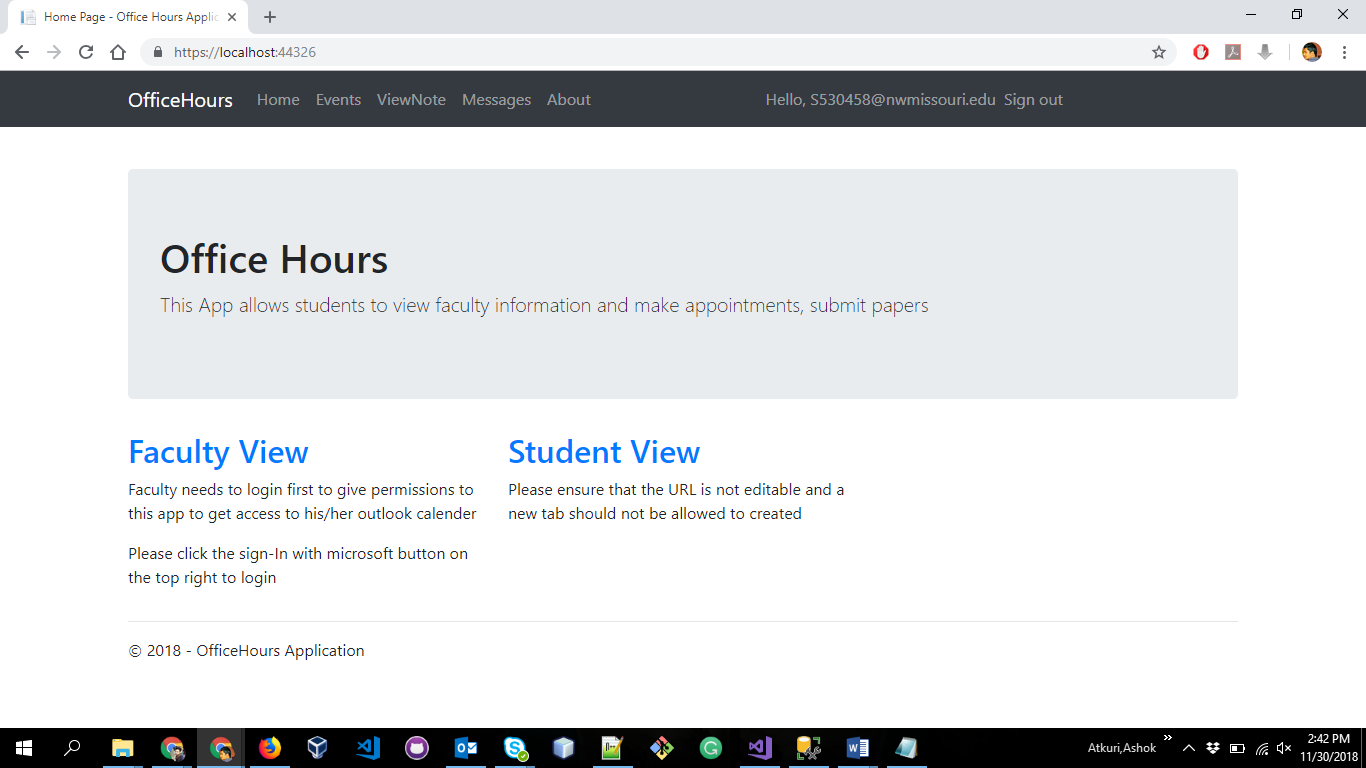
**Faculty**

**Home:**

