

תיק פרויקט

בתכנון ותכנות מערכות במסלול שירותי רשת אינטרנט

שנת לימודים תשפ"א

שם התלמיד: אורי דאבוש

בית ספר: מקיף עירוני א' אשקלון

תעודת זהות: 212945760

שם המורה: רפאל בן קימון

תוכן עניינים

4	רקע
4	קהל היעד
5	תיאור המערכת
6	תיאור דרישות המערכת
6	זיהוי והגדרת האתגרים המדעיים והטכנולוגיים
7	הצגת החלופות האפשריות להתמודדות עם האתגרים
7	הנמקת הבחירה ההולמת
7	תיאור המודולים והקשר ביניהם
8	תיאור האלגוריתמים המרכזיים של המערכת
8	כניסה לאתר
8	Login
10	Logout
11	Sign Up
14	מנהל המערכת
14	Admin Data
16	Block User
18	Create User
21	צ'אט וממשק IExecutioner
21	צ'אט
23	ממשק IExecutable
24	Evaluate Executioner
25	Help Executioner
26	Joke Executioner
28	Time Executioner
29	Translate Executioner

31.....	המחלקה CommandExecutioner וביצוע הפקודות
34.....	שימוש ב-cache ושמירת תוצאות
35.....	מבנה מסד הנתונים
36.....	מבנה התיקיות
37.....	מדריך למשתמש באתר
40.....	מדריך למנהל האתר

רקע

באינטרנט ישנם כלים רבים בהם ניתן להשתמש בכדי להקל על חיי היום יום שלנו – החל מפתרון בעיות באינטרנט כמו פתרון תרגילים, תרגום טקסט, חיפוש מידע מסוים באינטרנט ודוגמאות נוספות רבות וכלה בכניסה לאתרים שונים לצורכי הנאה – משחקים, בדיחות, רשתות חברתיות וכו'.

כמות האתרים שקיימים באינטרנט רבה, ורק חלקם מצליחים להגיע לידי המשתמשים. כתוצאה מכך נוצר מצב בו לא כל מפתחי האתרים והתוכנות זוכים שהאתרים והתוכנות שלהם יהיו בשימוש, ולא כל המשתמשים זוכים לחוויות המשתמש שהם קיוו לה, כיוון שלא הגיעו לאתר המתאים ביותר לצרכיהם.

ממצב זה עולה הצורך בריכוז של כלים שונים באינטרנט במקום אחד, כך יתאפשר למפתחים השונים להפיץ את פיתוחיהם כראוי ובנוסף למשתמשים השונים להגיע לכלי שחיפשו.

המערכת שפיתחתי עונה על מספר צרכים:

לטובת מפתחי האתרים והתוכנות – המערכת שפיתחתי מהווה מקום בו יוכלו להשתמש בכדי להפיץ את האתר או התוכנה שלה ובכדי שיגיעו ליותר משתמשים. הכנסת האתר או התוכנה שלהם למערכת שלי תגרום להרבה יותר משתמשים להגיע אליה.

לטובת המשתמשים – המערכת שפיתחתי מהווה ריכוז של כלים שונים וכך יוכלו לחפש ולמצוא את הכלי המתאים ביותר בשבילם.

קהל היעד

מבחינת מפתחי תוכנה או אתרים, המערכת פונה לכאלו שרוצים להפיץ את האתר או התוכנה שלהם ולגרום לה להגיע ליותר משתמשים.

מבחינת משתמשים, המערכת פונה למשתמשים שרוצים לחפש את הכלי המתאים ביותר עבורם, ובנוסף למשתמשים שרוצים להשתמש בכמה כלים בנוחות מבלי להסתבך.

תיאור המערכת

המערכת מורכבת משני חלקים עיקריים – צד המשתמש וצד המנהל (admin).

המשתמש הלא מחובר לא יכול לעשות הרבה – הוא יכול לצפות בדף הבית ובדף האודות, ליצור משתמש או להתחבר למשתמש קיים.

המשתמש המחובר לעומת זאת יכול להשתמש במערכת עצמה – הוא יכול לדבר בצ'אט מתעדכן עם בוט שקיימים בו כלים שונים.

מנהל האתר (משתמש מיוחד שמתחבר על ידי שם משתמש וסיסמא admin) יכול לראות את הנתונים השונים שנאגרו במערכת – נתונים על המשתמשים הקיימים (שמות המשתמשים והסיסמאות) ונתונים על ה-cache של הפקודות שקיים עד כה (פירוט בהמשך). בנוסף מנהל האתר יכול לחסום משתמש (כלומר למחוק את שם המשתמש והסיסמא שלו ממסד הנתונים) על ידי דף פשוט בו הוא מכניס את שם המשתמש (יפורט בהמשך) וליצור משתמש חדש.

תיאור דרישות המערכת

- שימוש בטכנולוגיית ASP.NET בשפת C#.
- שימוש במסד נתונים SQL.
- עיצוב UX/UI ברמה גבוה באמצעות קבצי CSS.
- הפרדה בין מודול מנהל למשתמש.
- חבילת NuGet (NuGet package) ששמה Z.Expressions.Eval בגרסה 4.0.42.

זיהוי והגדרת האתגרים המדעיים והטכנולוגיים

לפני שמימשתי את התוכנית כבר ידעתי C# וכבר פיתחתי כמה אפליקציות בממשקים דומים ל-ASP.NET (כמו WPF), לכן היה לי קל יותר להתחיל בפרויקט ולא הייתי צריך ללמוד את הכל מהתחלה. עם זאת, היה עליי ללמוד ולהכיר כלים שונים, חדשים ומגוונים ב-ASP.NET.

את המערכת בניתי לפי החלקים השונים:

ראשית, בניתי את ה-"שלד" של המערכת – דפי ההתחברות, הרשמה, הדפים הראשיים (דף ראשי, אודות וכו').

לאחר מכן בניתי את ה-code behind של דפים אלו – לוגיקת ההתחברות וההרשמה.

אחרי שבניתי את ה-"שלד" של המערכת הוספתי את הדפים שקשורים למנהל המערכת – דפי הסטטיסטיקות וחסימת המשתמש.

אחר כך עברתי למימוש המערכת עצמה – בניתי את דף ה-Chat ומימשתי את הלוגיקה הבסיסית שלו (אותה אפרט בהמשך).

אחרי שסיימתי לממש את הלוגיקה הבסיסית שלו, עברתי לממש את הלוגיקה המורכבת שלו ולהוסיף מחלקות שונות (אפרט בהמשך).

במערכת שלי יש שני מסדי נתונים – מסד נתונים אחד עבור המשתמשים הקיימים במערכת (שם משתמש וסיסמא), ומסד נתונים שני המשמש כ-cache עבור הפקודות השונות (פירוט בהמשך).

הצגת החלופות האפשריות להתמודדות עם האתגרים

על מנת לתת פתרון טכנולוגי למערכת שפתחנו בדקנו את החלופות הבאות:

טכנולוגיה	יתרון	חסרון
ASP.NET Core	מבוסס מבנה אחיד ומבנה קל ונוח למימוש	מסורבל בכל הקשור לביצוע שינויים במבנה
ASP.NET MVC	מקל על ביצוע הפרויקט מבחינת הפרדת הרשויות	דורש הבנה ברמה גבוהה על מנת לממש
ASP.NET Framework	פרויקט מובנה ברמה בסיסית, מאפשר שליטה טובה יותר של המתכנת	דורש כתיבת קוד רבה
HTML	בסיסי	אינו מתאים לדרישות הפרויקט
Bootstrap	מעוצב, קל לשימוש	אינו מתאים לדרישות הפרויקט

הנמקת הבחירה ההולמת

בחרתי לעשות את הפרויקט בטכנולוגיית ASP.NET Framework בשפת C# מכיוון שבטכנולוגיה זו ניתן לאפשר למשתמש חוויית גלישה טובה יותר ומתקדמת יותר ממה שהוא יכול לקבל במערכות האחרות.

תיאור המודולים והקשר ביניהם

במערכת ישנם שני מודולים עיקריים - מודול המנהל ומודול המשתמש.

הקשר בין המודולים מתבטא בכך שלמנהל יש דף הנפרד משאר המשתמשים שבו הוא יכול לצפות בנתונים, דבר שלא ניתן לעשות באמצעות משתמש רגיל במערכת. בנוסף למנהל יש אפשרות של ניהול משתמשים בה הוא יכול לחסום משתמשים (כלומר למחוק את שם המשתמש והסיסמא ממסד הנתונים).

תיאור האלגוריתמים המרכזיים של המערכת

האלגוריתמים המרכזיים במערכת נבנו בצורה כזו שיתנו את המענה ההולם למקום ולפעולה שהמשתמש אמור לבצע באותו דף באתר שבו הוא נמצא.

כניסה לאתר

Login

המשתמש יכניס את שם המשתמש והסיסמא שלו. כאשר המשתמש מסיים ולוחץ על הכפתור "login" צד השרת בודק במסד הנתונים האם המשתמש קיים והאם זו הסיסמא שלו. אם המשתמש אכן קיים, צד השרת מעדכנת את Session["username"] להיות שם המשתמש שהוכנס ומפנה את המשתמש לדף הבית. אם המשתמש אינו קיים מוצגת הודעת שגיאה (והוא נשאר בדף ההתחברות). לאחר ההתחברות המשתמש התפריט (navbar) שיוצג עבורו ישתנה ויתאים למשתמש המחובר.

