roTracker

שם התלמידה: דניאל וייסמן

ת.ז. של התלמידה: 207421256

שם בית הספר ועיר: אורט ע"ש נעמי שמר גן יבנה

שם המנחה: אנטולי פיימר

מועד הגשת המסמך: יוני 2017

תוכן עניינים:

רקע ותיאור מוצר מוגמר..... 3

מבט אישי על העבודה ועל תהליך פיתוחה..... 4

תיאור לממשק..... 6

סביבת עבודה..... 9

תיאור המודלים..... 10

הרחבה על פרוטוקול מרכזי..... 13

תיעוד הקוד..... 15

האלגוריתם המרכזיים בקוד..... 18

ביבליוגרפיה ותודות מיוחדות..... 27

רקע

לצד תחומים רבים שמנסים להתאים את עצמם למציאות הדינמית של היום, נמצא גם החינוך. פרויקטים חינוכיים רבים, כמו רפורמת ה-30% שיזם שי פירון, מנסים להמציא את הגלגל מחדש בכל מה שקשור לבית הספר. הם מעודדים את התלמידים להיעזר במשאבים האין-סופיים הקיימים ברשת, ובכך מגדילים להם את מרחב היצירתיות והעצמאות. כך, הופכת סביבת הלימודים של התלמידים לאחת המייצרת תכנים חדשים באופן תמידי ומהיר, דבר המקשה הן על התלמידים שנוטים להתפזר בעבודתם והן על המעקב של המורים אחר ההתקדמות של תלמידיהם.

שבועיים של חשיבה עברו מאז שנתבקשנו להתחיל לחשוב על רעיון לפרויקט. בעוד כל תלמידי הכיתה מחפשים אחר מקורות השראה שונים, סרקתי את החדר, תוך התמקדות אחר מסך המחשב של כל אחד מהם ופתאום תהיתי לעצמי- " אולי פשוט אעשה מעקב אחר כל הפרויקטים של הכיתה?"

על מנת להפוך את המעקב אחר הפרויקטים ליותר ידידותי עבור כל קהל המשתמשים בכלל ועבור מסגרת כמו בית הספר בפרט, השתמשתי בשיטת ה"קנבן".

מקור המילה "קנבן" הוא בשפה היפנית, ופירושה "כרטיס" או "שלט". אולם, בעולם התעשייתי המילה קיבלה משמעות שונה במעט- מערכת בקרה לשליטה על שרשרת אספקה. המתודה משרתת באופן מלא את שיטת ה"יצור הרזה" (Lean Management) שמתמקדת בייעול זרימת המידע והחומר. השיטה מחלקת את העבודה השלמה שיש לבצע על מנת להגיע למוצר המוגמר ל"שלבים", כאשר כל שלב גם הוא מחולק, ל"משימות". חברות רבות שהשתמשו בשיטה גילו שהיא סייעה להם לתקשר באופן ענייני ולהגיע לתוצאה המתבקשת באופן איכותי יותר. כיום, אפליקציות "קנבן" להורדה ניתן למצוא בכל רחבי הרשת, משימוש לחברות גדולות ועד לשימוש פרטי לניהול לוח הזמנים השבועי. שיערתי כי ה"קנבן", היא בידיוק אחת מהכלים שיכולים להוות מקפצה אחת קדימה למערכת החינוך. כך, למורים תהיה גישה נוחה ל"תמונת המצב" של הפרויקטים עליהם הם אמונים ולתלמידים תהיה חלוקה מסודרת למשימות נקודתיות המוגבלות בזמן.

תיאור מוצר מוגמר

שם המוצר: "ProTracker"

מטרה מרכזית למוצר: להוות פלטפורמה ידידותית למעקב אחר פרויקטים בבית הספר, הסגורה לתלמידים ומורים בלבד.

מבט אישי על העבודה ועל תהליך פיתוחה

תהליך העבודה היה רצוף מכשולים ואתגרים. לראשונה התמודדתי עם יצירת פרויקט שלא רק עושה שימוש במבנה נתונים, אלא גם משלב מספר שפות תכנות ביחד.

את רובן לא הכרתי לפני, ולכן נאלצתי להתמודד עם היכרות הסינטקס שלהן והדרך הנכונה לשלב אותן על מנת לאפשר לזרימה "חלקה" של התהליכים בפרויקט.

בעיה נקודתית אתה התמודדתי:

על מנת להקל על ניהול ארבע הפעולות המרכזיות שצריכות לעבוד על כל אובייקט בפרויקט (CRUD- CREATE, READ, UPDATE and DELETE), הבדלה בין שלושת סוגי המשתמשים השונים שיש לי: Admin, Teacher ו-Student ולתת לכל אחד מהם הרשאות גישה שונות, השתמשתי במחלקות UserIdentity ו- ApplicationUser (היורשת מ-UserIdentity). שתיהן נמצאות בכלי פיתוח של מייקרוסופט אשר נקרא "Entity FrameWork"- שאוטומטית ממפה טבלאות של מסדי נתונים למחלקות. מחלקות אלה נמצאות בהיבט המקיף של המערכת.

אולם, יצרתי גם מחלקות של תפקידים העונים על הצרכים הספציפיים של הפרויקט שלי, ואינם כל כך כללית ברמת המשתמש של המערכת.

הייתי זקוקה לקשר בניהם על מנת שאוכל לדעת, למשל, באותו הרגע האם המשתמש של המערכת אשר מנסה לגשת למידע מסוים באתר הוא מסוג "מורה" או מסוג "תלמיד".

דבר ראשון שעשיתי, היה לדאוג לקשר של ירושה בין סוגי המשתמשים הספציפיים שיצרתי לזהויות הכלליות של המערכת. כך, שכל מחקת משתמש שיצרתי ירשה מהמחלקה ApplicationUser.

דבר שני, היה ליצור שאילתות בתוך ה- IdentityModels שיסייעו לי לגשת למאפיינים ספציפיים שאני יצרתי למשתמש תלמיד או מורה, בהנחה שזיהיתי שמשתמש המערכת הוא אכן אחד מאלה.

```
24      public virtual ICollection<Course> Courses { get; set; }
25      //Create claims
        2 references
26      public async Task<ClaimsIdentity> GenerateUserIdentityAsync(UserManager<ApplicationUser> manager)
27      {
28          // Note the authenticationType must match the one defined in CookieAuthenticationOptions.AuthenticationType
29          var userIdentity = await manager.CreateIdentityAsync(this, DefaultAuthenticationTypes.ApplicationCookie);
30          // Add custom user claims here
31          userIdentity.AddClaim(new Claim("Teacher", this.IsAdmin.ToString()));
32          userIdentity.AddClaim(new Claim("Id", this.Id));
33          userIdentity.AddClaim(new Claim("FirstName", this.FirstName));
34          return userIdentity;
35      }
```

