**ONLINE BUS BOOKING**

**Theory:-**

Online Bus Booking System is Web Based application,That works within a centralised network,It Provides facility to reserved seats,different types of enquiry which need an instant and quick reservation. Buses may be used for scheduled bus transport, scheduled coach transport, school transport, private hire, or tourism; promotional buses may be used for political campaigns and others are privately operated for a wide range of purposes, including rock and pop band tour vehicles.

**This project is modularized as the following:**

* **Management of Route**
* **Trip Details**
* **Bus Details**
* **Bus Ticketing**

**1. Management of Routes**

This module includes information about how we can Manage the routes for a particular bus service. In the case of the Route management module we must know the details about route number, number of stops ,fare stages and running time of the particular bus. Moreover, we want to manipulate and store this information successfully.

**2. Trip information**

Each journey is identified as a trip. Each ticket must contain the trip no so that calculation of passenger can be done easily. Here in this section we want to know the start time and route no of the bus this information can be manipulated and stored successfully.

**3. Bus Detail**

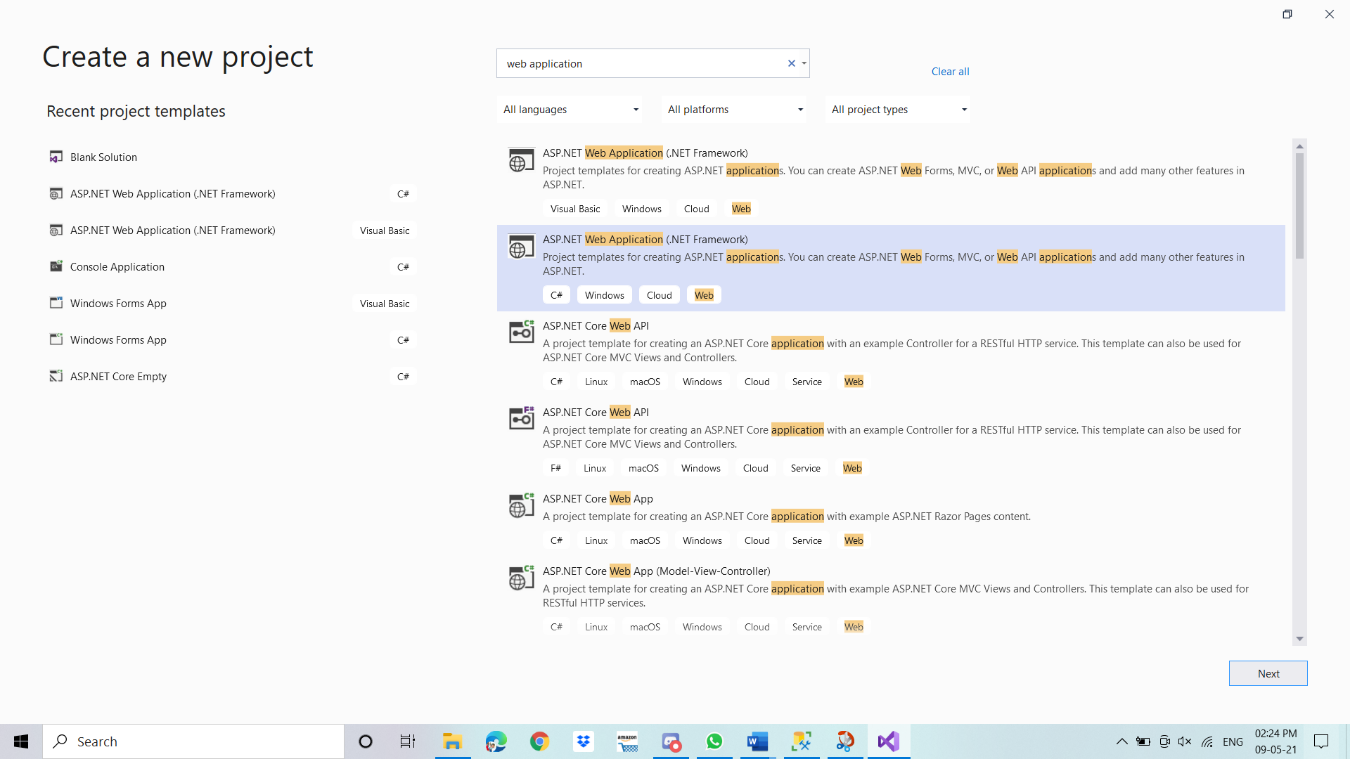
In this module all bus details are stored and manipulated, in the bus detail module contains minimum charge, type, depot, fare increment, bus number, and passenger’s states (child or adult) are manipulated and stored.

**4. Bus Ticketing**

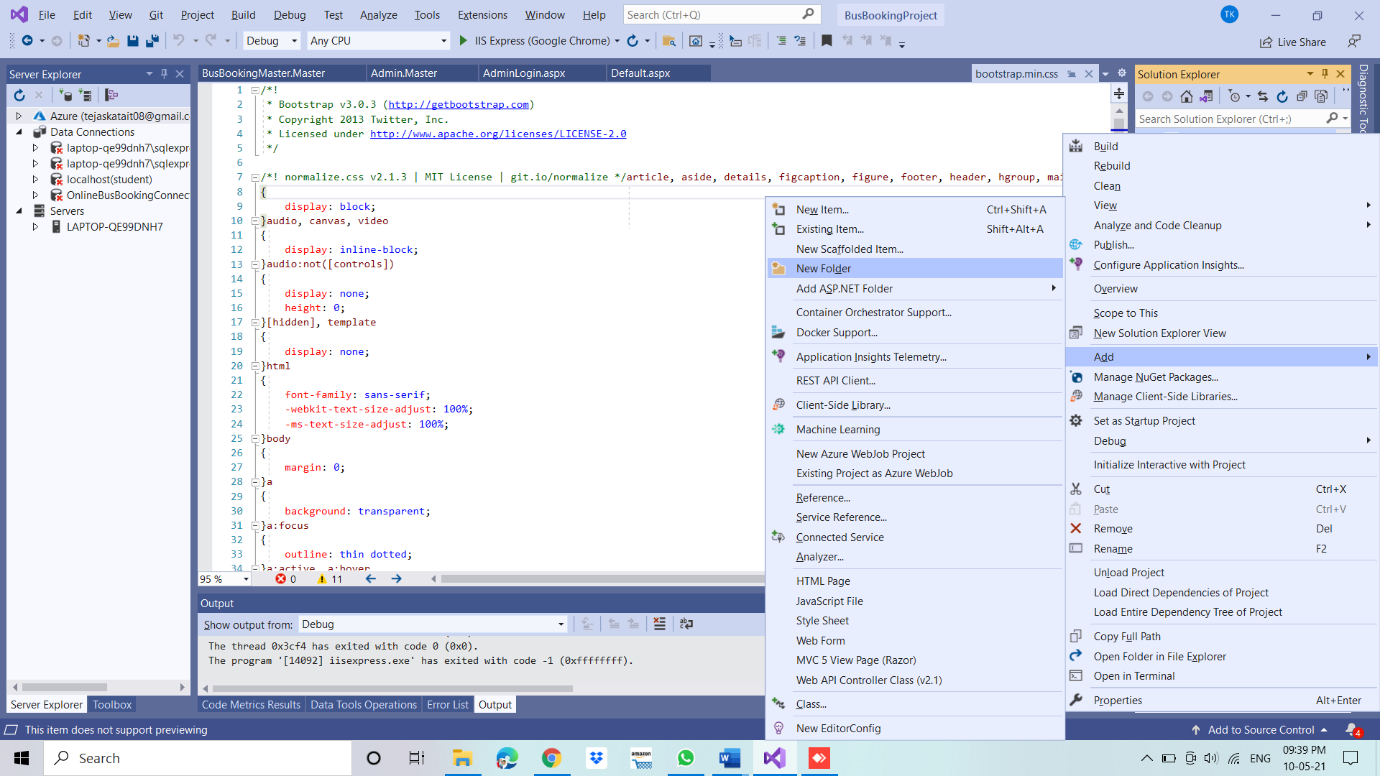
Ticketing is the most important module in this Project which uses all the tables together and calculates fare for the passengers. Venting the tickets is done using the route number, bus type, beginning stop, end stop, ticket number, persons(Adult/child)rate, date and time also we want to print all this information. In order to do the calculation data has to be pulled out from stops, bus, trip and route. Number of passengers & the states are entered by the Venter and to produce the tickets.

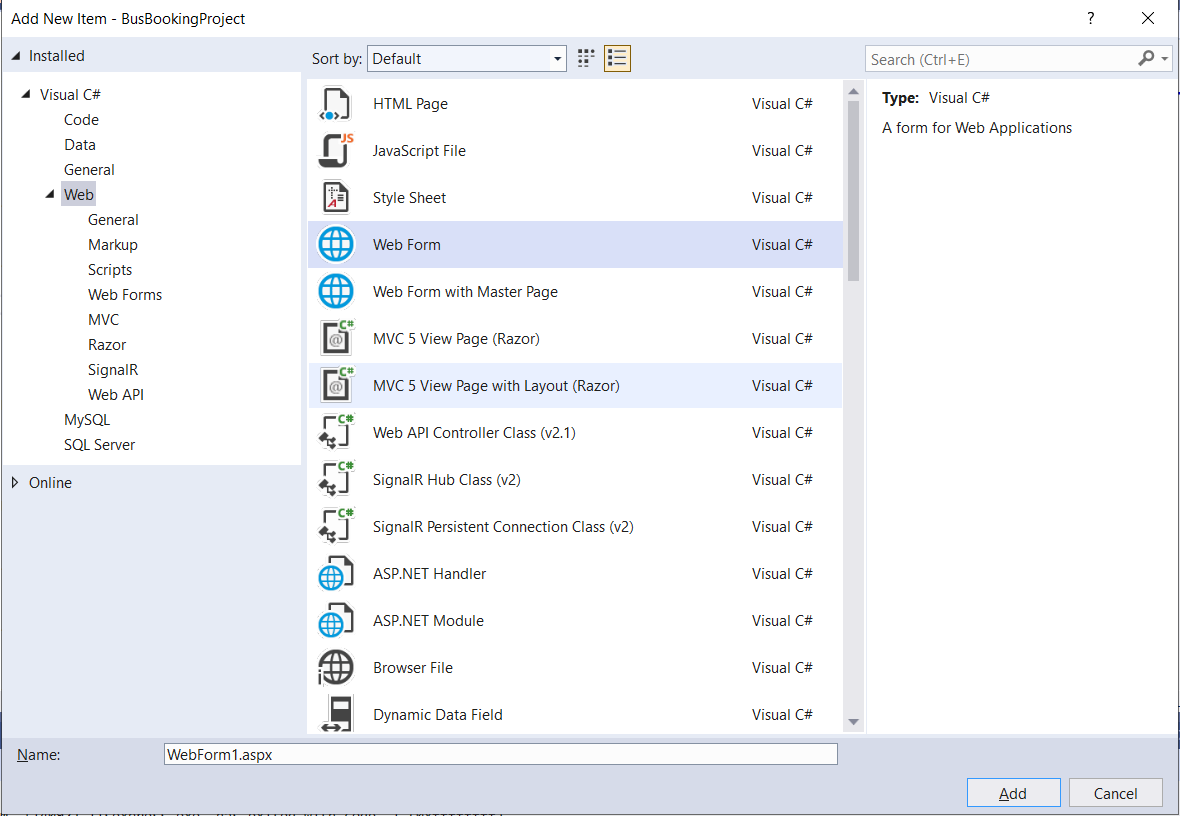
**Steps:**

1. In Visual Studio select new project
2. Select ASP.Net web application template
3. Name the project as BusBookingProject

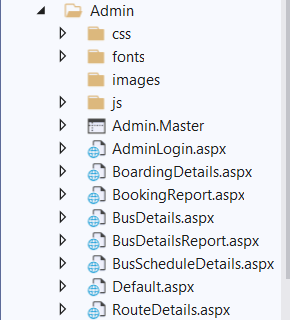


1. Now we create a separate Admin module in which we create all functionality of Admin.
2. Add new Folder name Admin in Project.