דף ה-Login (Login.aspx):

```
<%@ Page Title="Login" Language="C#" MasterPageFile="~/Site.Master"
AutoEventWireup="true" CodeBehind="Login.aspx.cs" Inherits="AI_Project.Login" %>

<asp:Content ID="BodyContent" ContentPlaceHolderID="MainContent" runat="server">
    <h1><strong>Welcome.</strong> Please login.</h1>

    <fieldset style="margin: 5px 5px 5px 5px">
        <asp:TextBox ID="tbUsername" CssClass="tb" runat="server" placeholder="Username"
        />
        <asp:RequiredFieldValidator runat="server" ControlToValidate="tbUsername"
        ErrorMessage="Field is required." />
        <br />
        <br />
        <asp:TextBox ID="tbPassword" CssClass="tb" runat="server" placeholder="Password"
        TextMode="Password" />
        <asp:RequiredFieldValidator runat="server" ControlToValidate="tbPassword"
        ErrorMessage="Field is required." />
        <br />
        <br />
        <asp:Button ID="sendButton" runat="server" Text="Login"
        OnClick="sendButton_Click" />
        <asp:Button ID="clearButton" runat="server" Text="Clear"
        OnClick="clearButton_Click" />
        <br />
    </fieldset>

</asp:Content>
```


:(Login.aspx.cs) code behind-7

```
using System;
using System.Web.UI;
using System.Data.SqlClient;

namespace AI_Project
{
    public partial class Login : System.Web.UI.Page
    {
        private SqlConnection con;
        protected void Page_Load(object sender, EventArgs e)
        {
            string conStr = String.Format("Data
Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=\"{0}\\DATABASE.mdf\";Integrated
Security=True", Server.MapPath("~/App_Data"));
            con = new SqlConnection(conStr);
        }

        protected void clearButton_Click(object sender, EventArgs e)
        {
            tbPassword.Text = "";
            tbUsername.Text = "";
        }

        protected void sendButton_Click(object sender, EventArgs e)
        {
            con.Open();
            string query = "select * from Users where username=@username and
password=@password";
            SqlCommand cmd = new SqlCommand(query, con);
            string username = tbUsername.Text.Trim(), password = tbPassword.Text.Trim();
            cmd.Parameters.AddWithValue("@username", username);
            cmd.Parameters.AddWithValue("@password", password);
            var reader = cmd.ExecuteReader();
            if (reader.HasRows)
            {
                Session["username"] = username;
                if (username == "admin" && password == "admin")
                {
                    Session["admin"] = true;
                }
                Response.Redirect("Default.aspx");
            }
            else
            {
                ClientScript.RegisterStartupScript(Page.GetType(), "validation", "<script
language='javascript'>alert('Invalid Username and Password')</script>");
            }
            con.Close();
        }
    }
}
```

Logout

בפני המשתמש המחובר יוצג כפתור "logout" (ב-navbar). כאשר המשתמש ילחץ עליו, צד השרת יפעיל את המתודה Session.Abandon שמוחקת את כל הנתונים ב-Session, בין היתר את Session["username"]. לאחר מכן המשתמש יופנה לדף הבית.

לאחר התנתקות המשתמש התפריט ישתנה בחזרה לתפריט של המשתמש הלא מחובר.

ההתנתקות נעשית בעזרת form שנקרא Logout (זאת בכדי להתאים לממשק של ה-navbar).

דף ה-Logout (Logout.aspx) הוא דף ריק, כיוון שהדבר היחיד שהוא עושה זה מרוקן את Session ומפנה לדף הבית.

ה-code behind (Logout.aspx.cs):

```
using System;

namespace AI_Project
{
    public partial class Logout : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            Session.Abandon();
            Response.Redirect("Default.aspx");
        }
    }
}
```

Sign Up

המשתמש יכניס את שם המשתמש והסיסמא שהוא רוצה למשתמש החדש שלו. צד השרת בודק האם המשתמש קיים. אם כן, הוא מציג הודעת שגיאה. אם לא, הוא יוצר את המשתמש על ידי עדכון מסד הנתונים עם הפרטים שהוכנסו.

בנוסף צד השרת מאמת את שם המשתמש והסיסמא. אימות הפרטים נעשה על ידי RequiredValidator שבודק שהשדות לא ריקים ועל ידי CustomValidator אותו אני יצרתי, שבודק האם שם המשתמש מכיל בין 3 ל-15 תווים שהם רק אותיות, מספרים, '-' ו-'_' והאם הסיסמא מכילה יותר מ-8 תווים שהם רק אותיות ומספרים.

דף ה-Sign Up (SignUp.aspx):

```
<%@ Page Language="C#" MasterPageFile="~/Site.Master" AutoEventWireup="true"
CodeBehind="SignUp.aspx.cs" Inherits="AI_Project.SignUp" %>

<asp:Content ID="BodyContent" ContentPlaceHolderID="MainContent" runat="server">

    <h1><strong>Welcome.</strong> Please sign up.</h1>

    <fieldset style="margin: 5px 5px 5px 5px">

        <asp:TextBox ID="tbUsername" CssClass="tb" runat="server" placeholder="Username"
        />
        <asp:RequiredFieldValidator runat="server" ControlToValidate="tbUsername"
        ErrorMessage="Field is required." />
        <asp:CustomValidator runat="server" ControlToValidate="tbUsername"
        ID="UsernameValidator" Display="Static"
        OnServerValidate="UsernameValidator_ServerValidate" />
        <br />
        <br />
        <asp:TextBox ID="tbPassword" CssClass="tb" runat="server" placeholder="Password"
        TextMode="Password" />
        <asp:RequiredFieldValidator runat="server" ControlToValidate="tbPassword"
        ErrorMessage="Field is required." />
        <asp:CustomValidator runat="server" ControlToValidate="tbPassword"
        ID="PasswordValidator" Display="Static"
        OnServerValidate="PasswordValidator_ServerValidate" />
        <br />
        <br />
        <asp:Button ID="signupButton" runat="server" Text="Sign Up"
        OnClick="signupButton_Click" />
        <asp:Button ID="clearButton" runat="server" Text="Clear"
        OnClick="clearButton_Click" />
        <br />

    </fieldset>

</asp:Content>
```

:(SignUp.aspx.cs) code behind-n

```

using System;
using System.Data.SqlClient;
using System.Text.RegularExpressions;

namespace AI_Project
{
    public partial class SignUp : System.Web.UI.Page
    {
        private SqlConnection con;
        protected void Page_Load(object sender, EventArgs e)
        {
            string conStr = String.Format("Data
Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=\"{0}\\DATABASE.mdf\";Integrated
Security=True", Server.MapPath("~/App_Data"));
            con = new SqlConnection(conStr);
        }

        protected void clearButton_Click(object sender, EventArgs e)
        {
            tbPassword.Text = "";
            tbUsername.Text = "";
        }

        private bool IsUserExist(string username)
        {
            con.Open();
            string query = "select * from Users where username=@username";
            SqlCommand cmd = new SqlCommand(query, con);
            cmd.Parameters.AddWithValue("@username", username);
            var reader = cmd.ExecuteReader();
            bool res = reader.Read();
            con.Close();
            return res;
        }

        protected void signupButton_Click(object sender, EventArgs e)
        {
            if (!Page.IsValid) return;
            string username = tbUsername.Text.Trim(), password = tbPassword.Text.Trim();
            if (IsUserExist(username))
            {
                ClientScript.RegisterStartupScript(Page.GetType(), "validation", "<script
language='javascript'>alert('Username already exists.')</script>");
                return;
            }
            con.Open();
            string query = String.Format("insert into Users(username, password)
values('{0}','{1}')" , username, password);
            SqlCommand cmd = new SqlCommand(query, con);
            cmd.ExecuteNonQuery();
            con.Close();

            Session["username"] = tbUsername.Text.Trim();
            Response.Redirect("Default.aspx");
        }
    }
}

```

```

protected void UsernameValidator_ServerValidate(object source,
System.Web.UI.WebControls.ServerValidateEventArgs args)
{
    string username = args.Value;
    if (!Regex.IsMatch(username, @"^[a-zA-Z0-9_-]{3,15}$")) {
        UsernameValidator.ErrorMessage = "Username must contain letters, digits
and '-' and '_', and its length must be between 3 to 15 characters.";
        args.IsValid = false;
    }
}

protected void PasswordValidator_ServerValidate(object source,
System.Web.UI.WebControls.ServerValidateEventArgs args)
{
    string password = args.Value;
    if (!Regex.IsMatch(password, @"^[a-zA-Z0-9]{8,}$"))
    {
        PasswordValidator.ErrorMessage = "Password can contain only letters and
digits and its length must be at least 8 characters.";
        args.IsValid = false;
    }
}
}
}

```

מנהל המערכת

Admin Data

בפני המנהל מוצגים שני כפתורים – View Users ו-View Commands. בהתאם לכפתור שהמנהל לוחץ עליו, צד השרת מציג את הנתונים בטבלה המתאימה במסד הנתונים.

דף ה- Admin Data (AdminData.aspx):

```
<%@ Page Title="Admin Page" Language="C#" MasterPageFile="~/Site.Master"
AutoEventWireup="true" CodeBehind="AdminData.aspx.cs" Inherits="AI_Project.AdminData" %>

<asp:Content ID="BodyContent" ContentPlaceHolderID="MainContent" runat="server">
    <h2><%: Title %> </h2>

    <asp:Button ID="ShowUsers" runat="server" Text="View Users"
OnClick="ShowUsers_Click"/>
    <asp:Button ID="ShowCommands" runat="server" Text="View Commands"
OnClick="ShowCommands_Click" />

    <div runat="server" id="dataDiv" visible="false">
        <asp:GridView ID="SqlDataGV" runat="server"/>
    </div>

    <div runat="server" visible="false">
        <asp:SqlDataSource
            id="UsersData"
            runat="server"
            DataSourceMode="DataReader"
            ConnectionString=""
            SelectCommand="select * from Users" />

        <asp:SqlDataSource
            id="CommandsData"
            runat="server"
            DataSourceMode="DataReader"
            ConnectionString=""
            SelectCommand="select * from Commands" />
    </div>
</asp:Content>
```

:(AdminData.aspx.cs) code behind-ה

```
using System;
using System.Web.UI;

namespace AI_Project
{
    public partial class AdminData : Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            string conStr = String.Format("Data
Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=\"{0}\"\\DATABASE.mdf\";Integrated
Security=True", Server.MapPath("~/App_Data"));
            UsersData.ConnectionString = conStr;
            CommandsData.ConnectionString = conStr;
        }

        protected void ShowUsers_Click(object sender, EventArgs e)
        {
            dataDiv.Visible = true;
            SqlDataGV.DataSourceID = "UsersData";
        }

        protected void ShowCommands_Click(object sender, EventArgs e)
        {
            dataDiv.Visible = true;
            SqlDataGV.DataSourceID = "CommandsData";
        }
    }
}
```

Block User

המנהל יכול לחסום משתמש מכניסה לאתר, כלומר למחוק את הרשומות שלו ממסד הנתונים (את שם המשתמש והסיסמא). נשים לב שהמשתמש לא יכול לחסום את עצמו או משתמש לא קיים.