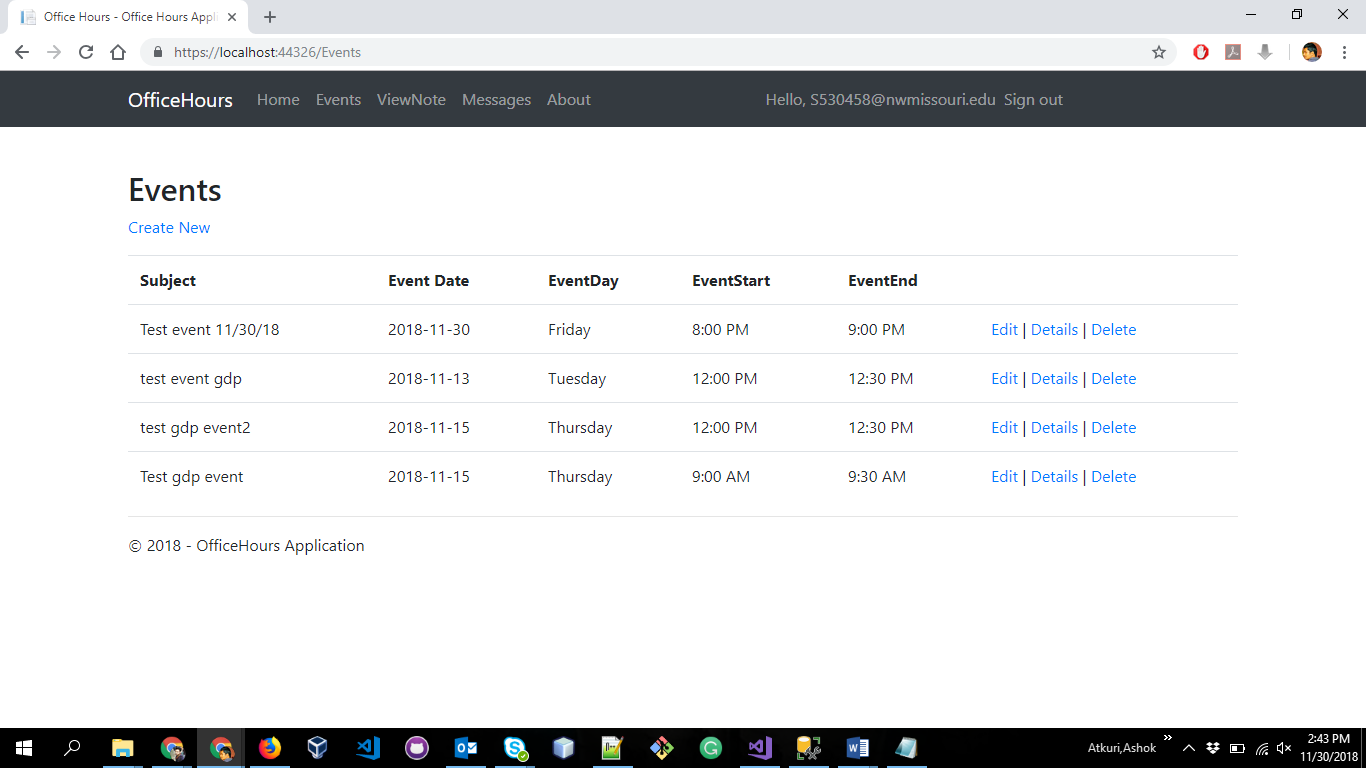
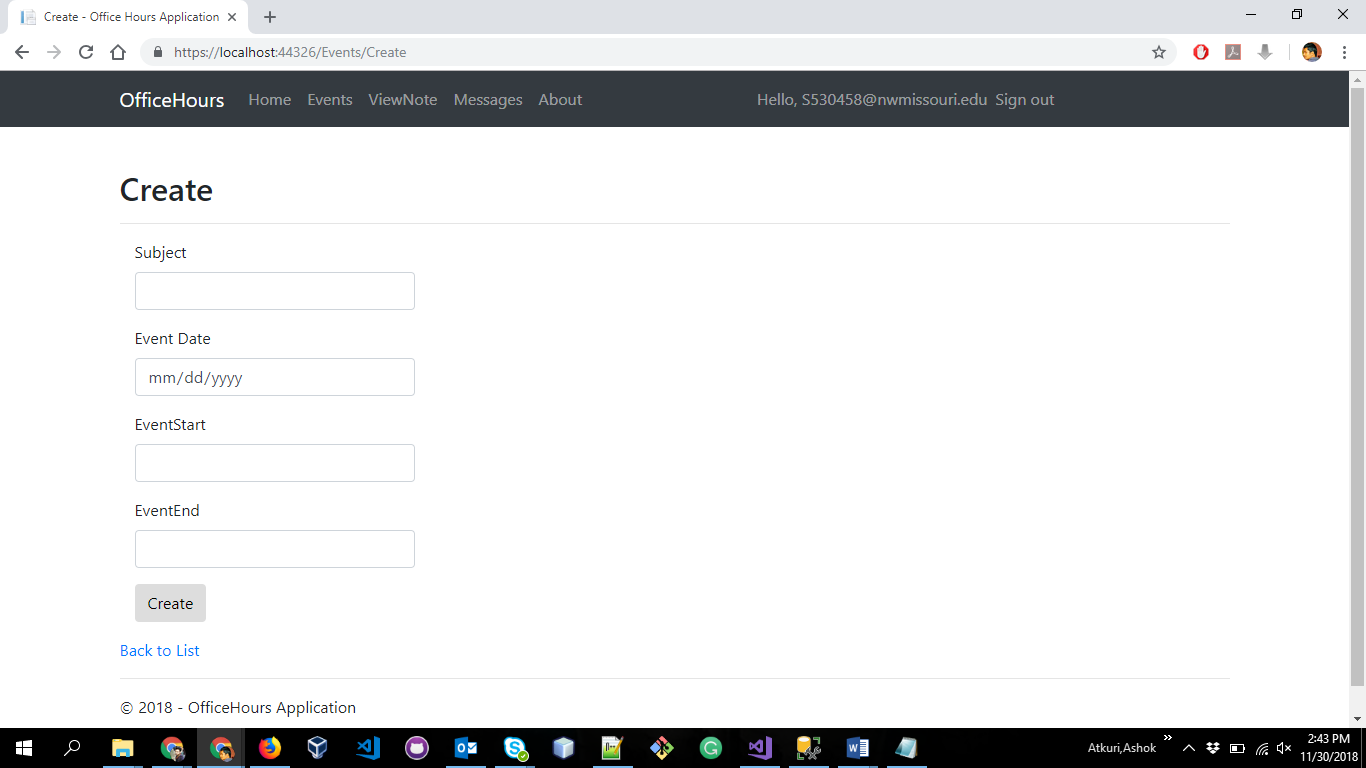


