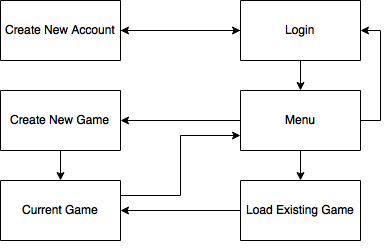
**Design**

**System Structure Diagram**

****

The first form will be the “Login” form in order for the player to be able to save and monitor their progress they must login to an existing account. If they are a new player and currently do not have an account there is a button to link them to the “Create New Account” form where they are able to enter in a new details such as a username and password so that they can logon in the future.

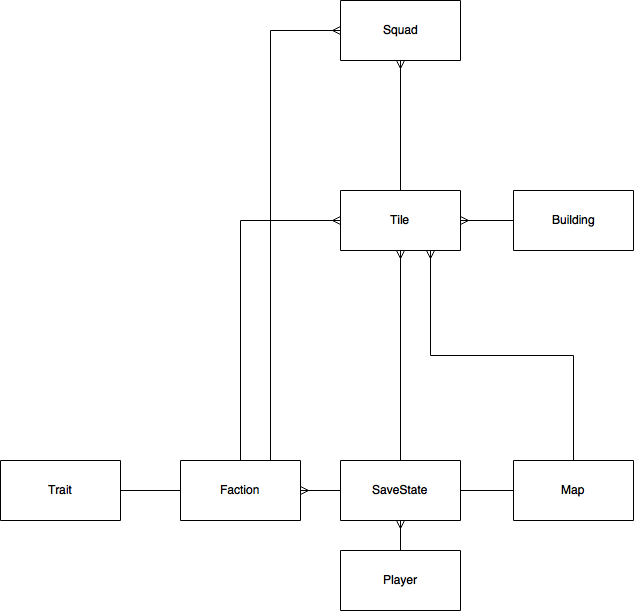
After logging on, the player is then redirected to the “Menu” form where there will be 2 buttons that the user can click, the first takes you to the “Create New Game” form and the second takes you to the “Load Existing Game” form.

The “Create New Game” form gives the player the ability to customize what they would like to play in the new game such as what faction that they would like to play as to what map they would like to play on. Additionally, they are able to choose what factions that they will be competing against and some information about their competitors. When they have finished choosing what they would like they can press the “Create Game” button which redirects them to the “Current Game” form.

Alternatively, the player can load an existing game within the “Load Existing Game” form. This form contains 3 buttons which load the game under different circumstances based on the save state that they contain. The form also includes some information about each save state in order to inform the player that they are selecting the right save state. After choosing which save state that they would like to load, they are redirected to the “Current Game” form.

Finally, the “Current Game” form is where most of the gameplay happens, the player is able to do a variety of things within this form such as choose what they would like to produce from a tile by clicking on the tile that it is located on as well as the ability to see information like the amount of food, gold produced per turn as well as the total happiness of the tile. When the player is happy with the choices that they have made they can then click next turn to see the outcome of their decisions, this process repeats until one of the three win conditions are met by any faction.