דף ה-Block User (BlockUser.aspx):

```
<%@ Page Title="Block User" Language="C#" MasterPageFile="~/Site.Master"
AutoEventWireup="true" CodeBehind="BlockUser.aspx.cs" Inherits="AI_Project.BlockUser" %>
```

```
<asp:Content ID="BodyContent" ContentPlaceHolderID="MainContent" runat="server">
    <h2><%: Title %> </h2>
    <asp:TextBox runat="server" ID="BlockUserTB" />
    <asp:RequiredFieldValidator runat="server" ControlToValidate="BlockUserTB"
ErrorMessage="Field is required." ForeColor="Red" />
    <br />
    <asp:Button runat="server" ID="BlockButton" Text="Block User"
OnClick="BlockButton_Click" />
</asp:Content>
```

ה-code behind (BlockUser.aspx.cs):

```
using System;
using System.Web.UI;
using System.Data.SqlClient;

namespace AI_Project
{
    public partial class BlockUser : System.Web.UI.Page
    {
        private SqlConnection con;
        protected void Page_Load(object sender, EventArgs e)
        {
            string conStr = String.Format("Data
Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=\"{0}\\DATABASE.mdf\";Integrated
Security=True", Server.MapPath("~/App_Data"));
            con = new SqlConnection(conStr);
        }

        private bool IsUserExist(string username)
        {
            con.Open();
            string query = "select * from Users where username=@username";
            SqlCommand cmd = new SqlCommand(query, con);
            cmd.Parameters.AddWithValue("@username", username);
            var reader = cmd.ExecuteReader();
            bool res = reader.Read();
            con.Close();
            return res;
        }

        protected void BlockButton_Click(object sender, EventArgs e)
        {
            string username = BlockUserTB.Text.Trim();
            if (username == "admin")
```



```

    {
        ClientScript.RegisterStartupScript(Page.GetType(), "validation", "<script
language='javascript'>alert(\"Admin can't be blocked.\")</script>");
        return;
    }
    if (!IsUserExist(username))
    {
        ClientScript.RegisterStartupScript(Page.GetType(), "validation", "<script
language='javascript'>alert(\"Username doesn't exists.\")</script>");
        return;
    }
    con.Open();
    string query = "delete from Users where username=@username";
    SqlCommand cmd = new SqlCommand(query, con);
    cmd.Parameters.AddWithValue("@username", username);
    cmd.ExecuteNonQuery();
    con.Close();
    BlockUserTB.Text = "";
    ClientScript.RegisterStartupScript(Page.GetType(), "validation", "<script
language='javascript'>alert('User has been blocked successfully.')</script>");
}
}
}

```

Create User

מנהל האתר יוכל ליצור משתמש על ידי הזנת שם משתמש וסיסמא (שלא כמו ב-Sign Up, הוא לא יחובר למשתמש זה ויועבר לדף הראשי). נשים לב שהמנהל לא יכול ליצור משתמש עם שם משתמש קיים.

נשים לב בנוסף שמנהל המערכת יכול לעקוף את ההגבלות שדרשנו עבור סיסמאות (זאת מכיוון שיש לו הרשאות לעשות זאת ואנחנו סומכים על מנהל האתר, אם היינו רוצים לא לאפשר לו את המעקף הזה היינו יכולים לחסום זאת בקלות על ידי אותה לוגיקה של ה-sign up).

דף ה-Create User (CreateUser.aspx):

```
<%@ Page Title="Create User" Language="C#" MasterPageFile="~/Site.Master"
AutoEventWireup="true" CodeBehind="CreateUser.aspx.cs" Inherits="AI_Project.CreateUser"
%>

<asp:Content ID="BodyContent" ContentPlaceHolderID="MainContent" runat="server">
    <div id="LoginPage">
        <asp:Table runat="server" Width="100%">

            <asp:TableRow>
                <asp:TableCell Width="40%">
                    <asp:Label ID="lblUsername" runat="server" Text="Username: " Font-
Size="XX-Large"/>
                </asp:TableCell>
                <asp:TableCell Width="60%">
                    <asp:TextBox ID="tbUsername" CssClass="tb" runat="server"
placeholder="Username" Width="50%"></asp:TextBox>
                    <asp:RequiredFieldValidator runat="server"
ControlToValidate="tbUsername" ErrorMessage="Field is required." ForeColor="Red" />
                </asp:TableCell>
            </asp:TableRow>
            <asp:TableRow Height="10"/>
            <asp:TableRow>
                <asp:TableCell Width="40%">
                    <asp:Label ID="lblPassword" runat="server" Text="Password: " Font-
Size="XX-Large"/>
                </asp:TableCell>
                <asp:TableCell Width="60%">
                    <asp:TextBox ID="tbPassword" CssClass="tb" runat="server"
placeholder="Password" Width="50%" TextMode="Password"/>
                    <asp:RequiredFieldValidator runat="server"
ControlToValidate="tbPassword" ErrorMessage="Field is required." ForeColor="Red" />
                </asp:TableCell>
            </asp:TableRow>
            <asp:TableRow Height="10"/>
            <asp:TableRow>
                <asp:TableCell Width="50%">
                    <asp:Button ID="sendButton" runat="server" text="Create User"
Width="50%" OnClick="sendButton_Click"/>
                </asp:TableCell>
            </asp:TableRow>
        </asp:Table>
    </div>
</asp:Content>
```

```

        <asp:TableCell Width="50%">
            <asp:Button ID="clearButton" runat="server" text="Clear" Width="50%"
OnClick="clearButton_Click"/>
        </asp:TableCell>
    </asp:TableRow>

</asp:Table>
</div>
</asp:Content>

```

:(CreateUser.aspx.cs) code behind-7

```

using System;
using System.Web.UI;
using System.Data.SqlClient;

namespace AI_Project
{
    public partial class CreateUser : System.Web.UI.Page
    {
        private SqlConnection con;
        protected void Page_Load(object sender, EventArgs e)
        {
            string conStr = String.Format("Data
Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=\\{0}\\DATABASE.mdf\\";Integrated
Security=True", Server.MapPath("~/App_Data"));
            con = new SqlConnection(conStr);
        }

        protected void clearButton_Click(object sender, EventArgs e)
        {
            tbPassword.Text = "";
            tbUsername.Text = "";
        }

        private bool IsUserExist(string username)
        {
            con.Open();
            string query = "select * from Users where username=@username";
            SqlCommand cmd = new SqlCommand(query, con);
            cmd.Parameters.AddWithValue("@username", username);
            var reader = cmd.ExecuteReader();
            bool res = reader.Read();
            con.Close();
            return res;
        }

        protected void sendButton_Click(object sender, EventArgs e)
        {
            string username = tbUsername.Text.Trim(), password = tbPassword.Text.Trim();
            if (IsUserExist(username))
            {
                ClientScript.RegisterStartupScript(Page.GetType(), "validation", "<script
language='javascript'>alert('Username already exists.')</script>");
                return;
            }
            con.Open();
            string query = String.Format("insert into Users(username, password)
values('{0}','{1}')" , username, password);

```

```
SqlCommand cmd = new SqlCommand(query, con);
cmd.ExecuteNonQuery();
con.Close();
ClientScript.RegisterStartupScript(Page.GetType(), "validation", "<script
language='javascript'>alert('User created successfully.')</script>");
    }
}
```

צ'אט וממשק IExecutioner

צ'אט

בפני המשתמש מוצג מסך צ'אט עם תיבת טקסט וכפתור שליחה. ההודעות שיכתבו בעתיד על ידי המשתמש ותגובות הבוט אליהם יוצגו במסך זה. המשתמש יכתוב הודעה, ישלח אותה (ע"י הכפתור), המערכת תחשב את התגובה לפי הלוגיקה שאתאר בהמשך ותציג את התגובה על המסך.