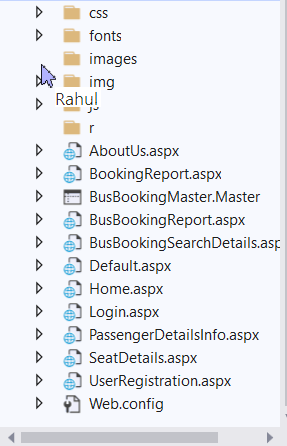




1. As shown in the above photos follow these steps to create new webform and folders
2. Create folders and webforms in Admin section as follows:



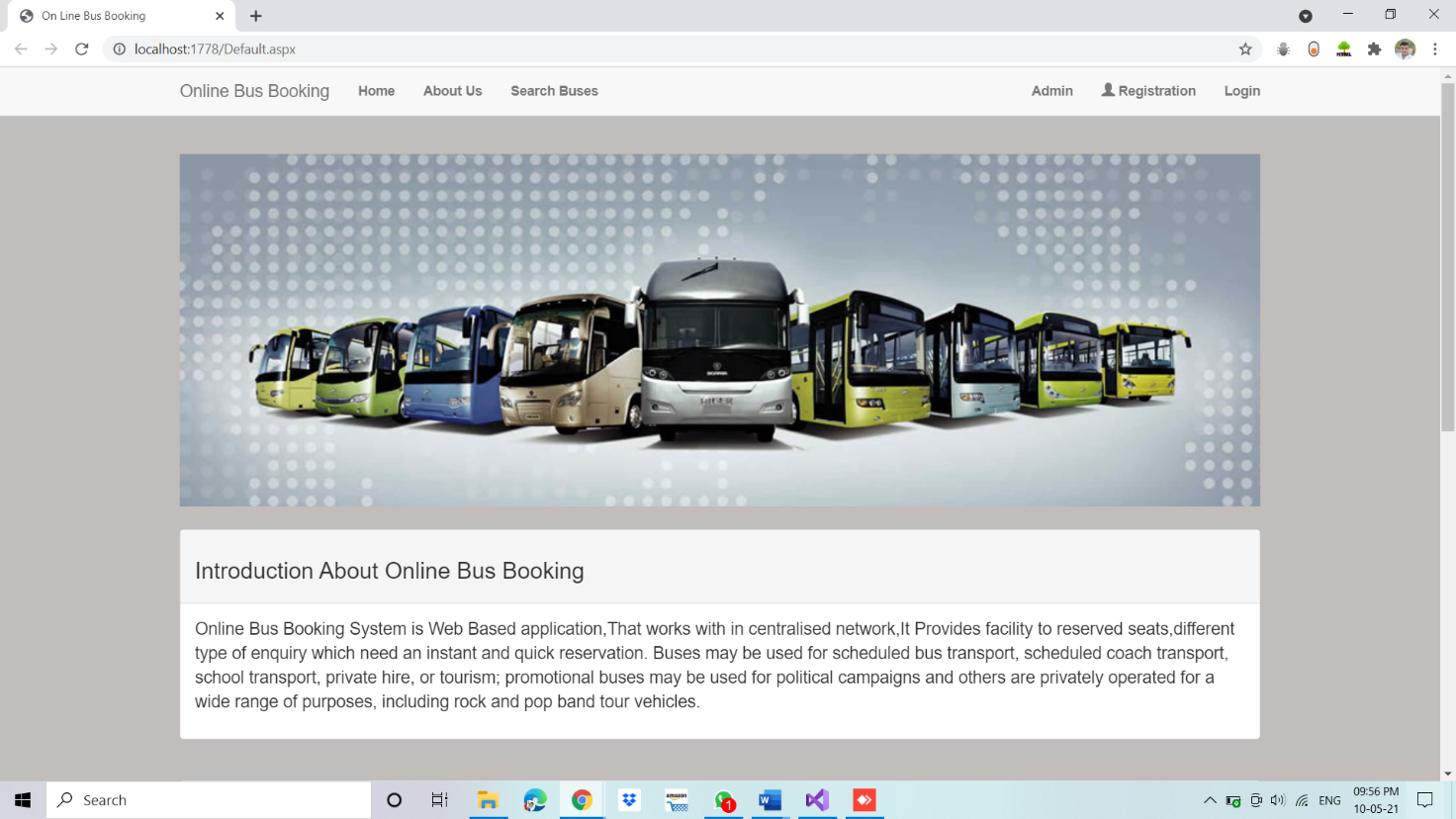
1. Now we will create our user module where main startup pages and other webform will be available.as shown in below picture:



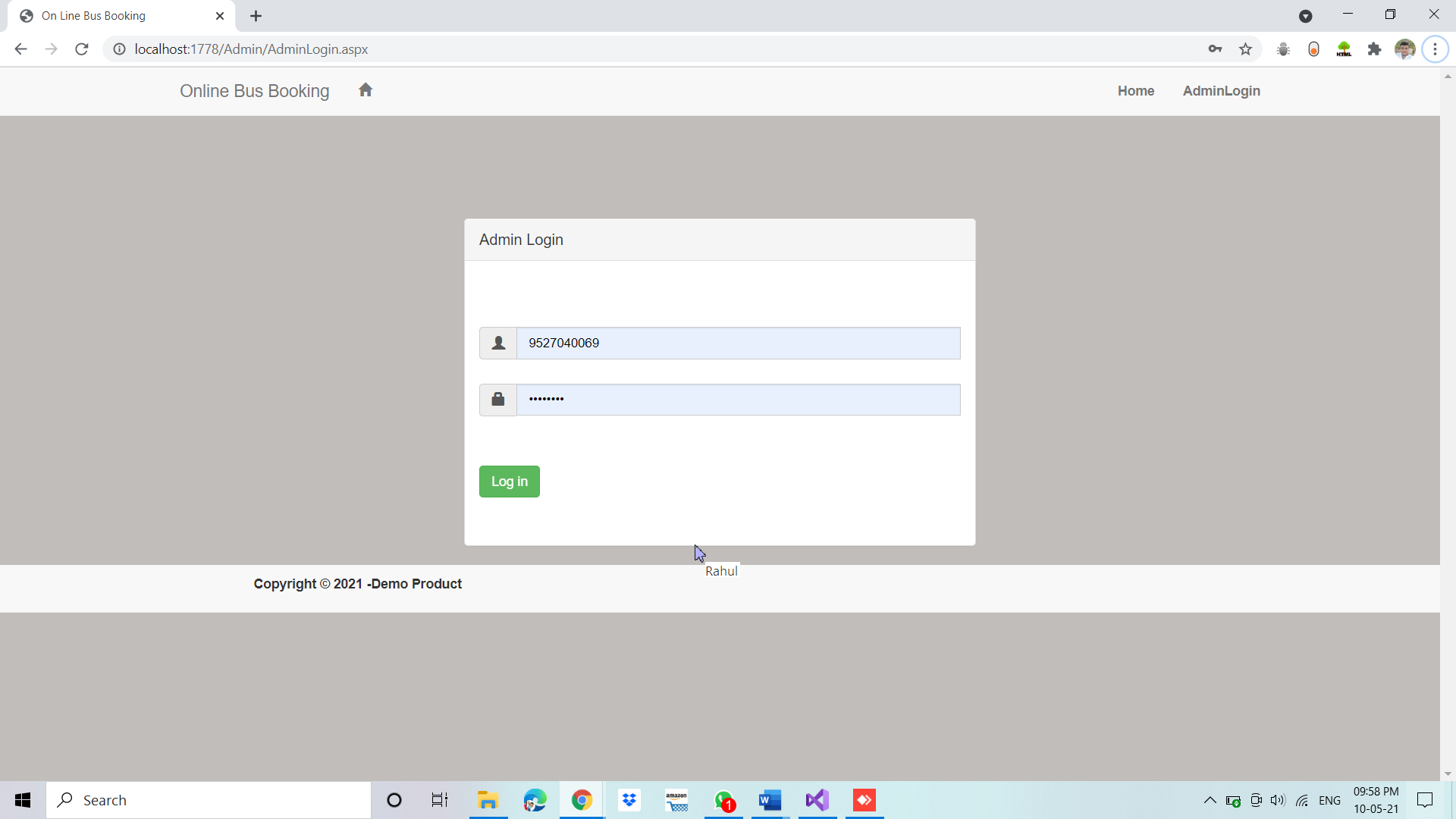
Now from here you can create your own project or you just open it in your visual studio. Database query is also available there.

Now execute your project by setting Default.aspx webform as startup page...

