# Computing Design

# Introduction

The design of this project will revolve around a teacher windows form side where teachers are able to add or remove students, see individual student scores, the class average scores, the students IDs and which multiple the students are doing the best work on. The teacher will be required to login based on this they will be able to see their class and their class scores. As well as this the will also be a student side of the project. This will include a student login that will open the game from here the student can select the multiple they would like to work on and

# Languages and frame works in creating this project

In this project I will be utilising:

* C# - Windows forms, XNA
* SQL – server management

# Design

## Initial form

Teacher

Button

Student

Button

Allows students to select if they are a student. Takes to student side of the project

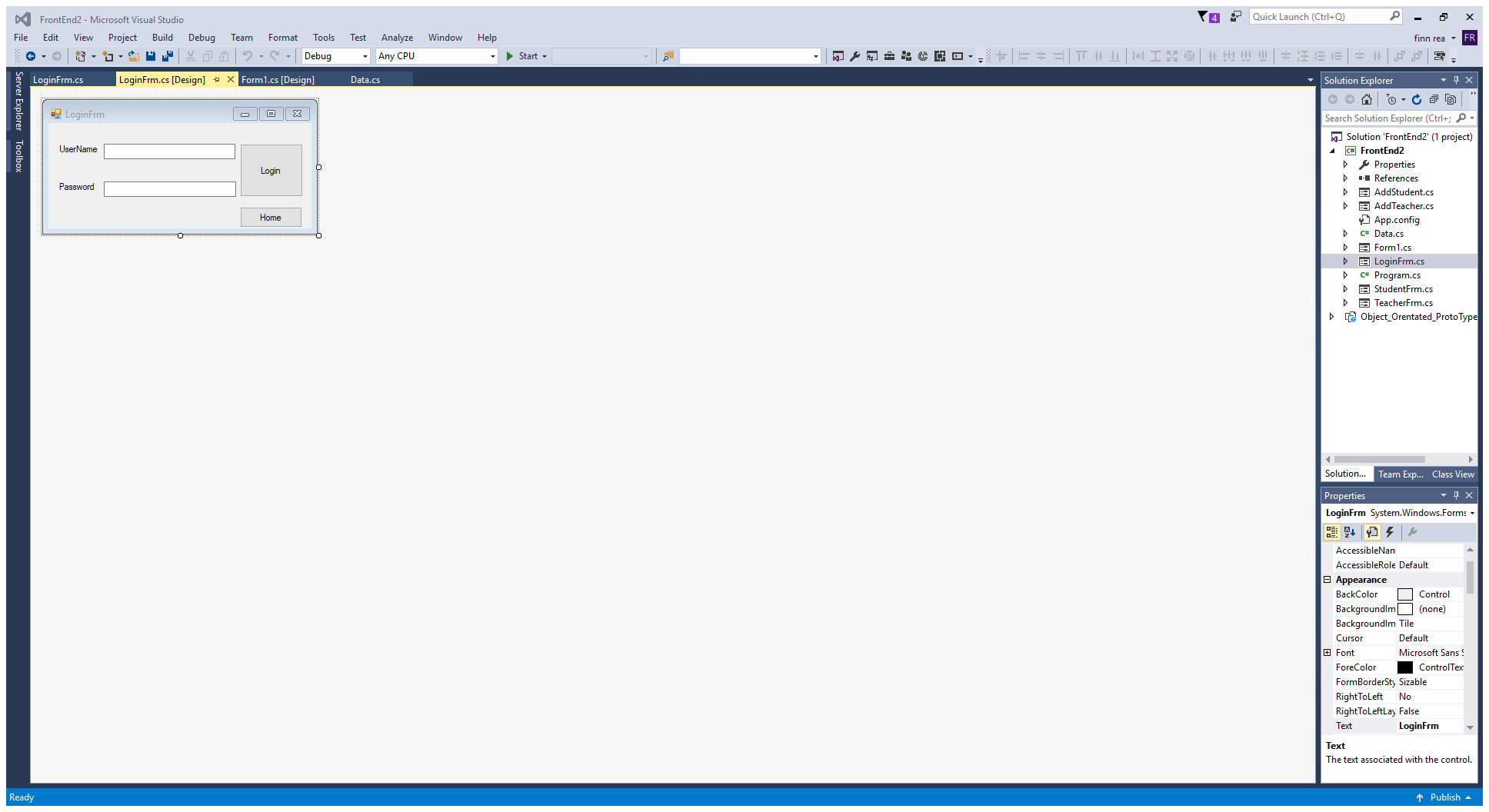
Allows teachers to select if they are a teacher. Takes to Teacher side of the project

## Teacher side design

When click, the program checks the username and password, if incorrect then the user can re-try entering their username and password.

### Initial form - Login

Where user can Input username



Where user can Input password. When characters are input the char “\*” is displayed, this reduces others seeing others passwords.

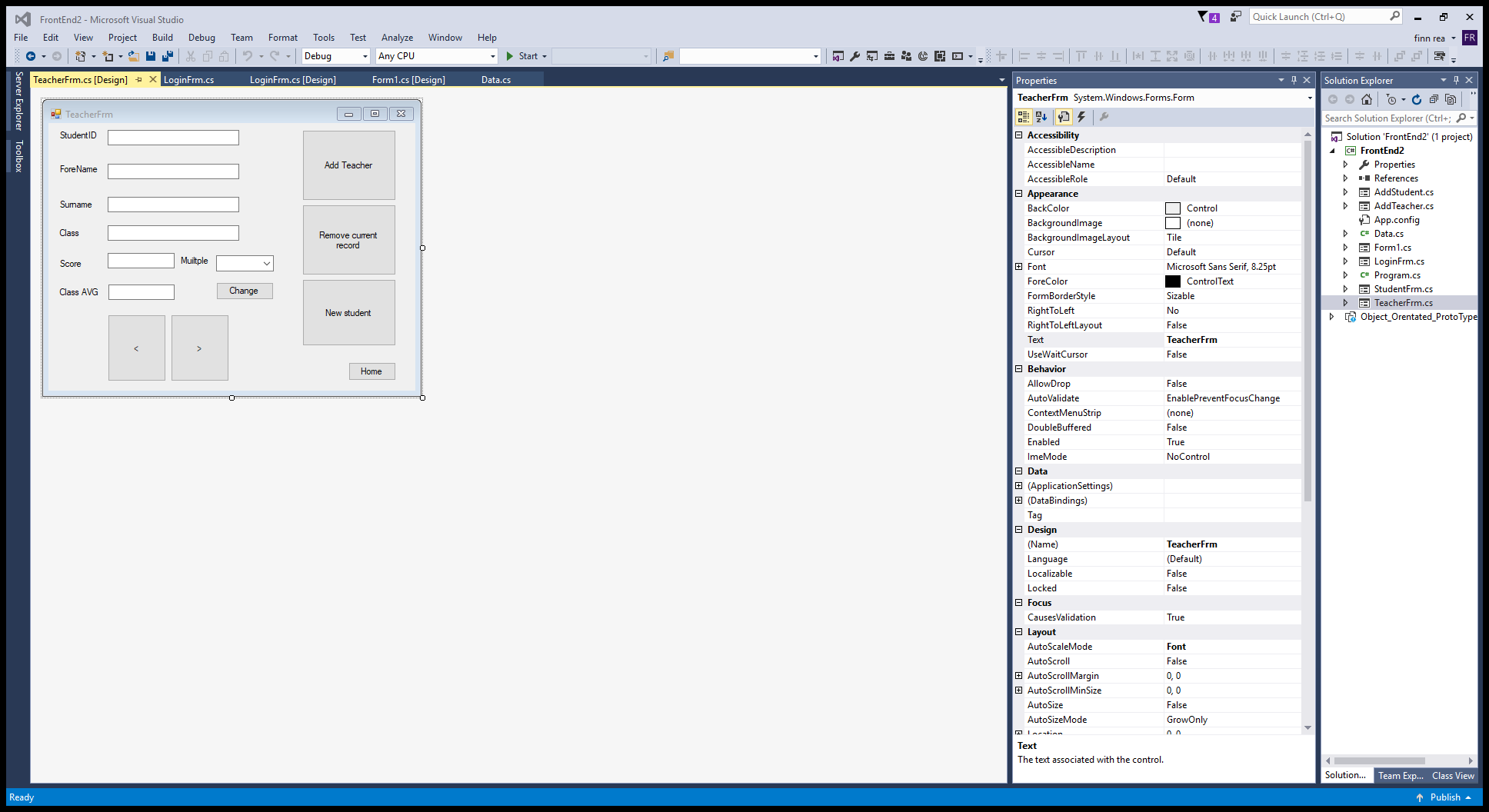
This take the user back to the first form when clicked.

This is the login form, at this stage the teach will need to input their username and password. This will allow for Id to be found, and which class belongs to them. If the Admin account is used then then class will not be specified and all accounts can be seen.

### Teacher form

Displays the forename of the student.

This displays the student ID.



This removed the current record that is selected. It removes password, username and account.

Displays the class of the student.

This displays the surname of the student.

This allows the user to choose which multiple they would like to see the scores of the children.

This displays the classes average score.

This is only accessible when the admin account is opened. This take the user to the add teacher form.



When clicked this takes the user to the add student form.

When clicked it changes the multiple being viewed.

This changes which student is being displayed. The students are displayed in order of ID. The left arrow decreases the student ID. The right increase it.

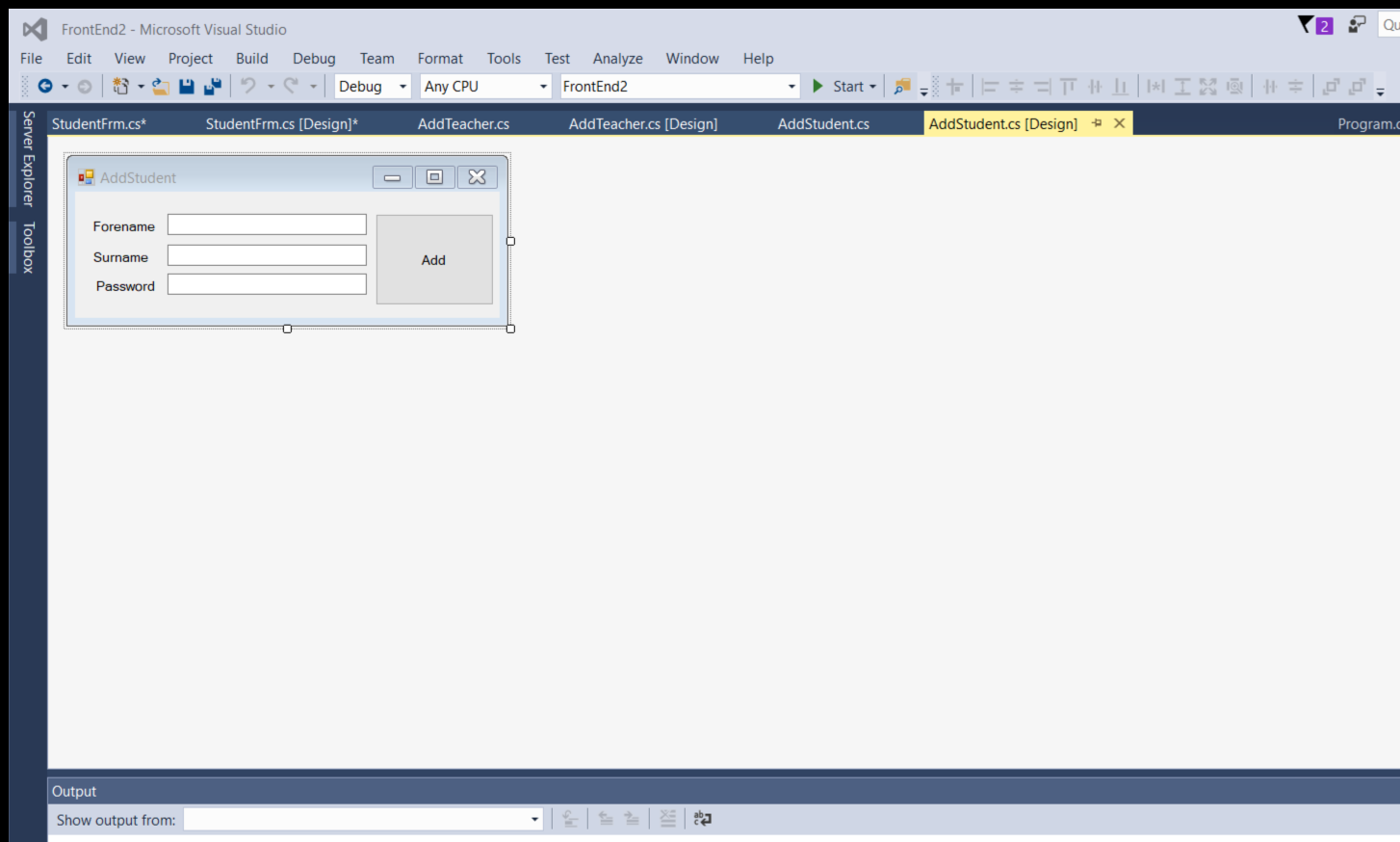
When clicked this takes the user back to the home screen.

When clicked this takes the user to ViewDisplayForm

This form is used by the teacher to navigate through their students, being able to see how well each individual student is doing on each individual multiple. From here the teacher can navigate through adding or removing accounts. If it is the admin account that has been accessed, then all students can be accessed and they are able to add teachers.

### Add student

This is where the new student’s first name is input.



This will add a new student to the teacher’s class. On click a new username and password will be created for the student. This is based on first name and surname e.g. john smith – jsmith1. If and pre-existing usernames and passwords exist then the end number is increased until it is unique.

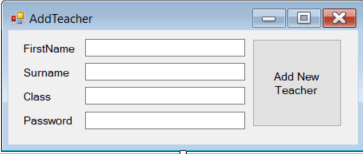
This is where the student’s surname is input.

This is where the student’s password is input.

### Add teacher – Admin

This is where the surname of the teacher is input.

This is where the first name of the teacher is input

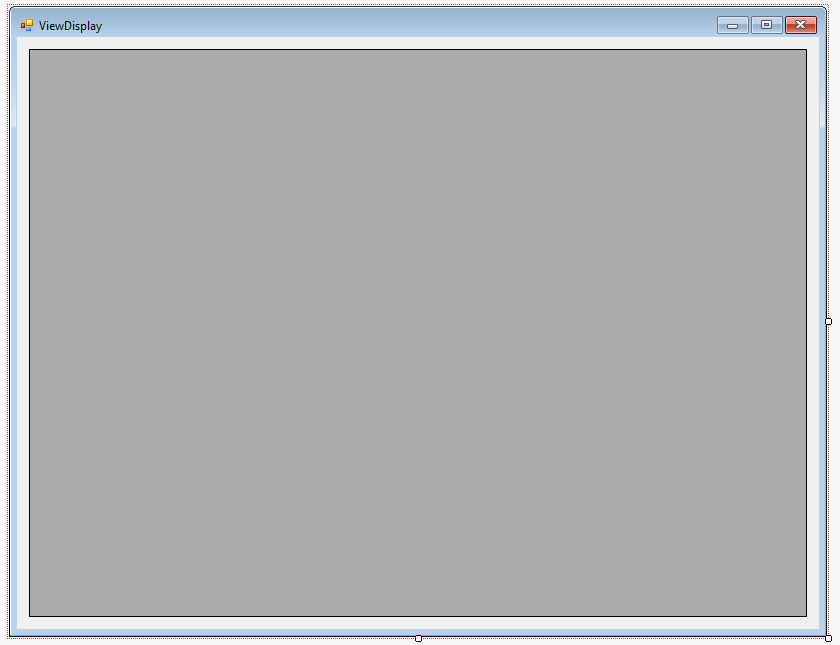


This is where the class that belongs to them is added

When clicked this adds a new teacher to the teacher table in the database. They will gain a username password in the same way the students is, the password will be hashed before added.

This is where the password of the teacher is input

### ViewDisplayForm



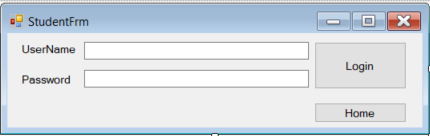
Displays the scores of all students in the class

## Student Side

When click, the program checks the username and password, if incorrect then the user can re-try entering their username and password.

### Login form

Username input



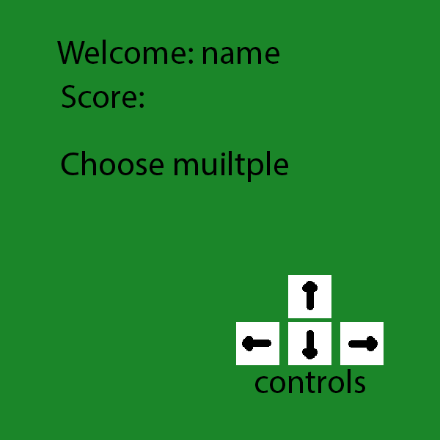
Where user can Input password. When characters are input the char “\*” is displayed, this reduces others seeing other people seeing passwords.

Return to home screen

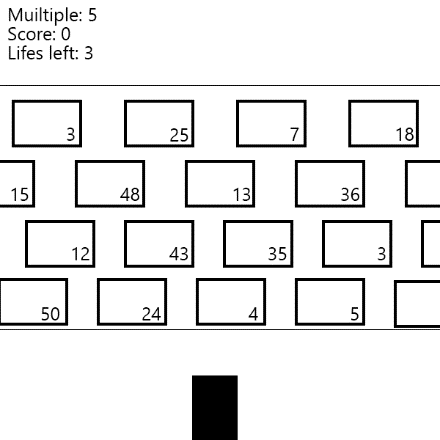
### Pause screen

This is the pause screen for the student game. Once the student has logged in their name will be displayed to add a user-friendly element. The score of each multiple will also be shown when scrolling through which multiple to select. Once the multiple they want to select has been selected they can press the start button to start the game.

Start



Start game



This displays the user’s “stats”. These being the score, the multiple working on and how many life’s the user has left.

This will be the user’s character, movement up and down the screen will be controlled with the arrow keys.

This will be the “logs” the user has to jump across. They will only be able to so if the multiple of the log is one of the ones the user selected. If not, the user will lose a life.

## State-transition diagram

## Database - ALevelDataBase

### Table - Teachers

|  |  |  |  |
| --- | --- | --- | --- |
| Identifier | Data type | Scope | Description |
| TeacherID | Int |  | Primary Key – unique for each student |
| First name | Varchar(100) |  | Name of teacher |
| Surname | Varchar(100) |  | Surname or teacher |
| Class | int |  | Class of theirs |

### Table – Students

|  |  |  |  |
| --- | --- | --- | --- |
| Identifier | Data type | Scope | Description |
| StudentId | int |  | Primary key – unique for key student |
| Firstname | Varchar(100) |  | Firstname of student |
| Surname | Varchar(100) |  | Last name of student |
| Class | int |  | Which class the student belongs to |

### Table – Users

|  |  |  |  |
| --- | --- | --- | --- |
| Identifier | Data type | Scope | Description |
| TeacherId | int |  | Foreign key – Links to Table Teachers on TeacherId = TeacherId |
| Username | Varchar(100) |  | Username of Teacher |
| Pass | Varchar(100) |  | Password of teacher |

### Table - StudentUsers

|  |  |  |  |
| --- | --- | --- | --- |
| Identifier | Data type | Scope | Description |
| StudentId | int |  | Foreign key – Links to Table Students, on StudentId = StudentId |
| Username | Varchar(100) |  | Username of Student |
| Pass | Varchar(100) |  | Password of Student |

### Table - Scores

|  |  |  |  |
| --- | --- | --- | --- |
| Identifier | Data type | Scope | Description |
| ScoreId | int |  | Foreign key and primary key – Links to Table Students, on StudentId = ScoreId |
| Score2 | int |  | Holds score for multiple of 2 |
| Score3 | int |  | Holds score for multiple of 3 |
| Score4 | int |  | Holds score for multiple of 4 |
| Score5 | int |  | Holds score for multiple of 5 |
| Score6 | int |  | Holds score for multiple of 6 |
| Score7 | int |  | Holds score for multiple of 7 |
| Score8 | int |  | Holds score for multiple of 8 |
| Score9 | int |  | Holds score for multiple of 9 |

## Front End – Teacher side

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identifier | Data type | Scope | Parameters passed | Description |
| Form 1 | Constructor | Public |  | Constructor for Form 1 |
| StudentBTN | Button | Private |  | Creates a new instance of student form. This is done when clicked. It also closes current form (Form1) |
| StudentFormOpen | Windows form | Public? |  | A new instance of student form, this is then opened. |
| TeacherBTN\_Click | Button | Private |  | Creates a new instance of login form. This is done when clicked. It also closes current form (Form1) |
| LoginFormOpen | Windows form | Public |  | This create a new windows form instance of LoginForm, This is then opened. |

### Form 1

### Data class

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identifier | Data type | Scope | Parameters passed | Description |
|  |  |  |  |  |
| Connection | String - Constant | private |  | This is a connection path followed to the back end database and to know to used Database “”. For this connection I initially used the OleDbConnection class a my way of connecting to the backend and using statement Provider=SQLOLEDB; Data Source=.\SQLEXPRESS; Initial Catalog=ALevelDatabase; User ID= sa; Password= Pa$$w0rd |
| Connection2 | String – Constant | private |  | This a different Connection to the same data based. For this one I used SQL client this was to allow for the parameterization of certain commands.  Data Source=.\SQLEXPRESS; Initial Catalog=ALevelDatabase; User ID=sa; Password=Pa$$w0rd |
| SQL\_COMMAND\_TEACHERS | String-constant | private |  | This is a select statement, to select all teachers in the teacher table  SELECT \* FROM Teachers |
| SQL\_COMMAND\_STUDENTS | String-constant | Private |  | This selects all students from the student table as well as all scores from the score table and joined them to the students where score ID is equal to student ID  SELECT \* FROM Students Inner join Scores on Students.StudentId = Scores.ScoreID Order by StudentId ASC |
| Sql\_Command\_Login | String-constant | Private |  | This is a where statement to check if a password and username exist in a database from either Users table or student Users table.  where Username = @User and Pass = @Pass |
| SQL\_COMMAND\_SCORE | String - constant | private |  | A select statement to select all studetns in the student table and all scores where score ID is equal to StudentID.  select \* From Scores Inner join Students on Students.StudentId = Scores.ScoreID |
| DaTblTeachers | OleDbDataAdapter | Private |  | Used as a bridge to the backend database to lift data from the backend dataset, it is used in conjunction with SQL\_COMMAND\_TEACHERS to lift information from the teacher table. It includes certain SQL commands. This then fills the data table TblTeachers. |
| DaTblStudents | OleDbDataAdapter | Private |  | This is used in conjunction with SQL\_COMMAND\_STUDENTS to lift information from the backend database tables students and scores to fill the data table TblStudents. |
| DaTblScores | OleDbDataAdapter | private |  | This is used in conjunction with the SQL\_COMMAND\_SCORE to lift data from the scores table in the backend data base to fill the data table TblScores. |
| TblStudents | DataTable | private |  | This will hold table information for students that has been retrieved from the backend. |
| TblTeachers | DataTable | private |  | This will hold table information for teachers that has been retrieved from the backend |
| TblLogin | DataTable | private |  | This will hold table information for Login that has been retrieved from the backend |
| TblScores | DataTable | private |  | This will hold table information for Scores that has been retrieved from the backend |
| Data | Constructor | Public | String username  String password  String table | This is where data adapters are created from here information from the backend is input into different data tables based of the commands and adapter. It also checks admin account and checks if login is correct. |
| TempLoginSelect | String | Public |  | This is an empty that is filled which will be used to create a SQL command to select information depending of on if it’s a student usernames table or teacher username table. |
| conn | SqlConnection | Public |  | This establishes a connection between the backend database and frontend based on a connection. |
| TempID | String | public |  | Used to check if column values returned are not null this prevents the user login without correct username and password |
| cmd | SqlCommand | Public |  | Command created from conn and TemploginSelect to create a select command and where it’s connecting to. |
| DaTblLogin | SqlDataAdapter | Public |  | Connection to database based on cmd |
| GetSqlCommand() | String | public |  | Method to return SQL\_COMMAND\_TEACHERS |
| ReturnTeachersTbl() | DataTable | public |  | Method to pass data table TblTeachers when called by other objects. |
| ReturnTeacherAdapter() | OleDbDataAdapter | Public |  | Method to return the OleDbDataAdapter DaTblteachers to other objects. |
| ReturnStudentsTbl() | DataTable | public |  | Method to pass data table TblStudents when called by other objects. |
| ReturnStudentAdapter() | OleDbDataAdapter | Public |  | Method to return the OleDbDataAdapter DaTblStudents to other objects. |
| ReturnScoresTbl() | DataTable | public |  | Method to pass data table TblScores when called by other objects. |
| ReturnScoresAdapter() | OleDbDataAdapter | Public |  | Method to return the OleDbDataAdapter DaTblScores to other objects. |
| ReturnLoginTbl() | DataTable | public |  | Method to retun the DataTable TblLogin to other objects. |

### Login Form

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identifier | Data type | Scope | Parameters passed | Description |
| Success | bool | Public |  | When Creating the object Student it needs to check if an exception in caught. |
| LoginBTN | Button | private |  | On click event, attempts to log into the backend with username and password provided.  If false allows user to re-try. |
| PasswordTXT | Textbox | public |  | Where user can input password. |
| UsernameTXT | Textbox | Public |  | Takes username input. |
| Students | Data | Public |  | This is the object from the class Data.  Uses users password and username to try and connect to the backend. |
| FrmTeacher | TeacherFrm | public |  | If username and password are correct then the user proceeds to the next form when logging in. |
| BTNHome | Button | private |  | Onclick event return to home page. |
| Newform | Form1 | public |  | Return to Form1. |
| BtnHome\_Click  LoginBTN\_Click | Event Proceedure | private | object sender, EventArgs e | All on click event procedure described by buttons |

### TeacherFrm

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identifier | Data type | Scope | Parameters passed | Description |
| Students | Data | Private |  | This is initially null, the data object created in LoginFrm is passed into this |
| TempTableTeachers | DataTable | Private |  | Hold temporary table info of the Teachers table from the backend to be manipulated. |
| TempTableStudents | DataTable | Private |  | Hold temporary table info of the Students table from the backend to be manipulated. |
| TempDaTblStudents | OleDbDataAdapter | Private |  | Temporary connection to filter the TempTableStudents Into students into the class the dependant on which teacher has logged in. |
| TempDaTableTeachers | OleDbDataAdapter | Private |  | Temporary connection to filter the TempTableTeachers Into students into the class the dependant on which teacher has logged in. |
| Connection | String | Private |  | This will be used when doing the aggregate, needs to connect to backend.  Data Source=.\SQLEXPRESS; Initial Catalog=ALevelDatabase; User ID=sa; Password=Pa$$w0rd |
| Username | string | Private |  | Used to check if admin account is being used |
| Password | String | Private |  | Used to check if admin account is being used. |
| TeacherFRM | Constructer | public | Data inStudents, string inUserName, string inPassword | Constructs form and adds value to Students, UserName and Password |
| TeacherFrm\_Load | Void - procedure | private |  | Instructions carried out when the form loads |
| Multiples | Int[] – int array | Public |  | A set of numbers to be added into the combo box CMBMuiltiples |
| FilterTbl | Void – procedure | Public | DataTable Table, OleDbDataAdapter DaTbl, string Sql, int? Filter | Filters a datatable based on Sql statement and what to filter to, e.g. select \* from Teachers where TeacherID =?. Used to select the teachers and all students in that teachers’ class. |
| TXTId | TextBox | private |  | This is used to display the Id of the the student currently being viewed |
| TxtFirstName | TextBox | private |  | This is used to display the first name of the student currently being viewed |
| TxtSurname | TextBox | private |  | This is used to display the surname of the student currently being viewed |
| TxtClass | TextBox | private |  | This displayed the class of the student currently being viewed |
| TXT\_Score | TextBox | private |  | This is displayed the selected muiltiple score of the current student being viewed |
| Txt\_AllScores | TextBox | private |  | This displayes the the Class Average score based on the current muiltiple chosen |
| CMBMuiltiple | Combobox | Private |  | This allows the teacher to select which multiple score they would like to view of the students |
| BTN\_AddTeacher | Button | Private |  | On click this displays the form AddTeacher. It is only accessible when the Admin account is accessed. |
| BTNNewStudent | Button | private |  | On click this displayes the form AddStudent |
| BTN\_Remove | Button | private |  | On click this removes the current record being viewed from the students table. It does this based on ID. A message box is displayed to ask if they would commit to the change. |
| Result | DialogResult | Public |  | Store the result from user decision from a message box |
| Connection | SqlConnection | Public |  | Connection to Database AlevelDatabase |
| connectionString | string | public |  | Path to backend  Data Source=.\SQLEXPRESS; Initial Catalog=ALevelDatabase; User ID=sa; Password=Pa$$w0rd |
| CMD | SqlCommand | public |  | Command made from SqlConnection and SqlCommand |
| Form | TeacherFrm | Public |  | Once student is deleted form must be re-opened to update changes |
| BTNPrevious | Button | private |  | On click decreases the studentId selected by one |
| BTNNext | Button | private |  | On click increases the StudentId by one |
| BTN\_Change | Button | Private |  | On click this updates the multiple selected for the score and average score based on what is in the combo box – CMBMuiltiple. |
| Column | String | Private |  | Column header for new muiltiple to be selected |
| ChangeAVG() | void | private |  | Updates the average score Id based on the new multiple selected |
| CommandText | string |  |  | Sql select average command which is an aggregate function  Select AVG(" + string.Concat("Score", CMBMuiltiple.SelectedItem) + ") from Scores inner join Students on Students.StudentId = Scores.ScoreID where Class =" + TxtClass.Text |
| ScoreAverages | DataTable | private |  | New dattbale to hold the average score of the column selected |
| insertSQL | SqlConnection | private |  | New connection to backend database ALevelDatabase |
| myCommand | SqlCommand | private |  | New SQL command based on the connection and command insertSql and CommandText |
| DaTblAVG | SqlDataAdapter | private |  | To fill the datatable ScoreAverages |
| BTN\_Back | Button | private |  | On click takes user back to form 1 |
| newform | Form1 | public |  | Takes user back to Form1 |
| BTN\_GRD\_VIEW | Button | private |  | On click takes to grid view |
| BTNSave\_Click BtnPrevious\_Click BtnNext\_Click BTNNEWStudent\_Click BTN\_REMOVE\_Click BTN\_Change\_Click BTN\_AddTeacher\_Click BTNBack\_Click BTN\_GRD\_View\_Click | Event Procedure | private | object sender, EventArgs e | All on click event procedure described by buttons |

### AddTeacher

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identifier | Data type | Scope | Parameters passed | Description |
| Teachers | Data | Private |  | Passed in object (data) from Teacher form |
| AddTeacher | constructer | Public | Data inTeacher | Creates the form initializes the form |
| TXT\_Firstname | Textbox | private |  | Where the first name of the new teacher in input |
| TXT\_surname | Textbox | private |  | Where the surname of the new teacher is input |
| TXT\_Class | Textbox | Private |  | Where the class of the new teacher is input |
| TXT\_Password | Textbox | Private |  | Where the new password for the teacher is input |
| BTNADD | button | private |  | Commits new teacher to the backend |
| connectionString | string | Public |  | The connection path and data source to the backend  Data Source=.\SQLEXPRESS; Initial Catalog=ALevelDatabase; User ID=sa; Password=Pa$$w0rd |
| NewTeacherInsert | string | public |  | SQL command inserts a new record into Teacher table  INSERT INTO Teachers([First Name], [Surname Name], Class) values ( @FirstName, @Surname,@class) |
| NewLoginInsert | string | Public |  | SQL command inserts new Username and password for teachers into table – users  Insert INTO USERS values (@UserName, @Pass) |
| insertSQL | sqlConnection | public |  | Connection using ConnectionString to connect a connection to the backend |
| myCommand | SqlCommand | Public |  | Command NewTeacherInsert and insertSQL connected |
| myCommand | SqlCommand | Public |  | Command NewLoginInsert and insertSQL connected |
| Hash | Funtion - string | Public | String Text | This Hashes the string to before inserted into the backend |
| HashString | string | Public |  | Once each value of thestring is hashed it is added into the string |
| MyBytes | Byte[] | Public |  | Hold in input string split into bytes and as char |
| HashText | SHA256Managed | Public |  | This is a secure hashing algorithm generating a unique 32-byte password |
| HashedValues | Byte[] | Public |  | Once Mybytes is hashed each byte is added and held in to later to “reassembled” as a string |
| CreateUserName | Function - string | private | string FirstName, string Surname, List<string> CurrentlyUsed | This creates a unique username from firstname and lastname |
| UserName | string | Public |  | String to construct new username in |
| temp | int | Public |  | Added on to the end on the new username |
| Existing | bool | public |  | Check is the username is pre-existing |
| FindColumnValues | List<string> | public | String column | Uses column passed to all entists in the column from the table Users |
| Temptbl | Datatable | Public |  | Used to temporarly hold values from select |
| TempDaTbl | OleDbDataAdapter | public |  | Set connection to database |
| Command | String | public |  | SQL command to select all from table Users  select \* from Users |
| Connection | string | Public |  | Connection  Provider=SQLOLEDB; Data Source=.\SQLEXPRESS; Initial Catalog=ALevelDatabase; User ID= sa; Password= Pa$$w0rd |
| ListFromColumn | List<string> | public |  | Holds list of strings from selected column |
| BTNADD\_Click | Event procedure | public | object sender, EventArgs e | On click event described by button |

### Form – AddStudent

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identifier | Data type | Scope | Parameters passed | Description |
| Students | Data | private |  | Hold the object Data passed in |
| Class | int | Private |  | Holds which class the teacher is in, than that student is added to that class |
| AddStudent | Constructer | public | Data inStudents, int inClass | Creates the form |
| TXTSurname | Textbox | Public |  | Where surname is input |
| TXTForename | Textbox | Public |  | Where forename is input |
| TXTPassword | Textbox | public |  | Where the password is input |
| BTNAdd | Button | public |  | On click commits the new student to the backend using input forename and surname. |
| connectionstring | string | public |  | Path of connection to backend  Data Source=.\SQLEXPRESS; Initial Catalog=ALevelDatabase; User ID=sa; Password=Pa$$w0rd |
| CommandText | string | Public |  | SQL insert into the table Students, using first name, surname and the class the teacher is adding from.  INSERT INTO Students(FirstName, Surname, Class) values ( @FirstName, @Surname,@class) |
| CommandText2 | String | Public |  | SQL Insert into Scores Table with blank scores for each column  Insert into Scores (Score2,Score3, Score4,Score5,Score6, Score7,Score8,Score9) values(0, 0, 0, 0, 0, 0, 0, 0) |
| CommandText3 | string | public |  | SQL Insert into StudentUsers table with username and password.  Insert into StudentUsers values (@UserName, @Pass) |
| insertSQL | SQLConnection | public |  | Connection to database |
| MyCommand | SqlCommand | public |  | New command using CommandText and the insertSQL connection |
| MyCommand2 | SqlCommand | public |  | New command using CommandText2 and the insertSQL connection |
| MyCommand3 | SqlCommand | public |  | New command using CommandText3 and the insertSQL connection |
| Hash | Funtion - string | Public | String Text | This Hashes the string to before inserted into the backend |
| HashString | string | Public |  | Once each value of thestring is hashed it is added into the string |
| MyBytes | Byte[] | Public |  | Hold in input string split into bytes and as char |
| HashText | SHA256Managed | Public |  | This is a secure hashing algorithm generating a unique 32-byte password |
| HashedValues | Byte[] | Public |  | Once Mybytes is hashed each byte is added and held in to later to “reassembled” as a string |
| AddStudent\_leave | void | private |  | On leave opens the Teachers form |
| CreateUserName | Function - string | private | string FirstName, string Surname, List<string> CurrentlyUsed | This creates a unique username from firstname and lastname |
| UserName | string | Public |  | String to construct new username in |
| temp | int | Public |  | Added on to the end on the new username |
| Existing | bool | public |  | Check is the username is pre-existing |
| FindColumnValues | List<string> | public | String column | Uses column passed to all entists in the column from the table Users |
| Temptbl | Datatable | Public |  | Used to temporarly hold values from select |
| TempDaTbl | OleDbDataAdapter | public |  | Set connection to database |
| Command | String | public |  | SQL command to select all from table Users  select \* from StudentUsers |
| Connection | string | Public |  | Connection  Provider=SQLOLEDB; Data Source=.\SQLEXPRESS; Initial Catalog=ALevelDatabase; User ID= sa; Password= Pa$$w0rd |
| ListFromColumn | List<string> | public |  | Holds list of strings from selected column |
| BTNADD\_Click | Event procedure | public | object sender, EventArgs e | On click event described by button |

### Form-StudentFrm

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identifier | Data type | Scope | Parameters passed | Description |
| Using Object\_Orentated\_ProtoType1 |  |  |  | This allows for the referencing of the XNA game, so is able to be open within a new thread |
| Students | Data | private |  | Object, holds backend data, On attempting instantiation if error occurs doesn’t allow login |
| StudentFrm | constructer | Public |  | Creates the form |
| TXT\_UserName | Textbox | Private |  | Where the user inputs username |
| TXT\_Password | Textbox | private |  | Where the user inputs there password |
| BTN\_Login | Button | private |  | On click checks username and password with backend if valid takes the user to the game |
| BTNHome | Button | private |  | Returns user to Form1 |
| Success | bool | public |  | Validation for whether username and password are correct |
| WFile | StreamWriter | Public |  | This writes the first name, surname and current scores to a text file, to be read by the XNA project to know which user is using the game. This hass been used because values where unable to be passed into the XNA game so must be read from a text file |
| t | thread | Public |  | This opens the XNA game with a new thread using the different file path. |
| StartGame | Void - proceedure | public |  | This creates and starts Game1, it is called by the t to be opened. |
| game | Game1 | public |  | New instance of Game1 when clicked to be opened. |
| FilterTbl | Void – procedure | Public | DataTable Table, OleDbDataAdapter DaTbl, string Sql, int? Filter | Filters a datatable based on Sql statement and what to fiter to, e.g. select \* from Teachers where StudentId =?. Used to select the student that has logged in. |
| BTNhome | Button | private |  | On click returns user to Form1 |
| newform | Form1 | public |  | New instance of Form1. |
| BTN\_LOGIN\_CLICK  BTN\_HOME\_CLICK | Event procedure | public | object sender, EventArgs e | On click event described by button |

### ViewDisplayForm

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identifier | Data type | Scope | Parameters passed | Description |
| Scores | DataTable | Public |  | Holds all the students’ scores to be displayed and Student ID |
| GRD | DataGrid | Public |  | Displays all students’ scores, to allow for comparison |

## Game side – Students

### Game1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identifier | Data type | Scope | Parameters passed | Description |
| Using.windows.Forms |  |  |  |  |
| Using Data.sqlclient |  |  |  |  |
| Using Data.OleDb |  |  |  |  |
| GameState | Enum - enumeration | Public |  | This keeps a game state between PauseScreen and Playgame |
| SCREEN\_WIDTH | Constant Int | Private |  | How many pixels wide the game is |
| SCREEN\_HEIGHT | Constant int | Private |  | How many pixels high the game is |
| graphics | GraphicsDeviceManager | Private |  | Rendering and management of graphics when drawing |
| spriteBatch | SpriteBatch | Private |  | Used for drawing 2D textures |
| LogSprite1List | List<Log> | Private |  | This is a list of all logs the player can possibly jump on |
| FrogSprite | Frog | Private |  | This is the object frog aka the player |
| TBackGround | Texture2D | Private |  | Texture for the background of the game |
| BackGround | Rectangle | Private |  | The size of the background |
| TSplashScreen | Texture2D | Private |  | Texture of the pause screen |
| SplashScreen | Rectangle | Private |  | Size of the pausescreen |
| Not\_In\_River | Bool | Private |  | Check if frog is in the river |
| gamestate | GameState | Private |  | An enumerated state for whether the game is paused and displaying the pause screen or playing the game and displaying the play game screen |
| MyKeys | KeyboredState | Private |  | Gets key board input |
| LogFont | SpriteFont | Private |  | Font used on top of logs to display which number is attracted to the log |
| PauseScreenfont | SpriteFont | Private |  | Font used to display text on the pause screen |
| GameOver | Bool | Private |  | Validation to see if the player still lives has left to continue to play the game |
| MuiltipleWorkingOn | Int | Private |  | Becomes the multiple the user wants to work on |
| Rnd | Random | Private |  | Used to readjust a row if there isn’t a possible solution |
| OldKeys | Keyboardstate | Private |  | Get keyboard input only allows a key to be clicked once |
| LeftButton | Button | Private |  | Reduces MuiltipleWorkingOn when clicked |
| RightButton | Button | Private |  | Increase MuiltipleWorkingOn when clicked |
| StartButton | Button | Private |  | Changes the gamestate from paused screen to play game when clicked |
| NewStudent | Student | Private |  | New object |
| SetCompontents | Bool | Private |  | Checks if objects need to be re-loaded when changing from pausescreen to play game or if the user completed given circuit |
| OnceNewStudent | Bool | Private |  | Checks if aa new student needs to be created |
| LevelSpeed | Int | Private |  | How fast logs are moving and the level the student is on, increased by one every 150 points. |
| Game1 | Constucter | public |  | Creates a new instance of Game1 sets the screen size from the SCREEN\_WIDTH and SCREEN\_HEIGHT |
| Initialize | Void | protected |  | Initializes new objects that have been loaded by LoadContent() and query services |
| LoadContent | void | protected |  | Loads any textures and fonts reads file |
| OnExiting | void | protected |  | When the game is exited this is called, it will save, scores achieved by the pupil |
| DaTblStudents | OleDbDataAdapter | Public |  | Adapter to connect to the database |
| TblStudents | Datatable | public |  | Hold information gathered from the database |
| FirstName | string | Public |  | Used in conjunction with surname to save records to the correct student |
| Surname | String | public |  | Hold logged in student surname |
| temp | string | public |  | Sql column holder to run through all data columns from Score2 - score9 saving relevant information to each |
| SQL\_COMMAND\_STUDENTS | string | Public |  | “Select \* from Students Inner join Scores on Students.StudentId = Scores.ScoreID where FirstName = '" + FirstName + "' and Surname = '" + Surname + "'"  Selects all students and scores |
| CONNECTIONSTRING | String | Public |  | Connection path  Provider=SQLOLEDB; Data Source=.\SQLEXPRESS; Initial Catalog=ALevelDatabase; User ID= sa; Password= Pa$$w0rd |
| CONNECTIONSTRING2 | string | Public |  | Connection Path  Data Source=.\SQLEXPRESS; Initial Catalog=ALevelDatabase; User ID=sa; Password=Pa$$w0rd |
| CommandText | string | public |  | update Scores set Score2 = @Score2, Score3 = @Score3, Score4 = @Score4, Score5 = @Score5, Score6 = @Score6, Score7 = @Score7, Score8 = @Score8, Score9 = @Score9 where ScoreID = @ID  update statement to Updata all scores |
| insertSQL | SQLConnection | Public |  | New SQL connection using ConnectionString2 |
| myCommand | SQLCommand | public |  | Command using Commandtext and insertSQL connection to update all scores in the scores column on the specific ID |
| Update | void | protected |  | Runs through the game 60 times a second, running game logic. Where pause screen and play game are updated and the game is played |
| ValidationOfLogs | void | private | List<Log> ListOfLogs, int ValidateTo | This runs through row by row checking there is a solution log on each row. If a row doesn’t have any possible solutions it will change the first log to one that is. It does this by taking a random number and times it by validateTo. |
| Row\_Find | List<Log> - function | private | List<Log> Inital\_Log, int Row | If the index of the object mod 5 is equal to the row to find, it is added to a new list<log> then returns that list after running through entire list of initial lists of objects. |
| RowToFind | List<Log> | public |  | List of logs to be added to every time the index of the object mod 5 equals row to find. |
| Row\_Valid | Bool - function | private | List<Log> Row, int inValid | Runs through the list of log objects, check if any of there numbers are divisible by multiple picked by the student. Returns true if is returns false if none. |
| Valid | bool | public |  | Returned, set to true if one of the objects number is a solution. |
| Make\_Possible | void | private | List<Log> ListToChange, int ValidFor | Changes the first object logs number into a solution. |
| PauseScreenUpdate | void | private | GameTime gametime | Check if any buttons are being moussed over of clicked. |
| MousePos | MouseState | public |  | Input from mouse |
| PlayGame | void | private | GameTime gametime | Changes level speed based on students level. Moves the Game object Logs. Calls object updates, checking collisions. Checks lose state, and if the game should be over. |
| Draw | void | protected | GameTime gametime | Renders the different screens, rendering the pause screen or the play game based on the gamestate. |
| DrawPauseScreen | void | Private |  | Draws splash screen, buttons, and welcome text and pick multiple to work on text. |
| DrawGame | void | private |  | Draws players sprite, the background of the game, the score and level and log objects. |

### Class – Student

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identifier | Data type | Scope | Parameters passed | Description |
| Firstname | String | Private |  | First name of the student playing |
| Surname | String | private |  | Surname of the student playing |
| LivesLeft | Int | Private |  | How many lives the student still has |
| Scores | List<string> | private |  | Holds scores for each multiple |
| Level | int | private |  | The Level the student is on |
| student | Constructor | public |  | Set livesleft to 3 when created |
| ReadTextfile | void | public | String File | Reads information from text file, giving first and last name and scores |
| temp | string | public |  | Add need string of score to Scores |
| RFile | StreamReader | public |  | To read the text file |
| ResetLifes | void | public |  | Resets LivesLeft to 3 |
| RemoveLife | void | public |  | Minus one to LivesLeft |
| GetLifes | Int – function | public |  | Returns livesLeft |
| GetFirstName | String - Function | public |  | Returns FirstName |
| GetSurname | String-function | public |  | Retuns Surname |
| IncreaseScore | void | public | int Muiltiple, int IncreaseBy | Increase score of given multiple by IncreaseBy |
| temp | int | public |  | Temporally holds score to increase as a int to increase by score increase. The returns to string. |
| DrawWelcome | void | public | SpriteBatch spritebatch, SpriteFont Font, int X, int Y, int Muiltiple | On pause screen welcome statement and multiple currently selected to work on. |
| DrawScore | void | public | SpriteBatch spritebatch, SpriteFont Font, int Muiltiple | In play game screen draws Score of student, Lifes Left and Level |
| GetScore | Int - function | public | int Muiltiple | Returns score of requested multiple |
| GetLevel | Int - function | public | int inMuiltiple | Returns Level based on the score of the multiple. |

### Number - Class

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identifier | Data type | Scope | Parameters passed | Description |
| Number | int | private |  |  |
| Rnd | Random | public |  | Random number |
| SetNewNumber | void | public | int inNumber | Changes Number to inNumber |
| SetMuiltiple | void | public | Int TestNumber | Randomly generates a new number with a change of being a multiple of TestNumber between 1 and 10 times. Sets Number equal to this new number. |
| GetNumberInt | Int - function | public |  | Returns Number. |

### Log - class

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identifier | Data type | Scope | Parameters passed | Description |
| Sprite\_Texture | Texture2D | private |  | Texture of the sprite |
| Velocity | Vector2 | Private |  | Speed in positive direction |
| rLog | Regtangle | Private |  | Position and size |
| number | Number | Private |  | Calls Numbers class, and creates new object inside |
| Rnd | Random | Public |  | Random number |
| CollisionNumber | int | Private |  | The multiple the student has picked to work on |
| NotJumpedOn | Bool | Private |  | Whether the log has already been jumped on, this means score cant be gained from jumping on the same log twice. |
| LoadContent | void | public |  | Called in loadcontent of main game. Where sprite texture is set and RLog and number is create. |
| SetCollisionNumber | Void | public | Int Multiple | Where number is set, multiple picked by the student is known to allow for possibility of number being a multiple |
| Draw | void | public |  | Called in Draw in Game1 the log is rendered and number is put on top indicating if it is a multiple or not. |
| Update | Void | public | GameTime gametime, int ScreenWidth, int Speed | Called from game1 In update, it moves the logs in direction based on speed. Also checks if log has moved of the screen then loops it bas=ck to the other side. |
| Negative | bool | public |  | Checks if the log is moving in a positive direction (right) or negative one (left) |
| ChangeforValidation | Void | public | int ValidationNumber, int inCollitionNumber | Resets the logs numbers to be a multiple of the collision number which is the multiple the student chose. |
| ReturnRec | Rectangle – function | public |  | Returns RLog |
| LogSpeed | Int – function | public |  | Returns Velocity.x |
| GetNumber | Number – function | public |  | returns number |
| GetJumpedOn | Bool – function | public |  | Returns true or false |
| SetJumpedOn | void | public | bool inJumpedOn | Sets weather the logs has been jumped on. |

### Frog-class

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identifier | Data type | Scope | Parameters passed | Description |
| FROG\_WIDTH | Int | private |  | Width of sprite |
| FROG\_HEIGHT | int | private |  | Height of sprite |
| Sprite\_Texture | Texture2D | private |  | Texture of sprite |
| OldKeys | KeyBoardState | private |  | Gets keyboard input |
| MyKeys | KeyBoardState | private |  | Gets ketboard input |
| origin | Vector2 | private |  | Positon of the center of the sprite |
| Rotation | Float | private |  | Rotation of sprite |
| rFrog | rectangle | private |  | Position and size of sprite |
| FrogInsect | rectangle | private |  | Smaller rectangle where collision actually occurs. |
| CollitonNumber | int | private |  | Users chosen multiple to work on |
| LoadContent | void | public | ContentManager theContentManager, string theAssetName, Vector2 InitalPosition | Called in load content game1, from parameters passed, creates rFrog and loads the Frog texture |
| CollisionNumberPlus | void | public |  | Adds one to collision number |
| CollisionNumberMinus | void | public |  | Removes one from collisionnumber |
| SetcollitionNumber | void | public | int inNumber | Set CollisionNumber equal to inNumber |
| Draw | void | public | SpriteBatch SpriteBatch, SpriteFont Font | Called from Draw in Game1, renders the frog sprite, during Playgame |
| Update | void | public | GameTime gametime, int ScreenWidth, int ScreenHeight | Called in update game1, Where game logic is and the movement of the frog sprite. |
| Collision | Bool - function | public | List<Log> Log\_List1, int Screen\_width, int LogSpeed1, int LogSpeed2, int LogSpeed3, int LogSpeed4, int LogSpeed5, Student student,int muiltiple | Also called from update, this is where colisions are checked between frog and relevant logs. Increases in scores, and wether a log has already been jumped or not. |
| KeyPress | Bool | Public |  | Checks if forwards has been pressed or not |
| index | int | public |  | Used to find row to set as jumped on |
| RowToShowJumpedOn | List<log> | public |  | List of the object log, changes all objects in the list once jumped. |
| Death | void | public | int ScreenHeight, int ScreenLength | Resets sprites position |
| Row\_Find | List<log> - function | public | List<Log> Inital\_Log, int Row | Finds all objects on the row just been jumped on |
| RowToFind | List<log> | public |  | Added to everytime Initial\_Log 20 % index |
| GetRec | Rectangle - function | public |  | Returns rFrog |
| ResetPosition | void | public | int ScreenLength, int ScreenHeight | Resets position of the sprite |

### Class – Button

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Identifier | Data type | Scope | Parameters passed | Description |
| TButton | Texture2D | private |  | Texture of the object |
| RButton | Rectangle | private |  | The size and position |
| MouseState | MouseState | private |  | Mouse input |
| OldMouseState | MouseState | private |  | Old mouse input |
| PressedOrNot | State-own enumeration | private |  | Enumeration between two states |
| Colour | Color | private |  | Colour to highlight when moussed over |
| Change | bool | private |  | Check if button is being moussed over. |
| LoadContent | void | public | ContentManager theContentManager, string inTexture, int PosX, int PosY, int Width, int Height | Loads information about the object, e.g texture and rectangle |
| Draw | void | public | SpriteBatch SpriteBatch | renders the button |
| GetReg | Rectangle - function | public |  | Returns RButton |
| MousePressButton | void | Public | int MouseX, int MouseY | Checks if mouse is scrolling inside button then highlights, and if it has been pressed. |
| mousePosition | Point | public |  | Position of mouse |
| CheckPressed | Bool - function | public |  | Returns if mouse has been pressed or not |

## Entity relationship diagram

Teachers

**TeacherID** First name Surname Class

Student Users

*StudentID* Username Password

Students

**StudentID** First name Surname Class

Scores

***ScoreID*** Score2 Score3 Score4 Score5 Score6 Score7 Score8 Score9

Users

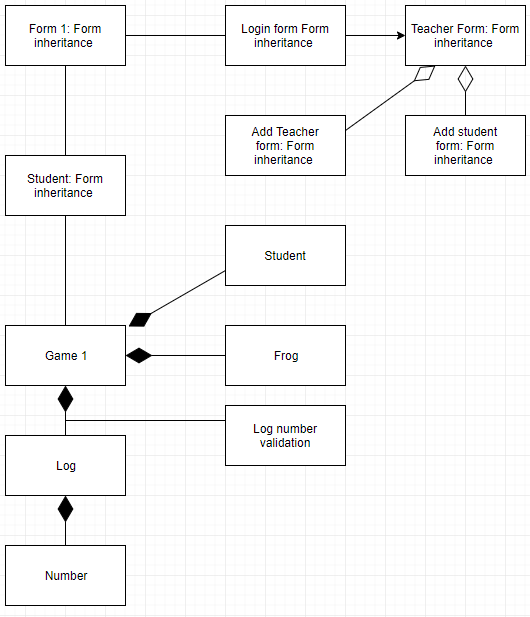
*TeacherId* Username Password

**Bold – Primary key**

*Italics – foreign key*

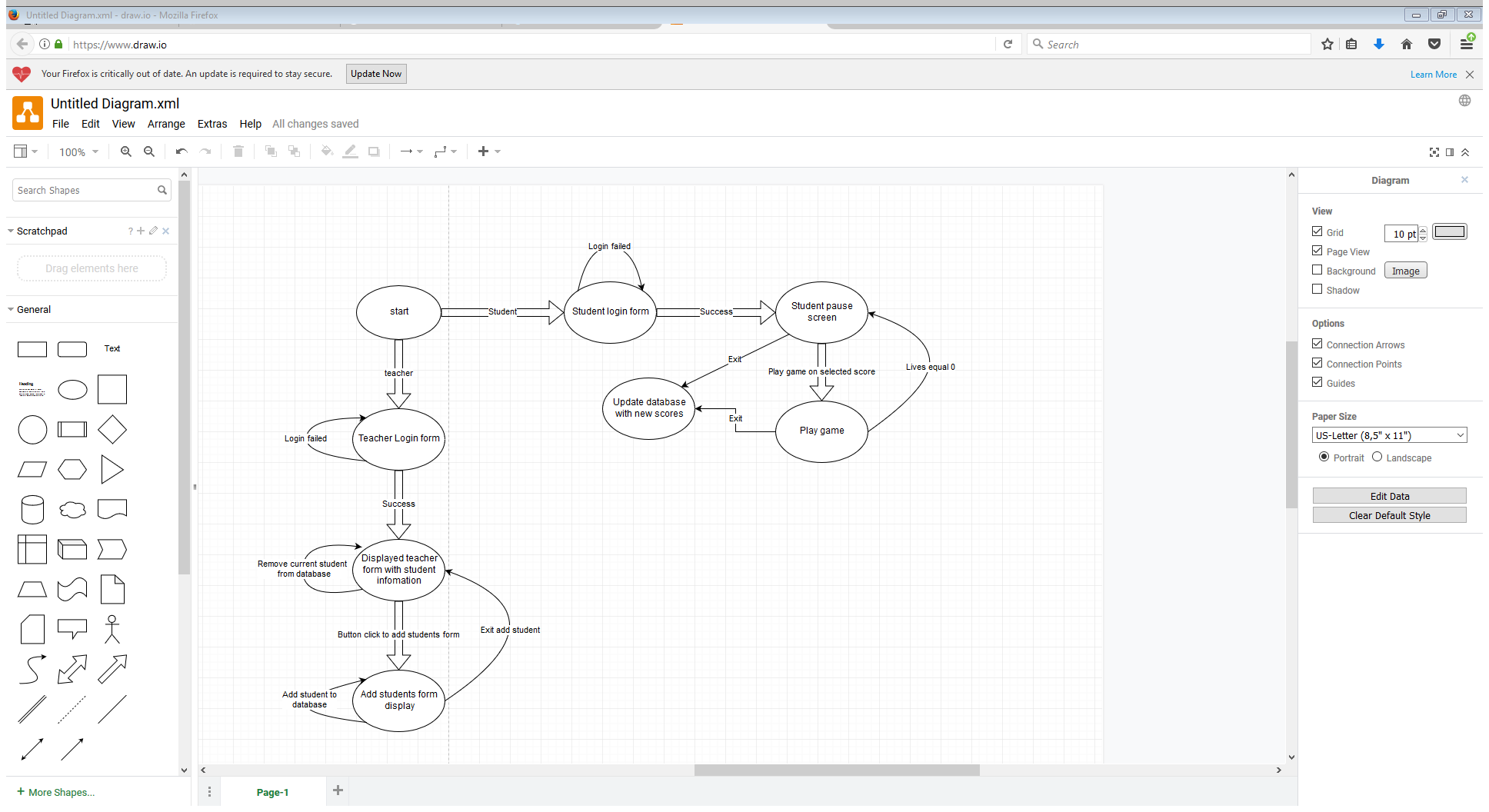
## Class diagram

Data

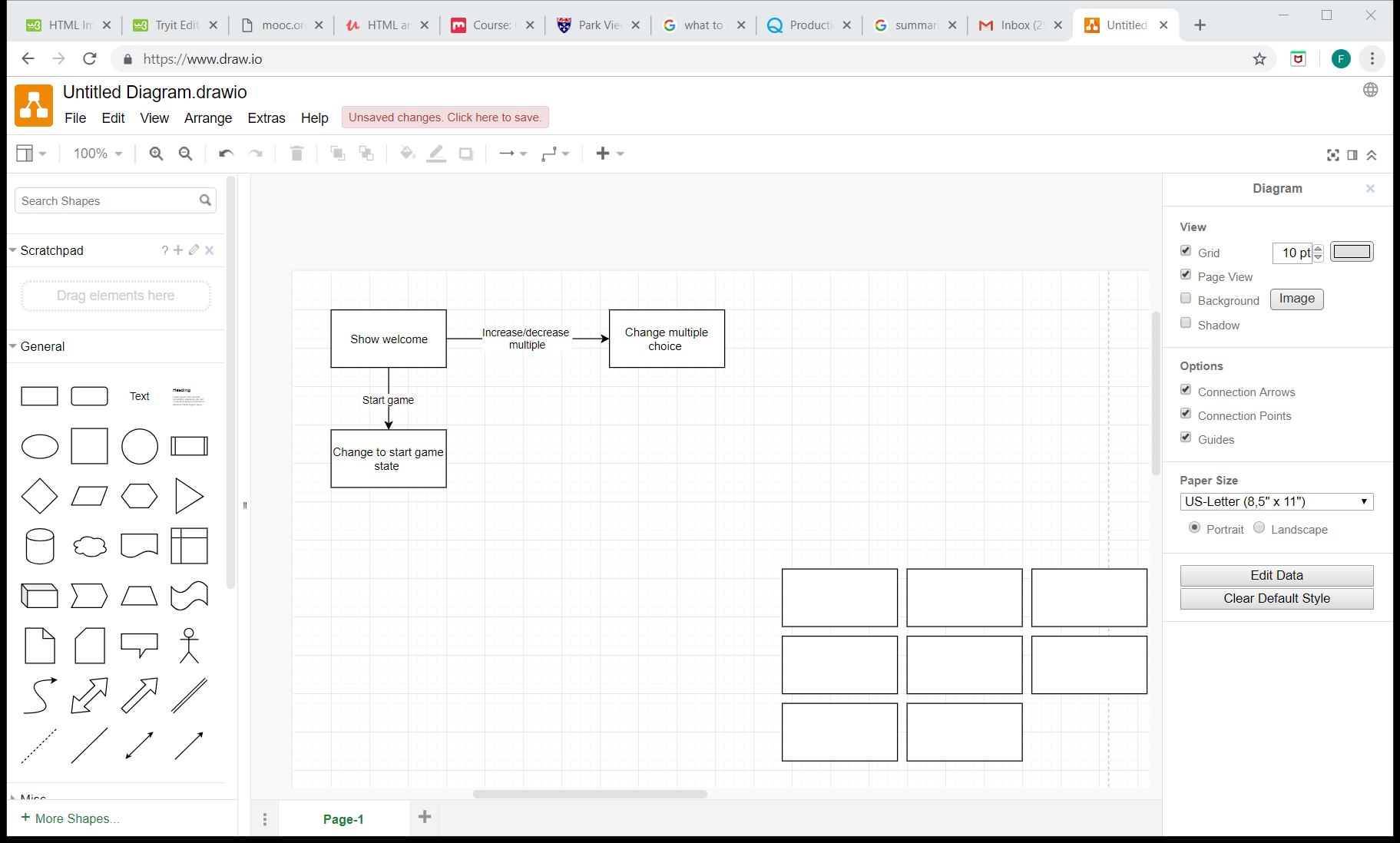


## State transition diagrams

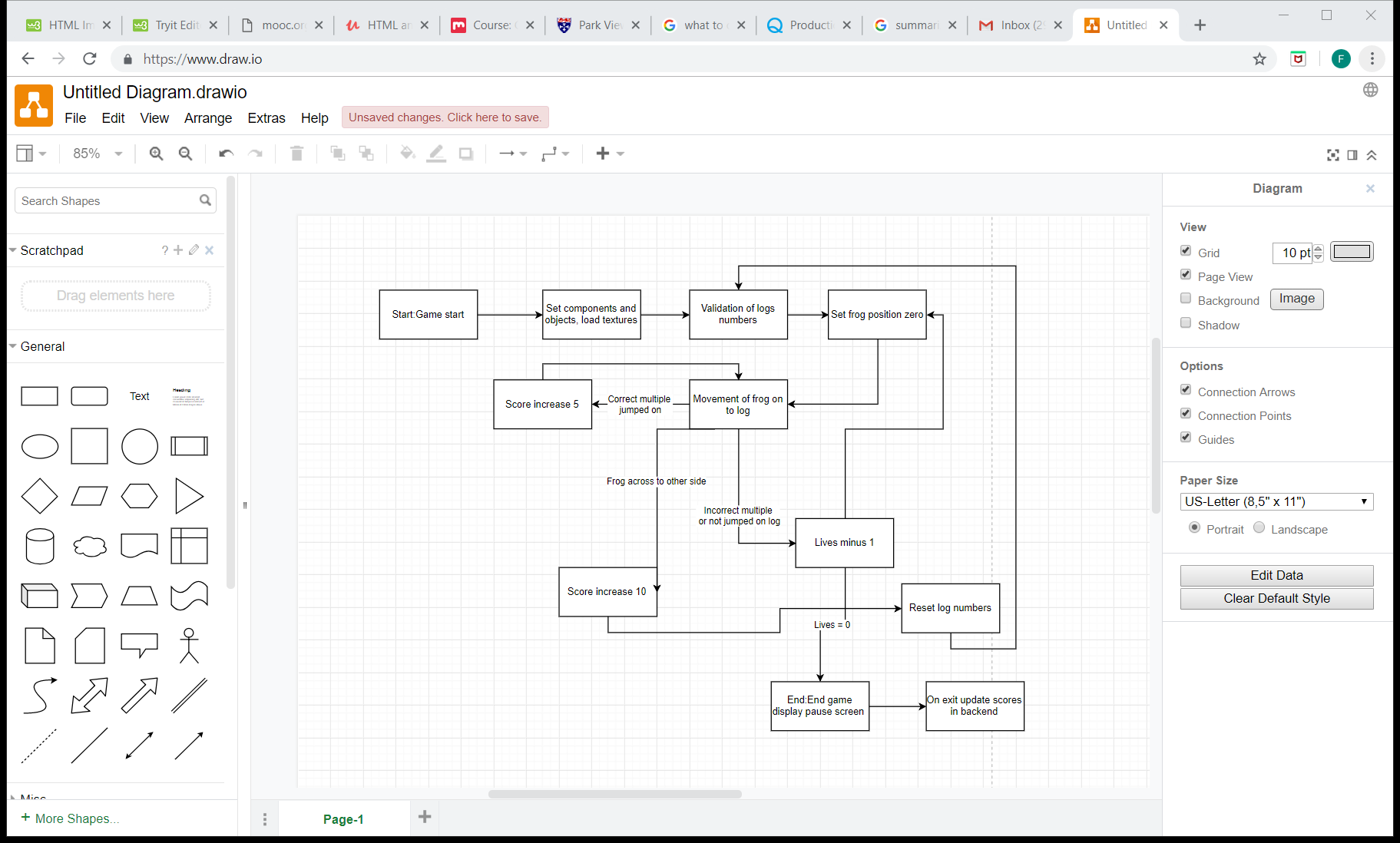
### Overview



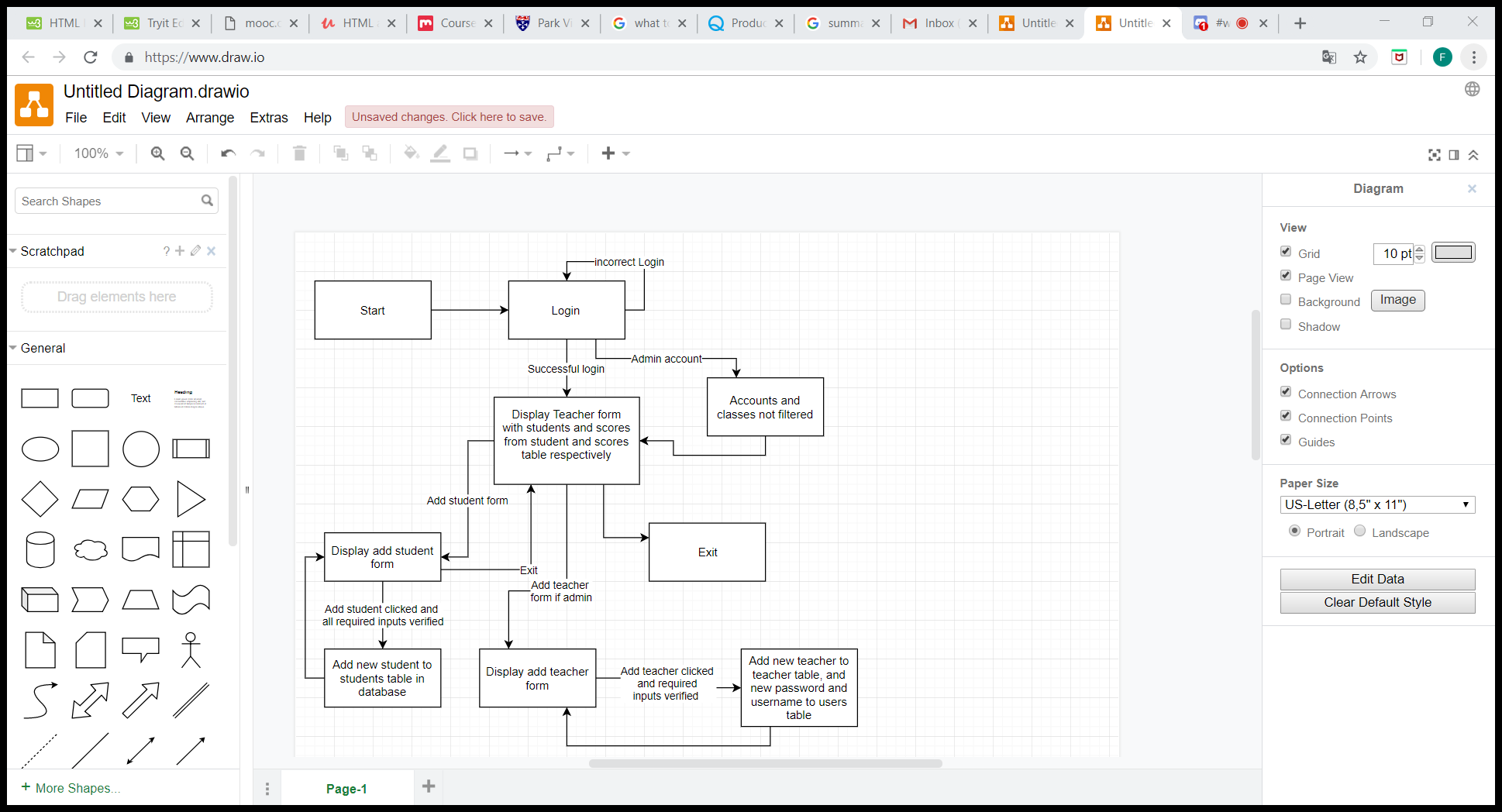
### Pause screen



### Play game



### Teachers side



## OOP design

|  |
| --- |
| Button - class |
| Public LoadContent: procedure  Public Draw: procedure  Public GetRec: Function  Public MousePressButton: procedure  Public CheckPressed: function |
| Private TButton: Texture2D  Private RButton: Rectangle  Private MouseState: MouseState  Private OldMouseState: MouseState  Private PressedOrNot: State  Private Colour: Color  Private Change:bool |

|  |
| --- |
| Frog – class |
| Public LoadContent: procedure  Public Collitionnumber\_Plus: procedure  Public CollitionnumberMinus: procedure  Public SetcollitionNumber: procedure  Public Draw: procedure  Public Update: procedure  Public Collision: Function  Public Death: procedure  Public RowToFind: Function  Public GetRec: Function  Public ResetPosition: procedure |
| Private FROG\_WIDTH: int  Private FROG\_HEIGHT: int  Private Sprite\_Texture: texture2d  Private OldKeys: KeyBoardState  Private MyKeys: KeyBoardState  Private origin: Vector2  Private Rotation: float  Private rFrog: rectangle  Private FrogIntersect: rectangle  Private CollisionNumber: int |

|  |
| --- |
| number |
| Public SetNewNumber: procedure  Public SetMuiltiple: procedure  Public GetNumberInt: function |
| Private number: int  Private Rnd: random |

|  |
| --- |
| Student |
| Public ReadTextFile: procedure  Public Resetlives: procedure  Public RemovesLife: procedure  Public Getlifes: Function  Public GetfirstName: Function  Public GetSurname: Function  Public Setlife: procedure  Public IncreaseScore: procedure  Public DrawWelcome: procedure  Public Drawscore: procedure  Public GetScore: Function  Public GetLevel: Function |
| Private FirstName: string  Private Surname: string  Private Livesleft: int  Private Scores: List<string>  Private Level: int |

|  |
| --- |
| Log |
| Public LoadContent: procedure  Public SetCollitionnumber: procedure  Public Draw: procedure  Public Update: procedure  Public ChangeForValidation: procedure  Public ReturnRec: Function  Public LogSpeed: Function  Public GetNumber: Function  Public GetJumpedOn: Function  Public SetJumpedOn: procedure |
| Private Sprite\_texture: Texture2D  Private velocity: Vector2  Private rLog: Rectangle  Private number: Number  Private Rnd: Random  Private CollisionNumber: int  Private NotJumpedOn: bool |

|  |
| --- |
| Data |
| Public: Data: Constructor  Public: Hash: Function  Public: GetSqlCommand: Function  Public: ReturnTeacherTbl: Function  Public: ReturnTeacherAdapter: Function  Public: ReturnStudentTbl: Function  Public: ReturnStudentAdapter: Function  Public: ReturnScoresTbl: Function  Public: ReturnScoreAdapter: Function  Public: ReturnLoginTbl: Function |
| Private Connection: string  Private Connection2: string  Private SQL\_COMMAND\_TEACHERS: string  Private SQL\_COMMAND\_STUDENTS: string  Private SQL\_COMMAND\_STUDENTS\_FILTER: string  Private sql\_Command\_Login: string  Private SQL\_COMMAND\_SCORE: string  Private DaTblStudents: OleDbDataAdapter  Private DaTblTeachers: OleDbDataAdapter  Private DaTblScores: OleDbDataAdapter  Private TblStudents: DataTable  Private TblTeachers: DataTable  Private TblLogin: DataTable  Private TblScores: DataTable |

|  |
| --- |
| Form 1 : Windows form |
| Private StudentBTN\_Click : procedure  Private TeacherBTN\_Click: procedure |
| Private StudentBTN: Button  Private TeacherBTN: Button |

|  |
| --- |
| LoginFrm: Windows Form |
| Private LoginBTN\_Click: Procedure  Private BTNHome\_Click: Procedure |
| Private LoginBTN: Button  Private BTNHome: Button  Private UsernameTXT: TextBox  Private PasswordTXT: TextBox  Public Success: bool  Public Students: Data  Public FrmTeacher: TeacherFrm  Public newform: Form1 |

|  |
| --- |
| TeacherFrm: Windows Form |
| Public TeacherFrm: Constructor  Private TeacherFrm\_Load: procedure  Public FilterTbl: Procedure  Private BTNSave\_Click: procedure  Private BtnPrevious\_Click: procedure  Private BtnNext\_Click: procedure  Private BTNNEWStudent\_Click: procedure  Private BTN\_REMOVE\_Click: procedure  Private BTN\_Change\_Click: procedure  Private ChangeAVG: procedure  Private BTN\_AddTeacher\_Click: procedure  Private BTNBack\_Click: procedure  Private BTN\_GRD\_View\_Click: procedure |
| Private Students: Data  Private TempTableTeachers: DataTable  Private TempTableStudents: DataTable  Private TempDaTblStudents: OldDbDataAdapter  Private TempDaTblTeachers: OldDbDataAdapter  Private Connection: String  Private Username: String  Private Password: String  Public Multiples: int[]  Public NewFrm: AddStudent  Public result: DiaLogResult  Public Connection: SqlConnection  Public connectionString: string  Public Form: TeacherForm  Public CommandText: string  Public ScoreAverages : DataTable  Public insertSQL: insertSql  Public myCommand: SqlCommand  Public DaTblAVG: SqlDataAdapter  Public newform: AddTeacher  Public newform: Form1  Public newform: ViewDisplay  Private BTNSave: Button  Private BtnPrevious: Button  Private BtnNext: Button  Private BTNNEWStudent: Button  Private BTN\_REMOVE: Button  Private BTN\_Change: Button  Private BTN\_AddTeacher: Button  Private BTNBack: Button  Private BTN\_GRD\_View: Button  Private TxtId: TextBox  Private TxtFirstName: TextBox  Private TxtSurname: TextBox  Private TxtClass: TextBox  Private TXT\_Score: TextBox  Private TXT\_AllScores: TextBox  Private CMBMuiltiple: ComboBox |

|  |
| --- |
| AddStudent: Windows Form |
| Public AddStudent: Constructor  Private BTNAdd\_click: Procedure  Public Hash: Function  Private AddStudent\_Leave: procedure  Private CreateUserNamePassWord: function  Public findColumnValues: function |
| Private Students: Data  Private Class: int  Public ConnectionString: string  Public commandText: string  Public CommandText2: String  Public CommandText3: String  Public myCommand: SqlCommand  Public myCommand2: SqlCommand  Public myCommand3: SqlCommand  Public HashString: string  Public MyBytes: byte[]  Public HashText: SHA256Managed  Public HashedValues: byte[]  Public NewTeacher: TeacherFrm  Public UserName: string  Public temp: int  Public Existing: bool  Public Temptbl: DataTable  Public TempdaTbl: OleDbDataAdapter  Public Command: String  Public connection: String  Public ListFromcolumn: List<string>  Public notnull: bool  Public count: int  Private BTNAdd: Button  Private TXTForname: TextBox  Private TXTPassword: Textbox  Private TXTSurname: Textbox |

|  |
| --- |
| AddTeacher: Windows Form |
| Public AddTeacher: Constructor  Private BTNAdd\_click: Procedure  Public Hash: Function  Private AddTeacher\_Leave: procedure  Private CreateUserNamePassWord: function  Public findColumnValues: function |
| Private Teachers: Data  Public ConnectionString: string  Public NewTeacherInsert: string  Public NewLoginInsert: String  Public myCommand: SqlCommand  Public myCommand2: SqlCommand  Public HashString: string  Public MyBytes: byte[]  Public HashText: SHA256Managed  Public HashedValues: byte[]  Public NewTeacher: TeacherFrm  Public UserName: string  Public temp: int  Public Existing: bool  Public Temptbl: DataTable  Public TempdaTbl: OleDbDataAdapter  Public Command: String  Public connection: String  Public ListFromcolumn: List<string>  Public notnull: bool  Public count: int  Private BTNAdd: Button  Private TXT\_FirstName: TextBox  Private TXT\_Password: Textbox  Private TXT\_Surname: Textbox  Private TXT\_Class: Textbox |

|  |
| --- |
| ViewDisplay: windows Form |
| Public ViewDisplay: constructor  Private ViewDisplay\_Load: procedure |
| Private GRD1: DataGrid |

|  |
| --- |
| StudentFrm: windows Form |
| Private BTN\_Login\_Click: procedure  Public StartGame: procedure  Public FilterTbl: Procedure  Private BTNHome\_Click: procedure |
| Private Students: Data  Public Success: Bool  Public WFile: StreamWriter  Public t: thread  Public game: Game1  Public newform: form1 |

## Algorithms/pseudo-code

### Login form

LoginBTN\_Click

If (username.input not null and Password not null)

then

bool success = true

Data Student = null

Try

Student = new data(username.input, password.input, table users)

Catch

Success = false

Massagebox(“incorrect username or password”)

If (success)

Then

New teacher form

Login form.hide

Else

Massagebox(“no input”)

### Data – class

Data: constructor (String username, string password, string table)

String connection

Connection create

Select all from StudentUsers table or Users table where Username = ? and Password = ?

If (rows = 1)

Else

Throw exception

Datatable Students = Select all from students join Scores

Datatable Teachers = select all from Teachers

Hash (string Text): function

Hashstring = null

Bytes bytes = Text.split

HashText = new HashingAlgorithm

HashedValue = HashText.Hash(bytes)

Foreach (value x in HashedValues)

Hashstring += X

Return HashString

### Teacher form

FilterTbl(DataTable Table, OleDbDataAdapter DaTbl, string Sql, int? Filter)

Table.Clear

DaTbl.SelectCommand.parameters.clear

If filter = null

Then

DaTbl.Se;ectCommand.CommandText = “Select \* from Teachers”

Else

DaTbl.selectCommand.CommandTest = Sql

DaTbl.SelectCommand.parameters.add(@FilterTeacherId, int).value = filter

DaTbl.fill(Table)

BTN\_Remove\_Click

Messagebox (“Would you like to continue”, yes no)

If (yes)

Then

If(TextBoxId test = null)

Then

Messagebox(“unable to delete”)

sqlConnection Connection = new SqlConnection

string Connectionstring = “Data Source=.\SQLEXPRESS; Initial Catalog=ALevelDatabase; User ID=sa; Password=Pa$$w0rd”

using (Connection(ConnectionString))

using(SqlCommand CMD)

Connction.Open

Messagebox(“connection made”)

CMD.Command = "DELETE Scores FROM Students INNER JOIN Scores ON Scores.ScoreID = Students.StudentId where Students.StudentId = @ID";

CMD.parameter(“@ID”, TEXTBOXID.Text)

CMD.Exectute

Messagebox.show(“USER:” TEXTBOXFIRSTNAME.text “removed”)

Reload form

### Add student

BTNAdd\_Click(): procedure

If (TXTForename not null and TXTPassword not null and TXTSurname not null)

Then

Connection = connection to Database

CommandText = insert in student table

CommandText = insert in Scores table

CommandText = insert in StudentUsers table

New SqlConnection

Try

Execute sqlInsert

Catch

Show exeption

Show new username

Hash (string Text): function

Hashstring = null

Bytes bytes = Text.split

HashText = new HashingAlgorithm

HashedValue = HashText.Hash(bytes)

Foreach (value x in HashedValues)

Hashstring += X

Return HashString

CreateUsernamePassword: function (string FirstName, string SurName, List<string> CurrentllyUsed)

String username = null

Int temp = 1

Bool existing = true

While (existing)

Existing = false

Username = Firstname[0] + Surname + temp

For (int I = 0, I < currentlyused, i++)

If (Username = CurrentlyUsed[i])

Then

Temp + 1

Username = Firstname[0] + Surname + temp

Existing = true

End

End

Return Username

FindColumnValues: function (string column)

Command = “select all from StudentUsers table”

List<string> List

DataAdapter TempTableAdapter(command, connection)

Datatable Table

TempTableAdapter.fill(Table)

List.Add(Table[column].allrows)

Return List

### Add Teacher

**Algorithms here same as for Add students**

### Student

BTN\_Login\_click: Procedure()

Bool success = true

If (TXTPass not null and TXTUsername not null)

Then

New Data (TXTUsername, TXTPass)

If(error)

Show “Incorrect Login”

Success = false

else

FilterTbl(Student.ReturnScoreTable, Students.ReturnScoreAdapter, “Select \* from scores”)

Write text file(file path)

Writeline(Student.firstname)

Writeline(student.Surname)

For(i=0, i< 8, i++)

Write(Score[i])

Start Game

Else

Show “fill all fields”

### Game1

#### Validation of Logs

Sub procedure ValidateLogs List<Logs> ListOfLogs, int ValidateTo

For I = 0, I <5, I ++

If Row\_Valid(RowFind(ListOfLogs, I), ValidateTo)

Then

FindRow(ListOfLogs, i)[1] = ValidateTo\*Random(1,10)

End

Sub Function RowValid List<Log> Row, int inValid

Bool valid

For I = 0; i < 4; i++

If Row[i] MOD invalid = 0 then

Valid = true

End

Else

Return valid

Sub function RowFind List<Log> Inital\_Log, int Row

List<Log> RowTofind

For I = 0, I < 20, i++

If I mod 5 = Row

RowToFind add Initial\_Log[I]

End

Return RowToFind

OnExiting: procedure ()

String SQLCommad = “Select \* from Students Inner join Scores on Students.StudentId = Scores.ScoreID where FirstName = '" + FirstName + "' and Surname = '" + Surname + "'"

String SQLCommand = “update Scores set Score2 = @Score2, Score3 = @Score3, Score4 = @Score4, Score5 = @Score5, Score6 = @Score6, Score7 = @Score7, Score8 = @Score8, Score9 = @Score9 where ScoreID = @ID”

String connection

Using (Connection)

For (int I = 0; I < 8; i++)

String Column = “Score” + i

SQLCommand.Command(column) = Get Score(i)

End for

SQLCommand.Command(“ID”) = Get Id (CurrentStudentLoggedIn)

Try

openConnection

catch

Display (“Unable to save”)

#### Frog – Collision

Function bool Collision List<Log> Log\_List1, int Screen\_width, Student student,int muiltiple

Bool valid

Foreach Log in LogList

If frog Intersect = Log intersect and Log.number Mod CollisionNumber = 0 and between 180 high and 450px

then

Score += 5

Log = JumpedOn

Frog.velocity = log.logspeed

Return true

Else

Return false

End

#### Number Class – set muiltiple

SetMuilitiple: Proceedure(int Testnumber)

Number = Random number (1 - 10)

If (random number (1 - 3) == 1)

Number = number \* testnumber

else

number = (number \* Testnumber) + 1