דף הצ'אט (Chat.aspx):

```
<%@ Page Language="C#" MasterPageFile="~/Site.Master" AutoEventWireup="true"
CodeBehind="Chat.aspx.cs" Inherits="AI_Project.Chat" %>

<asp:Content ID="BodyContent" ContentPlaceHolderID="MainContent" runat="server">
    <div class="Container">
        <asp:Label ID="lblChat" runat="server" CssClass="ChatLabel"
Text="<b><u>Bot:</u></b> Hi! I am chat-bot. Write <b>help</b> for information about
commands." Width="100%" Height="90%"/>
        <div style="display: flex; height: auto; width: 100%">
            <asp:TextBox ID="tbMessage" CssClass="MessageBox" runat="server" Width="90%"
Height="10%" AutoCompleteType="Disabled"/>
            <asp:Button ID="btnSend" CssClass="SendButton" runat="server" Text="Send"
Width="10%" Height="10%" OnClick="btnSend_Click"/>
        </div>
    </div>
</asp:Content>
```

ה-code behind (Chat.aspx.cs):

```
using System;
using System.Data.SqlClient;
using System.Text.RegularExpressions;
using System.Web.UI;

namespace AI_Project
{
    public partial class Chat : Page
    {
        private CommandExecutioner executioner;
        private SqlConnection con;
        protected void Page_Load(object sender, EventArgs e)
        {
            string conStr = String.Format("Data
Source=(LocalDB)\\MSSQLLocalDB;AttachDbFilename=\\{0}\\DATABASE.mdf\\;Integrated
Security=True", Server.MapPath("~/App_Data"));
            executioner = new CommandExecutioner(conStr, Server.MapPath("~/App_Data"));
            con = new SqlConnection(conStr);
        }

        private string GetBotResponse(string command)
        {
            return executioner.GetResponse(command);
        }
    }
}
```

```

private void AddMessage(string message)
{
    lblChat.Text += "<br>";
    lblChat.Text += message;
}

protected void btnSend_Click(object sender, EventArgs e)
{
    string command = tbMessage.Text;
    if (command != "")
    {
        AddMessage("<b><u>You:</u></b> " + command);
        string response = GetBotResponse(Regex.Replace(command, @"\s+", "
    ")).Trim());
        AddMessage("<b><u>Bot:</u></b> " + response);
        tbMessage.Text = "";
    }
}
}

```

ממשק IExecutioner

בכדי לממש את לוגיקת ביצוע הפעולות של הבוט השתמשתי בתבנית עיצוב שנקראת "Command". התבנית נועדה לעטוף (או להסתיר) את כל המידע הרלוונטי להרצת מתודה מסוימת (המופע ממנו קוראים לה, שמה, והפרמטרים שנשלחים לה) בתוך אובייקט יחיד. לשם כך, יצרתי ממשק שנקרא IExecutioner.

```
namespace AI_Project.Executioners
{
    interface IExecutioner
    {
        // a method that excecutes the command, return null if the command is wrong
        string Execute(string command);
        string Info { get; }

        bool ToLearn { get; }
    }
}
```

הממשק והמחלקות שמממשות אותו שמורות בתיקייה (package) Executioners.

המתודה Execute מקבלת string שהוא הפקודה, מבצעת אותה ומחזירה string שהוא התגובה (מה שיוצג למשתמש). אם הפקודה שהתקבלה לא נתמכת על ידי האובייקט, יוחזר null. בנוסף יש 2 מאפיינים:

המאפיין Info מסוג string מחזיר שורת הסבר על הפקודה.

המאפיין ToLearn מסוג bool מחזיר האם צריך לשמור את התוצאות של הפקודה ב-cache (יפורט בהמשך).

כעת אפרט את המחלקות השונות שמממשות את הממשק.

Evaluate Executioner

פקודה שמטרתה היא לחשב ביטויים מתמטיים (תרגילים) במספרים שלמים (integers). פורמט הפקודה הוא:

eval arithmetic_expression

המחלקה עושה שימוש ב-NuGet package ששמה Z.Expressions.Eval בגרסה 4.0.42, חבילה האחראית על פישוט הביטויים וחישובם.

המחלקה שומרת את התגובות לפקודות שלה ב-cache ולכן מאפיין ה-ToLearn שלה מחזיר true תמיד.

המחלקה EvaluateExecutioner:

```
using System;
using System.Linq;
using Z.Expressions;

namespace AI_Project.Executioners
{
    public class EvaluateExecutioner : IExecutioner
    {
        public bool ToLearn { get { return true; } }
        public string Info
        {
            get
            {
                return "<b>eval arithmetic_expression</b> - Evaluates the given
expression (pay attention that those are int expressions, so / is integer division and %
is integer modulo).";
            }
        }

        public string Execute(string command)
        {
            var args = command.Split(new char[] { ' ' }, 2);
            if (args.Length != 2 || args[0] != "eval") return null;
            string result = null;
            try
            {
                result = args[1].Execute<int>().ToString();
            } catch (Exception)
            {
                result = null;
            }
            return result;
        }
    }
}
```


Help Executioner

פקודה שמטרתה היא להציג הודעת עזרה עבור המשתמש. פורמט הפקודה הוא:

help

המחלקה עושה שימוש במאפיין HelpString של המחלקה CommandExecutioner (תפורט בהמשך) ולכן מקבלת עצם מהסוג שלו בבנאי שלה.

המחלקה לא שומרת את התגובות לפקודות שלה ב-cache (כיוון שיתכן ונרצה לשנות את הודעת ה-help) ולכן מאפיין ה-ToLearn שלה מחזיר false תמיד.

המחלקה HelpExecutioner:

```
namespace AI_Project.Executioners
{
    public class HelpExecutioner : IExecutioner
    {
        private CommandExecutioner excecutioner;
        public bool ToLearn { get { return false; } }
        public HelpExecutioner(CommandExecutioner excecutioner)
        {
            this.excecutioner = excecutioner;
        }
        public string Execute(string command)
        {
            if (command != "help")
            {
                return null;
            }
            return excecutioner.HelpString;
        }

        public string Info
        {
            get
            {
                return "<b>help</b> - shows a list of commands you can use.";
            }
        }
    }
}
```

Joke Executioner

פקודה שמטרתה היא להציג בדיחה אקראית למשתמש. פורמט הפקודה הוא:

joke

המחלקה עושה שימוש במאגר של בדיחות, שהוא קובץ json הנמצא בתיקייה App_Data והיא מקבלת את הנתיב לתיקייה בו הקובץ נמצא בבנאי שלה.

המחלקה לא שומרת את התגובות לפקודות שלה ב-cache (כיוון שאנחנו רוצים בדיחה שונה בכל פעם) ולכן מאפיין ה-ToLearn שלה מחזיר false תמיד.

המחלקה JokeExecutioner:

```
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.IO;

namespace AI_Project.Executioners
{
    public class JokeExecutioner : IExecutioner
    {
        private string appDataDir;
        public string Info
        {
            get
            {
                return "<b>joke</b> - Tells a random (hilarious) joke.";
            }
        }
        public bool ToLearn { get { return false; } }
        public JokeExecutioner(string add)
        {
            appDataDir = add;
        }

        public class Joke
        {
            public string body;
            public string id;
            public int score;
            public string title;
        }

        Random rand = new Random();

        public string Execute(string command)
        {
            if (command != "joke") return null;
            using (StreamReader r = new StreamReader(Path.Combine(appDataDir,
                "jokes.json")))
            {
            }
        }
    }
}
```

```
{
    string json = r.ReadToEnd();
    List<Joke> items = JsonConvert.DeserializeObject<List<Joke>>(json);
    Joke jo = items[rand.Next(items.Count)];
    return jo.title + "<br/>" + jo.body;
}
}
```

Time Executioner

פקודה שמטרתה היא להציג את הזמן הנוכחי בפורמט שהמשתמש בוחר. פורמט הפקודה הוא:

Time format_string

המחלקה עושה שימוש במאפיין Now של המחלקה DateTime ובמתודה ToString שלה. אם לא התקבל פורמט, הפורמט ברירת המחדל הוא "dd.MM.yyyy hh:mm:ss tt". לעוד הסבר בנוגע לפורמטים השונים יש להיכנס לאתר הזה.

המחלקה לא שומרת את התגובות לפקודות שלה ב-cache (כיוון שאנחנו רוצים לקבל את הזמן העדכני לכל קריאה ולא זמן אחיד לכולם) ולכן מאפיין ה-ToLearn שלה מחזיר false תמיד.

המחלקה TimeExecutioner:

```
using System;

namespace AI_Project.Executioners
{
    public class TimeExecutioner : IExecutioner
    {
        public bool ToLearn { get { return false; } }
        public string Info
        {
            get
            {
                return "<b>time format_string</b> - Tells the current time using the  
given format string. " +
                    "For more information about the format string, visit " +
                    "<a href=\"https://docs.microsoft.com/en-us/dotnet/standard/base-  
types/custom-date-and-time-format-strings\">this website. </a>" +
                    "If used without format string, it prints the output with the format  
string \"dd.MM.yyyy hh:mm: ss tt\".";
            }
        }

        public string Execute(string command)
        {
            var args = command.Split(new char[] { ' ' }, 2);
            if (args.Length == 1 && args[0] == "time")
            {
                return DateTime.Now.ToString("dd.MM.yyyy hh:mm:ss tt");
            }
            if (args.Length > 2 || args[0] != "time")
            {
                return null;
            }
            return DateTime.Now.ToString(args[1]);
        }
    }
}
```

Translate Executioner

פקודה שמטרתה היא לתרגם טקסט בין שפות לבחירת המשתמש פורמט הפקודה הוא:

translate from_lang to_lang string_to_translate

המחלקה עושה שימוש ב-WebClient והיא מתרגמת את הטקסט בין השפות על ידי שליחת בקשת GET מהכתובת: "https://translate.googleapis.com/translate_a/single" תוך הכנסת הפרמטרים המתאימים.

המחלקה לא שומרת את התגובות לפקודות שלה ב-cache (כיוון ששרת ה-SQL לא תומך בקידודים שמאפשרים שמירה של שפות שונות כמו עברית) ולכן מאפיין ה-ToLearn שלה מחזיר false תמיד.

המחלקה TranslateExecutioner:

```
using System;
using System.Net;
using System.Web;

namespace AI_Project.Executioners
{
    public class TranslateExecutioner : IExecutioner
    {
        public bool ToLearn { get { return false; } }
        public string Execute(string command)
        {
            var args = command.Split(new char[] { ' ' }, 4);
            if (args.Length != 4 || args[0] != "translate")
            {
                return null;
            }
            return Translate(args[3], args[2], args[1]);
        }

        private string Translate(string word, string toLanguage, string fromLanguage)
        {
            var url =
                $"https://translate.googleapis.com/translate_a/single?client=gtx&sl={fromLanguage}&tl={toLanguage}&dt=t&q={HttpUtility.UrlEncode(word)}";
            var webClient = new WebClient
            {
                Encoding = System.Text.Encoding.UTF8
            };
            var result = webClient.DownloadString(url);
            try
            {
                result = result.Substring(4, result.IndexOf("\\", 4,
                    StringComparison.Ordinal) - 4);
                return result;
            }
        }
    }
}
```

```
        catch
        {
            return null;
        }
    }

    public string Info
    {
        get
        {
            return "<b>translate from_lang to_lang string_to_translate</b> -  
Translates a string between two languages.";
        }
    }
}
}
```

המחלקה CommandExecutioner וביצוע הפקודות

המחלקה CommandExecutioner אחראית על ביצוע הפקודות. דף ה-Chat מחזיק אובייקט של מחלקה זו ומשתמש במתודה GetResponse שלו בכדי לקבל את תגובת הבוט לפקודה מסוימת.