* So this is our startup page:

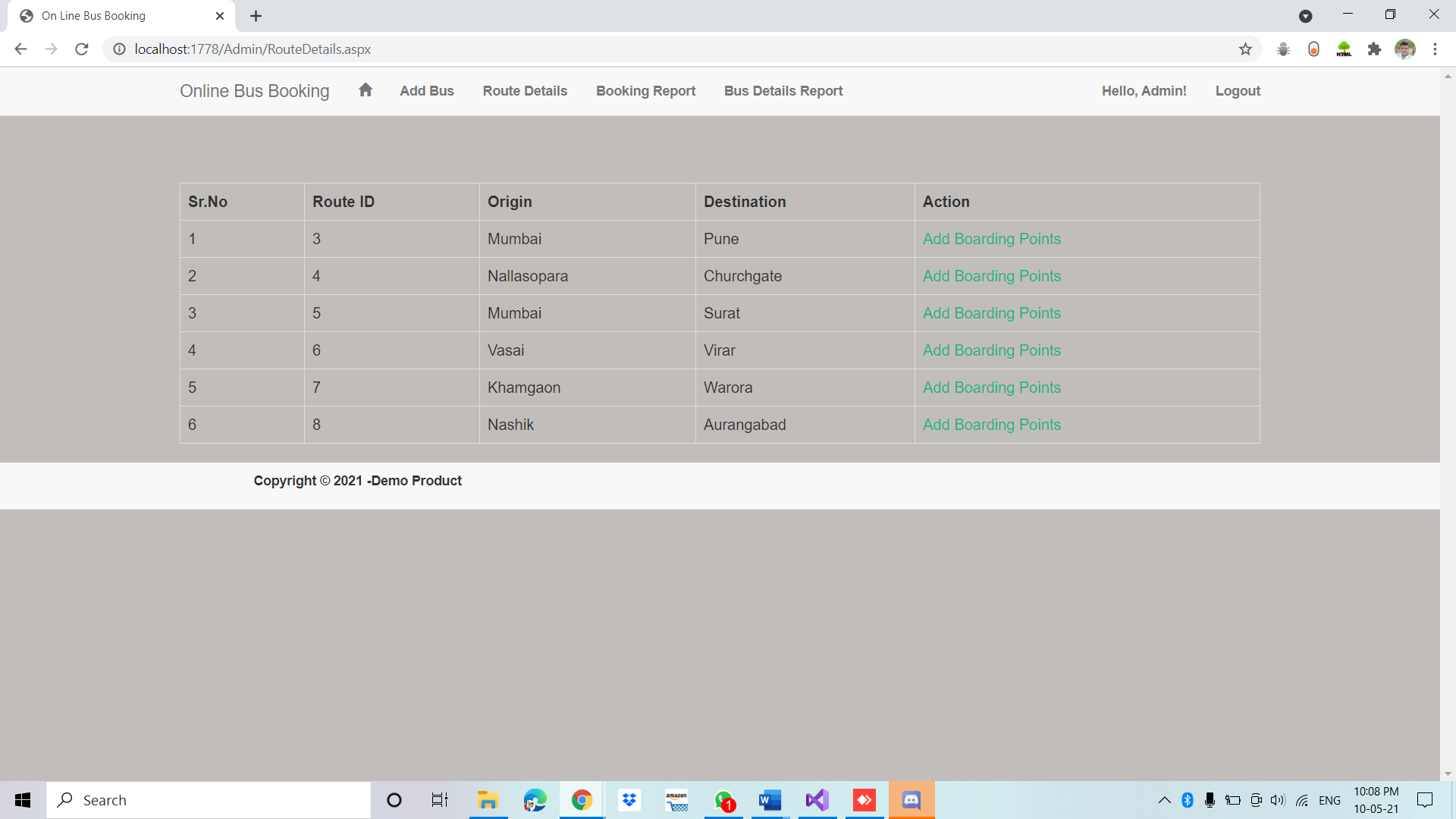


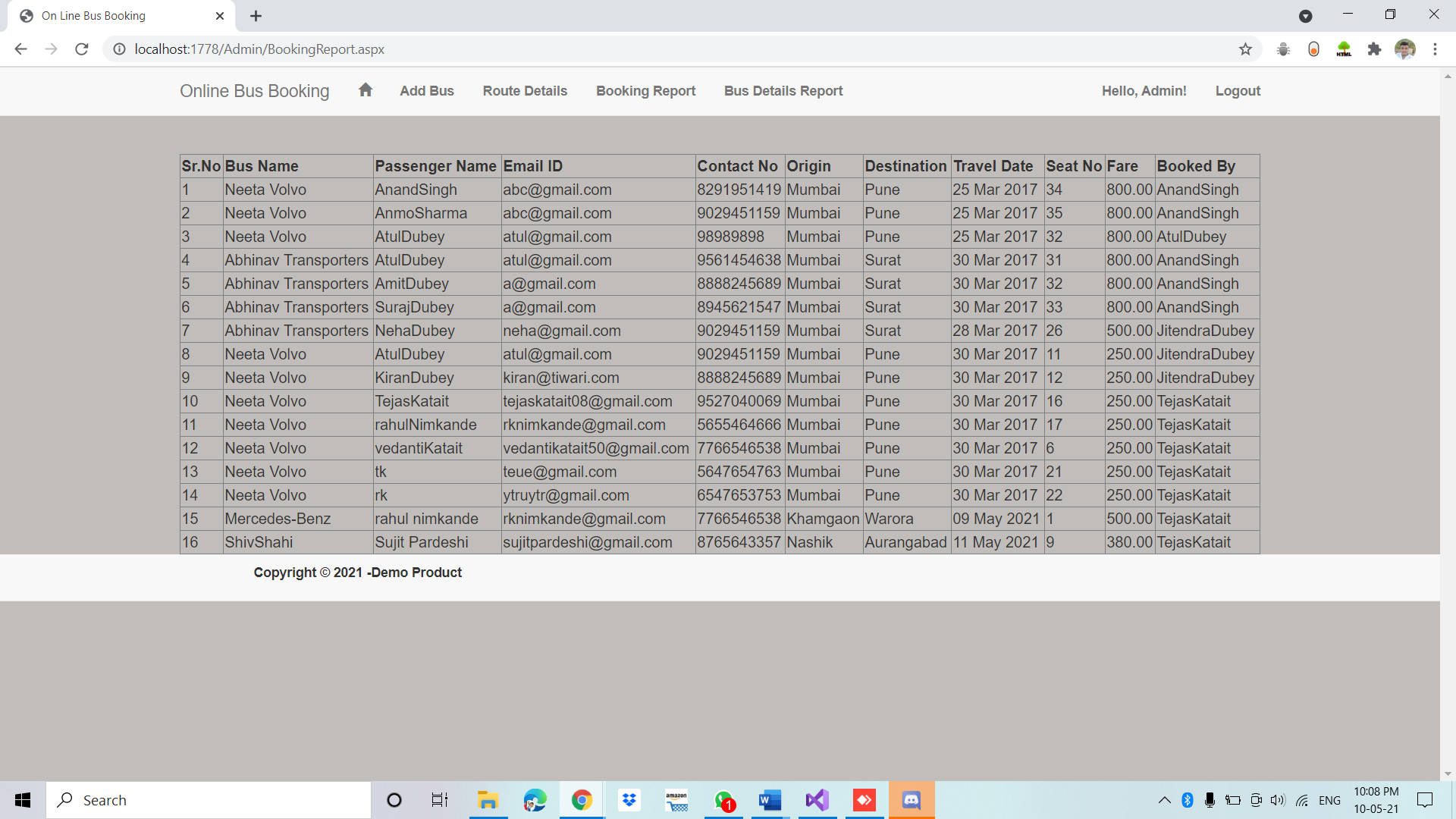
* Now by clicking on the Admin button you will redirect to the admin section.

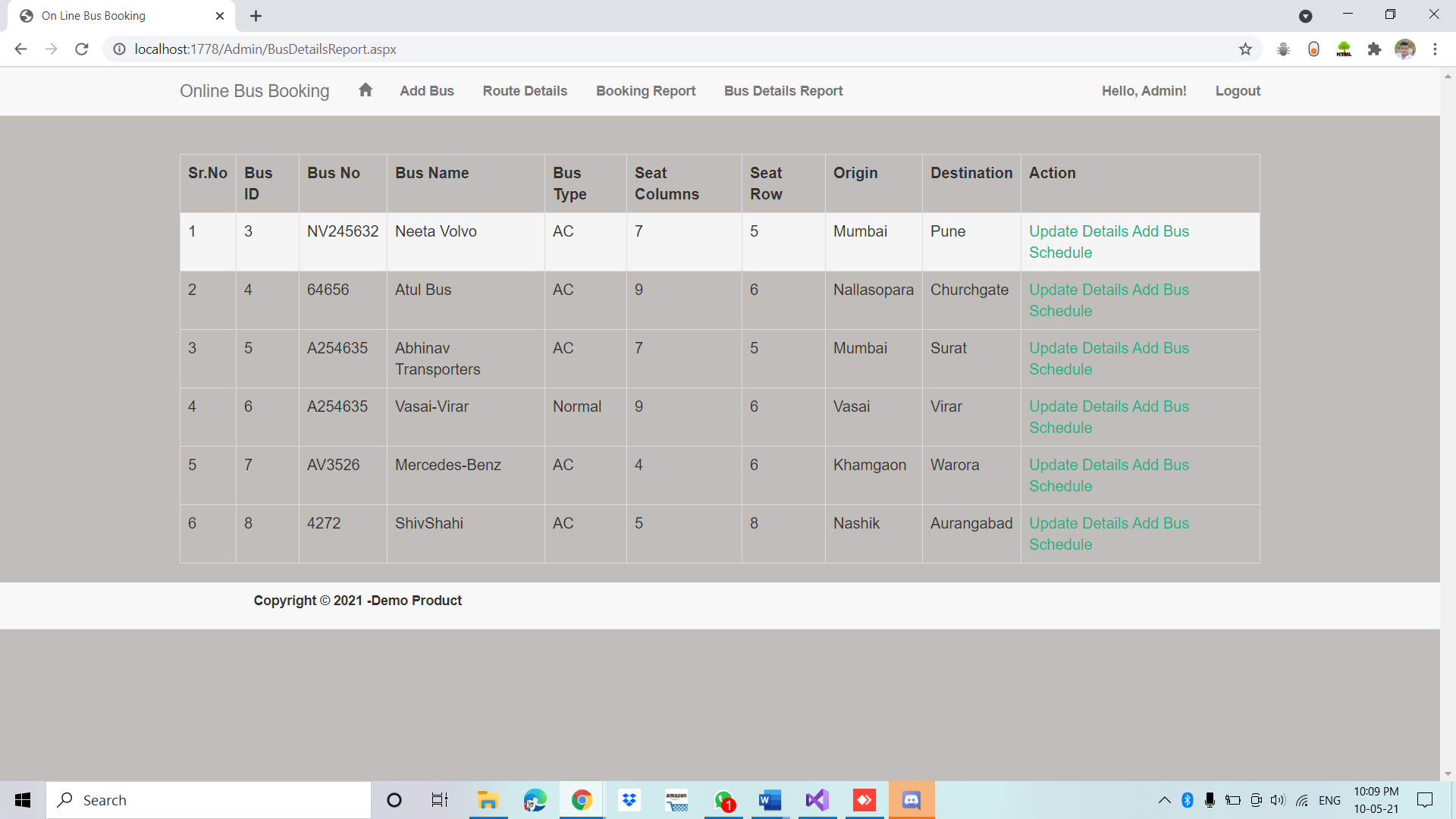


* After login, you will get options like add bus, Route details, Booking Report, Bus details Report.