המתודות השונות מחזירות מה- HttpContext , לפי דרישת השאילתה את הערך הראשון המתאים. במידה ולא נמצא ערך מתאים, היא מחזירה ערך דיפולטיבי.

```
namespace ProTracker.Core
{
    8 references
    public static class IdentityUtils
    {
        //Returns whether the current user us Teacher (IsAdmin==true) or not
        6 references
        public static bool IsTeacher()
        {
            return bool.Parse(HttpContext.Current.GetOwinContext().Authentication.User.Claims.FirstOrDefault(x => x.Type == "Teacher")?.Value);
        }
        //Returns the current user's Id
        2 references
        public static string GetUserId()
        {
            return HttpContext.Current.GetOwinContext().Authentication.User.Claims.FirstOrDefault(x => x.Type == "Id")?.Value;
        }
        //Returns the current user's Name
        0 references
        public static string GetUserName()
        {
            return HttpContext.Current.GetOwinContext().Authentication.User.Claims.FirstOrDefault(x => x.Type == "FirstName")?.Value;
        }
    }
```

תיאור הממשק למשתמש:

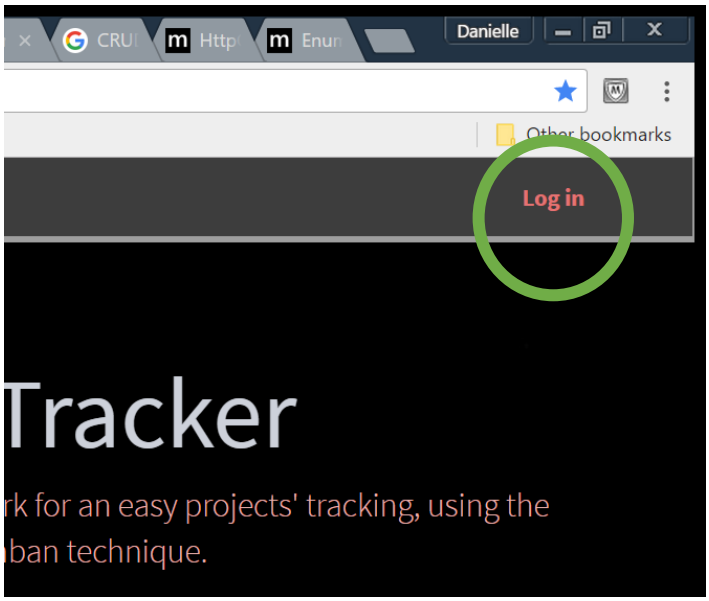
הדגשה חשובה: מדובר על מערכת סגורה, כך שאם אין ברשותך מייל וסיסמא שהוענקו בידי בית הספר, אתה לא אמור להיכנס למערכת. אי אפשר "להירשם" לאתר.

לדוגמה:

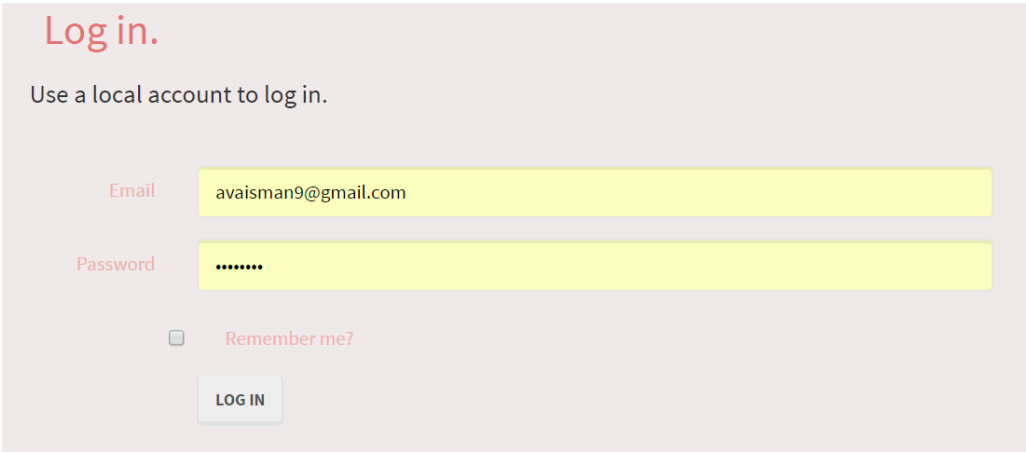
למורה TACHER- אי-מייל admin@protracker.com, סיסמא 44284428

למנהל ADMIN- אי-מייל avaisman9@gmail.com, סיסמא 44284428

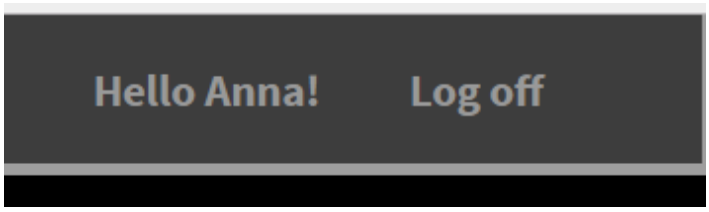
שלב ראשון- הקלק על תווית ה-Log in



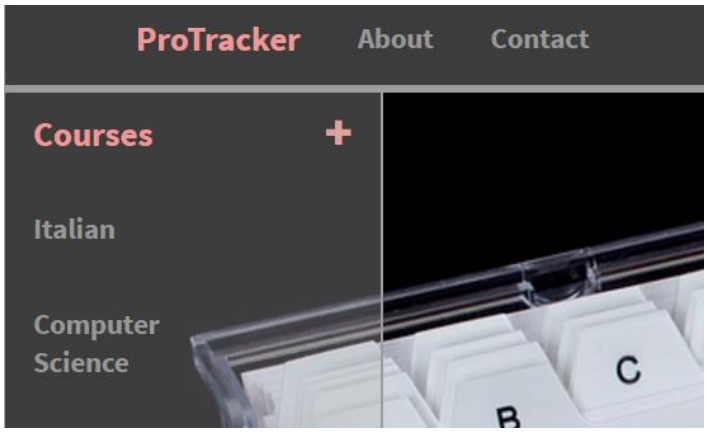
שלב שני- הזן את הפרטים הרלוונטיים לכניסה למערכת, ולחץ על "Log in"



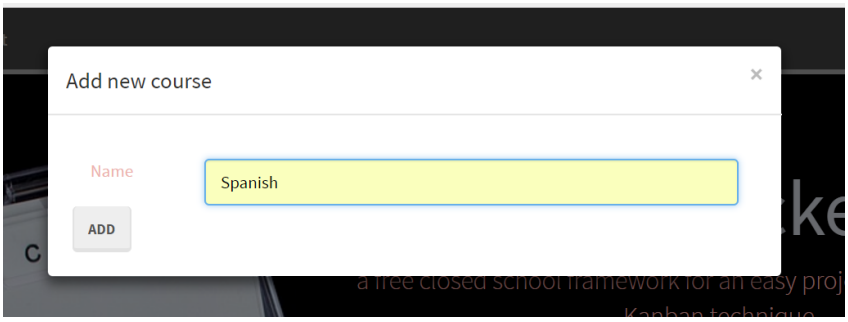
שלב שלישי- אם ברצונך לצאת מהמערכת, הקלק על "Log off" בצד ימין למעלה



בצד שמאל יופיע תפריט תחת הכותרת "קורסים" מדובר על רשימת כל הקורסים שבהם המורה או התלמיד לוקחים בו חלק.