אובייקט ה-CommandExecutioner שומר רשימה של IExecutioner-ים כשדה שלו והם הפקודות השמורות שקיימות במערכת (מאותחלות בבנאי שלו).

ביצוע הפקודה מתבצע כך:

1. בדיקה ב-cache האם הפקודה קיימת. אם כן – החזרה של התגובה השמורה ב-cache + הודעה שמראה שהפתרון הגיע מה-cache (רק לצורכי הדגמה, בפועל בוודאי שנמחק אותה).
2. אם הפקודה לא קיימת ב-cache, מעבר על כל הפקודות ברשימת הפקודות, ביצוע כל אחת מהם (על ידי קריאה למתודת ה-Execution שלהם כפי שמוגדרת בממשק) ובדיקה אם התוצאה היא null. אם כן, מעבר לפקודה הבאה. אם לא – החזרת התוצאה.
3. בדיקה האם משתנה ה-toLearn מראה true – משתנה זה הוא משתנה שמועבר לפונקציה ExecuteSavedCommand כפרמטר שמועבר על ידי רפרנס (out bool toLearn) וכך כאשר המתודה תשנה אותו גם המשתנה המקורי ישתנה. המתודה משנה אותו לפי מאפיין ה-ToLearn של אובייקט ה-IExecutioner שהצליח לבצע את הפקודה. אם המשתנה אכן מראה true, מתבצעת שמירה של הפקודה והתגובה אליה ב-cache. אחרת, הוא לא נשמר.
4. החזרת התוצאה שהתקבלה, והודעה שגיאה אם לא התקבלה תוצאה (כל הפקודות השמורות החזירו null ולכן הפונקציה ExecuteSavedCommand החזירה null גם היא).

המחלקה :CommandExecutioner

```

using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Text.RegularExpressions;
using AI_Project.Executioners;

namespace AI_Project
{
    public class CommandExecutioner
    {
        private SqlConnection con;
        private List<IExecutioner> executioners;
        public CommandExecutioner(string conStr, string appDataDir)
        {
            con = new SqlConnection(conStr);

            executioners = new List<IExecutioner>
            {
                new TranslateExecutioner(),
                new HelpExecutioner(this),
                new TimeExecutioner(),
                new EvaluateExecutioner(),
                new JokeExecutioner(appDataDir)
            };
        }

        public void Learn(string command, string response)
        {
            con.Open();
            string query = String.Format("insert into Commands(command, response)
values('{0}','{1}')" , command.Trim().Replace("'", "\""), response.Trim().Replace("'",
"\\'"));
            SqlCommand cmd = new SqlCommand(query, con);
            cmd.ExecuteNonQuery();
            con.Close();
        }

        private List<string> GetValuesByColumnName(SqlDataReader reader, string name)
        {
            List<string> vals = new List<string>();
            while (reader.Read())
            {
                vals.Add(reader[name].ToString());
            }
            return vals;
        }

        public string HelpString
        {
            get
            {
                string s = "Some commands you can use: <br/>";
                for (int i = 0; i < executioners.Count; ++i)
                {
                    s += executioners[i].Info;
                }
            }
        }
    }
}

```



```

        if (i != executioners.Count - 1) s += "<br/>";
    }
    return s;
}

private string ExcecuteSavedCommand(string command, out bool toLearn) {
    foreach (var e in executioners)
    {
        var res = e.Execute(command);
        if (res != null)
        {
            toLearn = e.ToLearn;
            return res;
        }
    }
    toLearn = false;
    return null;
}

Random rand = new Random();

public string GetResponse(string command)
{
    command = Regex.Replace(command, @"\s+", " ").Trim();
    string response;
    con.Open();
    string query = "select * from Commands where command=@command";
    SqlCommand cmd = new SqlCommand(query, con);
    cmd.Parameters.AddWithValue("@command", command);
    var reader = cmd.ExecuteReader();
    if (reader.HasRows)
    {
        List<string> responses = GetValuesByColumnName(reader, "response");
        reader.Close();
        response = responses[rand.Next(responses.Count)] + "<br/>(This is from
data base!!!)"; // database part can be removed
        con.Close();
    }
    else
    {
        reader.Close();
        con.Close();
        response = ExcecuteSavedCommand(command, out bool toLearn);
        if (response != null)
        {
            if (toLearn) Learn(command, response);
        }
        else
        {
            response = "Wrong command. Use <b>help</b> to read about the commands
format.";
        }
    }
    return response;
}
}
}

```

שימוש ב-cache ושמירת תוצאות

חלק מן הפקודות יכולות להיות מסובכות וביצוען יכול לקחת זמן רב (כמו לדוגמא חישוב מתמטי ארוך או עיבוד קובץ גדול). בכדי לבצע הקלה כלשהי, ברצוננו לשמור את הבעיות והפתרונות בדרך כלשהי וכך בפעם הבאה שיבקשו את הבעיה, נבדוק האם היא קיימת במאגר בו שמרנו את הנתונים ואם כן נחזיר ישר את הפתרון מבלי לבצע את הפעולה הארוכה שוב. מנגנון זה נקרא cache והוא מנגנון נפוץ מאוד בימינו.

במערכת שלי, נעשה שימוש ב-cache ב-CommandExecutioner. ה-cache הוא טבלה במסד הנתונים שנקראת Commands ובה יש עמודה של Command ועמודה של Response. כל פעולה של IExecutioner שמאפיין ה-ToLearn שלו הוא true תבוצע כרגיל בפעם הראשונה, ובפעם הבאה שאותה הפקודה תיכתב הפתרון ימצא ב-cache (כלומר בטבלה) ויוחזר למשתמש ללא ביצוע הפעולה בפועל שנית.


מנגנון ה-cache חוסך זמן רב וחישובים רבים והוא מועיל מאוד בהרבה תחומים. אמנם בפרוייקט שלי רוב הפקודות היו משתנות ולכן לא היו צריכות שימוש ב-cache (כמו למשל date או joke), אך עדיין נעשה בו שימוש (לדוגמא בפקודה eval).

מבנה מסד הנתונים


כפי שהסברתי, מסד הנתונים בנוי משתי טבלאות:

טבלת ה-Users בה שמורים הנתונים עבור כל משתמש.

הגדרת הטבלה:


	Name	Data Type	Allow Nulls	Default
	Id	int	<input type="checkbox"/>	
	username	varchar(50)	<input type="checkbox"/>	
	password	varchar(50)	<input type="checkbox"/>	
			<input type="checkbox"/>	

הנתונים השמורים בטבלה (כרגע):



	Id	username	password
	6	admin	admin
	7	123	123
	NULL	NULL	NULL

טבלת ה-Commands בה שמורות הפקודות שהתבצעו (ה-cache).

הגדרת הטבלה:

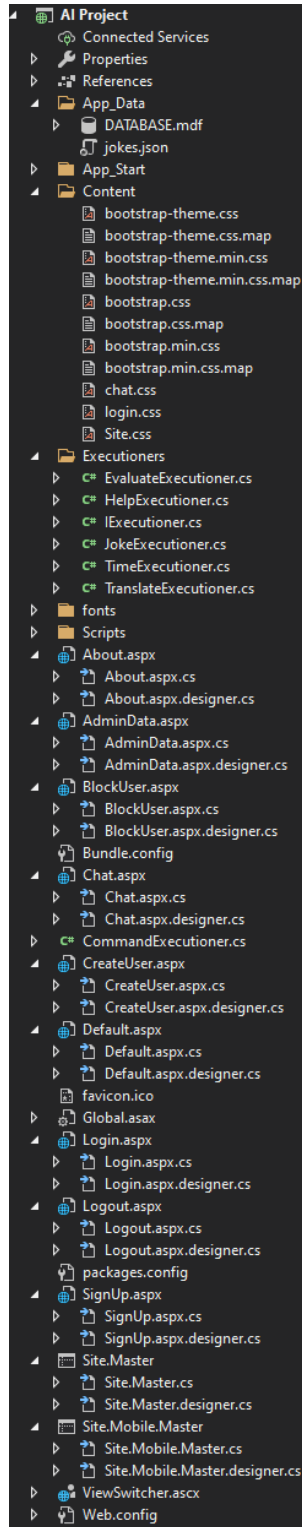
	Name	Data Type	Allow Nulls	Default
	Id	int	<input type="checkbox"/>	
	command	varchar(MAX)	<input type="checkbox"/>	
	response	varchar(MAX)	<input type="checkbox"/>	
			<input type="checkbox"/>	

הנתונים השמורים בטבלה (כרגע):

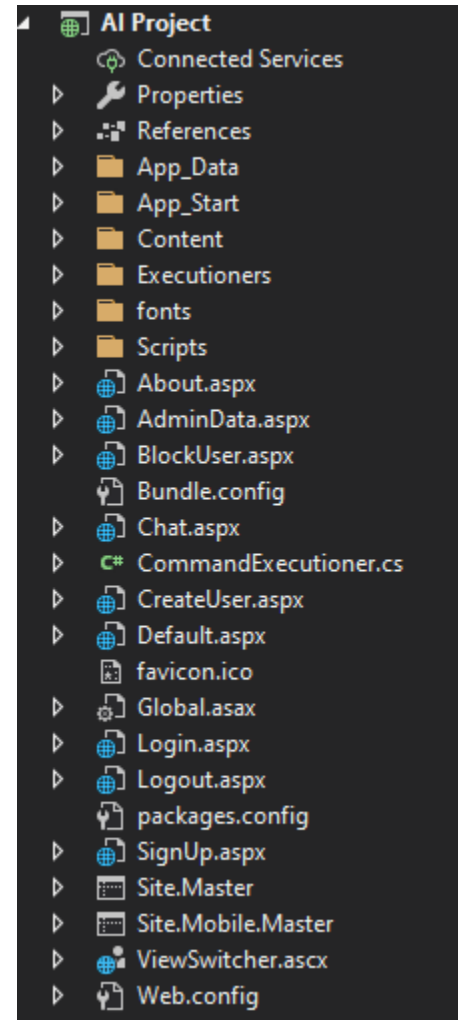
	Id	command	response
	1	hi	hi
	2	hello	hello
	3	what's your na...	my name is cha...
	4	eval 1+1	2
	5	eval 1+2	3
	NULL	NULL	NULL