**Home office hours**

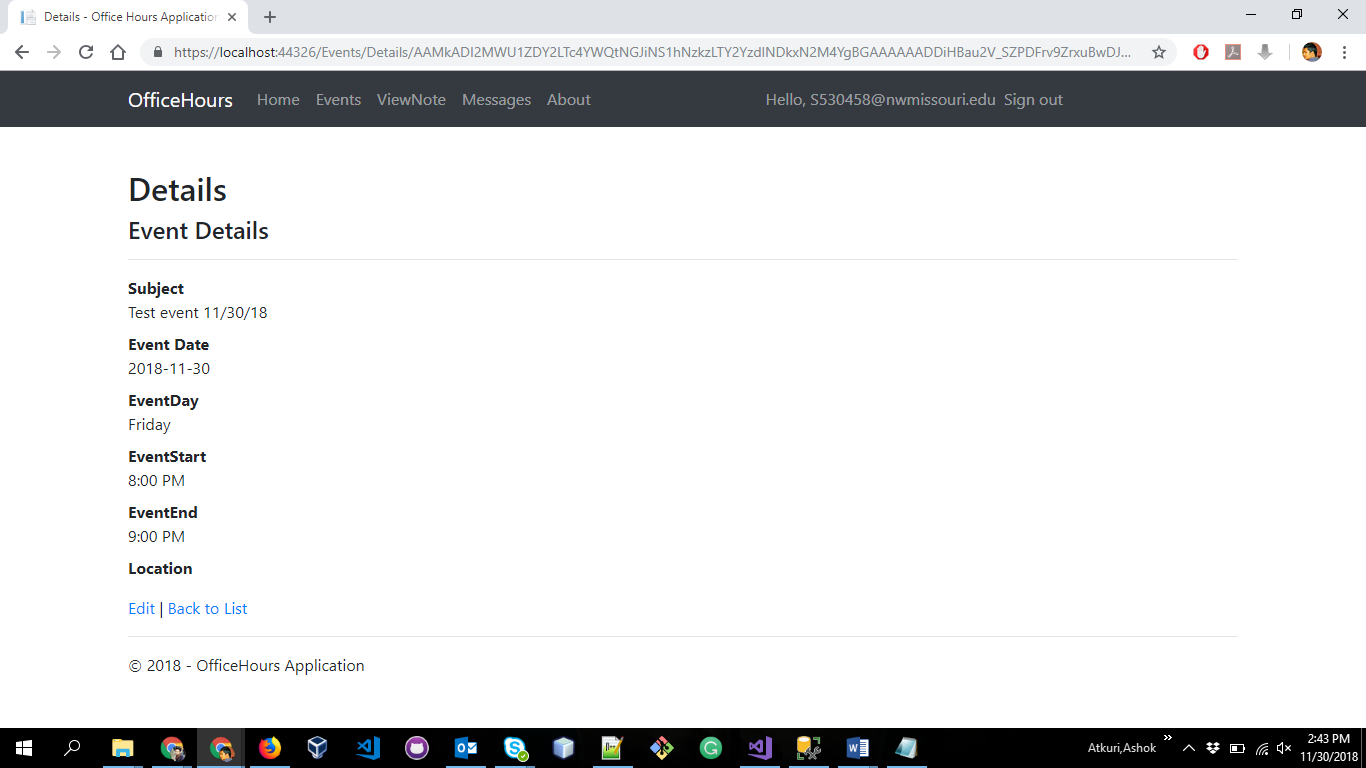
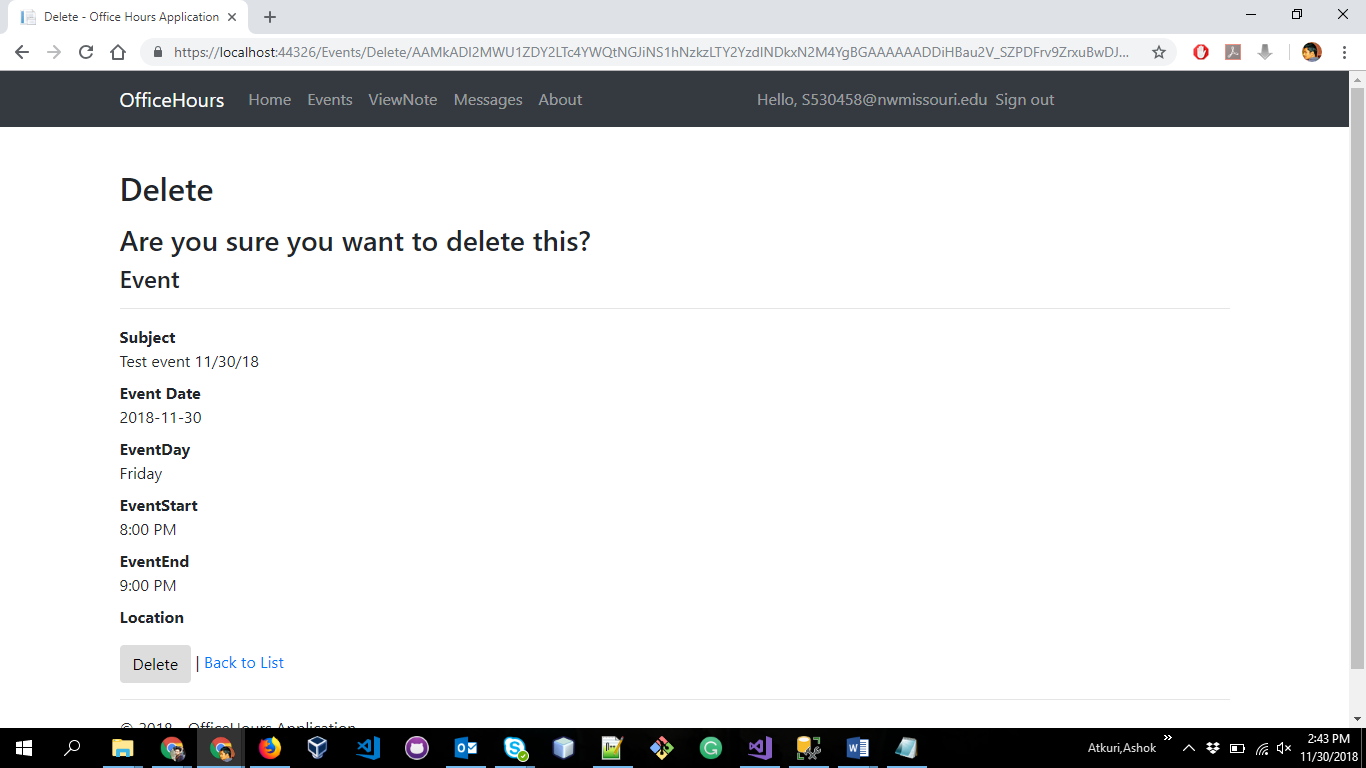
**Login page**

  
  