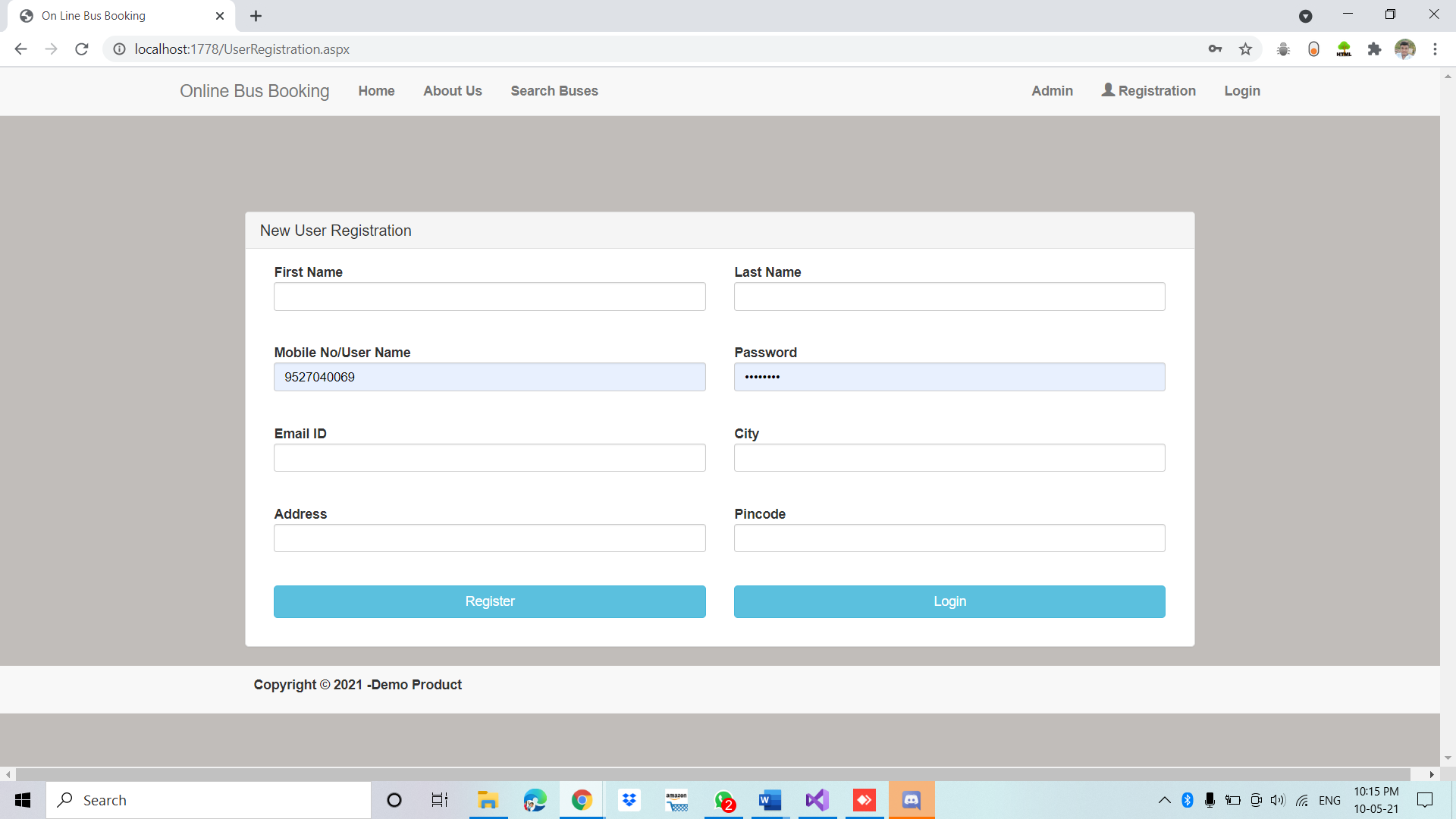


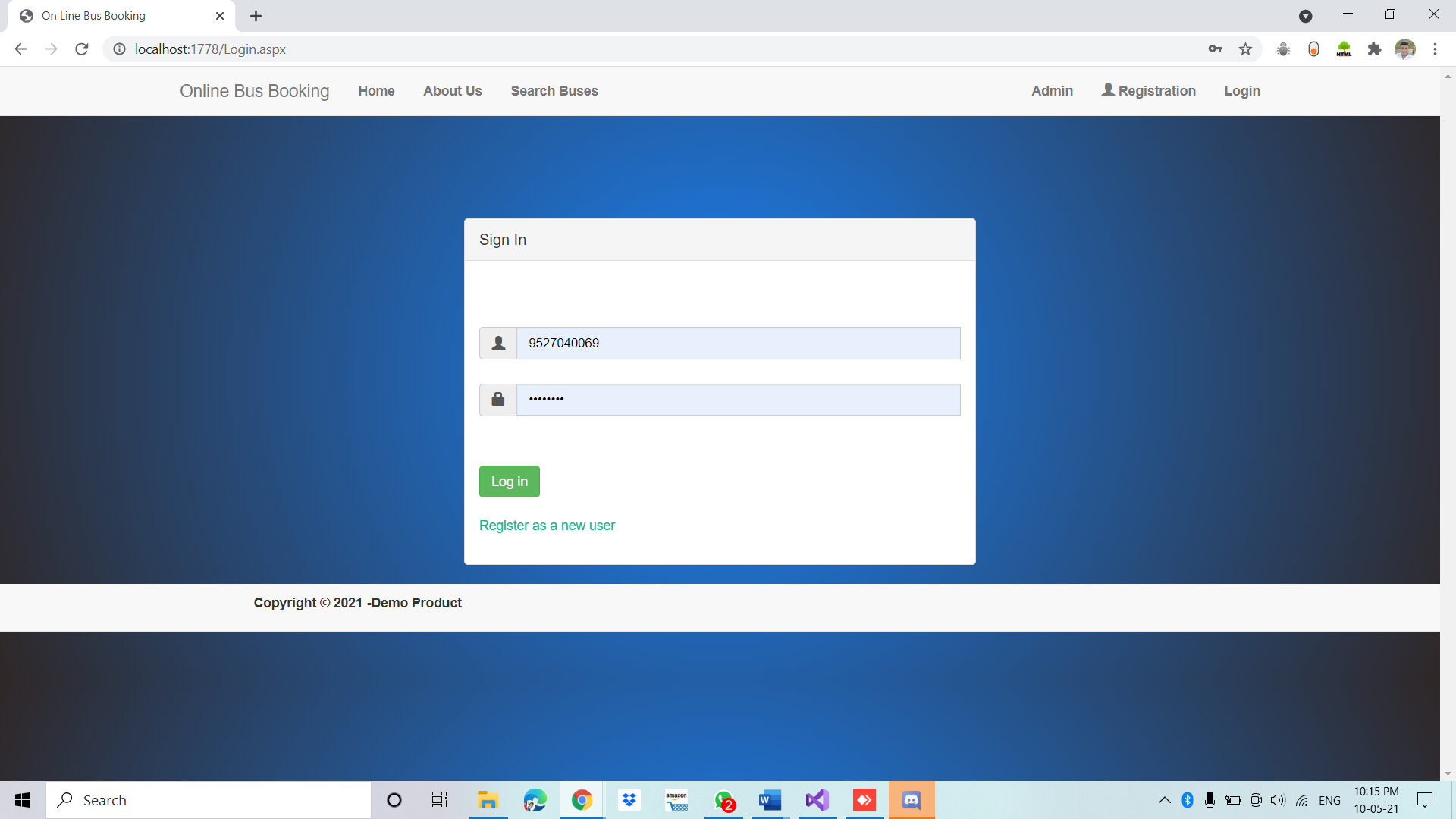




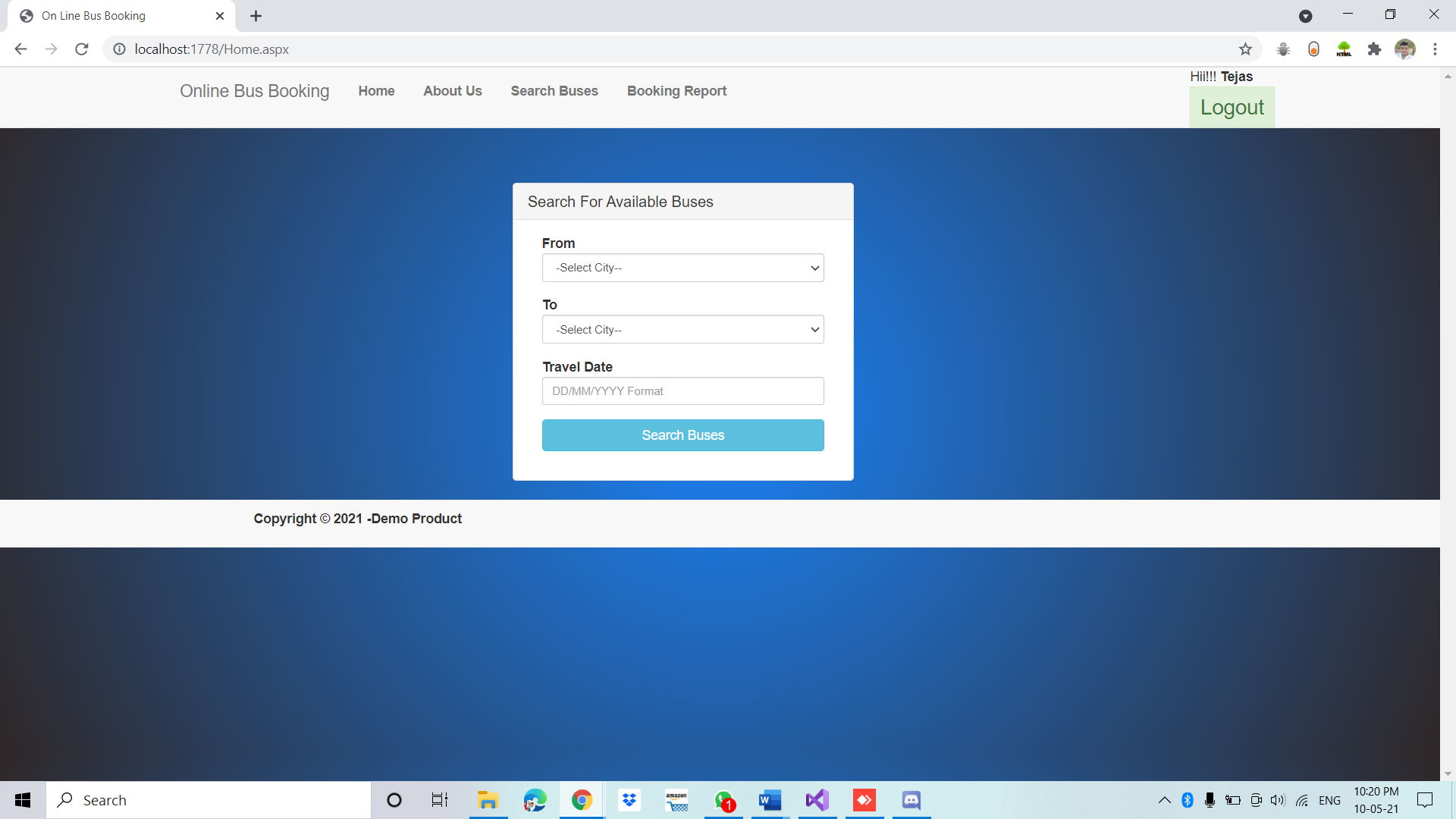


* Now in the User module default page you will get registration page and login page.



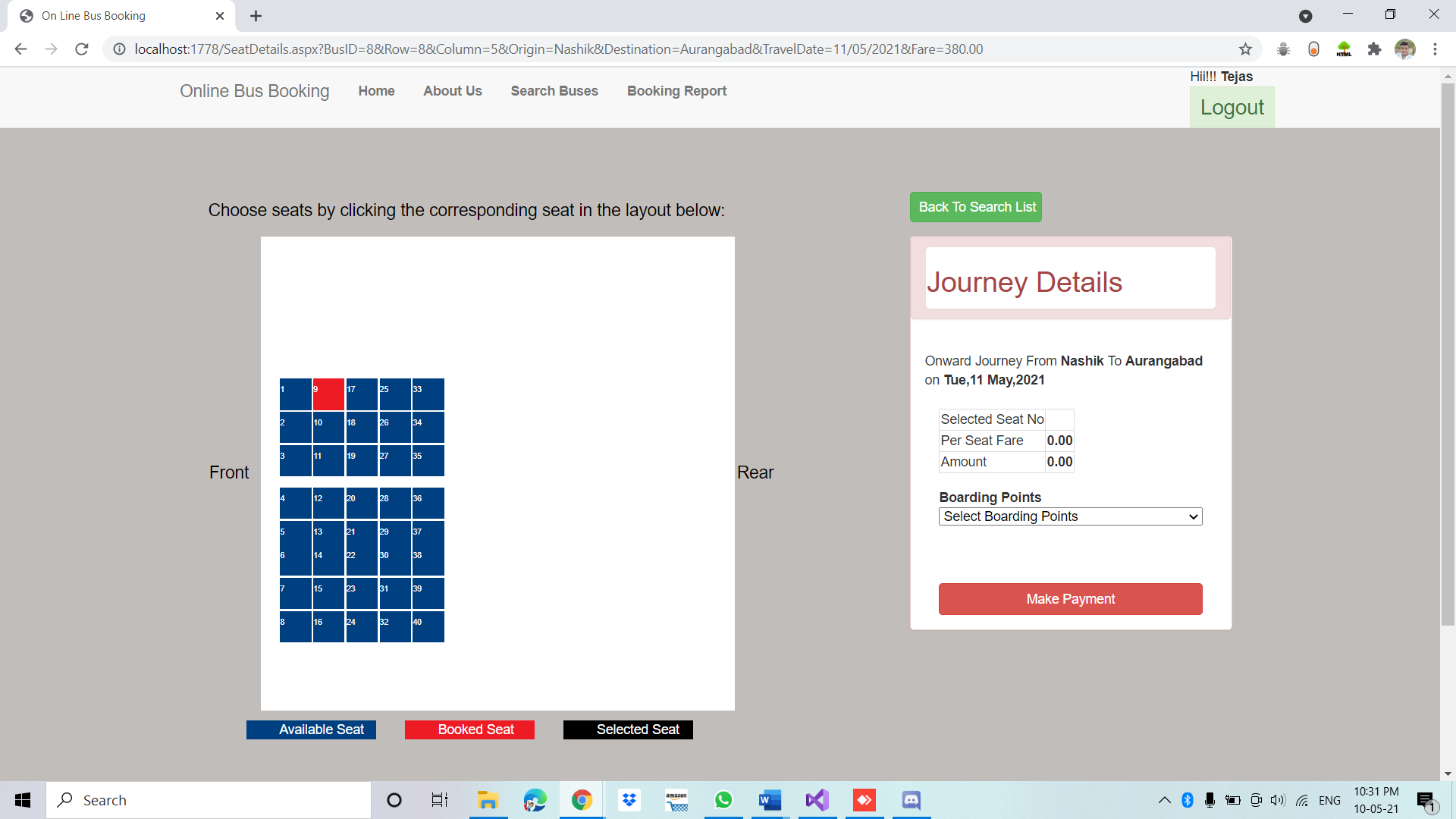


* After registering successfully you will get a username and password then login using it.
* After login you will get options like search buses, Booking Report.