מבנה התיקיות

מצב פתוח:

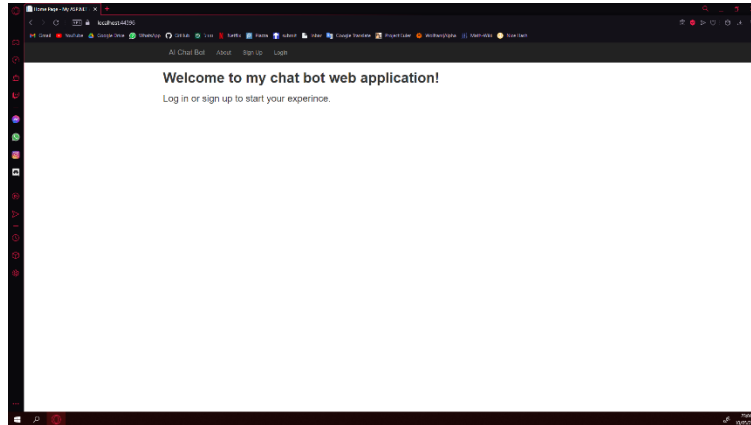


מצב סגור:

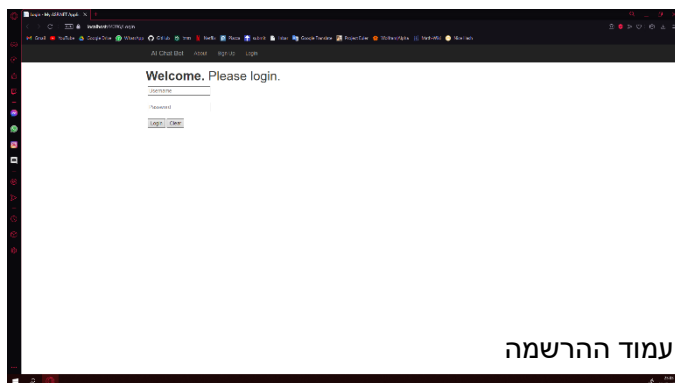


מדריך למשתמש באתר

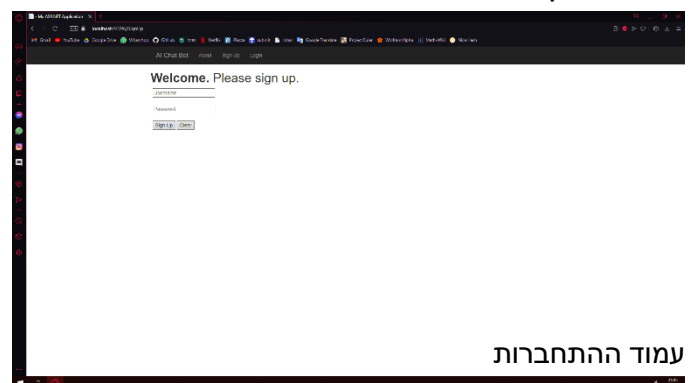
המשתמש הלא מחובר מגיע לדף הבית הראשי:



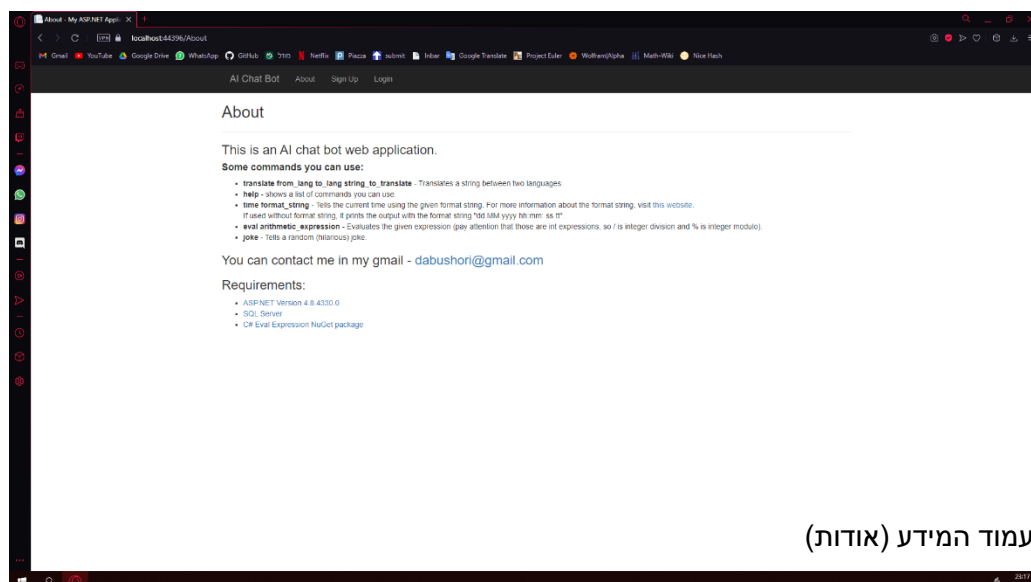
מכאן הוא יכול לעבור לכמה עמודים.



עמוד ההרשמה

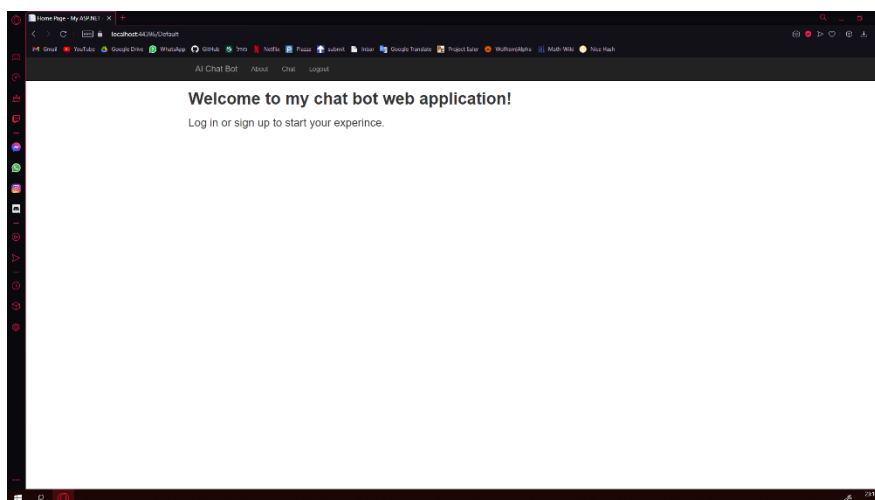


עמוד ההתחברות



עמוד המידע (אודות)

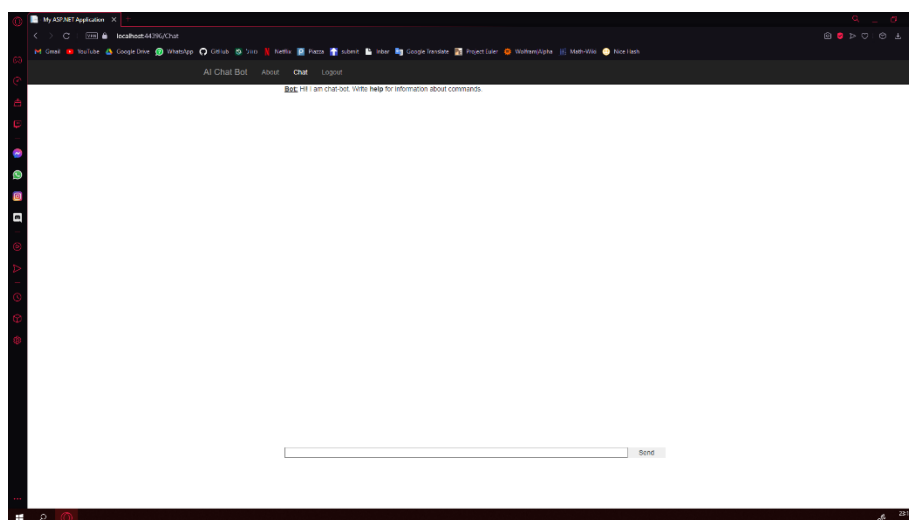
לאחר יצירת משתמש או התחברות למשתמש קיים (שאינו המנהל), התפריט משתנה והמשתמש מועבר לדף הראשי בחזרה:



כעת נפתחו בפניו אפשרויות של צ'אט והתנתקות, ונעלמו האפשרויות של יצירת משתמש והתחברות למשתמש קיים.