**Faculty Home Page:**  
  


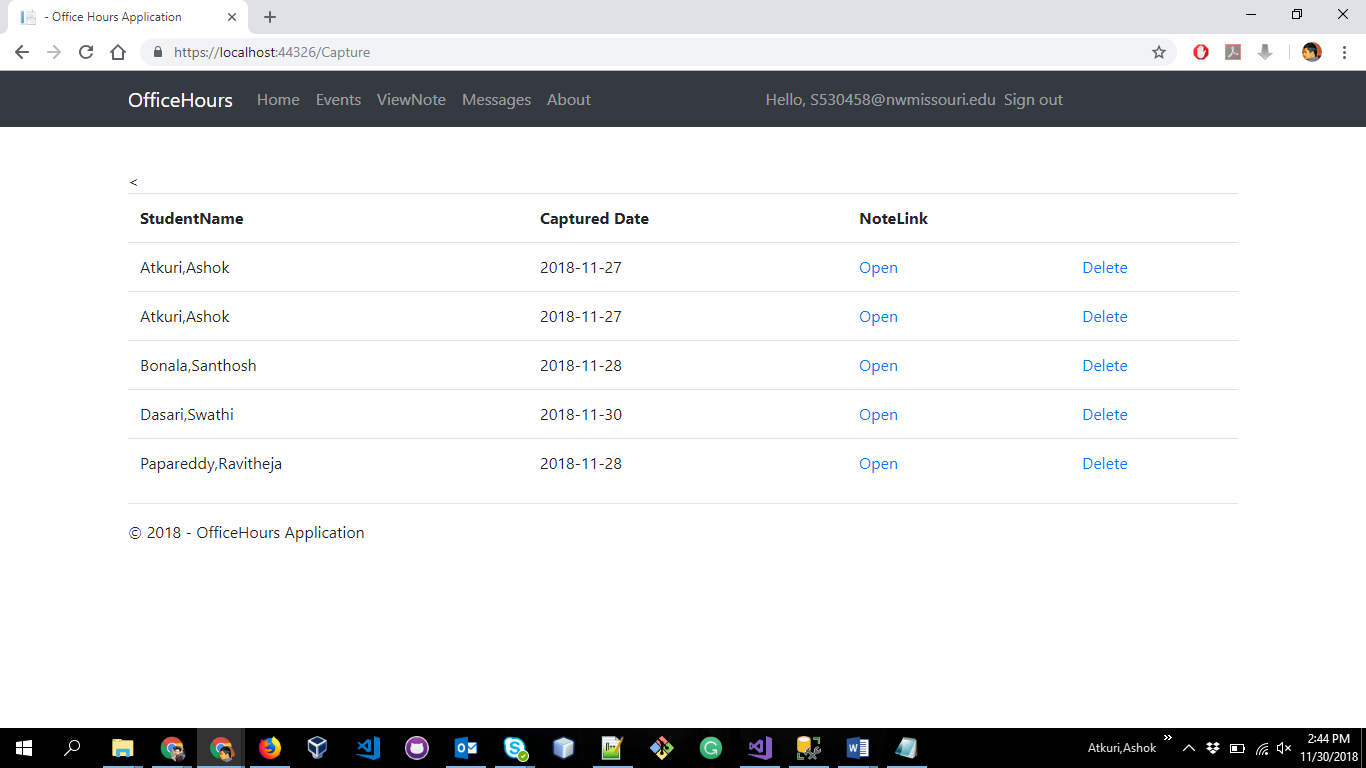
**Events List:**

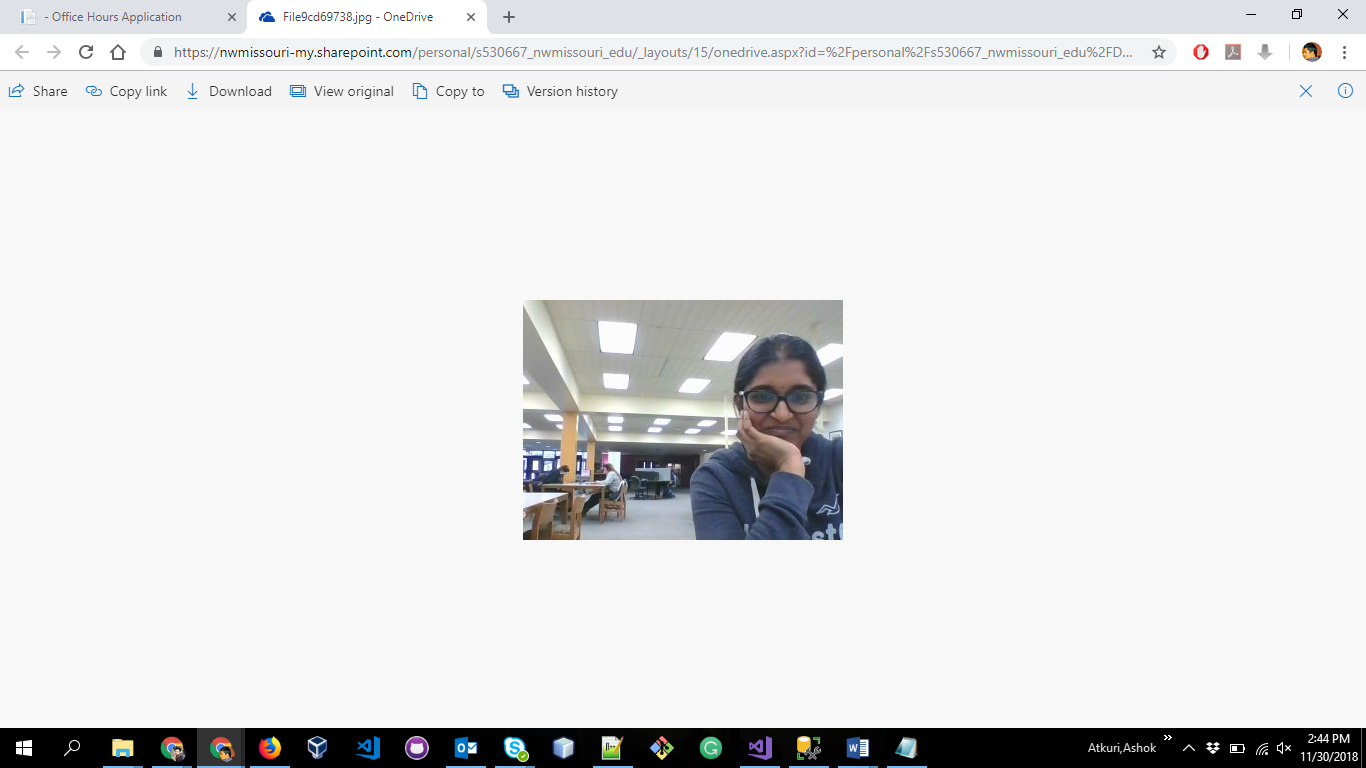
  
  