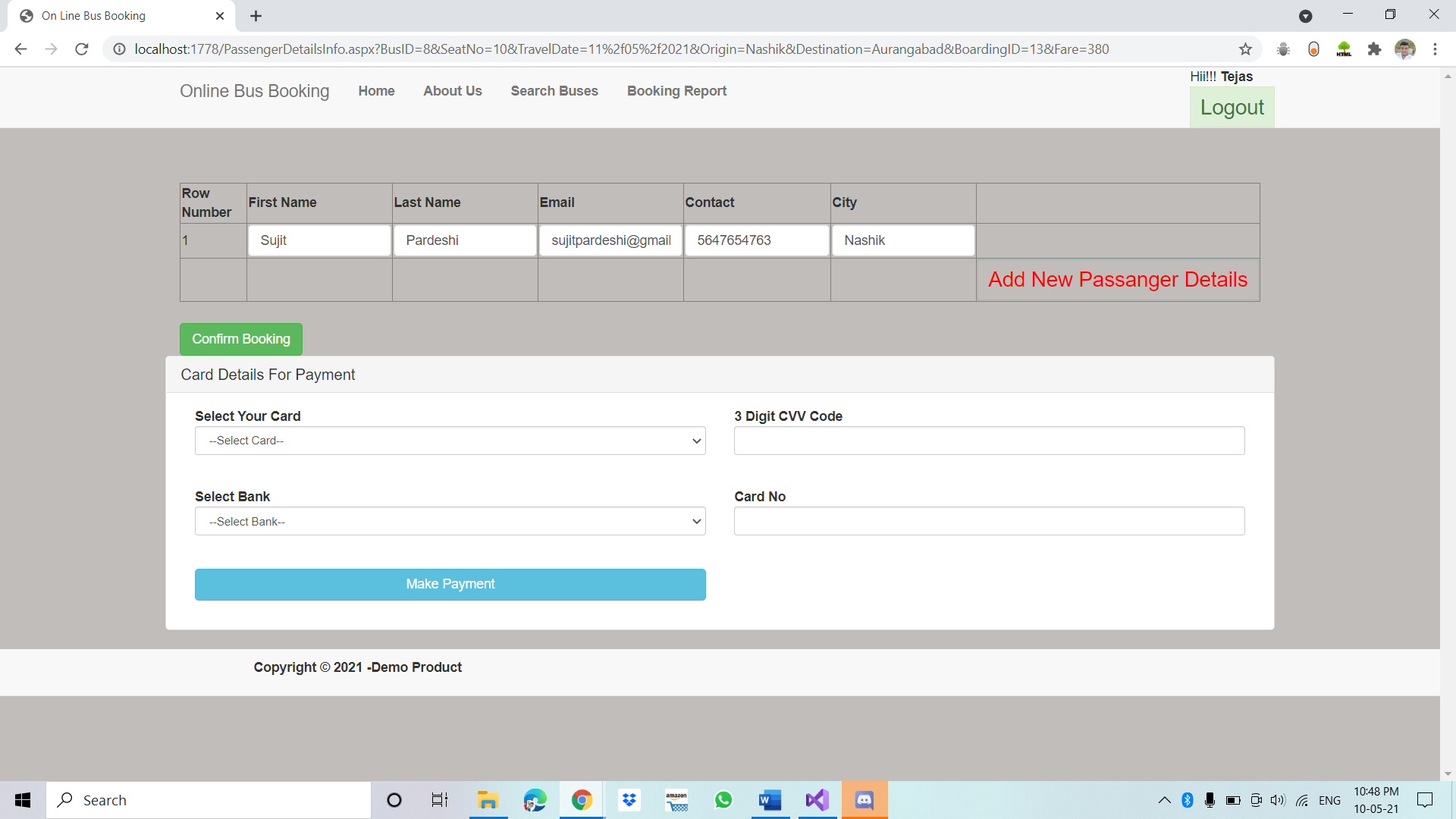


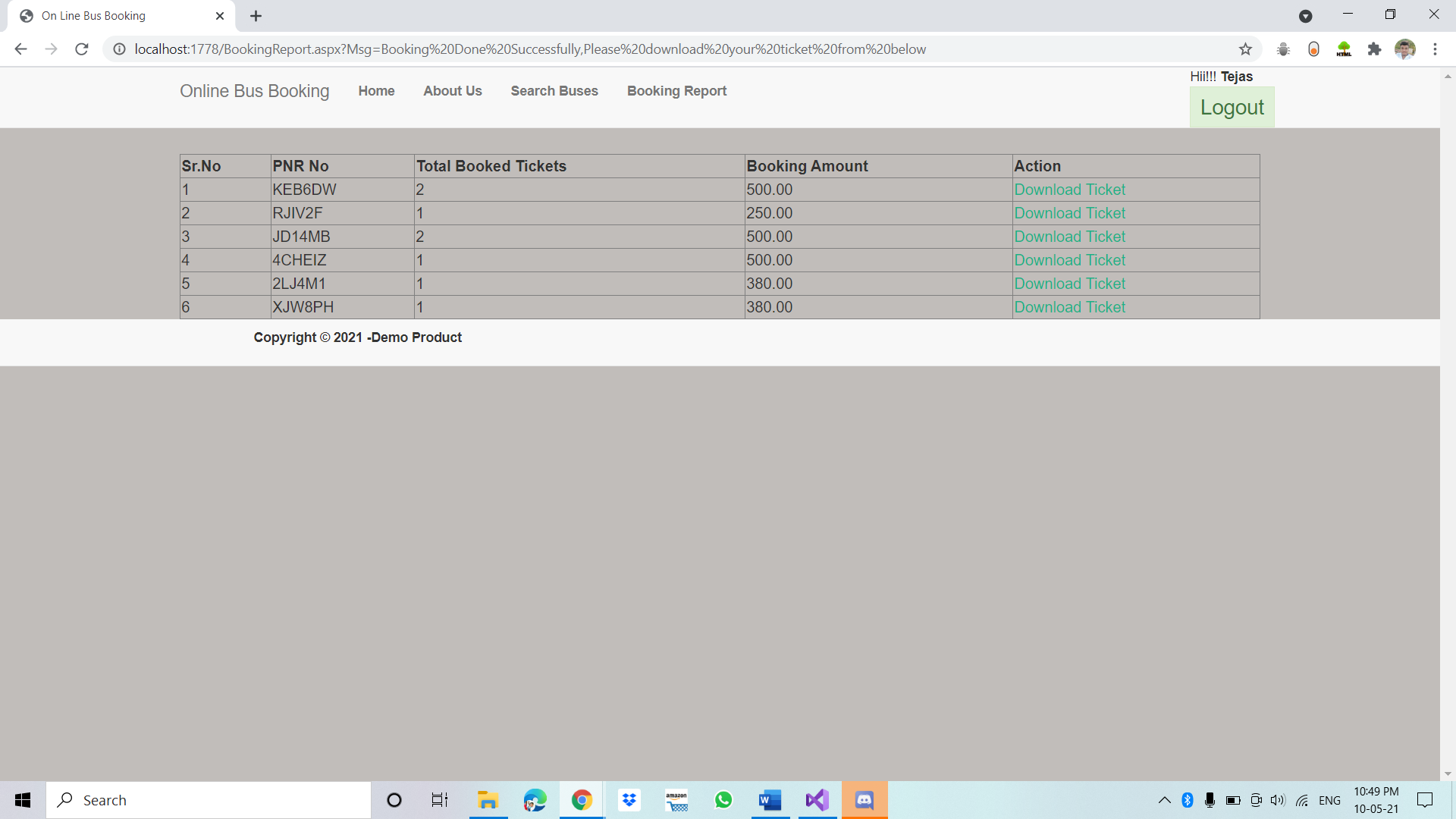


* After searching your desired bus you will get a select option and then you will proceed towards your booking.



* Then you will have to select available seats and then boarding point and then enter your details and confirm your booking and proceed your payment.





**Conclusion:** We can Successfully created a Online Bus Booking System in Asp.net C# You can book a bus ticket using this web application, and also you can add your Bus and Bus details in this web Application.

**Folders Details in detail**

**CSS**

1. Using the **Bootstrap.min.css** file in your ASP.NET project can provide several benefits. It can help you to:

* Quickly create responsive and mobile-friendly designs for your website or web application, without having to write complex CSS code from scratch.
* Save time and effort by leveraging the pre-built CSS classes and components provided by Bootstrap, such as navigation menus, forms, and buttons.
* Ensure consistency and compatibility across different browsers and devices, as Bootstrap has been extensively tested and optimised to work well on a wide range of platforms.
* Improve the overall user experience of your website or web application, as Bootstrap provides a modern and visually appealing design aesthetic that can help to engage and retain users.

1. The **style.css** file is typically used to add custom CSS styles to an ASP.NET project. It can help you to:

* Customise the visual appearance and layout of your website or web application, such as setting the font, colour, and spacing of various elements.
* Override or extend the default styles provided by frameworks or libraries used in your project, such as Bootstrap.
* Create unique and distinct branding for your website or web application, by incorporating your own logo, graphics, and other design elements.
* Improve the user experience by making the site more visually appealing and intuitive to navigate.

**FONTS**

**JS**

**AdminMaster**

This is a code for an ASP.NET Master Page which defines the overall layout and design of a website. It includes a navigation bar at the top with links to different pages, a content area where the actual page content will be displayed, and a footer. The code also includes references to CSS files for styling and JavaScript files for adding interactivity to the page. The if statements in the code are used to show or hide certain links in the navigation bar depending on whether the user is logged in or not.

**AdminLogin**

**AdminLogin.aspx**

This is an ASP.NET code for an admin login page. It has a master page named "Admin.Master" which is specified in the MasterPageFile attribute of the Page directive.

The content of the page is divided into two content placeholders: "head" and "ContentPlaceHolder1".

The "head" placeholder contains a style element that defines some CSS rules for the layout of the login form.

The "ContentPlaceHolder1" placeholder contains the login form. The form has two input fields for the admin id and password, respectively. There are also two required field validators, one for each input field, that ensure that the user enters a value before submitting the form.

Additionally, there is a checkbox for the "Remember me" option and a submit button for submitting the login form.

The login form also has a label that can be used to display error messages to the user.

**AdminLogin.cs**

This ASP.NET code is an event handler for the "btnLogin" button click event. It first checks whether the input values for the username and password fields match the predefined values ("admin" and "admin" respectively). If the input matches, it sets the session variable "UserName" to "Admin" and redirects the user to the "BusDetailsReport.aspx" page. If the input does not match, it displays an alert message using the ScriptManager class.

**BoardingDetails**

**BoardingDetails.aspx**

This is an ASP.NET web form page that adds boarding details to a database. It uses a Master Page file "Admin.Master" for page layout and design.

The page contains a panel with a form to add new boarding details. It has two input fields - "Boarding Place Name" and "Boarding Place Expected Time Of Arrival". These fields are implemented using two TextBox controls (txtPlace and txtArrival), and each has a RequiredFieldValidator control (rfvFirstName and rfVMobileNo respectively) that ensures that they are not left empty when the form is submitted.

The form also contains a button (btnAddBoardingDetails) that when clicked, calls the btnAddBoardingDetails\_Click method in the code-behind file "BoardingDetails.aspx.cs" to add the boarding details to the database.

The ValidationSummary control (vsRegister) is used to display a summary of all validation errors that occur when the form is submitted.

There is also a div element (divMessage) that displays messages to the user after the form is submitted.

**BoardingDetails.cs**

This is a C# code-behind file for the "BoardingDetails.aspx" page of an ASP.NET web application.

The code defines a class BoardingDetails that inherits from System.Web.UI.Page. The class contains a global variable connString that holds the connection string to the database.

In the Page\_Load event handler, the code checks if the page is being loaded for the first time or being loaded in response to a postback. If it's the first time, the code checks if the Session["UserName"] is not null, if it is null then the code redirects the user to the AdminLogin.aspx page.

The AddBoardingDetails method is called when the user clicks the "Add Boarding Details" button. The method retrieves the values entered by the user for the boarding place name and expected time of arrival, and executes a stored procedure called "addBordingDetails" that inserts the boarding details into the database. The stored procedure takes four parameters: RouteID, PlaceName, PlaceTime, and BusID. The AddBoardingDetails method returns an integer value indicating the number of rows affected by the stored procedure.

The btnAddBoardingDetails\_Click event handler is called when the user clicks the "Add Boarding Details" button. The method calls the AddBoardingDetails method and checks if the return value is -1, which indicates that the boarding details have been successfully added to the database. If the value is -1, a success message is displayed using the ScriptManager.RegisterClientScriptBlock method. If the value is not -1, an error message is displayed.

**BookingReport**

**BookingReport.aspx**

This is an ASP.NET web page that displays a GridView control, which shows a report of booking details for a bus service.

The page includes a reference to a Master Page file, which provides a consistent layout for all pages in the application. The GridView control is defined with the ID "gdTicketReport" and is set to allow paging with a page size of 20.

The GridView has several BoundField columns which display the Bus Name, Passenger Name, Email ID, Contact No, Origin, Destination, Travel Date, Seat No, Fare, and Booked By information for each booking record. Additionally, there is a TemplateField column that displays a sequential number for each row using the Container.DataItemIndex property.

Overall, this page is used to display a report of booking details for bus services, allowing administrators to easily view and manage the booking records.

**BookingReport.cs**

This is a C# code-behind file for an ASP.NET web page named "BookingReport.aspx". The code defines a partial class called "BookingReport" that inherits from the System.Web.UI.Page class.

The namespace for the code is "BusBookingProject.Admin".

The code contains a global variable named "connString" which is an instance of the SqlConnection class. It is initialized with the connection string from the web.config file.

The Page\_Load event handler is used to check if the user has logged in or not. If the user has logged in, it calls the "bingBookingReport" method to bind the booking report data to a gridview. If the user has not logged in, it redirects them to the login page.

The "bingBookingReport" method is a private method that retrieves data from the database using a stored procedure named "ispGetBookingReportByAdmin". The data is then bound to a gridview named "gdTicketReport". If there is no data, the gridview is populated with a message indicating that no records were found.

**BusDetails**

**BusDetails.aspx**

This appears to be an ASP.NET web form that allows an admin user to add bus details to a database. The form contains several text boxes, drop-down lists, and validation controls to ensure that required fields are filled out and that input values are valid. The form is using a Bootstrap layout to create a responsive design, which adjusts to different screen sizes. The form also has a master page file that provides a common layout and functionality to multiple pages. The CodeBehind attribute specifies the name of the C# code file that contains the server-side code for handling events and data binding. The Inherits attribute specifies the fully qualified name of the class in the code file that the page is associated with.

**BusDetails.cs**

This is an ASP.NET code for a web page that allows the user to add or update bus details.

The Page\_Load method is called when the page is loaded, and it checks whether the page is loaded for the first time (not a postback). If so, it checks whether the user is logged in (by checking whether the Session["UserName"] is null). If the user is not logged in, the page redirects to the login page. If the user is logged in, the method checks whether there is a BusID value in the query string. If there is, the FillData method is called to populate the page with the existing bus details and the Save button text is changed to "Update". If there is no BusID value, the Save button text is set to "Insert".

The FillData method retrieves the bus details from the database based on the BusID parameter and populates the textboxes on the page with the retrieved values.

The AddBusDetails and UpdateData methods are used to add new bus details and update existing ones, respectively. Both methods use stored procedures to perform the database operations.

The btnSave\_Click method is called when the user clicks the "Save" button. If the button text is "Insert", the AddBusDetails method is called to add the new bus details. If the button text is "Update", the UpdateData method is called to update the existing bus details. After the operation is completed, the method displays a success or error message using the ScriptManager.RegisterClientScriptBlock method. If the operation was successful, the method also clears the textboxes on the page.

**BusDetailsReport**

**BusDetailsReport.aspx**

This ASP.NET code creates a web page that displays a table (GridView) of bus details. The table has several columns including Bus ID, Bus No, Bus Name, Bus Type, Seat Columns, Seat Row, Origin, Destination, and Action. The data in each column is populated by binding to data from a data source (which is not shown in the code).

The code also includes some additional functionality such as pagination, row formatting (using the OnRowDataBound event), and the ability to add hyperlinks to update the bus details or add a bus schedule. The HiddenField controls are used to store values that can be accessed by other parts of the code. The layout of the web page is determined by a master page (Admin.Master) which is specified in the Page directive.

**BusDetailsReport.cs**

This is a C# code for an ASP.NET web page named "BusDetailsReport". The code-behind file for this web page is "BusDetailsReport.aspx.cs".

The code declares a SqlConnection object named "connString" with the connection string retrieved from the web.config file.

The Page\_Load method checks whether the web page is being loaded for the first time or it is a postback request. If it's a postback request, nothing happens. If it's the first time the page is being loaded, it checks whether the user is logged in. If the user is logged in, the method "bindBusDetailsReport()" is called to populate the GridView named "gdBusDetails" with data. If the user is not logged in, the page is redirected to the login page.

The bindBusDetailsReport() method retrieves data from the database using a stored procedure named "ispGetBusDetailsForUpdation". The data is stored in a DataSet object named "dsGetData". If the DataSet contains data, it is bound to the GridView control. If the DataSet is empty, a message is displayed in the GridView control indicating that there are no records to display.

The gdBusDetails\_RowDataBound event is fired when a data row is bound to the GridView control. In this event, the method finds the HyperLink control named "hlinkUpdate", the HiddenField control named "hdnBusID", the HyperLink control named "hlinkAddSchedule", and the HiddenField control named "hdnRouteID" for each row of data. It sets the NavigateUrl property of the "hlinkUpdate" and "hlinkSchedule" controls with the BusID and RouteID values retrieved from the corresponding HiddenFields.

Overall, the code populates a GridView with data from a database and provides links to update and add schedules for each row of data.

**BusScheduleDetails**

**BusScheduleDetails.aspx**

This ASP.NET code defines a web page for adding bus schedules to a database. The page includes a form with several text fields for inputting details such as the available date, fare, total travel time, arrival time, and departure time. The form also includes validation controls to ensure that the required fields are filled in before the user can submit the form.

The code behind the page is handled by a C# class called "BusScheduleDetails" which inherits from the base class "Page". The class contains an event handler method called "btnSaveSchedule\_Click" that is triggered when the user clicks the "Save Bus Schedule" button. This method retrieves the values from the form fields, validates them, and then inserts them into the database. The page also includes a master page called "Admin.Master" which provides a common layout and styling for all pages in the admin section of the website.

**BusScheduleDetails.cs**

This is an ASP.NET code-behind file for a web page called "BusScheduleDetails.aspx".

The code defines a class named "BusScheduleDetails" that inherits from "System.Web.UI.Page". It contains a global variable "connString" that is a SqlConnection object that establishes a connection to a SQL Server database using a connection string from the web.config file.

The "Page\_Load" method checks if the page is being loaded for the first time or if it's a postback. If it's a postback, it does nothing. If it's the first time, it checks if the "UserName" session variable is not null. If it's not null, it does nothing. If it's null, it redirects the user to the "AdminLogin.aspx" page.

The "addBusScheduleData" method defines a SqlCommand object that calls a stored procedure named "ispAddBusSchedule" and passes some parameters to it. The method executes the stored procedure using the SqlConnection object and returns the number of rows affected by the execution.

The "btnSaveSchedule\_Click" method is called when the user clicks the "Save Schedule" button on the web page. It calls the "addBusScheduleData" method and checks the returned result. If the result is -1, it means the stored procedure was executed successfully and it displays an alert message using ScriptManager. If the result is not -1, it means an error occurred and it displays an error message using ScriptManager. It also clears some text boxes on the web page.

**RouteDetails**

**RouteDetails.aspx**

This is an ASP.NET code for displaying route details in a GridView control.

The code defines a GridView control named "gdRouteDetails" with several columns: "Sr.No", "Route ID", "Origin", "Destination", and "Action". The GridView is set to allow paging and display 20 items per page.

Each row of the GridView is populated with data from a data source, which is not shown in this code snippet.

The "Action" column contains a HyperLink control labeled "Add Boarding Points". When this link is clicked, the page will redirect to another page where boarding points can be added for the corresponding route.

The code also includes two hidden fields, "hdnRouteID" and "hdnBusID", which store the values of the corresponding Route ID and Bus ID for each row. These hidden fields can be accessed in the code-behind file for further processing.

Overall, this code provides a user interface for displaying and managing route details in a web application.

**RouteDetails.cs**

This is a code-behind file written in C# for an ASP.NET web page named "RouteDetails.aspx". This web page is used by the administrator of an online bus booking system to view the details of bus routes and add boarding points to them.

The first section of the code defines a global variable "connString", which is an instance of the SqlConnection class used to connect to the database using the connection string specified in the web.config file.

The Page\_Load event handler is executed when the web page is loaded. It first checks if the page is being loaded for the first time (i.e., it is not a postback) and if the user has logged in as an administrator by checking the "UserName" session variable. If the user is authenticated, it calls the bindBoardingDetails() method to retrieve and display the bus route details in a GridView control. If the user is not authenticated, it redirects them to the login page.

The bindBoardingDetails() method uses a stored procedure named "ispGetRouteDetails" to fetch the bus route details from the database and bind them to the GridView control. It first creates a new DataSet object "dsGetData" and a new SqlCommand object "sqlCmd". It then opens the database connection and sets the CommandType and CommandText properties of the SqlCommand object to execute the stored procedure. It then creates a new SqlDataAdapter object "sda" and calls the Fill() method to populate the DataSet with the result set returned by the stored procedure. If the DataSet contains rows, it sets the DataSource property of the GridView control to the first table in the DataSet and binds it to the GridView. If the DataSet is empty, it sets the DataSource property to null and sets the EmptyDataText property of the GridView to display a message indicating that no records were found.

The gdRouteDetails\_RowDataBound event handler is executed for each row of the GridView control during databinding. It checks if the row type is DataRow and finds the HyperLink, HiddenField controls from the row and sets the NavigateUrl property of the HyperLink control to redirect the user to the "BoardingDetails.aspx" page with the BusID and RouteID querystring parameters for the selected bus route, so the administrator can add boarding points to that route.

**AboutUs**

**AboutUs.aspx**

This is an ASP.NET web page named "AboutUs.aspx". It is using a Master Page file named "BusBookingMaster.Master" for layout and styling. The page contains a content placeholder with the ID "ContentPlaceHolder1", where the actual content of the page is placed.

Inside the content placeholder, there is a container with a class of "container" and a margin-top of 6%. Inside this container, there is a panel with a default styling. The panel contains a heading with a title "About Us" and a body with a paragraph of text explaining the features and benefits of the online bus reservation system provided by the website.

The paragraph highlights the user-friendly interface, wide selection of bus options, exceptional customer service, 24/7 availability, technological advancement, and dedication to making the user experience better. The content ends with a message of gratitude for using the online bus reservation system.

**BookingReport  
BookingReport.aspx**

This is an ASP.NET code file for a web page called "BookingReport.aspx". It is used to display a report of bus ticket bookings made through a bus booking system.

The code defines a GridView control called "gdTicketReport" which displays the booking information including PNR number, total booked tickets, and booking amount. It also includes a link button called "Download Ticket" for each booking, which when clicked, will display details of the booking in a table format.

The table is defined using the HTML "table" tag and contains information such as PNR number, total tickets, total amount, and passenger details. The passenger details are displayed using another GridView control called "gdPaxDetails" which displays the passenger's first name, last name, contact number, seat number, travel date, origin, destination, and boarding place.

The code also includes a hidden field called "hdnPNRNo" which is used to store the PNR number of the booking for which the details are being displayed. The details of the booking are displayed in a separate div element called "ticket" which is initially hidden using the "visible" attribute set to false.

Finally, the code includes an event handler method called "gdTicketReport\_RowCommand" which is triggered when the "Download Ticket" link button is clicked. This method retrieves the PNR number of the booking from the hidden field and displays the booking details in the "ticket" div element by setting its "visible" attribute to true.

**BookingReport.cs**

This is a code-behind file for an ASP.NET web page called BookingReport. The page displays a list of bookings made by the logged-in user, retrieved from a SQL Server database, and provides a button to download a PDF ticket for each booking.

The code includes the following main components:

Global Variable: A SqlConnection object is declared to connect to the database.

Page\_Load method: The method is called when the page loads for the first time. It first checks if the page is being loaded for the first time (i.e., not a postback). If it is not a postback, it checks if the user is logged in. If the user is logged in, it calls the bindPnrDetails method to retrieve the user's booking details and bind them to a GridView control called gdTicketReport. If there is a "Msg" parameter in the query string (i.e., a message to display to the user), it displays an alert box with the message. Finally, it checks if the user is not logged in and redirects them to the login page.

bindPnrDetails method: This method retrieves the user's booking details from the database using a stored procedure and fills a DataSet object. It then binds the first table of the DataSet to the gdTicketReport GridView control.

VerifyRenderingInServerForm method: This is an override method required for exporting the GridView to PDF.

printTicket method: This method is called when the user clicks the "Download Ticket" button in the gdTicketReport GridView control. It retrieves the details of the booking corresponding to the selected row and fills a GridView control called gdPaxDetails with the passenger details. It then generates a PDF ticket using the iTextSharp library and sends it to the user's browser.

gdTicketReport\_RowCommand method: This method is called when the user clicks a button in the gdTicketReport GridView control. It checks if the button is the "Download Ticket" button and calls the printTicket method to generate and download the ticket.

**BusBookingMaster**

**BusBookingMaster.aspx**

This is an ASP.NET Master Page with a code-behind file called "BusBookingMaster.master.cs". Master Pages in ASP.NET provide a way to create a consistent look and feel across multiple pages in a web application. The Master Page defines the common layout and design elements such as the header, footer, and navigation menus.

The code snippet defines the layout for the Master Page, which includes a navigation menu with links to the home page, about us page, search buses page, and booking report page (if the user is logged in). It also includes a registration and login link for users who are not logged in and a logout button for users who are logged in.

The code uses a Bootstrap CSS framework to style the layout and jQuery JavaScript library to provide interactivity to the navigation menu. The code also includes a footer with copyright information.

In the code-behind file "BusBookingMaster.master.cs", there may be additional logic and code to handle user authentication and other functionalities specific to the web application.

**BusBookingMaster.cs**

This is the code-behind file for an ASP.NET Master Page named "BusBookingMaster".

The class "BusBookingMaster" inherits from the System.Web.UI.MasterPage class, which means that it defines the behavior of the master page.

The "Page\_Load" event handler method is called every time the page loads, and it checks whether the "FName" key exists in the session state. If it does, the method sets the text of the "lblName" label to the value of the "FName" key.

The "btnLogout\_Click" event handler method is called when the "btnLogout" button is clicked, and it ends the current session by calling the "Session.Abandon" method, and then redirects the user to the "Login.aspx" page.

**BusBookingSearchDetails**

**BusBookingSearchDetails.aspx**

This is an ASP.NET web page that displays bus details retrieved from a data source in a GridView control. The page is based on a master page called "BusBookingMaster" and contains two content placeholders - "head" and "ContentPlaceHolder1".

The GridView control (gvBusDetails) is used to display the bus details in a tabular format. The GridView is populated with data using the Eval method to bind the data to the appropriate fields. The TemplateFields define the columns of the GridView and are used to display the Bus Name, Departure Time, Arrival Time, Available Seats, Fare, and a link to select the bus. The select bus link is actually a HyperLink control (hplnkSelect) that is used to trigger a server-side event when clicked.

The GridView also has an OnRowDataBound event (gvBusDetails\_RowDataBound) which is used to bind the hidden fields (hdnBusID, hdnSeatRow, hdnSeatColumn) to their respective values.

Finally, the page also contains a hyperlink (hlinkSearch) which allows the user to search for buses again.

**BusBookingSearchDetails.cs**

This is a C# code-behind file for an ASP.NET web page named "BusBookingSearchDetails.aspx".

The code defines a class named "BusBookingSearchDetails" that inherits from "System.Web.UI.Page". It contains the following methods:

Page\_Load: This method is executed when the page loads for the first time. It checks whether the page is being loaded for the first time or as a result of a postback. If it is being loaded for the first time, it calls the "bindSearchDetails" method.

bindSearchDetails: This method retrieves data from the database using a stored procedure named "ispGetAvailableBusDetails". The method passes parameters to the stored procedure based on the query string parameters "Origin", "Destination", and "TravelDate". It then populates a GridView control named "gvBusDetails" with the data retrieved from the database. If no data is returned, it displays an error message.

gvBusDetails\_RowDataBound: This method is executed for each row of the GridView control. It sets the NavigateUrl property of a HyperLink control named "hlnkSelect" based on the values of the row's HiddenField controls and query string parameters.

The code also declares a global variable named "connString", which is an instance of the SqlConnection class. It is used to connect to the database using a connection string specified in the web.config file.

Overall, this code retrieves data from a database based on user input, displays the data in a GridView control, and allows the user to select a row and navigate to a page where they can select a seat and purchase a bus ticket.

**Default**

**Default.aspx**

This code represents the HTML and ASP.NET markup for the content of the Default.aspx page of a web application for an online bus booking system. The code specifies the page's title, language, master page, and code-behind file.

The page contains two content placeholders, "head" and "ContentPlaceHolder1", which will be filled with content from the master page and this page, respectively.

The content of this page includes an image banner and two panels that provide information about the online bus booking system. The first panel provides an introduction to the system, while the second panel lists the benefits of using the system, such as fast and easy booking, secure payment options, and online cancellation and support.

The code also includes CSS classes to control the layout and styling of the page.

**Home**

**Home.aspx**

This is an ASP.NET page that contains a form to search for available buses. It uses jQuery UI datepicker to allow users to select the travel date and return date. The page has two dropdown lists for selecting the origin and destination cities.

The page also has a panel with a title "Search For Available Buses" and a "divMessage" control for displaying messages. The form has a submit button that triggers an event in the code-behind file to search for available buses based on the user input. The form also has input fields for selecting the travel date and return date, and a dropdown list for selecting the number of passengers.

**Home.cs**

This is an ASP.NET code for a web page named "Home.aspx" which allows users to search for bus booking details by selecting their origin and destination city and providing a travel date.

The code defines a web form class named "Home" that inherits from the "System.Web.UI.Page" class. The web form contains two dropdown lists named "ddlOrigin" and "ddlDestination" for selecting the origin and destination cities, respectively. It also contains a textbox named "txtDate" for entering the travel date and a button named "btnSearch" for initiating the search.

The code initializes a global variable named "connString" which holds the connection string to the database used to retrieve the list of cities. The Page\_Load event is used to bind the origin and destination cities to their respective dropdown lists on the first load of the page. The "bindOriginCity" and "bindDestinationCity" methods are called to retrieve the list of cities from the database and bind them to their respective dropdown lists.

The "getCity" method is used to retrieve the list of cities from the database by executing a stored procedure named "ispGetCity". The method returns a dataset containing the list of cities.

The "btnSearch\_Click" event is triggered when the user clicks the search button. The event retrieves the selected origin and destination cities and the travel date from the web form controls and redirects the user to a page named "BusBookingSearchDetails.aspx" with the search parameters passed as query string parameters. The search parameters are used to retrieve the bus booking details from the database and display them on the search results page.

**Login**

**Login.aspx**

This ASP.NET code defines a login page for a bus booking project.

The page uses a master page (defined in the MasterPageFile attribute) for consistent layout and styling across the project.

The code defines a centered form (using CSS) that contains a panel for the login form. The form has two input fields: one for the user's mobile number and another for their password. There is also a checkbox for "Remember Me" and a button to submit the form.

The page uses ASP.NET validation controls (RequiredFieldValidator and ValidationSummary) to ensure that the user enters a value for both the mobile number and password fields.

If the user clicks the "Log in" button, the code executes the btnLogin\_Click method defined in the Login.aspx.cs code-behind file. This method likely handles the login logic, such as checking the user's credentials against a database and redirecting the user to a different page if the login is successful.

If the user clicks the "Register as a new user" link, the page redirects to the UserRegistration.aspx page, where the user can create a new account.

**Login.cs**

This is a C# code for an ASP.NET web application that handles user login functionality.

The code defines a partial class named "Login" that inherits from "System.Web.UI.Page" class, which is the base class for ASP.NET web pages.

The class includes a private method "getUserData()" that uses ADO.NET objects like SqlConnection, SqlCommand, SqlDataAdapter, and DataSet to retrieve user data from a SQL Server database. The SQL query is executed as a stored procedure named "ispUserLogin" that takes two parameters: MobileNo and Password. The user data is returned in a DataSet object.

The protected method "btnLogin\_Click" is called when the user clicks the login button on the web form. The method calls the "getUserData()" method to retrieve user data and checks whether the DataSet contains any rows. If the DataSet contains rows, it sets some session variables such as "UserID", "FName", and "MobileNo" to store user data, and redirects the user to another web form called "PassengerDetailsInfo.aspx" if a query string named "BusID" exists; otherwise, it redirects to the "Home.aspx" web form. If the DataSet does not contain any rows, it displays an error message using the ScriptManager.RegisterClientScriptBlock method.

Overall, this code provides a basic user authentication mechanism that retrieves user data from a SQL Server database and uses session variables to store user data across web pages.

**PassengerDetailsinfo.aspx**

This is an ASP.NET web page for a bus booking system. The page allows users to enter passenger details such as first name, last name, email, contact number, and city, for each passenger. The passenger details are displayed in a grid view, and users can add new passenger details by clicking the "Add New Passenger Details" button. Once the passenger details are entered, users can confirm the booking by clicking the "Confirm Booking" button.

The page also has a panel for payment details, which is initially hidden. When the user clicks the "Confirm Booking" button, the payment details panel is displayed, and the user can select a card type and bank for payment.

The ASP.NET controls used in the page include GridView, TextBox, Button, Label, DropDownList, RequiredFieldValidator, and ValidationSummary.

The code-behind file for the page contains event handlers for the ButtonAdd\_Click and btnConfirmBooking\_Click events, which add new rows to the grid view and display the payment details panel, respectively.

**SeatDetails.cs**

This is a C# code for an ASP.NET web application that provides seat booking functionality for bus tickets. The code is defined within a partial class named "SeatDetails" that extends the "System.Web.UI.Page" class.

The code retrieves query string parameters passed through the URL to display information about the bus, such as the origin, destination, travel date, and fare. It uses these parameters to query the database to get the available boarding points, bind them to a dropdown list, and populate the list with the values.

The code also calls a stored procedure to get the list of booked seat numbers for the given bus and travel date, and appends them to a string that is later used to mark them as unavailable for booking.

Finally, when the user clicks the "btnPayment" button, the code retrieves the selected seat number and fare using hidden fields, and redirects the user to the passenger details page with the necessary parameters to complete the booking process. If the user is not logged in, they are redirected to the login page with the same parameters.

**UserRegistration**

**UserRegistration.aspx**

This code is a web form written in C# that creates a user registration form. The form contains fields such as first name, last name, mobile number, email ID, address, city, and pin code. The form also includes validation controls for each field to ensure that the required fields are not left blank.

**UserRegistration.cs**

This is an ASP.NET code for user registration page of a bus booking project. The code uses C# programming language and SQL Server database to store user information.

The code defines a partial class called UserRegistration which is a subclass of the Page class. The Page\_Load event is defined but not implemented, which means it doesn't do anything when the page is loaded.

The Regitration() method is defined to insert user registration details into the database. It creates an instance of the SqlCommand class, sets its CommandType property to StoredProcedure, and adds the user's details as parameters to the SqlCommand object. It then calls a stored procedure named "ispUserRegistration" which inserts the data into the database.

The btnSave\_Click event is defined to handle the click event of the "Save" button on the user registration form. It calls the Regitration() method to insert the user's details into the database, and based on the return value of the method, it displays a message to the user using the ScriptManager.RegisterClientScriptBlock method.

If the registration is successful, a message is displayed saying "User Registration has been done successfully". If the mobile number already exists in the database, a message is displayed saying "Mobile No already exist please try with another mobile no". If an error occurs, a message is displayed saying "Error Occur Please contact your system Administrator". The textboxes on the form are cleared after the registration is complete.

**Web.config**

This is an XML configuration file for an ASP.NET web application. It defines settings related to compilation, runtime, connection strings, and application settings.

Within the <system.web> section, it sets the debug attribute to true, which enables debugging information to be shown when an error occurs. It also sets the targetFramework attribute to 4.8, which specifies the .NET Framework version the application is targeting. The <httpRuntime> element sets the targetFramework attribute to 4.5.

The <customErrors> element is used to disable custom error messages, showing the detailed error information instead.

The <connectionStrings> section defines a connection string named OnlineBusBookingConnectionString that specifies the server and database to be used for data storage. The providerName attribute specifies the ADO.NET provider for SQL Server.

The <appSettings> section defines an application setting named ValidationSettings:UnobtrusiveValidationMode, which disables unobtrusive validation mode for form validation.