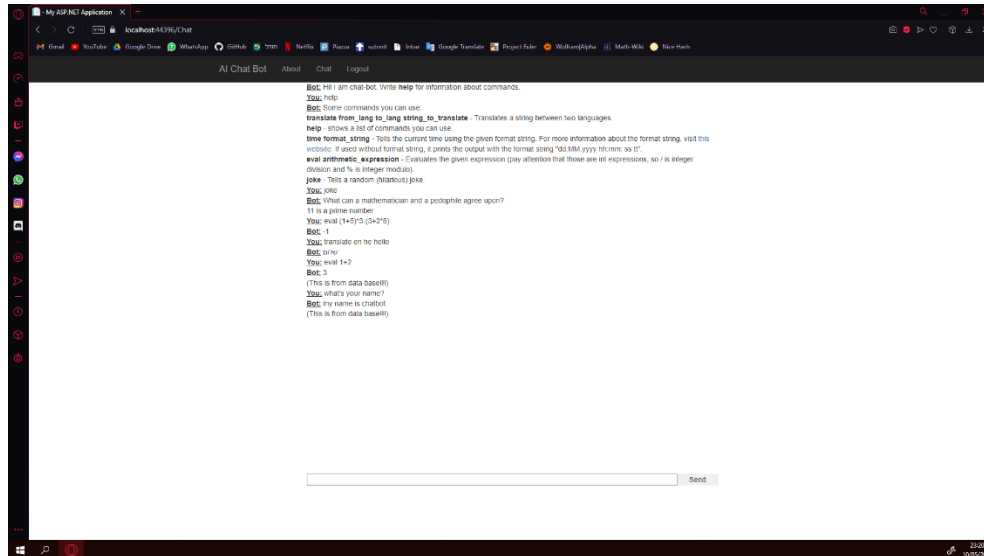
אם ילחץ המשתמש על כפתור ההתנתקות, יתנתק מהמשתמש הנוכחי ויחזור לדף הראשי כפי שמוצג בהתחלה.

אם ילחץ על צ'אט, יועבר לדף הצ'אט:



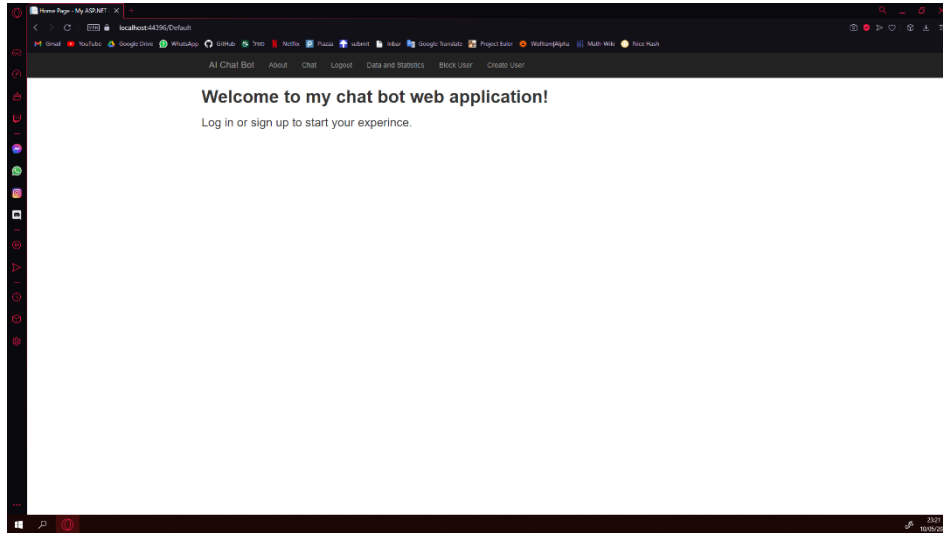
בדף זה נעשית כל ההתכתבות עם הבוט. המשתמש יכול להזין פקודות לתוך תיבת הטקסט ולשלוח אותם על ידי הכפתור או לחיצה על אנטר.

דוגמה להתכתבות עם הבוט:

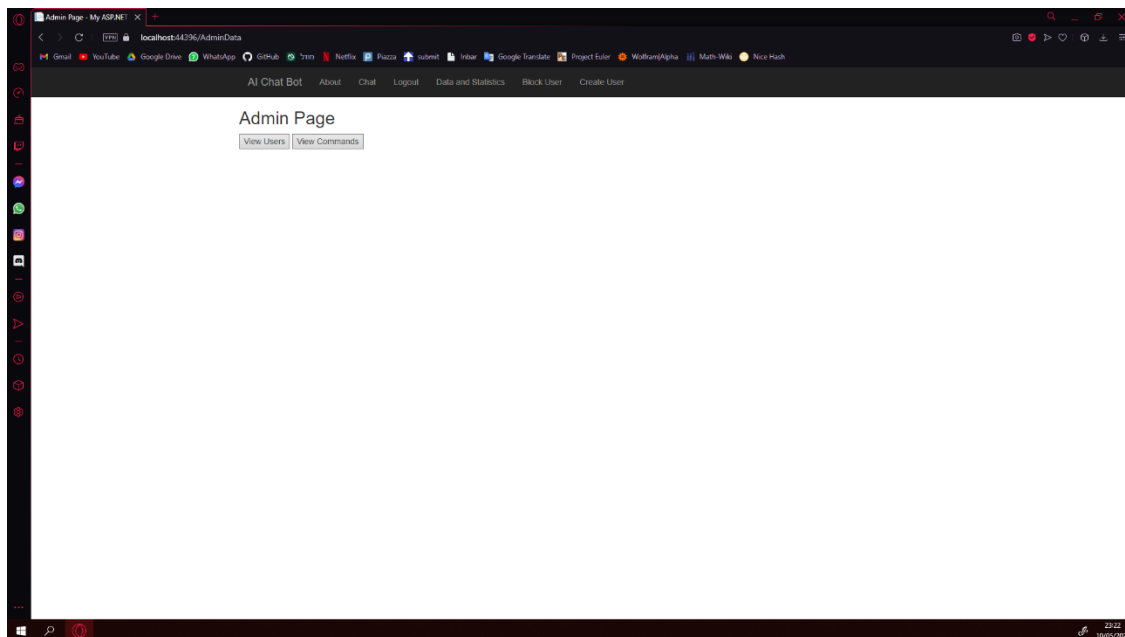


מדריך למנהל האתר

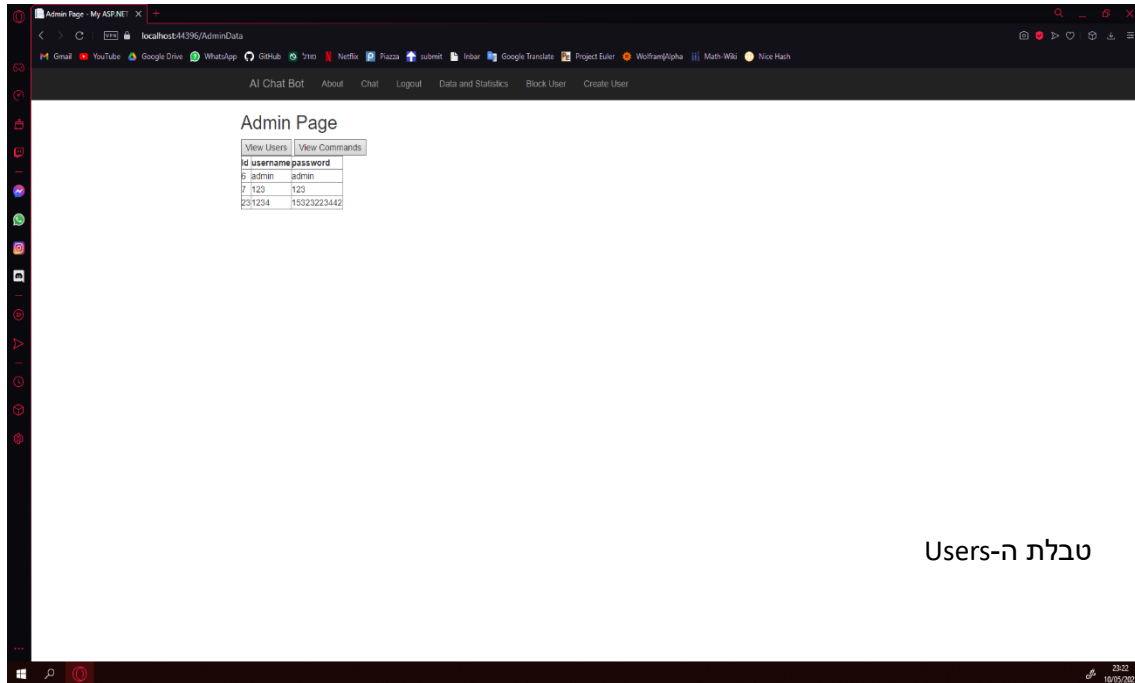
נניח שהמנהל התחבר למערכת (על ידי שימוש בשם המשתמש ובסיסמא admin). הוא יועבר למסך הבית ויראה את התפריט הבא:



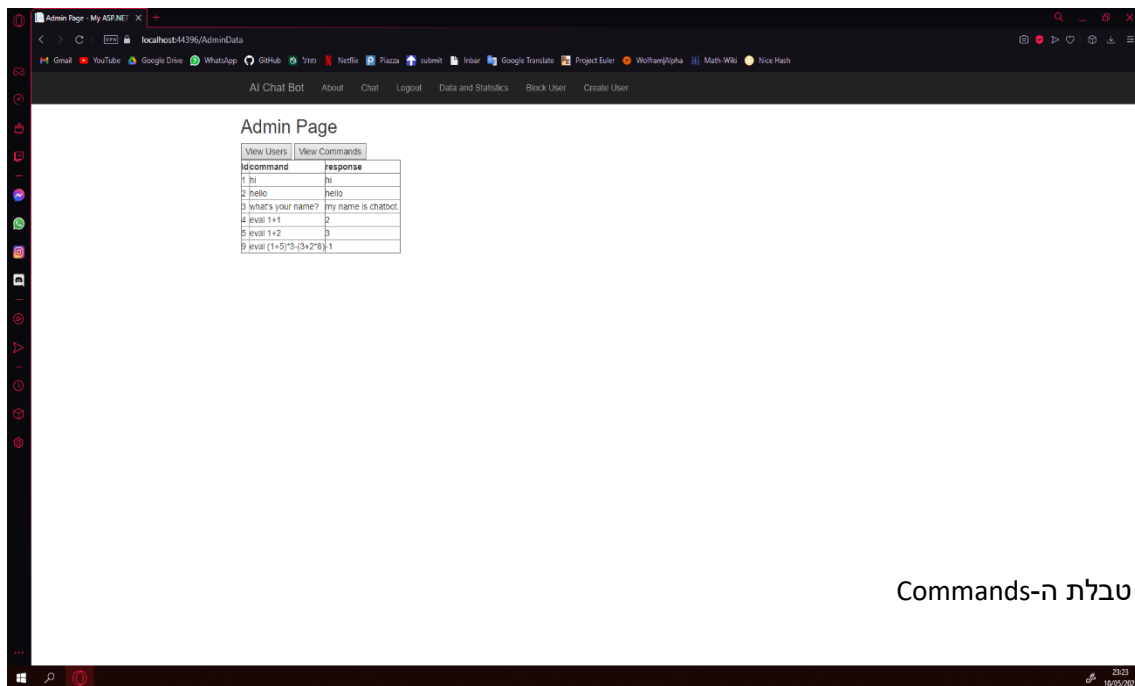
כעת פתוחות בפניו אופציות שכבר הראתי במדריך למשתמש (עמוד המידע, עמוד הצ'אט וההתנתקות). בנוסף פתוחות עבורו אפשרויות נוספות – מידע וסטטיסטיקות, חסימת משתמש ויצירת משתמש. כאשר המנהל יעבור לדף המידע והסטטיסטיקות הוא יראה את העמוד הבא:



כאשר ילחץ על כפתור, יוצג בפניו מאגר הנתונים המתאים.