**Create Event:**  


**Event Details**

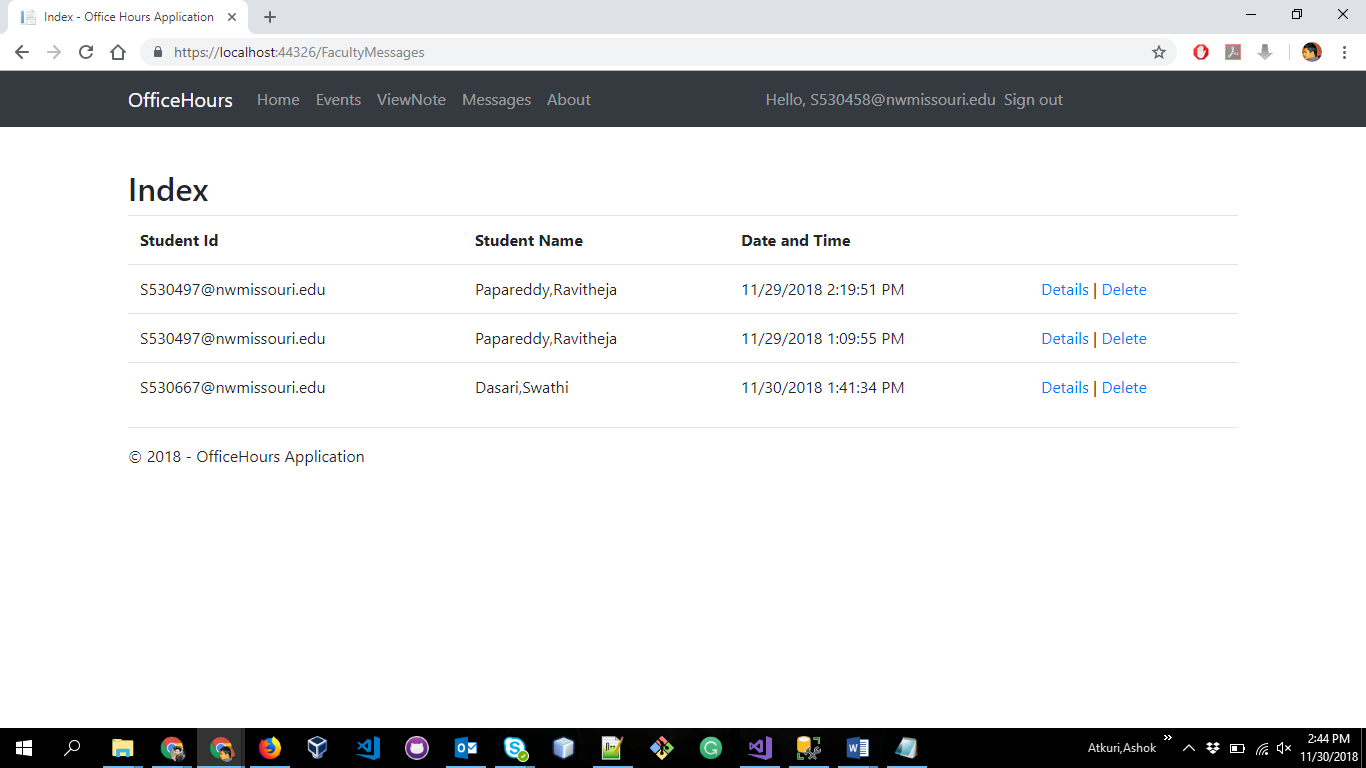
  
  
**Event Delete:**  
  


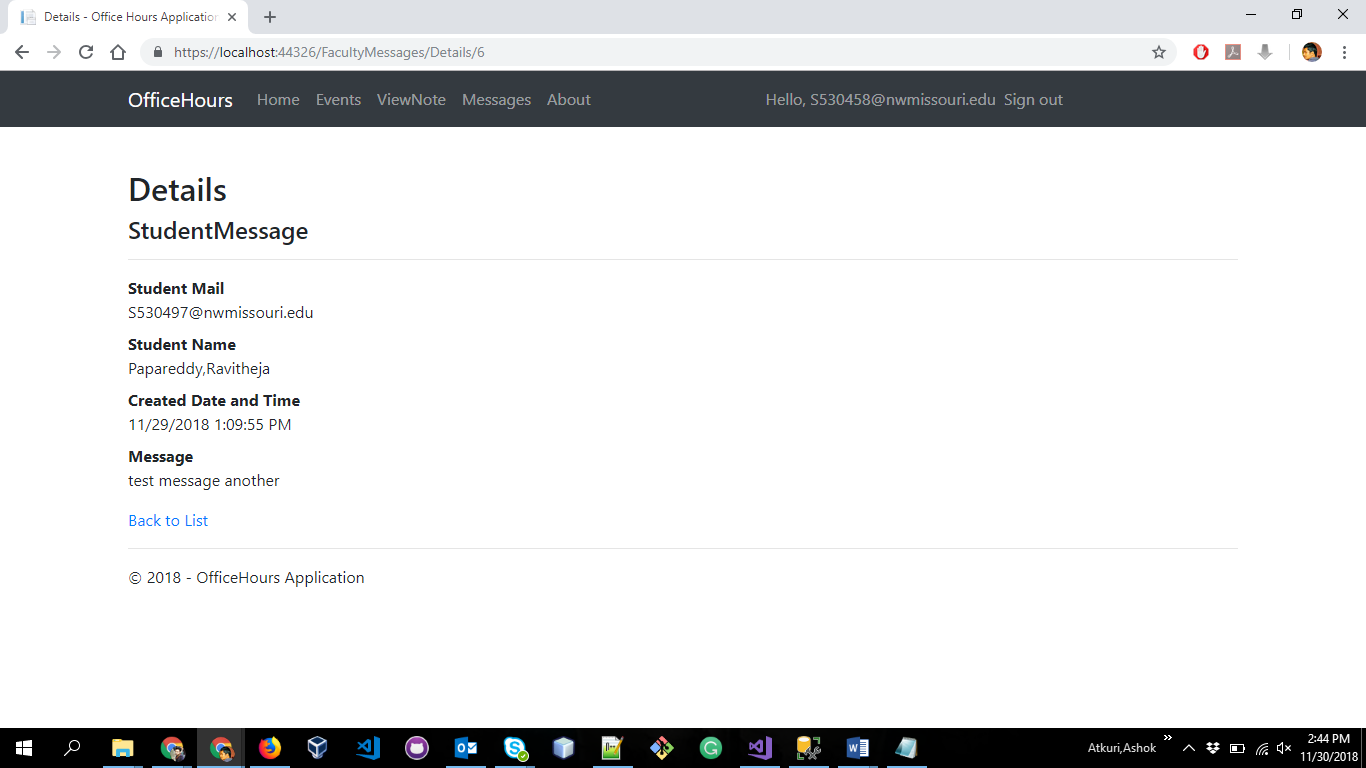
**View Notes(Pictures):**



**Note in new tab:**  


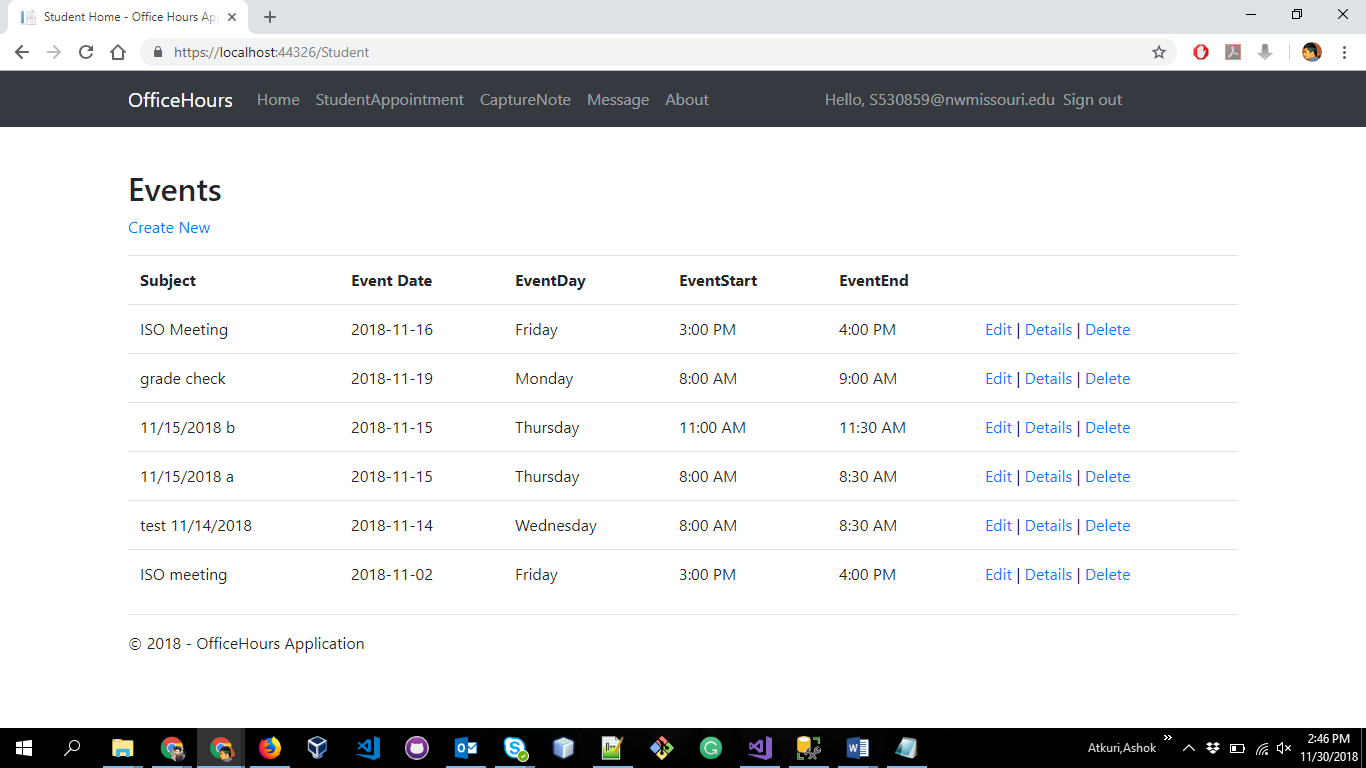
**Message View:**

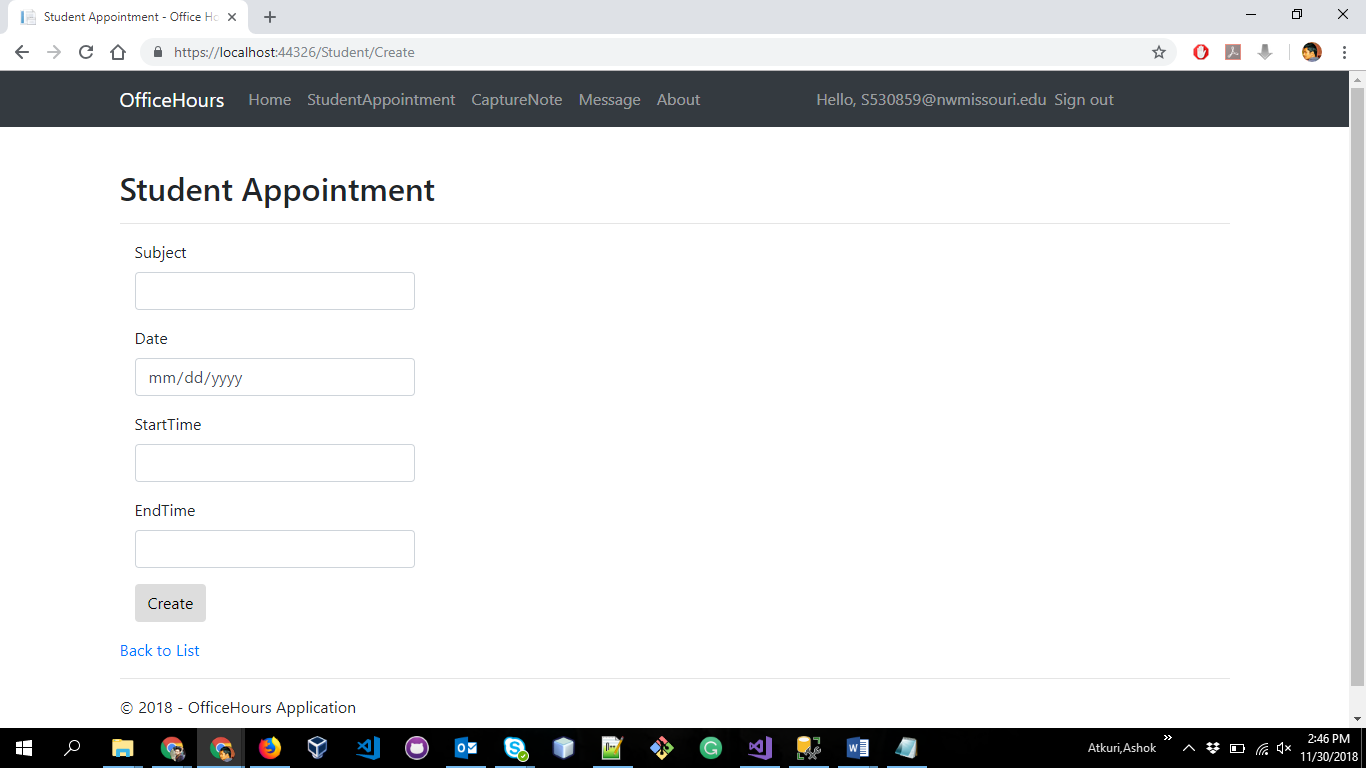


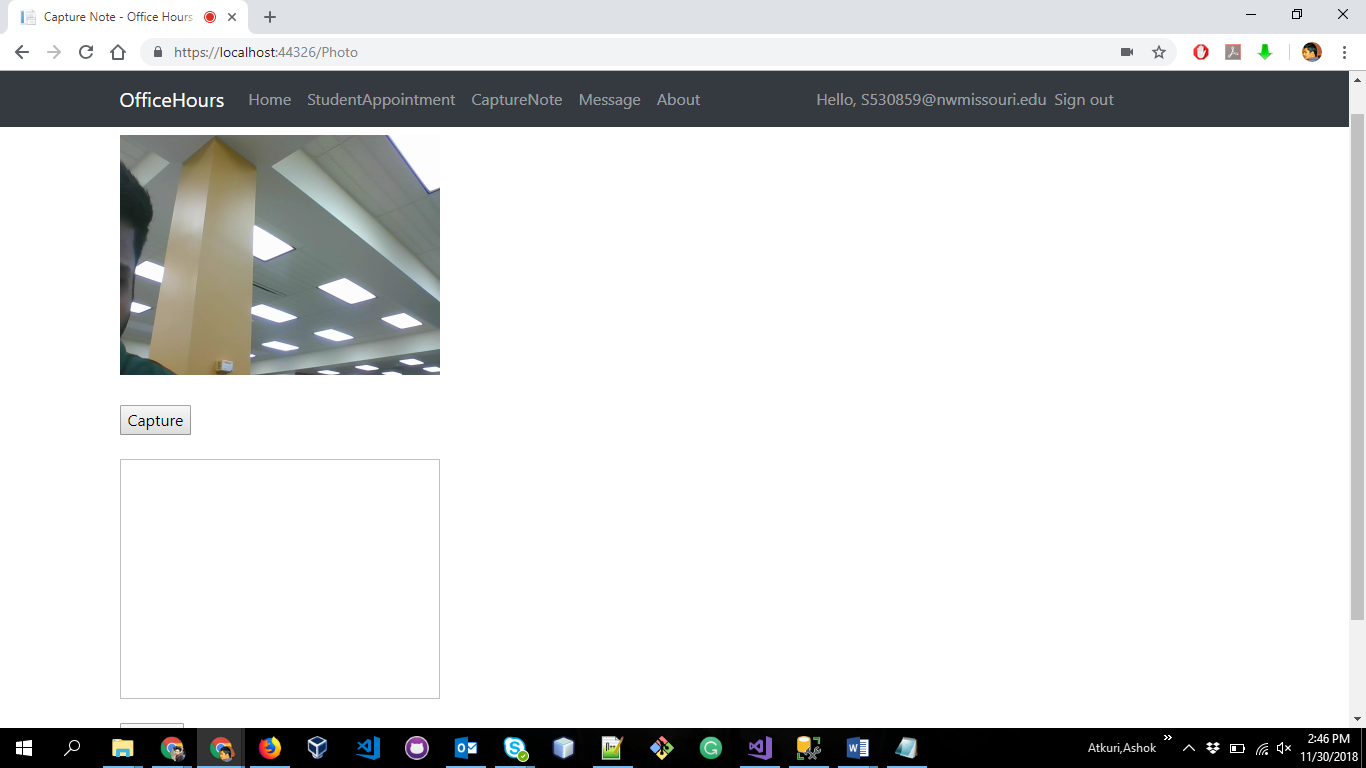
**Individual Message View:**

**Student View:**

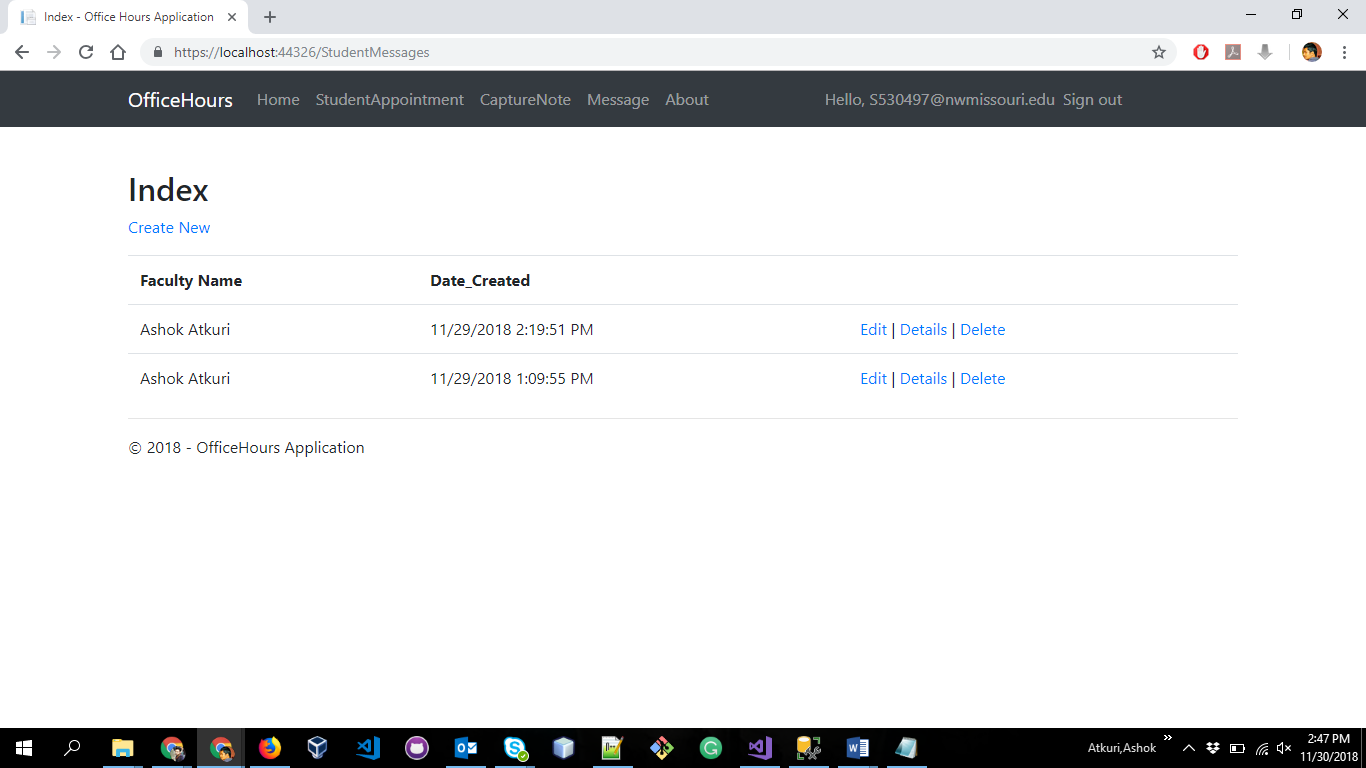
Events Home:

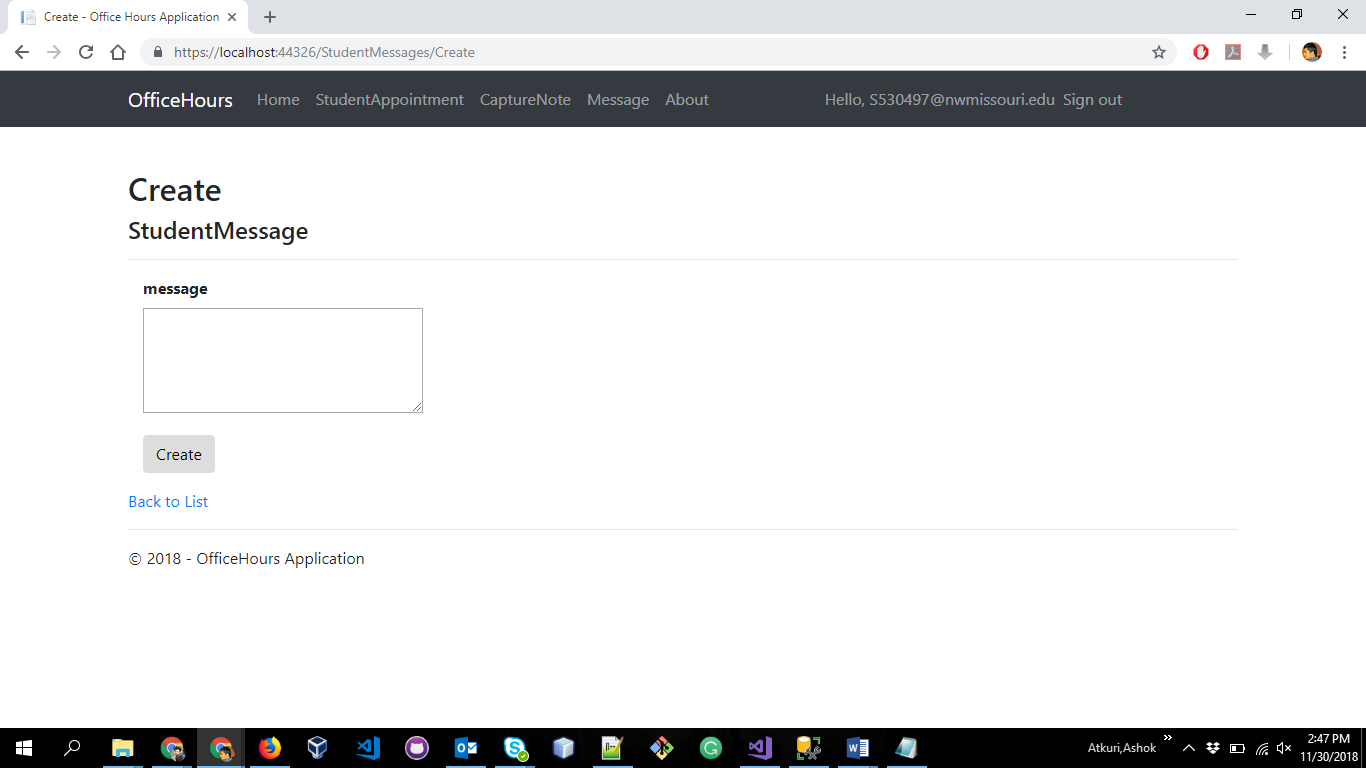


Book Appointment:  
  


Capture Note:

Message Home:



Create Message:  


1. **Installation guide:**

**Prepared by: Revanth Vallamsetty, Varshitha Guntakandla Date: 11/30/2018**

* Download and install Microsoft visual studio. Refer the below link:

<https://docs.microsoft.com/en-us/visualstudio/install/install-visual-studio?view=vs-2017>

* Download and install Microsoft SQL server 2017 professional with ssms/management studio . Refer the below link:

[https://www.microsoft.com/en-us/sql-server/sql-server-download](https://www.microsoft.com/en-us/sql-server/sql-server-downloads)

* Register the sample on the app registration portal.
* Create a new app at apps.dev.microsoft.com, or follow these detailed steps. Make sure to Copy down the Application ID assigned to your app, you'll need it soon.
* Add the Web platform for your app.
* Enter the correct Redirect URI. The redirect uri indicates to Azure AD where authentication responses should be directed - the default is [https://localhost:4279/](https://localhost:44326/).
* Add a new application secret via the "Generate new password", and save the result in a temporary location - you'll need it in the next step.

**3. Execution guide:**

**Prepared By: Revanth Vallamsetty , Ashok Atkuri Date : 11/30/2018**

1. To get the working files into a local computer, clone the repository.

**Repository link:**<https://github.com/RevanthVallamsetty/GDPOfficeHours>

**Steps to Clone:**

* Click the repository link mentioned above, select Clone this repository.
* Copy the clone command (either the SSH format or the HTTPS)
* From a terminal window, change to the local directory where you want to clone your repository.
* Enter the command git clone and the following link <https://github.com/RevanthVallamsetty/GDPOfficeHours>
* Enter the github credentials and the repository will be cloned into the local system.

2**.**  Click the officehours.sln file in visual studio and right click on the project and click on build.

3. Open the Package manager console present in tools and run below command in console:  
 nuget restore OfficeHours.sln.

4. Create a Database in SQL server and change the connection string in the Web.config file in the project. Then open the SQL server object explorer and connect to the database you created on the local machine before

5. .In the web.config file , Find the app key ida:ClientSecret and replace the value with the application secret you saved while installation/registering app in portal..Find the app key ida:ClientId and replace the value with the Application ID from the app registration portal, again in installation/registering app in portal..If you changed the base URL of the sample, find the app key ida:RedirectUri and replace the value with the new base URL of the project.

6. Then run the application by clicking on the button on the top menu bar labelled IIS express. This runs the app in the local IIS server. App will listen on <http://localhost:4279>

**4. Project itself (Copy and/or URL):**

**Prepared By: Varshitha Guntakandla Date : 11/30/2018**

[**https://github.com/RevanthVallamsetty/GDPOfficeHours**](https://github.com/RevanthVallamsetty/GDPOfficeHours)

**VI. Project Management Artifacts**

**1. Risk list, pros and cons**

**Prepared by: Varshitha Guntakandla Date: 11/30/2018**

Pros:

* User friendly website
* Easy steps to get started
* Good communication channel between students and professor
* Easy way to schedule a appointment
* Reduces lot of tedious process and save time to users

Cons:

* Not budget friendly
* Student logging into the Tab for using services
* Microsoft provides a easy login service by remembering the password as soon as the username is given. Other users have a option to misuse it.

**2. Summary of requirements that have been completed**

**Prepared by: Swathi Dasari Date: 11/30/2018**

Professor and student is able to register, signup and logout

System display professor schedule up to date and the professor can able to make his/her schedule.

System display their availability status indicator all the time and it display the professor schedule all the time.

System allow to book an appointment by the students with particular time and date.

System allow to take the pictures of important papers of the students before they submit.

System shows the slots as unavailable once the appointment is made at that time.

Student is able to leave a message to the faculty

The system is able to provide the number of appointments booked on particular day

The appointments made by the students are synchronized to the outlook calendar

The students gets notification mail when the professor accepts or decline appointment.

It provides security for the application.

All the credentials , photos , messages are stored into the database.

**3. Summary of requirements that have not been completed**

**Prepared by: Ashok Atkuri Date: 11/30/2018**

Making a video/ audio call to the professor

The system MUST show the slot as unavailable once an appointment is made at that time

**4. Conclusion**

**Prepared by: Varshitha Guntakandla Date: 11/30/2018**

Day to day scheduling and updating is a repetitive process for the university professor. The main problem occurs when the schedule of the advisor changes, they notify the changes by sticking the printed paper on the door. Another problem occurs while booking an appointment by the students always needs email Id and student should always drop a mail which may not be seen by other students. The project application provides details like schedule, booking an appointment by the student and sending message if the advisor is out of station. Dropping an email or booking appointment directly with voice rather than sending messages and easy access of the appointments for the advisor wherever he is. Providing security for the data throughout the process.