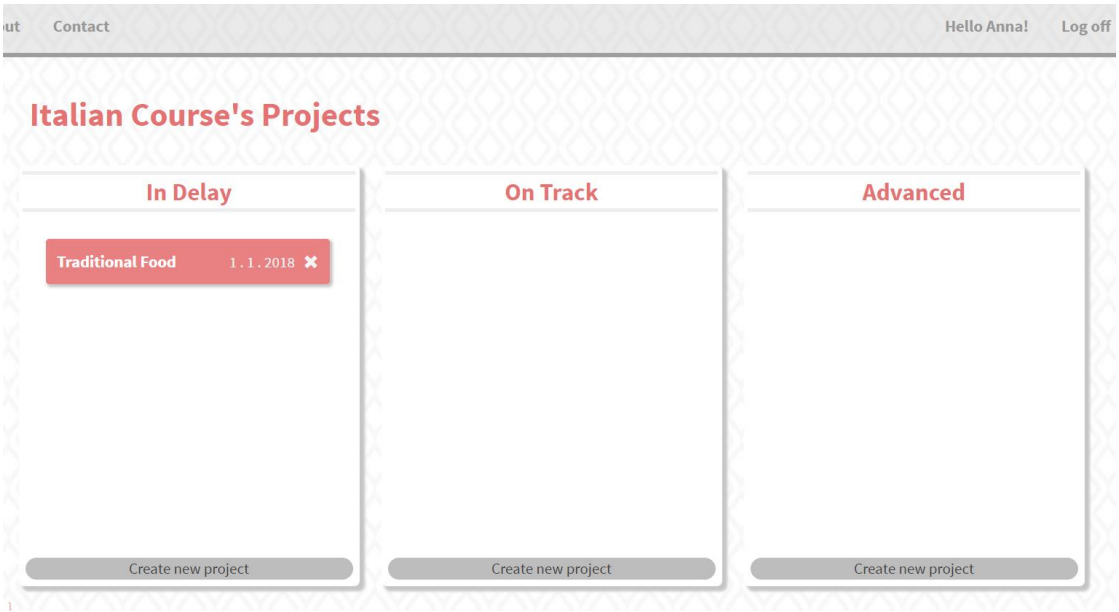


בלחיצה על הפלוס ניתן להוסיף קורס בצורה הבאה:



לאחר ההוספה יהיה ניתן לראות שהקורס נוסף לתפריט מידית:

שלב רביעי- בלחיצה על אחד משמות הקורסים בתפריט ניתן להיכנס ללוח ה"קנבן" של הקורס- יציג את כל הפרויקטים השייכים לקורס בהתאם לסטטוס ההתקדמות שלהם.



כרגע קיים לקורס איטלקית רק פרויקט אחד בשם "אוכל מסורתי". פרויקט זה נמצא בשלב "איחור". כמורה, המשתמש יכול להזיז את התווית הוורודה של הפרויקט ולקדם אותה שלב אחד קדימה או אחורה כרצונו. כתלמיד, המשתמש יכול רק לצפות בתמונת המצב.

בנוסף לכך, כמורה המשתמש יכול גם למחוק את הפרויקט לחלוטין, או להוסיף לעמודת השלב פרויקט חדש, באותו האופן שבו יצרנו קורס.

בלחיצה על שם הפרויקט ייפתח הן לתלמיד והן למורה לוח ה"קנבן" של הפרויקט- הכולל את כל המשימות השייכות לפרויקט בהתאם לסטטוס ההתקדמות שלהן.

סביבת עבודה

השפות שבהן השתמשתי :

C#, HTML, JAVASCRIPT, CSS, SQL

פירוט סביבת העבודה והכלים שנדרשו לפיתוח :

Visual Studio 2015

בסביבת עבודה זו נמצא עיקר הקוד שלי.

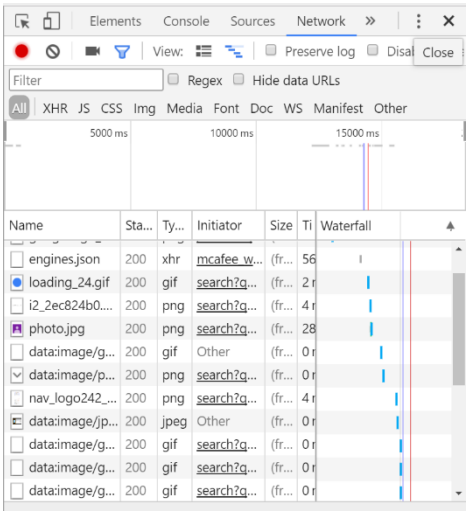
Microsoft sql Management Studio

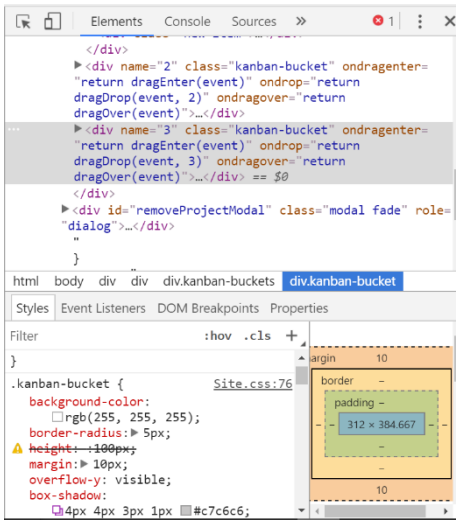
בסביבה זו בעיקר עקבתי אחרי שינויים בdatabase שלי בזמן ריצה.

פירוט הכלים הנדרשים לבדיקות :

Internet Information Service (IIS) manager

עד לאחסון האתר בשרת ופרסומו, האתר מאוכסן בשרת האינטרנטי של IIS





Ctrl+Shift+I in Chrome

באמצעות האופציה הזו היה לי קל יותר לעקוב אחר סטטוסי בקשות (404 NOT FOUND,

200 OK) ובכך לדעת היכן הקוד שלי "נפל". בנוסף, מאחר ולא היה לי ניסיון רב בעיצוב אתרים השתמשתי בכלי זה על מנת לעצב את האתר תוך שימוש בCSS

תיאור המודלים

על מנת לנהל באופן יעיל את הפרויקט השתמשתי בדפוס (pattern) ארכיטקטוני בשם MVC.

הדפוס הזה מחלק את האפליקציה לשלושה רכיבים לוגיים מרכזיים-

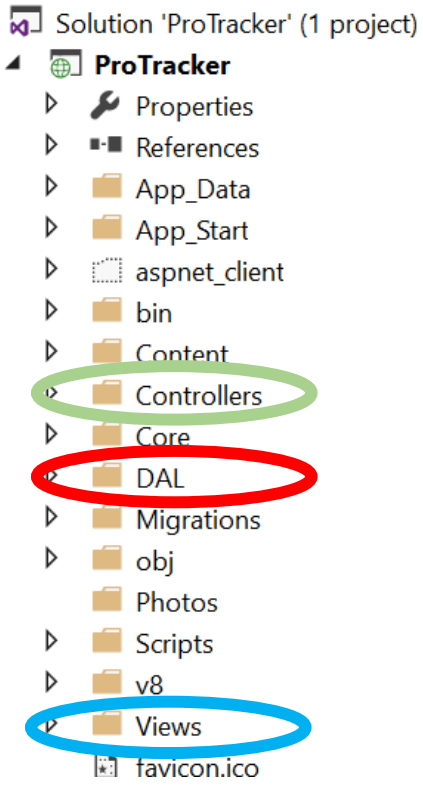
MODEL – ייצוג הנתונים שהמשתמש עובד איתם (כל המודלים שאציג בהמשך)

בפרויקט כל המודלים המרכזיים נמצאים תחת התיקייה "DAL" (DATA ACSSES LAYER)

VIEW- עיצוב ונראות האפליקציה (ה-front)

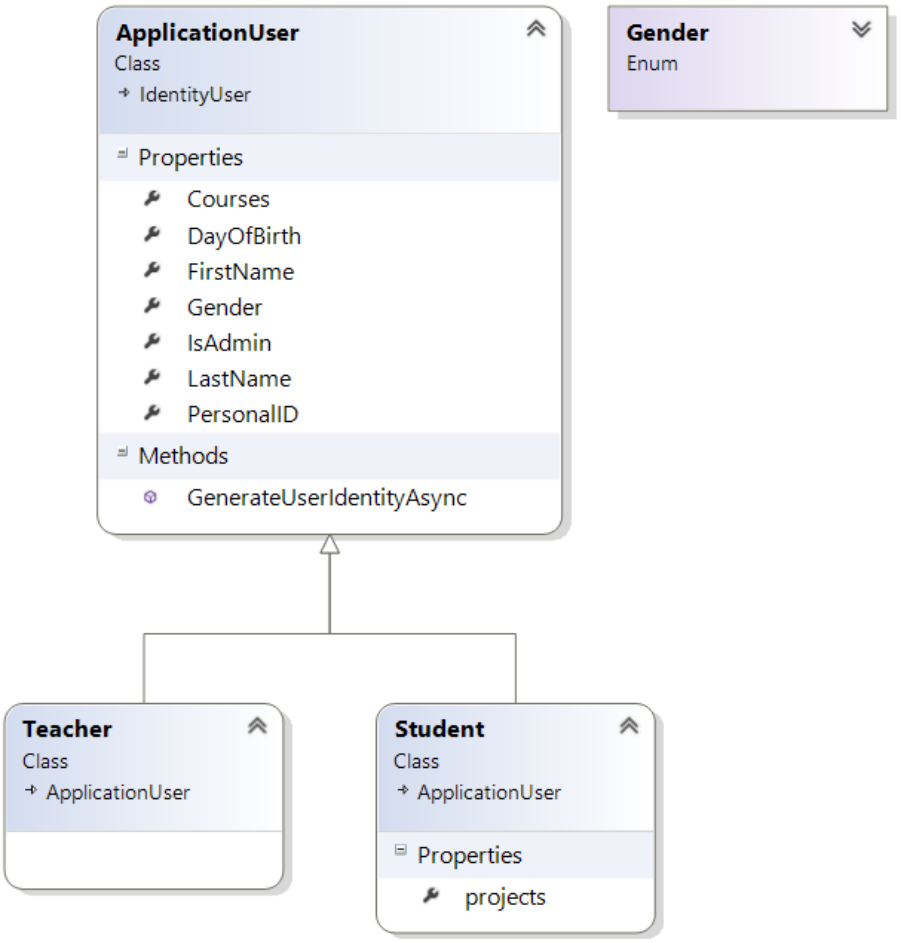
CNTROLLER- ה"דבק" המחבר בין המודל לייצוג. עיבוד כל הלוגיקה והבקשות הנכנסות, לתפעול הנתונים באמצעות רכיבי הדגם ויצירת אינטראקציה עם התצוגות להצגת הפלט הסופי.

MVC תומך בכל פונקציות ASP.NET הקיימות הקימות, כגון הרשאה ואימות, דפים ראשיים, כריכת נתונים, בקרת משתמשים, חברויות וניתוב.



מודלים הקשורים למשתמש:

כפי שציינתי קודם לכן, שני סוגי המשתמשים באתר- הן "Student" והן "Tacher" יורשים ממחלקת "ApplicationUser" של EntityFramework, בה נמצאים מאפייני משתמש כמו שם פרטי, שם משפחה, וכתובת אי-מייל.



מודלים הקשורים בתוכן האתר:

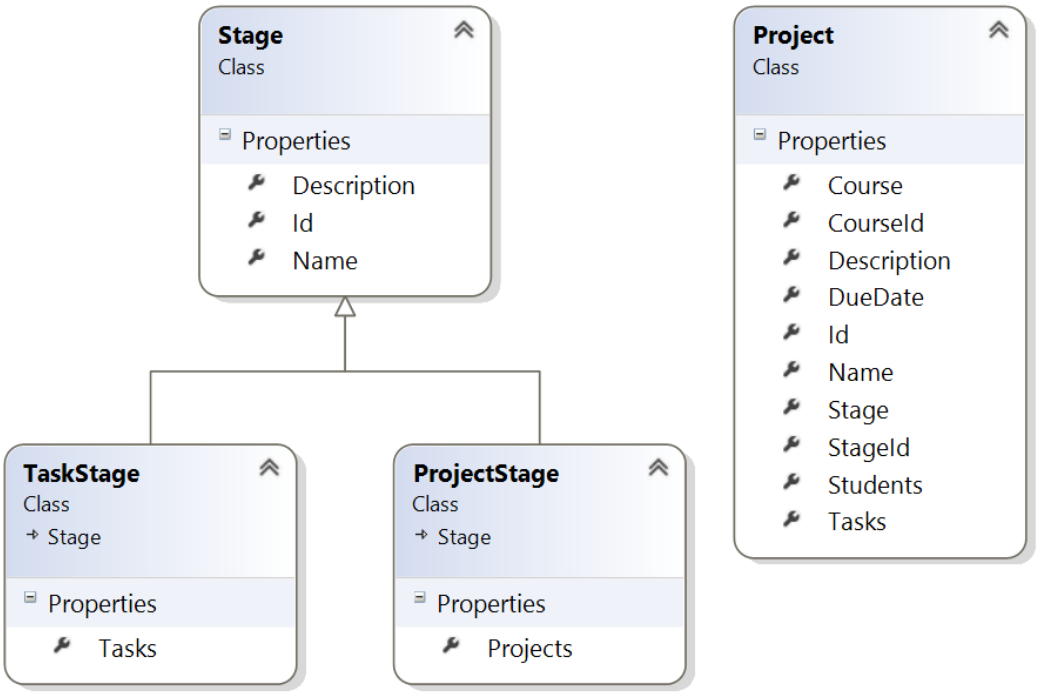
גם לפרויקט וגם למשימה בפרויקט יש "שלב שבו הם נמצאים", המכיל את שם השלב, את המזהה שלו ותיאור קצר אודותיו. לכן, ProjectStage ו-TaskStage יורשים ממחלקת "Stage".

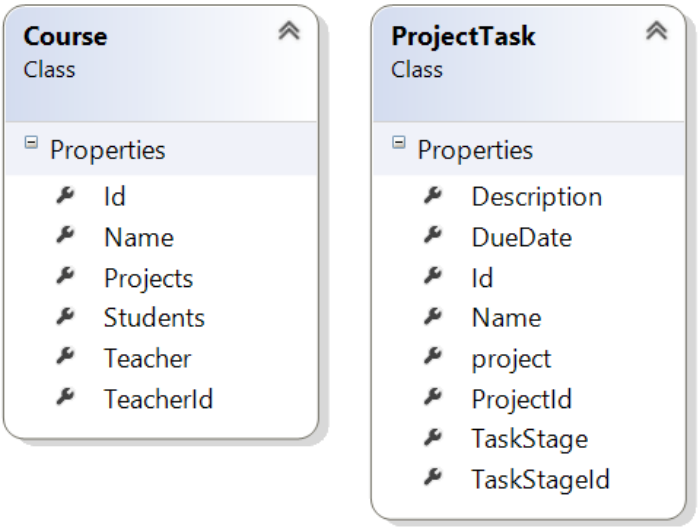
ProjectStage - מתארת את השלב בו הפרויקט נמצא. לכל שלב כזה קיימת רשימת פרויקטים המקושרים אליו.

TaskStage - מתארת את השלב בו המשימה בפרויקט נמצאת. לכל שלב כזה קיימת רשימת משימות המקושרות אליו.

מחלקת Course מתארת את הקורס השייך למספר תלמידים- לכן מכילה רשימת תלמידים, בהנהגת מורה אחד- לכן מכילה מורה אחד ומזהה אחד לו.

מחלקת ProjectTask מתארת משימה הקיימת בפרויקט, לכן מכילה קישור לפרויקט אליו היא קשורה. מלבד לכך, היא מכילה קישור לשלב שבו היא נמצאת- מסוג TaskStage





תיאור פרוטוקו

פרוטוקול מרכזי

פרוטוקול HTTP הוא הפרוטוקול המשמש להעברת דפי האינטרנט אליהם אנחנו גולשים בדפדפן.

בקשת GET

נועדה להביא פריט מידע כלשהו מהשרת באינטרנט שמסתתר מאחורי הכתובת.

ניתן לראות במתודה למעלה, המציגה את לוח ה"קנבן" של הקורס, שמדובר במתודת GET.

באופן כללי במודל MVC, ה-URL מתואר באופן הבא-

כך שאם הוא http://protracker/Courses/ShowProjectsKanban?courseId=1

אז אנו נקבל את מודל ה-View המקושר (באותו שם) לפעולה בשם ShowProjectsKanban שתקבל coursed=1, הנמצאת בתוך Controller בשם Course.

```
}
// GET: protracker/Courses/ShowProjectsKanban
0 references
public ActionResult ShowProjectsKanban(int courseId)
{
    using (ProTrackerContext db = new ProTrackerContext())
    {
        var stages = new List<ProjectStage>();
        stages = db.Stages.Include("Projects").OfType<ProjectStage>().ToList();

        foreach (var stage in stages)
        {
            stage.Projects = stage.Projects.Where(x => x.CourseId == courseId).ToList();
        }

        ViewBag.CourseId = courseId;

        return View(stages);
    }
}
```

בקשת POST

מאחר ואורכו של ה-URL מוגבל ל-2,000 תווים, משתמשים בבקשה מסוג POST לטפסים שמכילים מספר שדות, אותם המשתמש ממלא ואז לוחץ על כפתור לשליחת הטופס.

```
[HttpPost]
0 references
public ActionResult AddProject(Project project)
{
    try
    {
        if(!ModelState.IsValid)
            return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
        Course course = db.Courses.Find(project.CourseId);

        if (course != null)
        {
            db.Projects.Add(project);
            db.SaveChanges();

            project.Course = null;
            project.Stage = null;

            return Json(project);
        }
        //if course was not found
        return new HttpStatusCodeResult(HttpStatusCode.NotFound);
        //return null;
    }
    catch (Exception)
    {
        //return null;
        return new HttpStatusCodeResult(HttpStatusCode.InternalServerError);
    }
}
```

האלגוריתם המרכזי בקוד

מושגים הכרחיים להבנת האלגוריתם:

AJAX

בכל יישומי האינטרנט (Web Application) יש את צד הלקוח וצד השרת. ה"לקוח" הוא דפדפן אינטרנט, כמו Internet Explorer, והשרת הינו שרת יישומי אינטרנט במיקום מרוחק שיעבד בקשות אינטרנט וישלח תצוגה משולבת נתונים ללקוח. אולם, ליישומי האינטרנט או למסדי הנתונים אין חיבור חי וקבוע בינו לבין הדף המוצג בדפדפן הלקוח. רוב העיבוד ייעשה בשרת עצמו, ולא בדפדפן של הלקוח. כאשר מסד הנתונים ניגש לשרת, כך שכאשר יישום האינטרנט יהיה צריך לגשת למבנה הנתונים הוא יישלח את הדף לשרת והקוד הנמצא בשרת יטפל בדרישות ובקשות הלקוח.

החיסרון המרכזי של עיבוד בצד השרת הוא מהירות הביצועים- לקוח יצטרך לחכות עד עיבוד המידע והפעולות שמתרחשות בצד השרת ייגמרו, ורק אז יוכל לראות שיוניים בדף, לאחר שהשרת עיבד את הבקשה ושלח את הדף המעודכן ללקוח. לכן, משתמשים לרוב בתקשורת בין הצדדים רק כאשר הלקוח באמת צריך ליזום אותה על מנת להתקדם בשימוש באפליקציה.

לאור הקושי הברור, נולד הרעיון של Ajax- asynchronous JavaScript and XML

במסגרת הקונספט הזה, הלקוח פונה בצורה ישירה אל השרת ועובד עם הנתונים ישירות, כך שרק חלק מהתצוגה של הנתונים בדפדפן המשתמש משתנים בצורה אסינכרונית, ללא צורך לעדכן את הדף כולו. בקריאת Ajax הלקוח לא "ננעל" ומחכה לתשובה מהשרת ברגע שביצע בקשה, אלא יראה את התוצאה הסופית בדפדפן ברגע שהמשימה של השרת הושלמה.

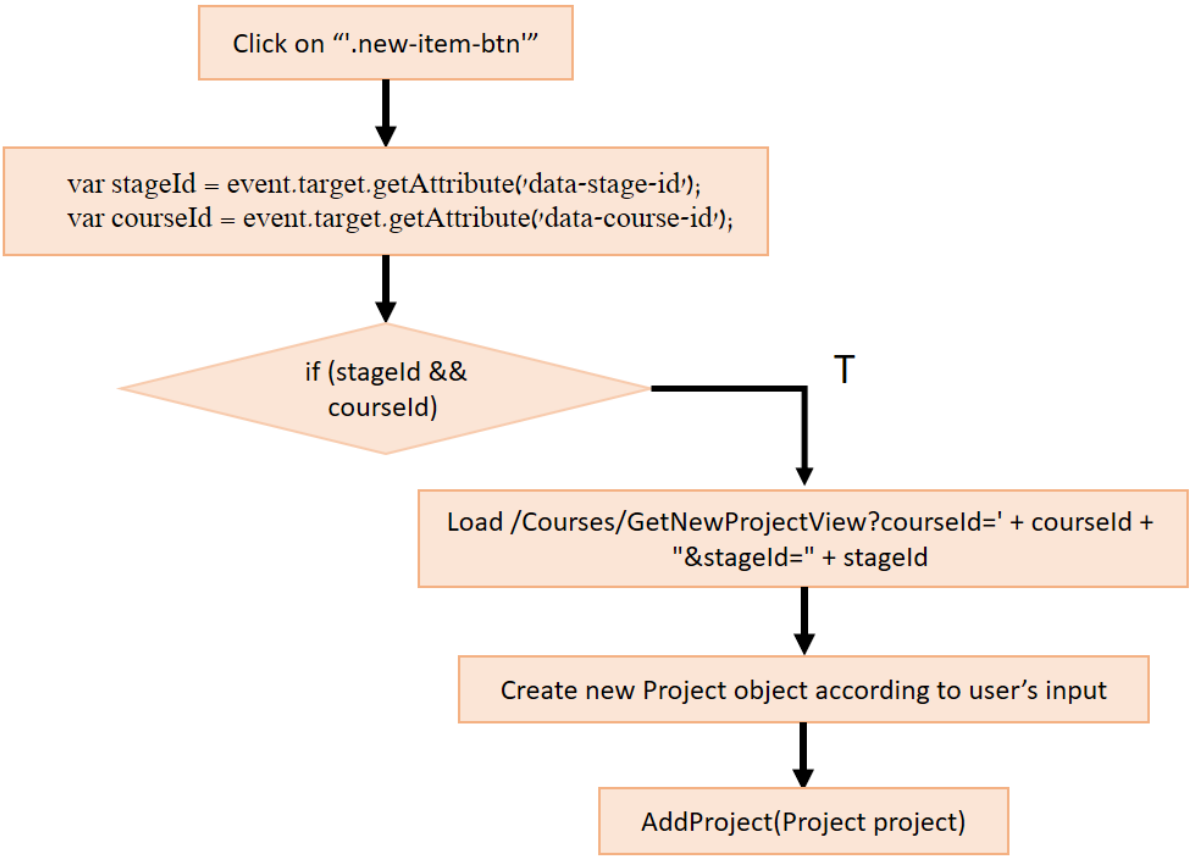
בנוסף, השתמשתי בתיקייה הנקראת jQuery, הנחשבת לפריצת דרך בכל הקשור לתכנות צד הלקוח. בתיקייה ישנם כלים שמאפשרים אנימציה, מעבר של אובייקטי HTML והופכים את השימוש ב-Ajax לנגיש יותר.

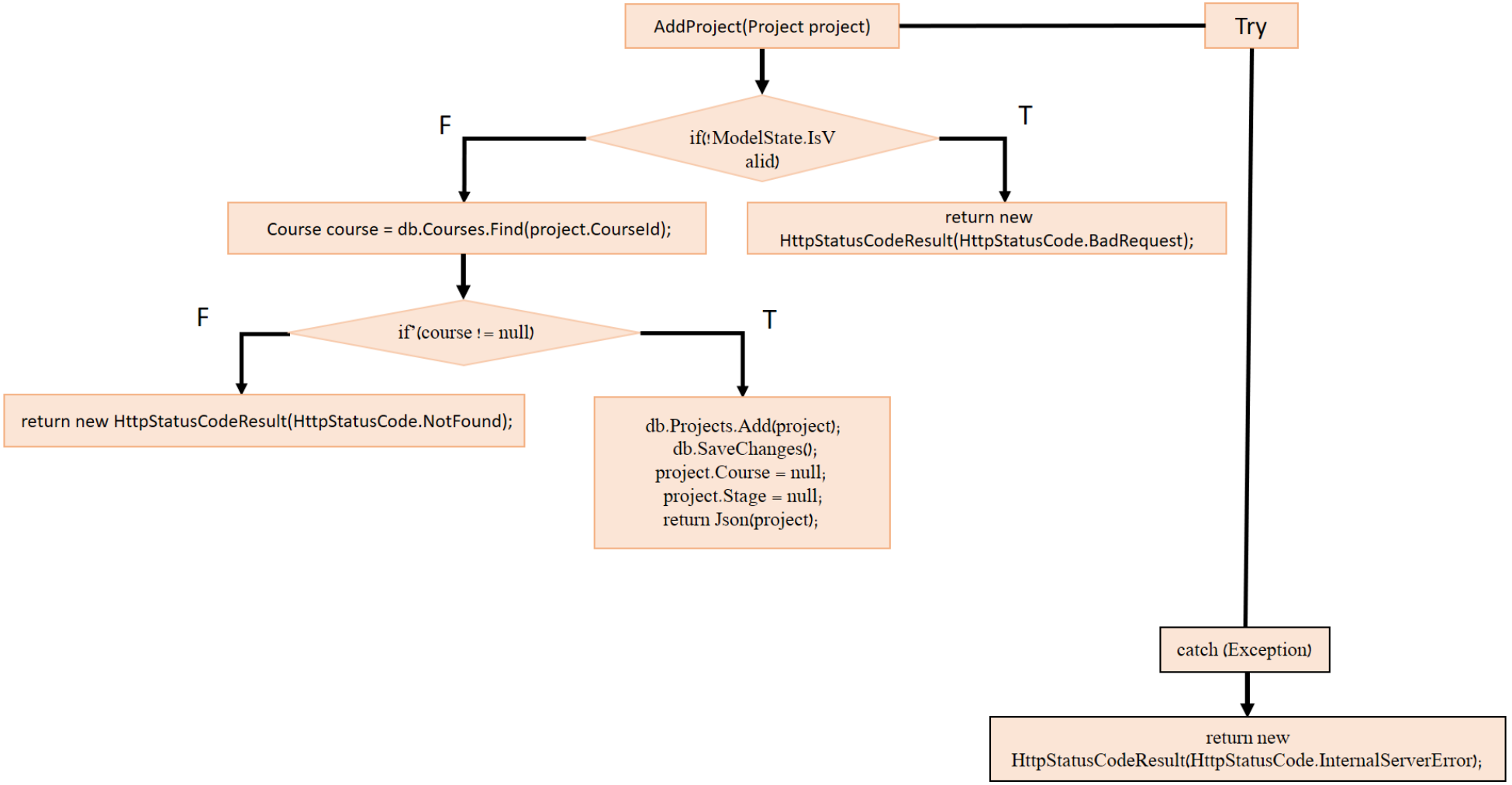
DRAG AND DROP

הכלי המרכזי עליו הפרויקט שלי מתבסס היא גרירת הכרטיסיות בין השלבים השונים בלוח ה"קנבן" – Drag and Drop.

הכלי מדמה הצבעה שבה המשתמש בוחר אובייקט וירטואלי על ידי "תופס" וגורר אותו למיקום אחר או על אובייקט וירטואלי אחר. באופן כללי, ניתן להשתמש בו ליצירת סוגים רבים של פעולות, או ליצור סוגי אסוציאציות רבים בין שני אובייקטים מופשטים.

לכאורה, מדובר בטכניקה פשוטה. אולם, יש צורך לדאוג מעבר לתצוגת הדף- לדאוג שהכרטיסייה לא "תיזרק" במקום רנדומלי שאינו שלב אחר, לעדכן את השלב של אותה משימה שהוזזה וכדומה.


Click on "'.new-item-btn'"
var stageId = event.target.getAttribute('data-stage-id');
var courseId = event.target.getAttribute('data-course-id');
if (stageId && courseId)
T
Load /Courses/GetNewProjectView?courseId=' + courseId + "&stageId=" + stageId
Create new Project object according to user's input
AddProject(Project project)


AddProject(Project project)
Try
if(!ModelState.IsV
alid)
F
T
Course course = db.Courses.Find(project.CourseId);
return new
HttpStatusCodeResult(HttpStatusCode.BadRequest);
if (course != null)
F
T
return new HttpStatusCodeResult(HttpStatusCode.NotFound);
db.Projects.Add(project);
db.SaveChanges();
project.Course = null;
project.Stage = null;
return Json(project);
catch (Exception)
return new
HttpStatusCodeResult(HttpStatusCode.InternalServerError);

```
public class Project
    //Project a teacher creates for one or more students
{
    [Key]
    [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
    7 references
    public int Id { get; set; }
    [Required, MaxLength(100)]
    4 references
    public string Name { get; set; }
    2 references
    public string Description { get; set; }
    [ForeignKey("CourseId")]
    [JsonIgnore]
    1 reference
    public virtual Course Course { get; set; }
    5 references
    public int CourseId { get; set; }
    [JsonIgnore]
    4 references
    public virtual ICollection<Student> Students { get; set; }
    [ForeignKey("StageId")]

    1 reference
    public ProjectStage Stage { get; set; }
    5 references
    public int StageId { get; set; }
    [JsonIgnore]
    2 references
    public virtual ICollection<ProjectTask> Tasks { get; set; }
    7 references
    public DateTime DueDate { get; set; }


}
```

```
namespace ProTracker.Models
{
   public class ApplicationUser : IdentityUser
    {
        [Required(ErrorMessage = "First name is needed."), MaxLength(50)]
        public string FirstName { get; set; }
        [Required(ErrorMessage = "Last name is needed."), MaxLength(50)]
        public string LastName { get; set; }
        public DateTime? DayOfBirth { get; set; }
        public bool IsAdmin { get; set; }
        public string PersonalID { get; set; }
        public Gender Gender { get; set; }
        public virtual ICollection<Course> Courses { get; set; }
        //Create claims
        public async Task<ClaimsIdentity>
GenerateUserIdentityAsync(UserManager<ApplicationUser> manager)
        {
            // Note the authenticationType must match the one defined in
CookieAuthenticationOptions.AuthenticationType
            var userIdentity = await manager.CreateIdentityAsync(this,
DefaultAuthenticationTypes.ApplicationCookie);
            // Add custom user claims here
            userIdentity.AddClaim(new Claim("Teacher",
this.IsAdmin.ToString()));
            userIdentity.AddClaim(new Claim("Id", this.Id));
            userIdentity.AddClaim(new Claim("FirstName", this.FirstName));
            return userIdentity;
        }
    }
    public enum Gender
    {
        Male,
        Female
  }
}
```

```
namespace ProTracker.Models
{
    public class Stage
    {
        [Key]
        [DatabaseGenerated(DatabaseGeneratedOption.Identity)]
        public int Id { get; set; }
        [Required, MaxLength(100)]
        public string Name { get; set; }
        [Required, MaxLength(150)]
        public string Description { get; set; }


    }
}
```

תיעוד קוד- קונטרולרים

```
public class CoursesController : Controller
    {
        private ProTrackerContext db = new ProTrackerContext();

        // GET: Courses
        public ActionResult Index()
        {
            var courses = db.Courses.Include(c => c.Teacher);
            return View(courses.ToList());
        }
        public ActionResult RemoveProjectFromCourse(int projectId, int
courseId)
        {
            var isTeacher = IdentityUtils.IsTeacher();
            if (isTeacher)
            {
                using (ProTrackerContext db = new ProTrackerContext())
                {
                    var project = db.Projects.FirstOrDefault(t => t.Id ==
projectId);
                    var course = db.Courses.FirstOrDefault(t => t.Id ==
courseId);

                    if (project != null && course != null)
                    {
                        course.Projects.Remove(project);
                        db.SaveChanges();
                    }
                }
                return View();
            }
            return View();
        }
        // GET: protracker/Courses/ShowProjectsKanban
        public ActionResult ShowProjectsKanban(int courseId)
        {
            using (ProTrackerContext db = new ProTrackerContext())
            {
                var stages = new List<ProjectStage>();
                stages =
db.Stages.Include("Projects").OfType<ProjectStage>().ToList();

                foreach (var stage in stages)
                {
                    stage.Projects = stage.Projects.Where(x => x.CourseId ==
courseId).ToList();
                }

                ViewBag.CourseId = courseId;

                return View(stages);
            }
        }
```

המשך בעמוד הבא

```
        [HttpPost]
        public ActionResult AddProject(Project project)
        {
            try
            {
                if(!ModelState.IsValid)
                    return new HttpStatusCodeResult(HttpStatusCode.BadRequest);
                Course course = db.Courses.Find(project.CourseId);

                if (course != null)
                {
                    db.Projects.Add(project);
                    db.SaveChanges();

                    project.Course = null;
                    project.Stage = null;

                    return Json(project);
                }
                //if course was not found
                return new HttpStatusCodeResult(HttpStatusCode.NotFound);
                //return null;
            }
            catch (Exception)
            {
                //return null;
                return new
HttpStatusCodeResult(HttpStatusCode.InternalServerError);
            }
        }
```

```
namespace ProTracker.Controllers
{
    [Authorize]
    public class MenuController : Controller
    {
        [ChildActionOnly]
        public ActionResult UserMenu()
        {
            using (ProTrackerContext db = new ProTrackerContext())
            {
                var courses = new List<Course>();
                var userId = IdentityUtils.GetUserId();

                if (IdentityUtils.IsTeacher())
                {

                    courses = db.Courses.Where(c => c.TeacherId ==
userId).ToList();
                }
                else
                {
                    courses = db.Courses.Include("Students").Where(c =>
c.Students.Any(s => s.Id == userId)).ToList();
                }
                return PartialView("~/Views/Shared/_MenuPartial.cshtml",
courses);
            }
        }

        [HttpPost]
        [ValidateAntiForgeryToken]
        public ActionResult AddNewCourse(Course course)
        {
            //add new course
            using (ProTrackerContext db = new ProTrackerContext())
            {
                var userId = IdentityUtils.GetUserId();
                course.TeacherId = userId;
                if (!db.Courses.Any(c => c.Name == course.Name))
                    {
                    db.Courses.Add(course);
                    db.SaveChanges();
                    }
                // else return view s
            }

            return UserMenu();
        }
    }
                                                                                }
```

app.js Script –תיעוד קוד

```
//when the user starts dragging of the object.
function dragStart(ev) {
    ev.dataTransfer.effectAllowed = 'move';
    ev.dataTransfer.setData("Text", ev.target.getAttribute('id'));
    ev.dataTransfer.setDragImage(ev.target, 0, 0);
    return true;

}
//when the mouse is first moved over the target element while a drag is
occuring.
function dragEnter(ev) {
    event.preventDefault();
    return true;
}
//when the mouse is moved over an element when a drag is occuring.
function dragOver(ev) {
    return false;
}
//After drop happened
function dragDrop(ev, stageId) {
    var projectId = ev.dataTransfer.getData("Text");

    $.ajax({
        method: "PUT",
        data: null,
        url: "/ProjectStage/AddProjectToStage?stageId=" + stageId +
"&projectId=" + projectId,
        success: function (AddProjectToStage) {
            ev.target.appendChild(document.getElementById(projectId));
        }
    });

    ev.stopPropagation();
}


//ready function
$(document).ready(function () {
    // // Attach listener to .modal-close-btn's so that when the button is
pressed the modal disappears
    $(document).on("click", "#newCourseForm", function (e) {
        $('#newCourseForm').modal('hide')
    });

    $('.new-item-btn').on('click', function (event) {

        var stageId = event.target.getAttribute('data-stage-id');
        var courseId = event.target.getAttribute('data-course-id');

        if (stageId && courseId) {
```

```
$(event.target.nextElementSibling).load('/Courses/GetNewProjectView?courseId='
+ courseId + "&stageId=" + stageId, function () {
            });
            $(this).hide();
        }
        //else
        //    $("#element").html(''); //clearing the innerHTML if checkbox is
unchecked
    });
    //new course
    $("#newCourseModal").on("submit", "#newCourseForm", function (e) {
        e.preventDefault();  // prevent standard form submission
        var form = $(this);
        $.ajax({
            url: form.attr("action"),
            method: form.attr("method"),  // post
            data: form.serialize(),
            success: function (partialResult) {
                $("#userMenu").html(partialResult);
                $('.modal-backdrop').remove();
            }
        });
    });
    //new project to kanban
    $(".new-item-partial").on("submit", "#newProjectForm", function (e) {
        e.preventDefault();  // prevent standard form submission
        var form = $(this);
        var items = form.parents('.kanban-bucket').find('.kanban-items');
        $.ajax({
            url: form.attr("action"),
            method: form.attr("method"),  // post
            data: form.serialize(),
            success: function (project) {
                var newProjectDiv = $('<div draggable="true"
ondragstart="return dragStart(event)" Name="Item" id="' + project.Id + '"
class="kanban-item"> \
                            <span> \
                                ' + project.Name + ' \
                            </span> \
                        </div>');
                $(items[0]).append(newProjectDiv);
                $(form).parents('.new-item').find('.new-item-btn').show()
                $(form).html('');
            }
        });
    });
    //clear modal cache, so that new content can be loaded
    $('#newCourseModal').on('hidden.bs.modal', function () {
        $(this).removeData('bs.modal');
    });

    $('#newCourseModal').on('hidden.bs.modal', function () {
        $(this).find("input,textarea,select").val('').end();

    });

                                                                        });
```

```
@{
//ViewBag.Title = "ShowProjectsKanban";
  //Layout = "~/Views/Shared/_Layout.cshtml";
    @model List<ProTracker.Models.ProjectStage>
}
<h1 class="header">
    Italian Course's Projects
</h1>
<body   background="//protracker/Photos/Kanban_Background.jpg">
</body>
        <div Name="Buckets" class="kanban-buckets">
            @foreach (var projectStage in Model)
            {
                <div name="@projectStage.Id" class="kanban-bucket"
ondragenter="return dragEnter(event)"
                     ondrop="return dragDrop(event, @projectStage.Id)"
                     ondragover="return dragOver(event)">
                    <div Name="Header" class="kanban-header">
                        <span>
                            @projectStage.Name
                        </span>
                    </div>
                    <div name="items" class="kanban-items">
                        @foreach (var project in projectStage.Projects)
                        {
                            <div draggable="true" ondragstart="return
dragStart(event)" Name="Item" id="@project.Id" class="kanban-item">
                                <span>
                                    @Html.ActionLink(@project.Name,
"ShowKanbanBoard", "Project", new { projectId = @project.Id }, new { })
                                </span>
                                <span class="date">
                                    @project.DueDate.Day .
@project.DueDate.Month . @project.DueDate.Year
                                    <span data-toggle="modal" data-
target="#removeProjectModal" class="glyphicon glyphicon-remove"></span>
                                </span>
                            </div>
                        }

                    </div>
                    <div class="new-item">
                        <div class="adding-btn">
                            <span class="new-item-btn" data-course-
id="@ViewBag.CourseId" data-stage-id="@projectStage.Id">
                                Create new project
                            </span>
                        </div>
                        <div class="new-item-partial"></div>
                    </div>
                </div>
            }

        </div>
```

```
<div id="removeProjectModal" class="modal fade" role="dialog">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
                <button type="button" class="close" data-
dismiss="modal">&times;</button>
                <h4 class="modal-title">Are you sure you want to delete this
project?</h4>
            </div>
            <div class="modal-body">
                @Html.Partial("_DeleteProjectPartial", new
ProTracker.Models.Project())
            </div>
        </div>
    </div>
</div>
```

הצעות לשיפור:

- הוספת צ׳אט בכל קנבן של קורס כדי להעשיר את תרומת האתר להספק העבודה בשוטף
- הוספת פונקציה של שליחת אימייל לקבוצת התלמידים שמשימה אחת או יותר שלהם בפיגור, באמצעות הפרוטוקול SMTP
- לשלב את המשתמשים עם פרופילי הרשתות חברתיות כמו Facebook , שכן Identity מאפשר זאת, על מנת לעודד תלמידים צעירים להשתמש באתר.

ביבליוגרפיה ותודות מיוחדות:

ספר "רשתות" גבהים
http://data.cyber.org.il/assembly/gvahim_assembly_book.pdf

ההבדל בין AJAX לתקשורת לקוח-שרת

/http://www.seguetech.com/client-server-side-code

מדריך Drag and Drop

https://www.tutorialspoint.com/html5/html5_drag_drop.htm

מדריכי שפות

/https://www.w3schools.com

פורום- פתרון בעיות נקודתיות

/https://stackoverflow.com

חשוב לי להדגיש שלא הייתי יכולה לעשות את הפרויקט ללא עזרתו של גיסי, חיים. הוא סייע לי מגיבוש הרעיון הראשוני ועד הפרטים הקטנים ביותר. יותר מכל, הוא לימד אותי כיצד לגשת לשגיאות בצורה נכונה- לא להירתע, אלא לפרק את תוכן השגיאה לפרטי מידע שיסייעו לי לחפש לה פתרון באינטרנט.

פעמים רבות רציתי "להרים ידיים" משום שלא הצלחתי ונתקלתי ביותר מידי מכשולים בכתיבת הפרויקט. בזכות חיים, היום אני כבר לא "מפחדת" כשאני רואה שגיאה אלא מנסה להתמודד אתה עם כל הכלים שברשותי. את השיעור שחיים לימד אותי אני לוקחת גם לחיים עצמם- כפי שחיים אמר לי באחת מהפעמים שנפגשנו לעבוד על הפרויקט- "יש רצון, אין פחד"