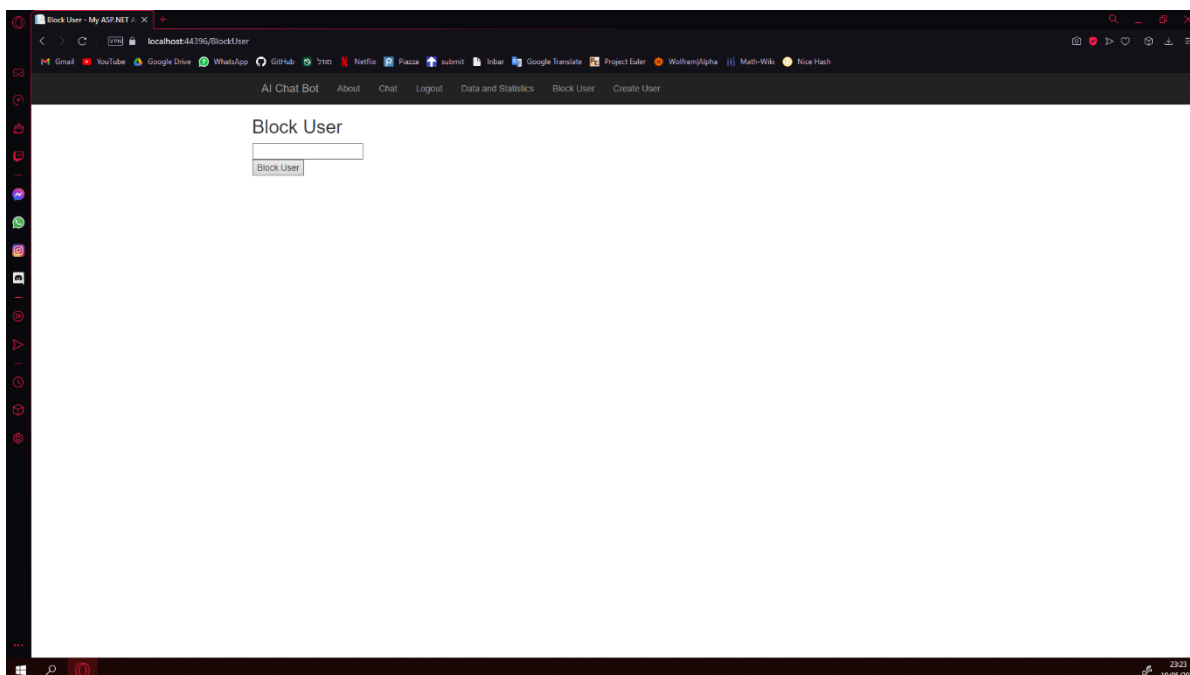


טבלת ה-Users

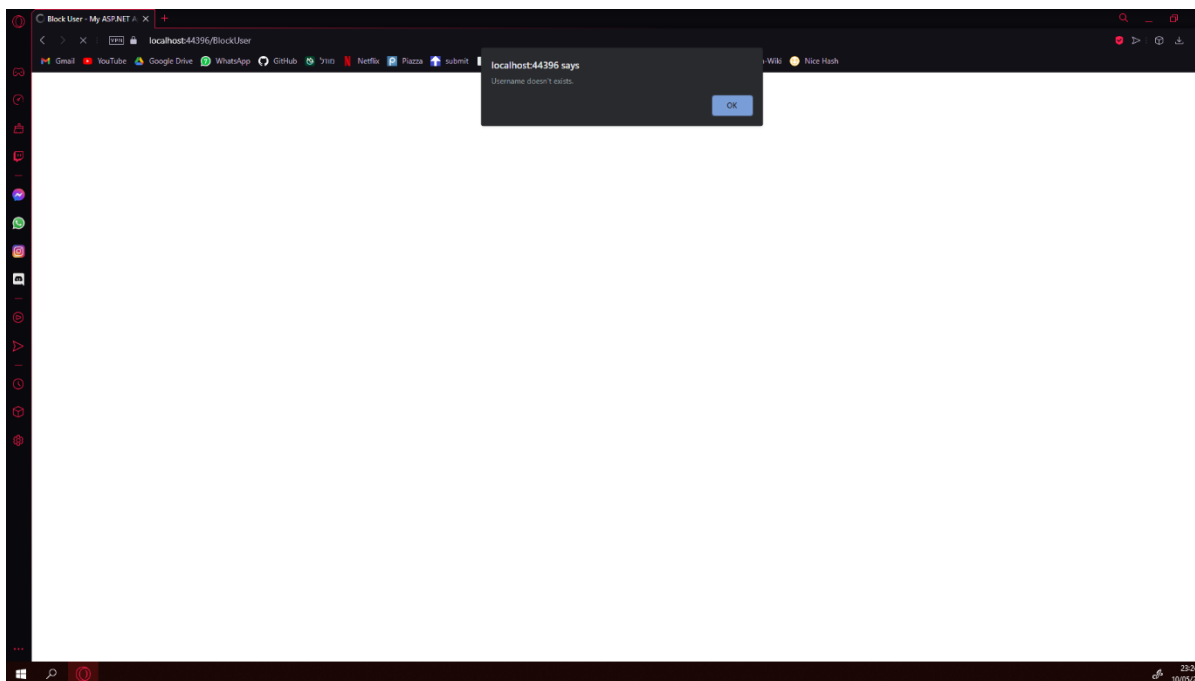


טבלת ה-Commands

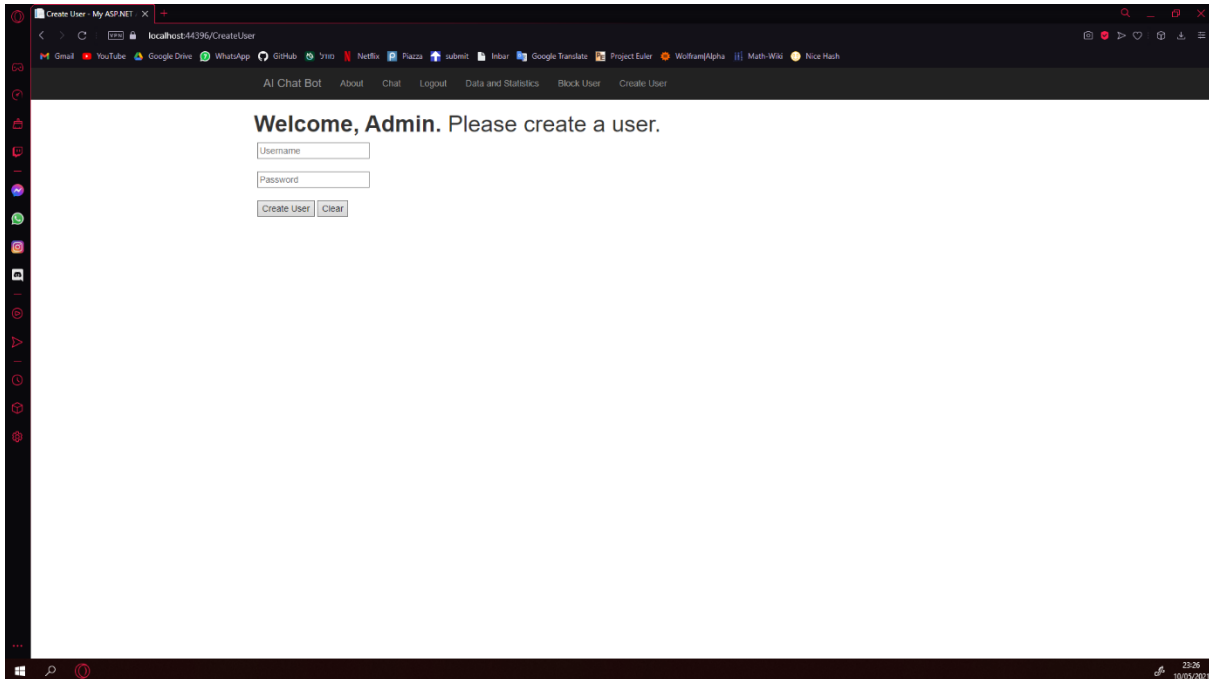
כאשר המנהל יעבור לדף חסימת המשתמש, יוצג בפניו הדף הבא:



כאן הוא יכול להקליד שם משתמש ולחסום אותו. במידת הצורך, יוצגו הודעות שגיאה או הודעות שהפעולה הצליחה. לדוגמא:



כאשר יעבור המנהל לדף יצירת המשתמש, יוצג בפניו הדף הבא:



בתיבות הטקסט הוא יוכל להזין מידע כרצונו וליצור משתמש. במקרה הצורך, תוצג הודעת שגיאה או הצלחה למסך. לדוגמא:

