

עבודת גמר

בתכנון ותכנות מערכות

במסלול "שירותי רשת" 2017

שם התלמיד: אביב בואנו

ת.ז: 314711045

שם המורה: מאיר דהאן

בית הספר: שש שנתי נחשון מ.א. חבל מודיעין

# תודות

ראשית ברצוני להודות לחבר, רום גביש, על התמיכה והעזרה בבניית ממשק יפה ונוח לשימוש על פי אדיאולוגיות עיצוביות של גוגל - 'מטריאל'.

כמו כן, אני מודה לאמי אשר עזרה לי באפיון המערכת ועזרה בפתירת בעיות אלגוריתמיות.

תודה אחרונה וחשובה, למשפחתי המדהימה אשר סיפקה עבורי סביבת למידה תומכת מועדדת ושקטה.

# תוכן עניינים

תודות..............................................................................................................................................2

תוכן עניינים .....................................................................................................................................3 מבוא...............................................................................................................................................4

תאור הפרויקט..................................................................................................................................4

מטרות הפרויקט ...............................................................................................................................5

קהל יעד ..........................................................................................................................................5

מבנה בסיס הנתונים.......................................... ................................................................................6

טבלאות מתוך מסד הנתונים ...............................................................................................................6

רשימת טבלאות: ...................................... ........................................................................................6

קשרי גומלין מתוך מסד הנתוני(DSD) :.......................................................................................................6

מבנה התיקיות: ........................................ ........................................................................................6

המחלקות: .............................................. ........................................................................................6

תרשימי UML: ......................................... ........................................................................................6

DAL- Data Access Layer: …………............... ....................................................................................6

BI – Business Intelligence: ....................... ........................................................................................6

פירוט חתימות ותוכן המחלקות:................... ........................................................................................6

רשימת טבלאות: ...................................... ........................................................................................6

רשימת טבלאות: ...................................... ........................................................................................6

רשימת טבלאות: ...................................... ........................................................................................6

רשימת טבלאות: ...................................... ........................................................................................6

רשימת טבלאות: ...................................... ........................................................................................6

רשימת טבלאות: ...................................... ........................................................................................6

# מבוא

**תאור הפרויקט**

מערכת לניהול בתי ספר אינטרנטית, המאפשרת מעבר מהיר של מידע בין אנשי הצוות בבתי ספר והמורים. המערכת נותנת לתלמיד דוחות על פעילותו הלימודית. המערכת מאפשרת למורה להזין ציונים והערות משמעת. בנוסף לכך יכול המורה לקבל דוחות על תלמיד באופן פרטני על פעילותו בשיעורים ובמבחנים על מנת למנף את הישגיו הלימודים. המורה יכול גם לקבל דוח כיתתי על מנת לראות את מצב הכיתה במאקרו. המורים והתלמידים מקבלים מערכת אישית היכולה להציג שינויי מערכת פרטניים אליהם, לדוגמא, במידה והמורה דני הודיע להנהלה כי אינו יכול להגיע ביום מסויים יופיע לכל תלמידיו ביטול שיעור במערכת. סגל ההנהלה של בית הספר יכול להוסיף אנשי סגל ותלמידים, לייצור כיתות, להוסיף ו/או למחוק שיעורים, לשנות שיעורים לתאריך מסוים(ביטול, מילויי מקום ועוד), בנוסף סגל ההנהלה יכול לערוך את כרטיס התאמות של התלמיד.

**מטרות הפרויקט**

* הפקת דוחות נתוני לימוד לתלמידים.
* הפקת דוחות נתוני לימוד של התלמידים למורים.
* הפקת דוחות כניסה למערכת לסגל ההנהלה.
* ניהול מערכת שעות יותר נוח ונגיש לבתי ספר.
* ניהול התאמות התלמידים.
* נגישות המורה לתלמידים, למורה קיימת אפשרות חיפוש תלמידים אשר הוא מלמד.
* נגישות סגל ההנהלה לכל הסגל והתלמידים, לסגל ההנהלה קיימת אפשרות חיפוש אנשי סגל ותלמידים.

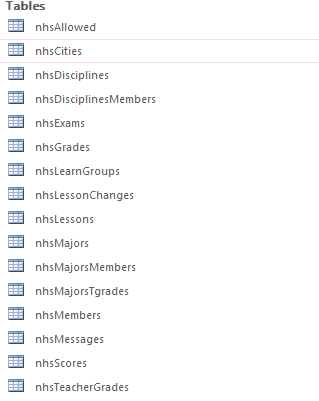
**קהל הייעד**

**מנהל –** סגל ההנהלה הבית ספרי, מזכירות ומנהלים/מנהלות בית ספר.

**מורה –** כל מורה בבית הספר.

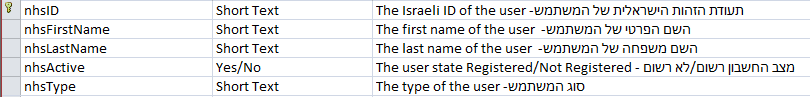
**תלמיד –** כל תלמיד בבית הספר.

# מבנה מסד הנתונים



**רשימת טבלאות**

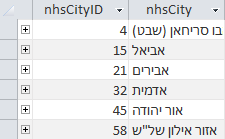
**טבלת מוזמנים להרשמה (nhsAllowed)**



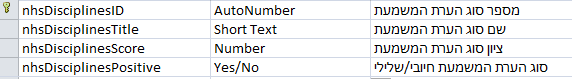


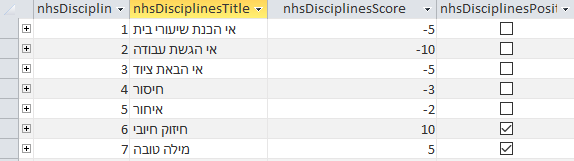
**טבלת ערים(nhsCities)**





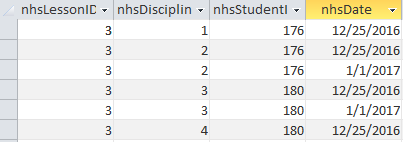
**טבלת סוגי הערות משמעת (nhsDisciplines)**



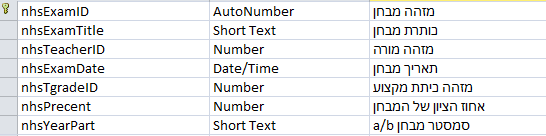


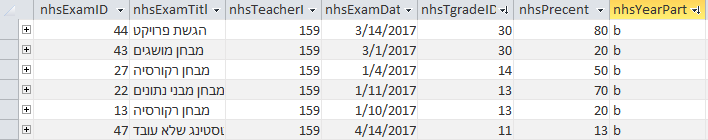
**טבלת סוגי הערות משמעת – משתמשים (nhsDisciplinesMembers)**





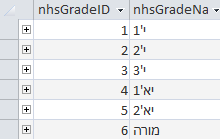
**טבלת מבחנים(nhsExams)**





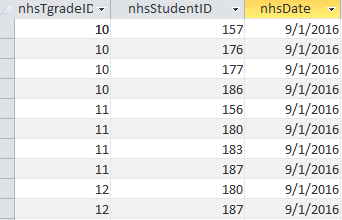
**טבלת כיתות אם (nhsGrades)**



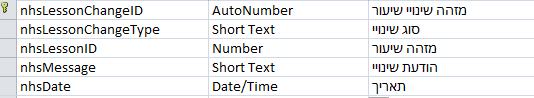


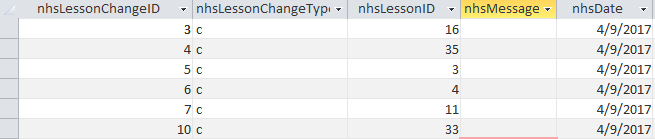
**טבלת כיתות מקצוע – תלמיד (nhsLearnGroups)**



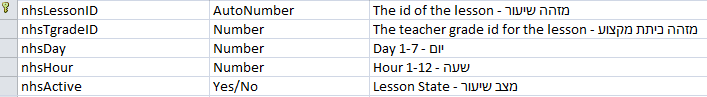


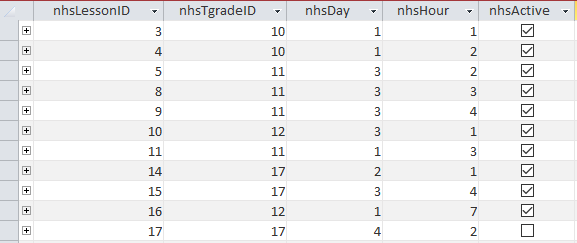
**טבלת שינויים בשיעורים(nhsLessonChanges)**





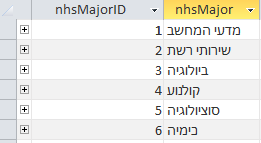
**טבלת שיעורים (nhsLessons)**





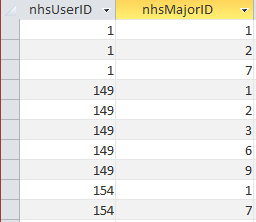
**טבלת מגמות (nhsMajors)**





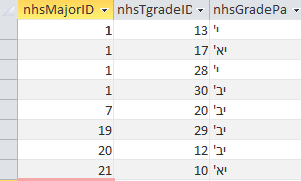
**טבלת מגמות - משתמשים (nhsMajorsMembers)**

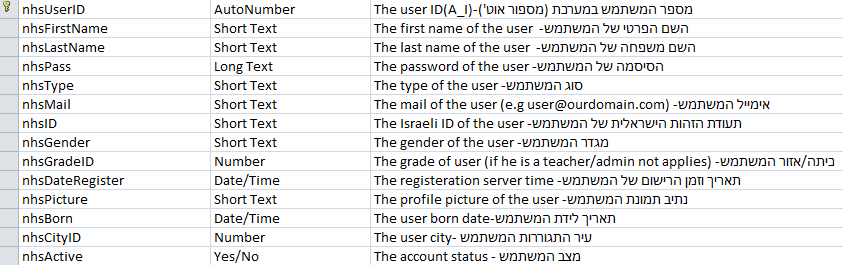


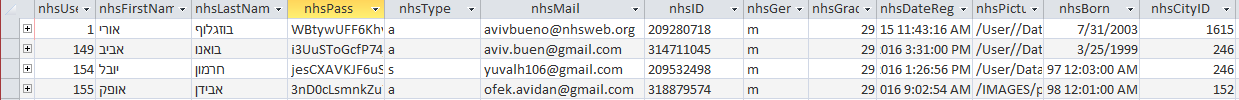


**טבלת מגמות – כיתות מקצוע(nhsMajorsTgrades)**

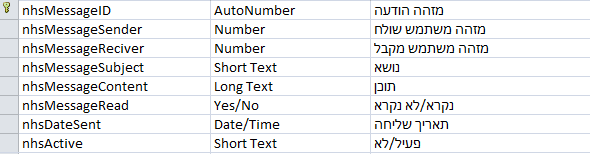


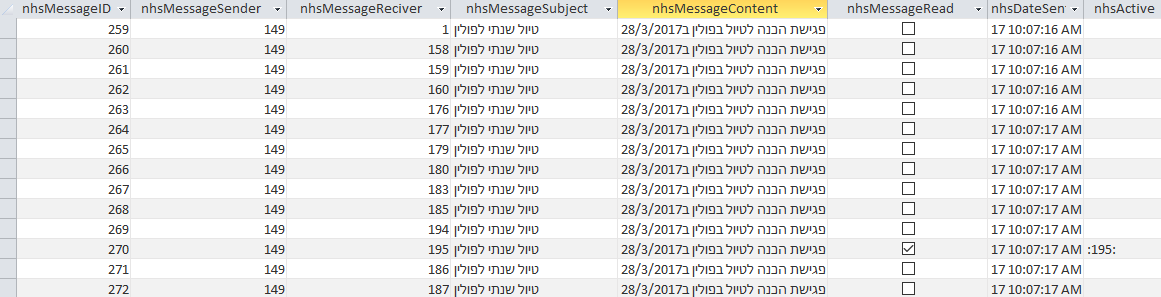


**טבלת משתמשים (nhsMembers)**

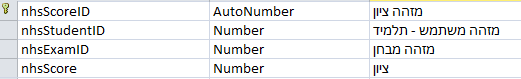


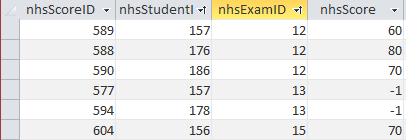
**טבלת הודעות (nhsMessages)**



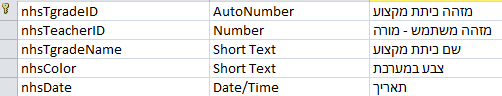


**טבלת ציונים (nhsScores)**

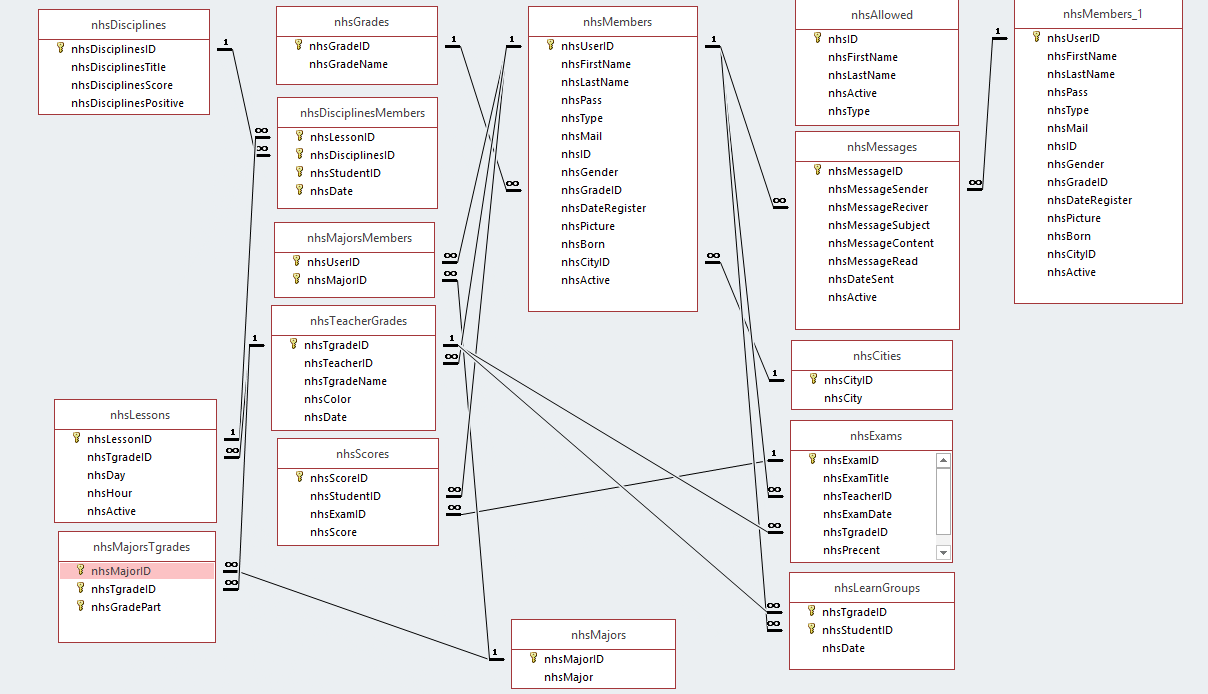




**טבלת כיתות מקצוע (nhsTeacherGrades)**



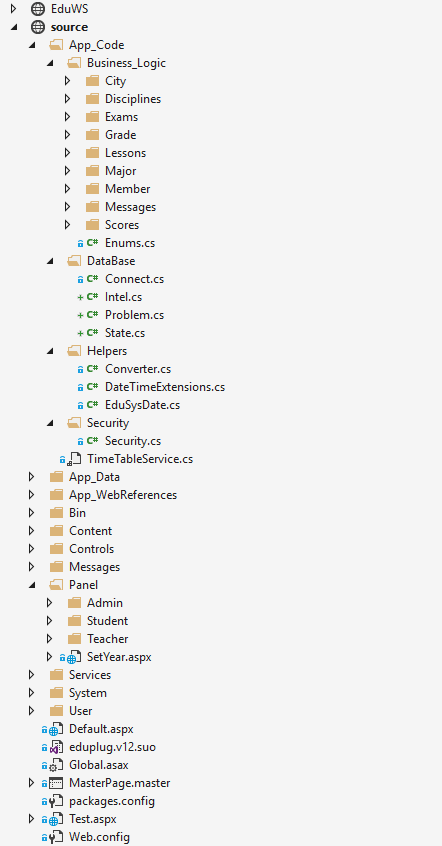


**קשרי**

**גומלין**

# מבנה התיקיות

שירות רשת להתאמות



Encryption layer

אחראית להצפנה חזקה של סיסמאות המשתמשים טרם כניסתם למסד הנתונים

BI – Business Intelligence layer

אחראית לכל הלוגיקה של המערכת

DAL – Data Access Layer

אחראית לתקשורת הלוגיקה עם מסדי הנתונים.

Web Interface

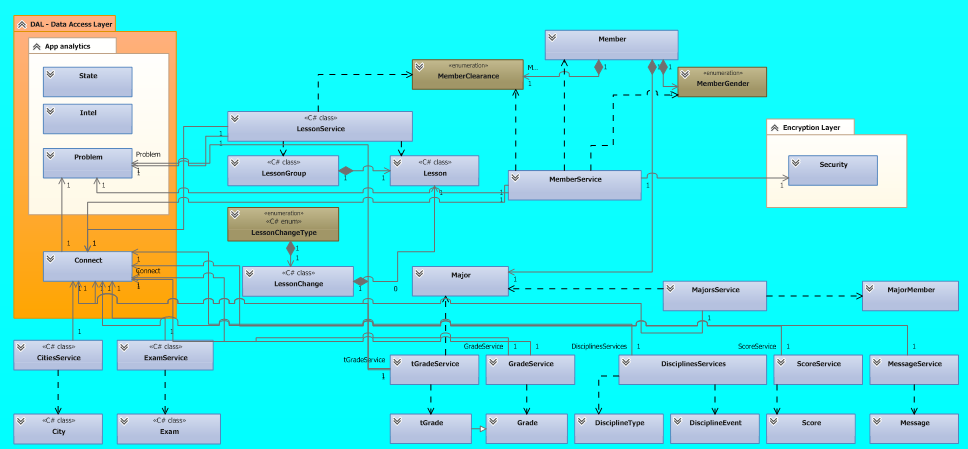
ממשק הWeb של הראשי של המערכת, מחולק לשלושה סוגי משתמשים

Web Interface

ממשק הכניסה/הרשמה/איפוס סיסמה

# המחלקות

**UML(tiny.cc/uml\_eduplug(:**



**DAL – Data Access Layer:**

Connect.cs :

/// <summary>

/// This class is connecting the application to the database,

/// mainly used in the business logic classes.

/// </summary>

public static class Connect

{

/// <summary>

/// Counts the queries that excuted for the page loaded

/// </summary>

public static int QueriesCount;

/// <summary>

/// Return's Result For The SQL Query(DataTable Object)

/// </summary>

public static DataTable GetData(string SqlQuery, string TableName)

/// <summary>

/// Return's Result For The SQL Query(DataSet Object)

/// </summary>

public static DataSet GetData(string SqlQuery, string TableName, bool WantFullDataSet)

/// <summary>

/// Return's Object Result (It's Mainly For Mathmatical Use)

/// </summary>

public static object GetObject(string SqlQuery)

/// <summary>

/// Executes The SqlQuery And Return's Boolean For Success Of The Query Execution

/// </summary>

public static bool InsertUpdateDelete(string SqlQuery)

/// <summary>

/// Executes The SqlQuery And Return's Boolean For Success Of The Query Execution

/// </summary>

public static int InsertUpdateDeleteState(string SqlQuery)

}

Intel.cs :

/// <summary>

/// Gathers intelligence on client and system, also provides help to UI

/// </summary>

public static class Intel

{

/// <summary>

/// Returns the project's root url in order to use the master page in all the web directories

/// </summary>

/// <returns>Project's root url</returns>

public static string GetFullRootUrl()

/// <summary>

/// Redirects the user to his panel

/// </summary>

public static void Redirect()

/// <summary>

/// Gets the ip address of the client

/// </summary>

/// <returns>String - IP Address of client</returns>

public static string GetIpAddress()

/// <summary>

/// Mark an entry as reviewed

/// </summary>

/// <param name="ip">IP of entry</param>

/// <param name="state">Rev/NoRev</param>

/// <returns>success</returns>

public static bool MarkRev(string ip,bool state)

/// <summary>

/// Gets all the entries

/// </summary>

/// <returns>All the entries</returns>

public static DataTable GetEduSense()

/// <summary>

/// Gets the user device OS

/// </summary>

/// <returns>Client device OS</returns>

public static string GetUserPlatform()

/// <summary>

/// Get the current device family of the client

/// </summary>

/// <returns>Client Device Family</returns>

public static string GetDeviceFamily()

/// <summary>

/// Gets the version of the mobile OS of client

/// </summary>

/// <param name="userAgent">The user agent of the request</param>

/// <param name="device">The device of the client</param>

/// <returns>The mobile version of the device</returns>

public static string GetMobileVersion(string userAgent, string device)

/// <summary>

/// Delete entry

/// </summary>

/// <param name="ip">IP</param>

/// <returns>success</returns>

public static bool DeleteVisit(string ip)

/// <summary>

/// Saves the visitor entry in the DB

/// </summary>

public static void SaveVisit()

/// <summary>

/// Uses an external API to resolve the current client country by the ip (The API - IP-API.COM)

/// </summary>

/// <returns>Client country</returns>

public static string GetCountry()

/// <summary>

/// Uses an external API to resolve the current client ISP(Internet Service Provider) by the ip (The API - IP-API.COM)

/// </summary>

/// <returns>Client ISP</returns>

public static string GetISP()

/// <summary>

/// Gets entries locations for graph

/// </summary>

/// <returns>Dictionary(string, object) of locations</returns>

public static Dictionary<string, object> GetLocations()

}

Problem.cs :

/// <summary>

/// Logging errors and track to txt files

/// </summary>

public static class Problem

{

/// <summary>

/// Log an Exception

/// </summary>

/// <param name="ex">Exception occured in system</param>

public static void Log(Exception ex)

/// <summary>

/// Save the error to app log

/// </summary>

/// <param name="error">Error content</param>

private static void AddLogApp(string error)

/// <summary>

/// Save the error to DB log

/// </summary>

/// <param name="error">Error content</param>

private static void AddLogDB(string error)

/// <summary>

/// Save the error to UI log

/// </summary>

/// <param name="error">Error content</param>

private static void AddLogUI(string error)

/// <summary>

/// Logs to file

/// </summary>

/// <param name="logMessage">Log Message</param>

/// <param name="w">TextWriter/StreamWriter</param>

public static void Log(string logMessage, TextWriter w)

}

State.cs :

/// <summary>

/// Deals with txt log

/// </summary>

public static class State

{

/// <summary>

/// Logs to file

/// </summary>

/// <param name="logMessage">Log Message</param>

/// <param name="w">TextWriter/StreamWriter</param>

private static void Log(string logMessage, TextWriter w)

}

**BI – Business Intelligence:**

**Models:**

City.cs :

/// <summary>

/// City Database Structure

/// </summary>

public class City

{

/// <summary>

/// The id of the city

/// </summary>

public int ID { get; set; }

/// <summary>

/// The name of the city

/// </summary>

public string Name { get; set; }

}

DisciplineEvent.cs :

/// <summary>

/// Discipline Event

/// </summary>

public class DisciplineEvent

{

/// <summary>

/// The id of the lesson

/// </summary>

public int LessonID { get; set; }

/// <summary>

/// The id of the disciplines type

/// </summary>

public int DisciplinesID { get; set; }

/// <summary>

/// The id of the student

/// </summary>

public int StudentID { get; set; }

/// <summary>

/// The date of the event

/// </summary>

public DateTime Date { get; set; }

}

DisciplineType.cs :

/// <summary>

/// Discipline Type

/// </summary>

public class DisciplineType

{

/// <summary>

/// The name of the type

/// </summary>

public string Name { get; set; }

/// <summary>

/// The id of the type

/// </summary>

public int ID { get; set; }

/// <summary>

/// The score

/// </summary>

public int Score { get; set; }

}

Exam.cs :

/// <summary>

/// Exam

/// </summary>

public class Exam

{

/// <summary>

/// The id of the exam

/// </summary>

public int ID { get; set; }

/// <summary>

/// The title of the exam

/// </summary>

public string Title { get; set; }

/// <summary>

/// The id of the teacher that conducts this exam

/// </summary>

public int TeacherID { get; set; }

/// <summary>

/// The date of the exam

/// </summary>

public DateTime Date { get; set; }

/// <summary>

/// The precent of the exam

/// </summary>

public int Precent { get; set; }

/// <summary>

/// The teacher grade id of exam

/// </summary>

public int TeacherGradeID { get; set; }

}

Grade.cs :

/// <summary>

/// Grade Structure

/// </summary>

public class Grade

{

/// <summary>

/// The id of the grade

/// </summary>

public int ID { get; set; }

/// <summary>

/// The name of the grade

/// </summary>

public string Name { get; set; }

}

tGrade.cs :

/// <summary>

/// Teacher Grade Structure

/// </summary>

public class tGrade : Grade

{

public int TeacherID { get; set; }

}

Lesson.cs :

/// <summary>

/// Lesson

/// </summary>

public class Lesson

{

/// <summary>

/// The color in the table

/// </summary>

public string Color { get; set; }

/// <summary>

/// The id of the lesson

/// </summary>

public int ID { get; set; }

/// <summary>

/// The hour of the lesson 1-12

/// </summary>

public int Hour { get; set; }

/// <summary>

/// The day of the lesson 1-6

/// </summary>

public int Day { get; set; }

/// <summary>

/// The id of the teacher

/// </summary>

public int TeacherID { get; set; }

/// <summary>

/// The id of the teacher grade

/// </summary>

public int TeacherGradeID { get; set; }

/// <summary>

/// The name of the grade

/// </summary>

public string Name { get; set; }

/// <summary>

/// The changes in the system for the lesson

/// </summary>

public List<LessonChange> Changes { get; set; }

}

LessonChange.cs :

/// <summary>

/// Lesson Change

/// </summary>

public class LessonChange

{

/// <summary>

/// The id of the lesson change

/// </summary>

public int ID { get; set; }

/// <summary>

/// The id of the lesson

/// </summary>

public int LessonID { get; set; }

/// <summary>

/// The date of the change

/// </summary>

public DateTime Date { get; set; }

/// <summary>

/// The change type

/// </summary>

public LessonChangeType ChangeType { get; set; }

/// <summary>

/// The message for the change

/// </summary>

public string Message { get; set; }

}

LessonGroup.cs :

/// <summary>

/// LessonGroup

/// </summary>

public class LessonGroup

{

/// <summary>

/// The lessons

/// </summary>

public List<Lesson> lessons;

/// <summary>

/// CTOR

/// </summary>

public LessonGroup()

}

Major.cs :

/// <summary>

/// Major structure

/// </summary>

public class Major

{

/// <summary>

/// The id of the major

/// </summary>

public int ID { get; set; }

/// <summary>

/// The name of the major

/// </summary>

public string Title { get; set; }

}

MajorMember.cs :

/// <summary>

/// Major Member

/// </summary>

public class MajorMember

{

/// <summary>

/// The user id of the user

/// </summary>

public int UserID { get; set; }

/// <summary>

/// The major id

/// </summary>

public int MajorID { get; set; }

}

Member.cs :

/// <summary>

/// Member structure for business logic

/// </summary>

public class Member

{

//GET/SET

/// <summary>

/// The DB id of the user (A\_I-PK) - DO NOT ENTER YOURSELF

/// </summary>

public int UserID { get; set; }

/// <summary>

/// The israeli/passport id of user

/// </summary>

public string ID { get; set; }

/// <summary>

/// The first name of the user

/// </summary>

public string FirstName { get; set; }

/// <summary>

/// The last name of the user

/// </summary>

public string LastName { get; set; }

public string Name

{

get

{

return this.FirstName + ' ' + this.LastName;

}

set

{

string[] val = value.Split(' ');

if (val.Length == 2)

{

FirstName = val[0];

LastName = val[1];

}

}

}

/// <summary>

/// The E-Mail of the user

/// </summary>

public string Mail { get; set; }

/// <summary>

/// The type of the user - The auth(Admin/Student/Teacher)

/// </summary>

public MemberClearance Auth { get; set; }

/// <summary>

/// The gender of the user

/// </summary>

public MemberGender Gender { get; set; }

/// <summary>

/// The date that the user was born in

/// </summary>

public DateTime BornDate { get; set; }

/// <summary>

/// The majors of the user(Computer Science, Movie Director, etc.)

/// </summary>

public List<Major> Majors { get; set; }

/// <summary>

/// The registeration time of the user

/// </summary>

public DateTime RegisterationDate { get; set; }

/// <summary>

/// The path that the user profile picture is located in

/// </summary>

public string PicturePath { get; set; }

/// <summary>

/// The id of the user grade

/// </summary>

public int GradeID { get; set; }

/// <summary>

/// The grade of the user

/// </summary>

public Grade Grade { get; set; }

/// <summary>

/// The city of the user

/// </summary>

public City City { get; set; }

/// <summary>

/// The user state active/not active

/// </summary>

public bool Active { get; set; }

//END GET/SET

}

Message.cs :

/// <summary>

/// City Database Structure

/// </summary>

public class Message

{

/// <summary>

/// The id of the message

/// </summary>

public int ID { get; set; }

/// <summary>

/// The id of the reciver

/// </summary>

public int ReciverID { get; set; }

/// <summary>

/// The id of the sender

/// </summary>

public int SenderID { get; set; }

/// <summary>

/// The subject of the message

/// </summary>

public string Subject { get; set; }

/// <summary>

/// The content of the message

/// </summary>

public string Content { get; set; }

/// <summary>

/// The read status of the message

/// </summary>

public bool Read { get; set; }

/// <summary>

/// The sent date

/// </summary>

public DateTime SentDate { get; set; }

/// <summary>

/// State

/// </summary>

public string State { get; set; }

/// <summary>

/// The name of the reciver

/// </summary>

public string ReciverName { get; set; }

/// <summary>

/// The name of the sender

/// </summary>

public string SenderName { get; set; }

/// <summary>

/// Guest

/// </summary>

public bool Guest { get; set; }

}

Score.cs :

/// <summary>

/// Score

/// </summary>

public class Score

{

/// <summary>

/// The id of the score

/// </summary>

public int ID { get; set; }

/// <summary>

/// The studetn

/// </summary>

public Member Student { get; set; }

/// <summary>

/// The exam

/// </summary>

public Exam Exam { get; set; }

/// <summary>

/// The value of the score

/// </summary>

public int ScoreVal { get; set; }

}

**Enumerations:**

Enums.cs :

/// <summary>

/// Enums for business logic

/// </summary>

public enum MemberClearance

{

Guest = 'g',

Student = 's',

Teacher = 't',

Admin = 'a'

}

public enum MemberGender

{

Male = 'm',

Female = 'f',

Unknown = ' '

}

public enum LessonChangeType

{

Cancel= 'c',

Fill= 'f',

Test= 't',

FinalTest = 'b',

Unknown = 'u'

}

**Services:**

CitiesService.cs :

/// <summary>

/// CitiesService

/// </summary>

public static class CitiesService

{

/// <summary>

/// Get all cities

/// </summary>

/// <returns>List(City) of all cities</returns>

public static List<City> GetAll()

/// <summary>

/// Get all cities in datatable

/// </summary>

/// <returns>DataTable</returns>

public static DataTable GetAllDT()

/// <summary>

/// Get all cities in DataSet

/// </summary>

/// <returns>DataSet</returns>

public static DataSet GetAllDS()

/// <summary>

/// Get city by id

/// </summary>

/// <param name="cityID">The id of the city</param>

/// <returns>City</returns>

public static City GetCity(int cityID)

/// <summary>

/// Add a new city

/// </summary>

/// <param name="c">City</param>

/// <returns>success</returns>

public static bool Add(City c)

/// <summary>

/// Updates city from service

/// </summary>

public static void UpdateCities()

/// <summary>

/// Updates city from service

/// </summary>

private static void updateDB()

}

DisciplinesServices.cs :

/// <summary>

/// DisciplinesServices

/// </summary>

public static class DisciplinesServices

{

/// <summary>

/// Gets all the types

/// </summary>

/// <returns></returns>

public static List<DisciplineType> GetAllTypes()

/// <summary>

/// Add new event

/// </summary>

/// <param name="lessonID">The id of the lesson</param>

/// <param name="studentId">The user id of the student</param>

/// <param name="disciplinesID">The disciplines type id</param>

/// <param name="date">The date of the event</param>

/// <returns></returns>

public static bool Add(int lessonID, int studentId, int disciplinesID,DateTime date) /// <summary>

/// Get all the preselected items

/// </summary>

/// <param name="lessonID">The lesson id</param>

/// <param name="date">The date of the lesson</param>

/// <returns></returns>

public static List<DisciplineEvent> GetSelected(int lessonID, DateTime date)

/// <summary>

/// Get student disciplines events by user id

/// </summary>

/// <param name="uid">User ID</param>

/// <returns></returns>

public static DataTable GetStudent(int uid)

/// <summary>

/// Get student disciplines events by user id

/// </summary>

/// <param name="uid">User ID</param>

/// <returns></returns>

public static DataTable GetStudent(int uid,DateTime date)

/// <summary>

/// Get student disciplines events by user id

/// </summary>

/// <param name="uid">User ID</param>

/// <returns></returns>

public static DataTable GetStudent(int uid,int tgid)

/// <summary>

/// Reset lesson disciplines

/// </summary>

/// <param name="lessonID">LessonID</param>

/// <param name="date">Date</param>

public static void ResetLesson(int lessonID, DateTime date)

}

ExamService.cs :

/// <summary>

/// ExamService

/// </summary>

public static class ExamService

{

/// <summary>

/// Get all exams

/// </summary>

/// <returns>List of Exams</returns>

public static List<Exam> GetAll()

/// <summary>

/// Add new exam

/// </summary>

/// <param name="exm">Exam obj</param>

/// <param name="tgid">Teacher grade id</param>

/// <returns>success</returns>

public static bool Add(Exam exm, int tgid)

/// <summary>

/// Get exam by id

/// </summary>

/// <param name="eid">Exam id</param>

/// <returns>Exam</returns>

public static Exam GetExam(int eid)

/// <summary>

/// Get exam id by exam object

/// </summary>

/// <param name="exm">Exam</param>

/// <returns></returns>

public static int GetExamID(Exam exm)

/// <summary>

/// Delete exam

/// </summary>

/// <param name="eid"></param>

/// <returns></returns>

public static bool Delete(int eid)

/// <summary>

/// Get precent left of teacher grade - irelavent

/// </summary>

/// <param name="tgid"></param>

/// <returns></returns>

public static int PrecentLeft(int tgid)

/// <summary>

/// Get precent left of teacher grade

/// </summary>

/// <param name="tgid">teacher grade id</param>

/// /// <param name="yearPart">year part</param>

/// <returns></returns>

public static int PrecentLeft(int tgid,string yearPart)

/// <summary>

/// Get exams by tgrade id

/// </summary>

/// <param name="tgid">Teacher grade id</param>

/// <returns></returns>

public static List<Exam> GetExamsByTgradeID(int tgid)

/// <summary>

/// Get exams by tgrade id

/// </summary>

/// <param name="tgid">Teacher grade id</param>

/// <param name="yearPart">Year part</param>

/// <returns></returns>

public static List<Exam> GetExamsByTgradeID(int tgid,string yearPart)

/// <summary>

/// Update exam in DB

/// </summary>

/// <param name="exam">Exam</param>

public static bool Update(Exam exam)

}

GradesService.cs :

/// <summary>

/// GradesService

/// </summary>

public static class GradesService

{

/// <summary>

/// Gets all the grades

/// </summary>

/// <returns></returns>

public static List<Grade> GetAll()

/// <summary>

/// Gets all the grades - DataTable

/// </summary>

/// <returns>DataTable</returns>

public static DataTable GetAllDT()

/// <summary>

/// Gets all the grades - DataSet

/// </summary>

/// <returns>DataSet</returns>

public static DataSet GetAllDS()

/// <summary>

/// Get grade by id

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

public static Grade Get(int id)

}

tGradesService.cs :

/// <summary>

/// tGradeService

/// </summary>

public static class tGradeService

{

/// <summary>

/// Gets all the tgrades

/// </summary>

/// <returns></returns>

public static List<tGrade> GetAll()

/// <summary>

/// Removes tgrade by id

/// </summary>

/// <param name="tgid"></param>

/// <returns></returns>

public static bool Remove(int tgid)

/// <summary>

/// Get id by obj

/// </summary>

/// <param name="grd">tGrade OBJ</param>

/// <returns></returns>

public static int GetID(tGrade grd)

/// <summary>

/// Add new tGrade to DB

/// </summary>

/// <param name="grd">tGrade</param>

/// <returns></returns>

public static bool Add(tGrade grd)

/// <summary>

/// Add students to tgarde

/// </summary>

/// <param name="tgid">Tgrade ID</param>

/// <param name="students">List of students</param>

/// <returns></returns>

public static bool AddStudents(int tgid, List<Member> students)

/// <summary>

/// Add student to tgarde

/// </summary>

public static bool AddStudent(int tgid, Member student)

/// <summary>

/// Add student to tgarde

/// </summary>

public static bool AddStudent(int tgid, int uid)

/// <summary>

/// Update tgrade

/// </summary>

/// <param name="tg"></param>

/// <returns></returns>

public static bool Update(tGrade tg)

/// <summary>

/// Get part grade of tgarde

/// </summary>

/// <param name="tgid">tgrade id</param>

/// <returns></returns>

public static string GetPartGrade(int tgid)

/// <summary>

/// Get part grade

/// </summary>

/// <param name="gname">grade</param>

/// <returns></returns>

public static string GetPartGrade(string gname)

/// <summary>

/// Get tgrade by id

/// </summary>

/// <param name="tgid">tGrade id</param>

/// <returns></returns>

public static tGrade Get(int tgid)

/// <summary>

/// Get tgrades of teacher

/// </summary>

/// <param name="tid"></param>

/// <returns></returns>

public static List<tGrade> GetTeacherTgrades(int tid)

/// <summary>

/// Get student count of tgrade

/// </summary>

/// <param name="tgid"></param>

/// <returns></returns>

public static int GetStudentCount(int tgid)

/// <summary>

/// Get students of tgrade

/// </summary>

/// <param name="tgid"></param>

/// <returns></returns>

public static List<Member> GetStudents(int tgid)

/// <summary>

/// Get the major of tgrade

/// </summary>

/// <param name="tgid"></param>

/// <returns></returns>

public static int GetMajor(int tgid)

}

LessonService.cs :

/// <summary>

/// LessonService

/// </summary>

public static class LessonService

{

//Days in week

public static int DaysInWeek = int.Parse(ConfigurationManager.AppSettings["DaysInWeek"].ToString());

//Lessons in day

public static int LessonsInDay = int.Parse(ConfigurationManager.AppSettings["LessonsInDay"].ToString());

/// <summary>

/// Gets all the lessons for the following id.

/// </summary>

/// <param name="teacherId">The teacher id</param>

/// <returns></returns>

public static List<Lesson> GetAll(int teacherId)

/// <summary>

/// Gets a lesson by lesson id

/// </summary>

/// <param name="lid">Lesson ID</param>

/// <returns></returns>

public static Lesson GetLesson(int lid)

/// <summary>

/// Get all students in lesson

/// </summary>

/// <param name="lid">Lesson id</param>

/// <returns></returns>

public static List<Member> GetAllStudents(int lid)

/// <summary>

/// Helper method

/// </summary>

/// <typeparam name="T"></typeparam>

/// <param name="arrs"></param>

/// <param name="rowIndex"></param>

/// <returns></returns>

private static T[] GetRow<T>(T[,] arrs, int rowIndex)

/// <summary>

/// Gets lessons by teacher

/// </summary>

/// <param name="tid"></param>

/// <returns></returns>

private static DataTable GetLessonsByTeacher(int tid)

/// <summary>

/// Gets lessons by student

/// </summary>

/// <param name="tid"></param>

/// <returns></returns>

private static DataTable GetLessonsByStudent(int sid)

/// <summary>

/// Gets lessons by GradePart

/// </summary>

/// <param name="tid"></param>

/// <returns></returns>

private static DataTable GetLessonsByGradePart(string gradePart)

/// <summary>

/// Gets the time table of the grade that contains this value in its name

/// </summary>

/// <param name="gradePart">Value to check</param>

/// <returns></returns>

/// <summary>

/// Gets the time table of the user

/// </summary>

/// <param name="uid">User ID</param>

/// <param name="memTable">User Clearance</param>

/// <returns></returns>

public static List<LessonGroup[]> GetTimeTable(int uid, MemberClearance memTable)

/// <summary>

/// Cancel change

/// </summary>

/// <param name="changeID"></param>

/// <returns></returns>

public static bool CancelChange(int changeID)

/// <summary>

/// Get the changes of a lesson

/// </summary>

/// <param name="lid">Lesson id</param>

/// <returns></returns>

public static List<LessonChange> GetChanges(int lid)

/// <summary>

/// Get time table - helper method

/// </summary>

/// <param name="dt">DataTable</param>

/// <returns></returns>

private static List<LessonGroup[]> GetDataTime(DataTable dt)

/// <summary>

/// Gets all the lessons of a tGrade

/// </summary>

/// <param name="tgid">tgrade id</param>

/// <returns></returns>

public static List<Lesson> GetLessons(int tgid)

/// <summary>

/// Delete lesson

/// </summary>

/// <param name="tgid"></param>

/// <param name="day"></param>

/// <param name="hour"></param>

/// <returns></returns>

public static bool DeleteLesson(int tgid, int day, int hour)

/// <summary>

/// Delete lesson

/// </summary>

public static bool DeleteLesson(int lid)

/// <summary>

/// Add lesson to db

/// </summary>

public static bool Add(Lesson lsn)

/// <summary>

/// Add new change to lesson

/// </summary>

/// <param name="change"></param>

/// <returns></returns>

public static bool AddChange(LessonChange change)

/// <summary>

/// Get lesson by params

/// </summary>

/// <param name="tgid"></param>

/// <param name="hour"></param>

/// <param name="day"></param>

/// <returns></returns>

public static Lesson GetLesson(int tgid,int hour,int day)

}

MajorsService.cs :

/// <summary>

/// Major Service

/// </summary>

public static class MajorsService

{

/// <summary>

/// Get all majors

/// </summary>

/// <returns></returns>

public static List<Major> GetAll()

/// <summary>

/// Get all majors - DataTable

/// </summary>

/// <returns>DataTable</returns>

public static DataTable GetAllDT()

/// <summary>

/// Get all majors - DataSet

/// </summary>

/// <returns>DataSet</returns>

public static DataSet GetAllDS()

/// <summary>

/// Get major by id

/// </summary>

/// <param name="majorID">Major id</param>

/// <returns>Major</returns>

public static Major Get(int majorID)

/// <summary>

/// Get connections of majors and members

/// </summary>

/// <returns></returns>

public static List<MajorMember> GetConnection()

private static List<Major> currentAll = GetAll();//All

private static List<MajorMember> currentConnections = GetConnection();//Connections

/// <summary>

/// Get majors of user

/// </summary>

/// <param name="uid">User ID</param>

/// <returns></returns>

public static List<Major> GetUserMajors(int uid)

/// <summary>

/// Update majors and connections with multithreading

/// </summary>

/// <returns></returns>

private static Task Update()

/// <summary>

/// Update majors and connections with multithreading

/// </summary>

/// <returns></returns>

private static void updateDB()

/// <summary>

/// connect major to tgrade for certain grade part

/// </summary>

/// <returns></returns>

public static bool SetMajorTgrade(int tgid,int mjrid,string gPart)

/// <summary>

/// Add new major

/// </summary>

/// <param name="m">Major</param>

/// <returns></returns>

public static bool Add(Major m)

/// <summary>

/// Get major id by name

/// </summary>

/// <param name="name"></param>

/// <returns></returns>

public static int GetMajorID(string name)

}

MembersService.cs :

/// <summary>

/// A service class for member class

/// DB Connector

/// </summary>

public static class MemberService

{

/// <summary>

/// Get allowed by id

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

public static Member GetAllowed(string id)

/// <summary>

/// Removes the user from the invite list

/// </summary>

/// <param name="id">Civilian ID</param>

/// <returns></returns>

public static bool RemoveFromAllowed(string id)

/// <summary>

/// Gets all the members from the DB

/// </summary>

/// <returns>A list of member object</returns>

public static List<Member> GetAll()

/// <summary>

/// Gets partially field member object for liter work loads that require the following:

/// UserID, FirstName, LastName, Name, Auth, Active

/// </summary>

/// <returns></returns>

public static List<Member> GetNames()

/// <summary>

/// Gets all the members from the DB

/// </summary>

/// <returns>A DataTable of members</returns>

public static DataTable GetAllDT()

/// <summary>

/// Get the current member from the DB

/// </summary>

/// <returns>A DataTable of members</returns>

public static DataTable GetCurrentDT()

/// <summary>

/// Login of the user - init for session - The session key is 'Member'

/// </summary>

/// <param name="email">Email to login</param>

/// <param name="pass">Password to login</param>

/// <returns>Whether the login was successful or not</returns>

public static bool Login(string email, string pass)

/// <summary>

/// Login of the user - init for session - The session key is 'Member'

/// </summary>

/// <param name="email">Email to login</param>

/// <param name="pass">Password to login</param>

/// <returns>Whether the login was successful or not</returns>

public static bool Login(string id, string pass,bool ids)

/// <summary>

/// Get's the current user from the session

/// </summary>

/// <returns>The current logged in user</returns>

public static Member GetCurrent()

/// <summary>

/// Validates Session Keys (Their Values)

/// </summary>

/// <param name="sessionNames">The keys</param>

/// <param name="c">The hyper text transfer protocol context</param>

/// <returns>Whether the keys are empty or not(if one is empty then it is false)</returns>

private static bool ValidateSessions(string[] sessionNames, HttpContext c)

/// <summary>

/// Template for inserting a new member into the DB

/// </summary>

private const string FullInsertTemplate = "INSERT INTO nhsMembers (nhsFirstName,nhsLastName,nhsPass,nhsType,nhsMail,nhsID,nhsGender,nhsGradeID,nhsPicture,nhsBorn,nhsDateRegister,nhsCityID,nhsActive) VALUES('{0}','{1}','{2}','{3}','{4}','{5}','{6}','{7}','{8}','{9}','{10}',{11},Yes)";

/// <summary>

/// Adds a user - ALL FIELDS MUST NOT BE EMPTY!

/// </summary>

/// <param name="m">The member to add</param>

/// <param name="pass">The password of the member</param>

public static void Add(Member m, string pass)

/// <summary>

/// Update the DB allowed table to active account -- just for tracking

/// </summary>

/// <param name="uid">The user id<

/// /param>

public static void UpdateAllowed(int uid)

/// <summary>

/// Updates the session to the following user

/// </summary>

/// <param name="m"></param>

public static void UpdateCurrent(Member m)

/// <summary>

/// Checks if the user is allowed to register - to prevent unwanted guests from registering

/// </summary>

/// <param name="fname">First Name</param>

/// <param name="lname">Last Name</param>

/// <param name="iuid">Israeli ID</param>

/// <returns>UserAllowed</returns>

public static bool IsAllowed(string fname, string lname, string iuid)

/// <summary>

/// Check if allowed exsit

/// </summary>

/// <param name="id"></param>

/// <returns></returns>

public static bool ExsitsAllowed(string id)

/// <summary>

/// Gets the user auth

/// </summary>

/// <param name="uid">The user id</param>

/// <returns></returns>

public static MemberClearance GetClearance(string uid)

/// <summary>

/// Gets the user id with the id of the user

/// </summary>

/// <param name="ID"></param>

/// <returns></returns>

public static int GetUID(string ID)

/// <summary>

/// Checks if the email already exsits in the database

/// </summary>

/// <param name="email">Email</param>

/// <returns>Whether the member exsit with the same email</returns>

public static bool Exsits(string email)

/// <summary>

/// Updates the user in the database - USER ID IS MUST

/// </summary>

/// <param name="m">The user</param>

/// <returns>State</returns>

public static bool Update(Member m)

/// <summary>

/// Updates the user password in the database - USER ID IS MUST - PK

/// </summary>

/// <param name="uid">The user id</param>

/// <param name="pass">The new password</param>

/// <returns>State</returns>

public static bool UpdatePassword(int uid, string pass)

/// <summary>

/// Removes the user by the user id PK

/// </summary>

/// <param name="uid">User ID</param>

/// <returns>Action State</returns>

public static bool RemoveFromActive(string uid)

/// <summary>

/// Gets a user by his user id PK

/// </summary>

/// <param name="uid"></param>

/// <returns></returns>

public static Member GetUser(int uid)

/// <summary>

/// Gets a user by his user id PK

/// </summary>

/// <param name="uid"></param>

/// <returns></returns>

public static Member GetUserPart(int uid)

/// <summary>

/// Logs out the current connected user

/// </summary>

public static void Logout()

/// <summary>

/// Returns a list of currently connected users

/// </summary>

/// <returns></returns>

public static List<Member> GetConnected()

/// <summary>

/// Adds a member to invite list(those who are allowed to register)

/// </summary>

/// <param name="m">Member to add</param>

/// <returns></returns>

public static bool AddAllowed(Member m)

/// <summary>

/// Adds a member to invite list(those who are allowed to register)

/// </summary>

/// <param name="fname">First Name</param>

/// <param name="lname">Last Name</param>

/// <param name="id">ID</param>

/// <param name="type">Clearence</param>

/// <returns></returns>

public static bool AddAllowed(string fname,string lname,string id,string type)

/// <summary>

/// Adds a member to invite list(those who are allowed to register)

/// </summary>

/// <param name="dt"></param>

/// <param name="indID"></param>

/// <param name="indFname"></param>

/// <param name="indLname"></param>

/// <returns></returns>

public static int AddAllowed(DataTable dt,int indID,int indFname,int indLname)

/// <summary>

/// Gets all the users from the following grade

/// </summary>

/// <param name="gid"></param>

/// <returns></returns>

public static List<Member> GetGrade(int gid)

/// <summary>

/// Gets all the users that in that grade incl. Teachers

/// </summary>

/// <param name="grade"></param>

/// <returns></returns>

public static List<Member> GetGradePart(string grade)

/// <summary>

/// Gets all the allowed to register list

/// </summary>

/// <returns>DataTable</returns>

public static DataTable GetAllowed()

/// <summary>

/// Gets the greeting for the panel

/// </summary>

/// <param name="m">Member</param>

/// <returns>Greeting for the panel</returns>

public static string GetGreeting(Member m)

/// <summary>

/// Gets the greeting for the panel

/// </summary>

/// <param name="auth">MemberClearance</param>

/// <param name="gen">MemberGender</param>

/// <returns>Greeting for the panel</returns>

public static string GetGreeting(MemberClearance auth,MemberGender gen)

/// <summary>

/// Get free hours of a teacher

/// </summary>

/// <param name="teacherId"></param>

/// <param name="day"></param>

/// <returns></returns>

public static List<int> GetFreeHours(int teacherId,int day)

/// <summary>

/// Get students of a teacher

/// </summary>

/// <param name="tid"></param>

/// <returns></returns>

public static DataTable GetStudents(int tid)

}

MessagesService.cs :

/// <summary>

/// Message Service

/// </summary>

public static class MessagesService

{

/// <summary>

/// Sends a message

/// </summary>

/// <param name="m">Message Object</param>

public static void SendMessage(Message m)

/// <summary>

/// Marks a message as read by message id

/// </summary>

/// <param name="mid">Message ID</param>

public static void MarkAsRead(int mid)

/// <summary>

/// Marks a message as read by message object(id)

/// </summary>

/// <param name="mid">Message</param>

public static void MarkAsRead(Message m) { MarkAsRead(m.ID); }

/// <summary>

/// Gets the user messages

/// </summary>

/// <param name="uid">User ID</param>

public static List<Message> GetAllUser(int uid)

/// <summary>

/// Gets all messages

/// </summary>

public static List<Message> GetAll()

/// <summary>

/// Deletes a message

/// </summary>

/// <param name="uid">User ID</param>

/// <param name="mid">Message ID</param>

public static void Delete(int uid, int mid)

/// <summary>

/// Gets the unread count of the user

/// </summary>

/// <param name="uid">User ID</param>

public static int GetUnreaedCount(int uid)

/// <summary>

/// Get unread messages of user

/// </summary>

/// <param name="uid"></param>

/// <returns></returns>

public static List<Message> GetUnreaed(int uid)

}

ScoreService.cs :

/// <summary>

// ScoreService

/// </summary>

public static class ScoreService

{

/// <summary>

/// Get all scores

/// </summary>

/// <returns></returns>

public static List<Score> GetAll()

/// <summary>

/// Get all scores of student

/// </summary>

/// <param name="sid"></param>

/// <returns></returns>

public static List<Score> GetAllStudent(int sid)

/// <summary>

/// Get all scores of student in specific tgrade

/// </summary>

/// <param name="sid"></param>

/// <param name="tgid"></param>

/// <returns></returns>

public static List<Score> GetAllStudent(int sid,int tgid)

/// <summary>

/// Get all scores of student in specific tgrade and year part

/// </summary>

/// <param name="sid"></param>

/// <param name="tgid"></param>

/// <param name="yearPart"></param>

/// <returns></returns>

public static List<Score> GetAllStudent(int sid, int tgid,string yearPart)

/// <summary>

/// Get all scores of student in specific tgrade with empty scores

/// </summary>

/// <param name="sid"></param>

/// <param name="tgid"></param>

/// <returns></returns>

public static List<Score> GetAllStudentWE(int sid,int tgid) /// <summary>

/// Get all scores of student in specific tgrade and year part with empty scores

/// </summary>

/// <param name="sid"></param>

/// <param name="tgid"></param>

/// <param name="yearPart"></param>

/// <returns></returns>

public static List<Score> GetAllStudentWE(int sid, int tgid,string yearPart)

/// <summary>

/// Get all the scores of an exam

/// </summary>

/// <param name="eid"></param>

/// <returns></returns>

public static List<Score> GetAllExam(int eid)

/// <summary>

/// Get score

/// </summary>

/// <param name="sid">Student ID</param>

/// <param name="eid">Exam ID</param>

/// <returns></returns>

public static Score GetScore(int sid, int eid)

/// <summary>

/// Get avg of exam

/// </summary>

/// <param name="eid">ExamID</param>

/// <returns></returns>

public static double GetAvgExam(int eid)

/// <summary>

/// Get all grade - DataTable

/// </summary>

/// <param name="sid"></param>

/// <returns></returns>

public static DataTable GetAllGrade(int sid)

/// <summary>

/// Get student avg

/// </summary>

/// <param name="sid"></param>

/// <returns></returns>

public static double GetStudentAvg(int sid)

/// <summary>

/// Get student avg

/// </summary>

/// <param name="sid"></param>

/// <param name="tgid"></param>

/// <returns></returns>

public static double GetStudentAvg(int sid,int tgid)

/// <summary>

/// Get student avg final

/// </summary>

/// <param name="sid"></param>

/// <param name="tgid"></param>

/// <returns></returns>

public static double GetStudentAvgFinal(int sid, int tgid)

/// <summary>

/// Get student avg final

/// </summary>

/// <param name="sid"></param>

/// <param name="tgid"></param>

/// <param name="yearPart"></param>

/// <returns></returns>

public static double GetStudentAvgFinal(int sid, int tgid,string yearPart)

/// <summary>

/// Add new score

/// </summary>

/// <param name="score"></param>

/// <returns></returns>

public static bool Add(Score score)

/// <summary>

/// Check if exsit

/// </summary>

/// <param name="sid"></param>

/// <param name="eid"></param>

/// <returns></returns>

public static bool Exsits(int sid, int eid)

/// <summary>

/// Reset scores

/// </summary>

/// <param name="eid"></param>

/// <returns></returns>

public static bool ResetScores(int eid) /// <summary>

/// Reset scores student

/// </summary>

/// <param name="sid"></param>

/// <param name="tgid"></param>

/// <returns></returns>

public static bool ResetScoresStudent(int sid,int tgid)

}

**Encryption:**

/// <summary>

/// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/// \*\*\*\*\*Security\*\*\*\*\*\*\*

/// \*\*\*\*\*TOP SECRET\*\*\*\*\*

/// \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

/// </summary>

/\* base64://8J2UsfCdlKXwnZSm8J2UsCDwnZSm8J2UsCDwnZSeIPCdlKDwnZSp8J2UnvCdlLDwnZSwIPCdlLHwnZSl8J2UnvCdlLEg8J2UovCdlKvwnZSg8J2Ur/CdlLbwnZSt8J2UsfCdlLAg8J2UnvCdlKnwnZSpIPCdlK3wnZSe8J2UsPCdlLDwnZS08J2UrPCdlK/wnZSh8J2UsCDwnZS08J2UpvCdlLHwnZSlIPCdlJ4g8J2UsPCdlLHwnZSv8J2UrPCdlKvwnZSkIPCdlKLwnZSr8J2UoPCdlK/wnZS28J2UrfCdlLHwnZSm8J2UrPCdlKs= \*/

public static class Security

{

public static int SALT\_SIZE = 32;//Salt size

public static int HASH\_SIZE = 64;//Hash size

public static int PBKDF2\_TTT = 512;//Hashing Iteration Count

/// <summary>

/// Encryption creator

/// </summary>

/// <param name="password">string to encrypt</param>

/// <returns></returns>

public static string CreateHash(string password)

private static byte[] PBKDF2(string password, byte[] salt)

private static bool SlowEqual(byte[] dbHash, byte[] passHash)

public static bool ValidatePassword(string password, string dbHash)

}

**EduWS – Internal Web Service:**

public class EduAdjustmentsService : System.Web.Services.WebService {

/// <summary>

/// Gets all the adjustments types

/// </summary>

/// <returns>List of adjustments</returns>

[WebMethod]

public List<Adjustment> GetAdjustmentsTypes()

/// <summary>

/// Gets the adjustment of the student with the following id

/// </summary>

/// <param name="id">ID of the student</param>

/// <returns>List of adjustments the student have</returns>

[WebMethod]

public List<Adjustment> GetAdjustmentsStudent(string id)

/// <summary>

/// Add a student to the system with an adjustment

/// </summary>

/// <param name="firstName">First Name</param>

/// <param name="lastName">Last Name</param>

/// <param name="id">ID</param>

/// <param name="adid">Adjustment ID</param>

/// <returns>The result of the action from the system</returns>

[WebMethod]

public string AddStudent(string firstName,string lastName,string id, string adid)

/// <summary>

/// Adds an adjustment to a student that already exsits in the system

/// </summary>

/// <param name="id">ID</param>

/// <param name="adid">Adjustment ID</param>

/// <returns>The result of the action from the system</returns>

[WebMethod]

public string AddAdjustmentToStudent(string id, string adid)

/// <summary>

/// Gets the adjustment of the student with the following id - in JSON

/// </summary>

/// <param name="id">ID of the student</param>

/// <returns>List of adjustments the student have</returns>

[WebMethod]

public string GetAdjustmentsStudentJSON(string id)

/// <summary>

/// Gets all the students from the system

/// </summary>

/// <returns>List of students</returns>

[WebMethod]

public List<Member> GetAllStudents()

/// <summary>

/// Gets all the students from the system - JSON

/// </summary>

/// <returns>List of students</returns>

[WebMethod]

public string GetAllStudentsJSON()

}

Adjustment.cs :

/// <summary>

/// Adjustment

/// </summary>

public class Adjustment

{

/// <summary>

/// Name of adjustment

/// </summary>

public string Name { get; set; }

/// <summary>

/// ID of adjustment

/// </summary>

public int ID { get; set; }

}

AdjustmentService.cs :

/// <summary>

/// AdjustmentService

/// </summary>

public static class AdjustmentService

{

/// <summary>

/// Get all adjustments

/// </summary>

/// <returns></returns>

public static List<Adjustment> GetAll()

/// <summary>

/// Get students adjustments

/// </summary>

/// <param name="sid"></param>

/// <returns></returns>

public static List<Adjustment> GetStudent(string sid)

/// <summary>

/// Add adjustment to student

/// </summary>

/// <param name="sid"></param>

/// <param name="adid"></param>

/// <returns></returns>

public static string Add(string sid, int adid)

}

# מפת האתר

ראשית, באתר קיימים שלושה סוגי הרשאות:

לכן לכל אחד

מהסוגים מפה

שונה.

**תלמיד:**

**מורה:**

**מנהל:**

# ביבליוגרפיה / רשימת מקורות / אתרי עזר

* **W3School** – "the world's largest web development site" <http://www.w3schools.com/>
* **jQuery** – "with less do more" <http://jquery.com/>
* **Codepen.io** - <http://codepen.io/>
* **Material.io** – “Material Design is a unified system that combines theory, resources, and tools for crafting digital experiences.” <http://material.io/>
* **Stack** **Overflow** -<http://stackoverflow.com/>
* **Kudevnet** - <https://www.youtube.com/user/kudvenkat>
* **JQuery DataTables** - <https://datatables.net/>
* **Github.com** – **Important – I used it as a source control for my project.** Find project in <https://github.com/avivbuen/eduplug>

## ReadMe

## EduPlug - A school managing app

http://eduplug.co.il

To setup the project you need to do the following by order:

1. Open the eduplug.sln file on visual studio.

2. Re-add all the web references because visual studio opens different port every time.

3. Run the project.

From here you can use the project in 3 different account types:

Manager:

ID: 314711045

Pass: avivadi1

Teacher:

ID: 324940014

Pass: avivadi1

Student:

ID: 318849262

Pass: avivadi1

Notice: In order to register a new user, you need to first login as a manager

and allow this man to register by typing the first name, last name and ID.