**Entity Relationship Diagram Update**

****

**Data Dictionary**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Table Name: Player | | | Primary Key: AccountID | | |
| Field Name | Data Type | Length | Validation | Example Data | Comment |
| AccountID | INTEGER | 2 |  | 1 | Unique Identifier. |
| Username | VARCHAR | 40 | Presence Check | BobSmith1348 | Allows the player to choose their own login name so it is easier to remember. |
| Password | VARCHAR | 25 | Presence Check | SecurePassword | Allows the player to choose their own password so it is easier to remember. |
| LastLogin | DATE | 10 |  | 22/07/2016 | Used when maximum numbers of accounts are in the database to delete accounts that have been inactive for more than 6 months. |
| Expert | BIT | 1 |  | 1 | Used to disable help messages if the player no longer requires them. |
| Wins | INTEGER | 2 |  | 5 | Tracks the win-loss rate of the player. |
| Losses | INTEGER | 2 |  | 0 | Tracks the win-loss rate of the player. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Table Name: Faction | | | Primary Key: FactionID | | |
| Field Name | Data Type | Length | Validation | Example Data | Comment |
| FactionID | INTEGER | 2 |  | 10 | Unique Identifier. |
| Name | VARCHAR | 40 |  | The Steampunkers |  |
| Type | VARCHAR | 13 |  | Technological | Identifies what their most likely win condition will be as they will be more proficient in this area. |
| Capital Name | VARCHAR | 20 |  | Fallen London | The default name for the faction’s capital which the player can change. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Table Name: SaveState | | | Composite Primary Key: SaveID, AccountID | | |
| Field Name | Data Type | Length | Validation | Example Data | Comment |
| SaveID | INTEGER | 1 |  | 2 | Limits the amount of saves that a player can have (3). |
| FileName | VARCHAR | 25 | Presence Check, Length Check | First Game | Allows player to choose a meaningful identifier for their saves instead of a number. |
| AccountID | INTEGER | 2 |  | 19 | To link up save states to an account but also restrict those who do not have the correct AccountID from accessing other’s saves. |
| MapID | INTEGER | 1 |  | 3 | To link up a map to the save state’s information. |
| FactionID | INTEGER | 2 |  | 4 | Identifies what faction the player is in control of. |
| NumberOfTurns | INTEGER | 2 |  | 7 | Amount of turns since game has started. |
| CurrentNumberOfFactions | INTEGER | 1 |  | 4 | Remaining factions as some may have been defeated. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Table Name: Map | | | Primary Key: MapID | | |
| Field Name | Data Type | Length | Validation | Example Data | Comment |
| MapID | INTEGER | 2 |  | 5 | Unique Identifier. |
| Name | VARCHAR | 9 |  | Continent | Description of what the map is to use in a combo box for the player. |
| XCoordinate | INTEGER | 2 |  | 20 | Sets up the map grid. |
| YCoordinate | INTEGER | 2 |  | 14 | Sets up the map grid. |
| NumberOfFactions | INTEGER | 1 |  | 4 | States starting number of factions. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Table Name: Tile | | | Primary Key: TileID | | |
| Field Name | Data Type | Length | Validation | Example Data | Comment |
| TileID | CHAR | 5 |  | (1,2) | Unique Identifier. |
| SaveID | INTEGER | 1 |  | 3 | Used to actively track updates to each tile. |
| AccountID | INTEGER | 2 |  | 11 | Used to actively track updates to each tile. (Composite Primary Key requires both fields.) |
| Name | VARCHAR | 40 |  | Fallen London |  |
| TotalFood | FLOAT | 4 |  | 56.1 | Amount of food in surplus for citizens to eat. |
| FoodPerTurn | FLOAT | 4 |  | -1.4 | Can be negative. |
| TotalGold | FLOAT | 6 |  | 1000.6 | Amount of gold the tile’s government has to spend on what the faction chooses. |
| GoldPerTurn | FLOAT | 5 |  | 20.2 | Can be negative. |
| TotalHappiness | INTEGER | 3 |  | 95 | How happy the people are under current leadership. |
| HappinessPerTurn | INTEGER | 2 |  | -1 | Can be negative. |
| FactionID | INTEGER | 2 |  | 7 | To identify current ownership of a tile. |
| MapID | INTEGER | 2 |  | 2 | Identifies what map the tile is located. |
| SettlementID | INTEGER | 1 |  | 0 | Identifies what level settlement is on tile. |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Table Name: Settlement | | | Primary Key: SettlementID | | |
| Field Name | Data Type | Length | Validation | Example Data | Comment |
| SettlementID | INTEGER | 2 |  | 1 | Unique Identifier. |
| Name | VARCHAR | 25 |  | Hamlet |  |
| Paradigm | VARCHAR | 13 |  | Advanced |  |
| GoldToConstruct | INTEGER | 3 |  | 14 | Stores the amount of gold for the settlement to be constructed. |
| FoodPerTurn | INTEGER | 2 |  | 5 | Additional Food Income. |
| GoldPerTurn | INTEGER | 2 |  | 5 | Additional Gold Income. |

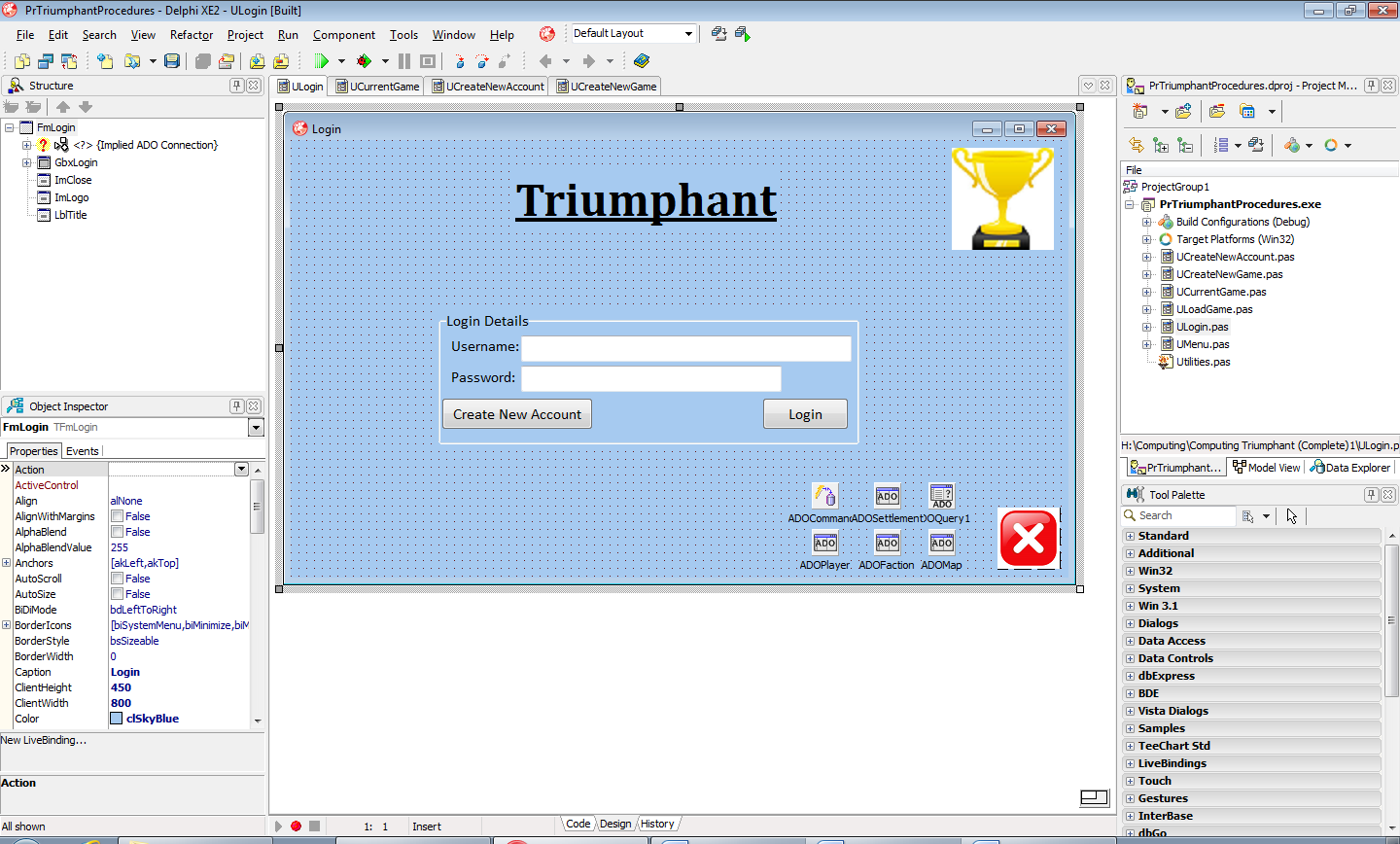
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Table Name: Squad | | | Primary Key: SquadID | | |
| Field Name | Data Type | Length | Validation | Example Data | Comment |
| SquadID | INTEGER | 2 |  | 5 | Unique Identifier. |
| Active | BIT | 1 |  | 1 | Value to determine whether they are at base (0) or active on the battlefield (1). |
| Objective | VARCHAR | 10 |  | Ambush | Command word for their objective for the player to choose what they want the squad to do (with a combo box). |
| TurnsToComplete | INTEGER | 2 |  | 7 | Storage for how many turns left an action is left to complete. |
| XCoordinate | INTEGER | 2 |  | 5 | Stores the squad’s co-ordinates to actively track them on the map. |
| YCoordinate | INTEGER | 2 |  | 6 | Stores the squad’s co-ordinates to actively track them on the map. |
| TileID | INTEGER | 2 |  | 2 | Identifies which tile the squad comes from. |
| FactionID | INTEGER | 2 |  | 4 | States what faction they belong to so that they can inherit a trait. |
| SaveID | INTEGER | 1 |  | 3 |  |
| AccountID | INTEGER | 2 |  | 11 |  |

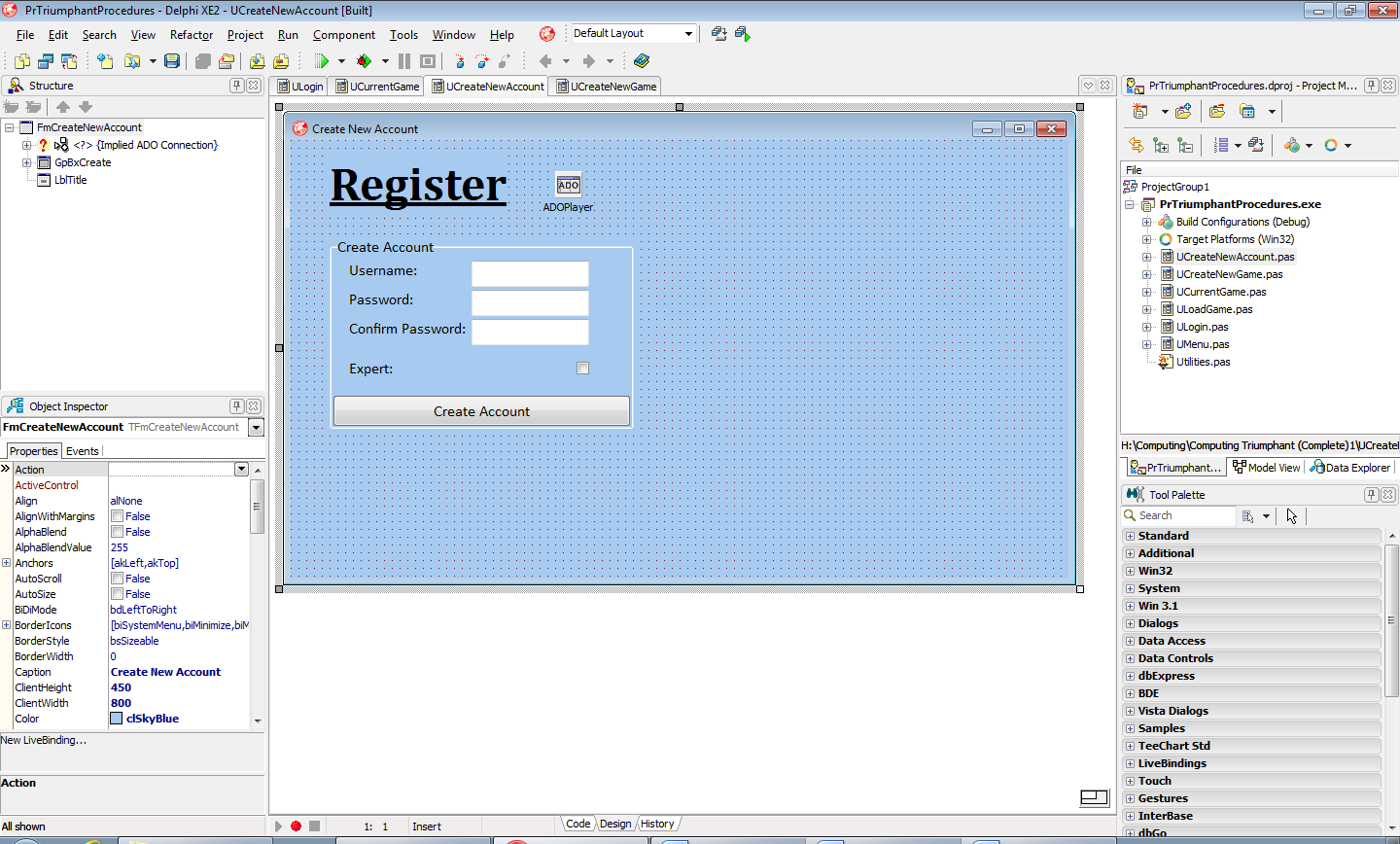
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Table Name: Trait | | | Primary Key: TraitID | | |
| Field Name | Data Type | Length | Validation | Example Data | Comment |
| TraitID | INTEGER | 2 |  | 4 | Unique Identifier. |
| Name | VARCHAR | 25 |  | Strength In Numbers |  |
| Description | VARCHAR | 40 |  | Increases damage by 50% for each nearby squad. | Describes how the trait impacts gameplay to the player. |
| FactionID | INTEGER | 1 |  | 1 | Links up a faction with a particular trait. |

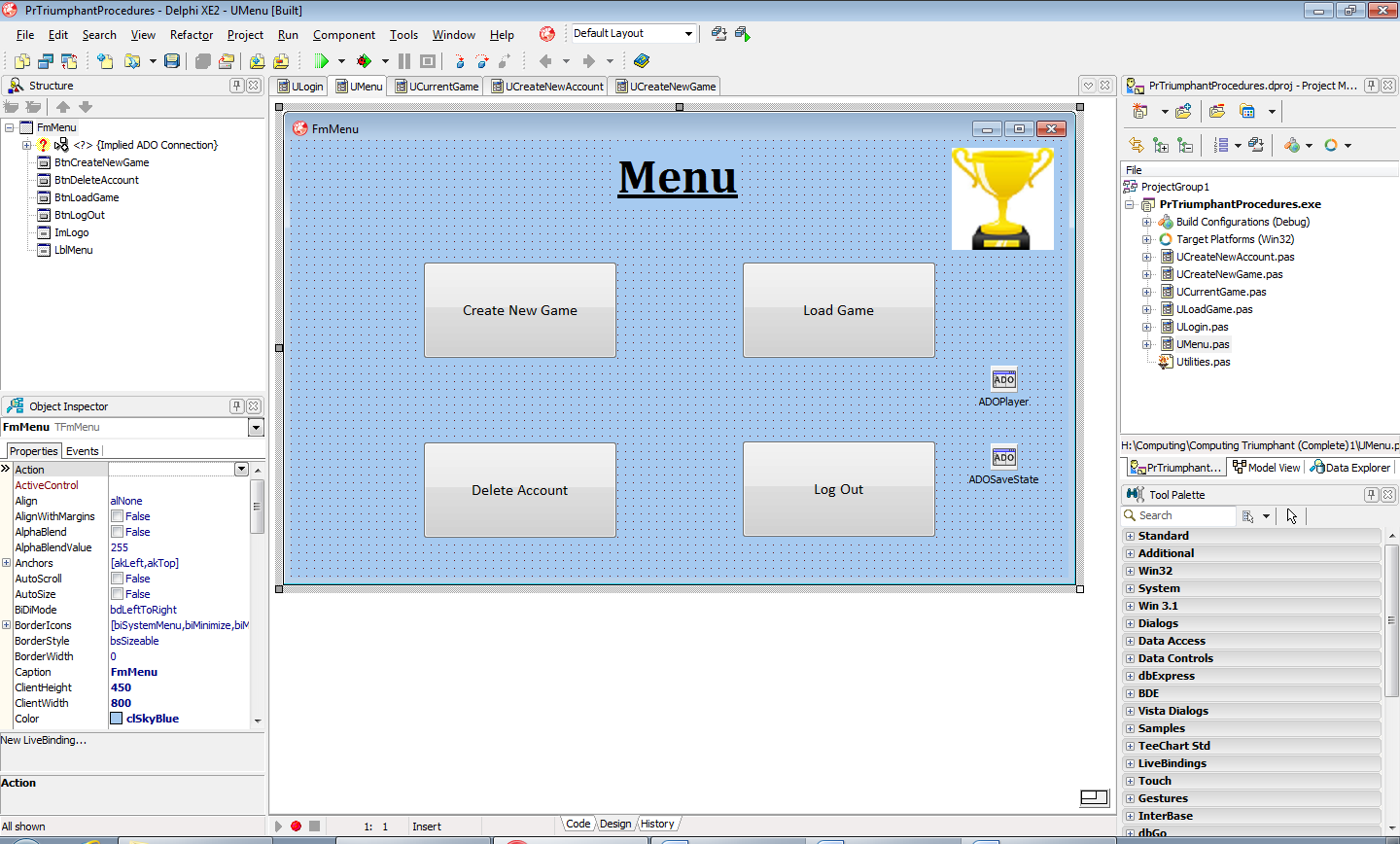
**Description of Algorithms**

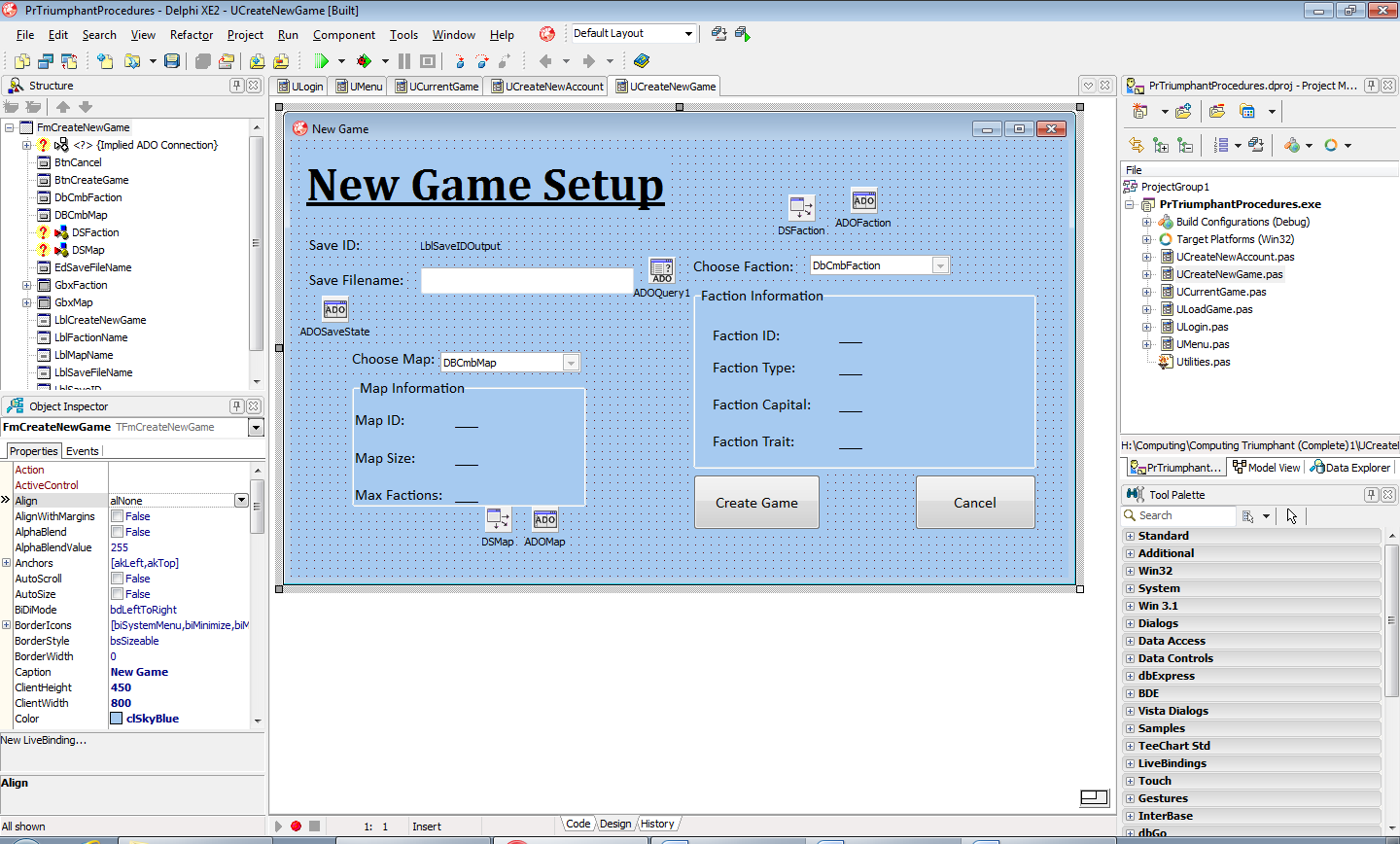
|  |
| --- |
| DDL: |
| Const  ConnStr='Provider=Microsoft.ACE.OLEDB.12.0; Data Source=Triumphant.accdb; Persist Security Info=False;';  procedure TFmLogin.FormCreate(Sender: TObject);  var cat:OLEVariant;  I, MaxFactions, MaxMaps, MaxSettlements: Integer;  begin  SetUpFactions(Faction);  SetUpMaps(Map);  SetUpTiles(TileSet);  SetUpSettlements(Settlement);  MaxFactions:=5;  MaxMaps:=5;  MaxSettlements:=4;  cat:=CreateOleObject('ADOX.Catalog');  if not FileExists('Triumphant.accdb') then  begin  cat.create('Provider=Microsoft.ACE.OLEDB.12.0; Data Source=Triumphant.accdb;');  ADOCommand1.ConnectionString:=ConnStr;  //Player Table DDL  ADOCommand1.CommandText:='CREATE TABLE Player(AccountID INTEGER,' +  'Username VARCHAR(40),Passcode VARCHAR(25),LastLogin DATE,' +  'Expert BIT,Wins INTEGER,Losses INTEGER,' +  'PRIMARY KEY(AccountID))';  ADOCommand1.Execute;  //Faction Table DDL  ADOCommand1.CommandText:='CREATE TABLE Faction(FactionID INTEGER,' +  'Name VARCHAR(40),Type VARCHAR(13),CapitalName VARCHAR(20),' +  'PRIMARY KEY(FactionID))';  ADOCommand1.Execute;  //Trait Table DDL  ADOCommand1.CommandText:='CREATE TABLE Trait(TraitID INTEGER,' +  'Name VARCHAR(25),Description VARCHAR(40),' +  'FactionID INTEGER,' +  'FOREIGN KEY(FactionID) REFERENCES Faction(FactionID),' +  'PRIMARY KEY(TraitID))';  ADOCommand1.Execute;  //Map Table DDL  ADOCommand1.CommandText:='CREATE TABLE Map(MapID INTEGER,' +  'Name VARCHAR(9),XCoordinate INTEGER,YCoordinate INTEGER,' +  'NumberOfFactions INTEGER,' +  'PRIMARY KEY(MapID))';  ADOCommand1.Execute;  //SaveState Table DDL  ADOCommand1.CommandText:='CREATE TABLE SaveState(SaveID INTEGER,' +  'FileName VARCHAR(25),' +  'AccountID INTEGER,MapID INTEGER,' +  'FactionID INTEGER,' +  'NumberOfTurns INTEGER,CurrentNumberOfFactions INTEGER,' +  'FOREIGN KEY(AccountID) REFERENCES Player(AccountID),' +  'FOREIGN KEY(MapID) REFERENCES Map(MapID),' +  'FOREIGN KEY(FactionID) REFERENCES Faction(FactionID),' +  'PRIMARY KEY(SaveID,AccountID))';  ADOCommand1.Execute;  //Settlement Table DDL  ADOCommand1.CommandText:='CREATE TABLE Settlement(SettlementID INTEGER,' +  'Name VARCHAR(25),Paradigm VARCHAR(13),' +  'GoldToConstruct INTEGER,FoodPerTurn INTEGER,GoldPerTurn INTEGER,' +  'PRIMARY KEY(SettlementID))';  ADOCommand1.Execute;  //Tile Table DDL  ADOCommand1.CommandText:='CREATE TABLE Tile(TileID CHAR(5),SaveID INTEGER,AccountID INTEGER,' +  'Name VARCHAR(40),' +  'TotalFood FLOAT,FoodPerTurn FLOAT,' +  'TotalGold FLOAT,GoldPerTurn FLOAT,' +  'TotalHappiness INTEGER,HappinessPerTurn INTEGER,' +  'FactionID INTEGER,MapID INTEGER,' +  'SettlementID INTEGER,' +  'FOREIGN KEY(FactionID) REFERENCES Faction(FactionID),' +  'FOREIGN KEY(MapID) REFERENCES Map(MapID),' +  'FOREIGN KEY(SettlementID) REFERENCES Settlement(SettlementID),' +  'FOREIGN KEY(SaveID,AccountID) REFERENCES SaveState(SaveID,AccountID),' +  'PRIMARY KEY(TileID,SaveID,AccountID))';  ADOCommand1.Execute;  //Squad Table DDL  ADOCommand1.CommandText:='CREATE TABLE Squad(SquadID INTEGER,' +  'Active BIT,Objective VARCHAR(10),TurnsToComplete INTEGER,' +  'XCoordinate INTEGER,YCoordinate INTEGER,FactionID INTEGER,' +  'TileID CHAR(5),SaveID INTEGER, AccountID INTEGER,' +  'FOREIGN KEY(TileID,SaveID,AccountID) REFERENCES Tile(TileID,SaveID,AccountID),' +  'FOREIGN KEY(FactionID) REFERENCES Faction(FactionID),' +  'PRIMARY KEY(SquadID))';  ADOCommand1.Execute;  //Adds array of TFaction into factions table in database  //maybe use array of TFaction and a for loop  FmLogin.Activate;  for I := 0 to MaxFactions do  begin  AdoFaction.Append;  AdoFaction['FactionID']:=I;  AdoFaction['Name']:=Faction[I].Name;  AdoFaction['Type']:=Faction[I].TypeOfFaction;  AdoFaction['CapitalName']:=Faction[I].CapitalName;  AdoFaction.Post;  end;  //Adds array of TMap into maps table in database  for I := 1 to MaxMaps do  begin  AdoMap.Append;  AdoMap['MapID']:=I;  AdoMap['Name']:=Map[I].Name;  AdoMap['XCoordinate']:=Map[I].XCoordinate;  AdoMap['YCoordinate']:=Map[I].YCoordinate;  AdoMap['NumberOfFactions']:=Map[I].NumberOfFactions;  AdoMap.Post;  end;  for I := 1 to MaxSettlements do  begin  AdoSettlement.Append;  AdoSettlement['SettlementID']:=I;  AdoSettlement['Name']:=Settlement[I].Name;  AdoSettlement['Paradigm']:=Settlement[I].Paradigm;  AdoSettlement['GoldToConstruct']:=Settlement[I].GoldToConstruct;  AdoSettlement['FoodPerTurn']:=Settlement[I].FoodPerTurn;  AdoSettlement['GoldPerTurn']:=Settlement[I].GoldPerTurn;  AdoSettlement.Post;  end;  end;  end; |

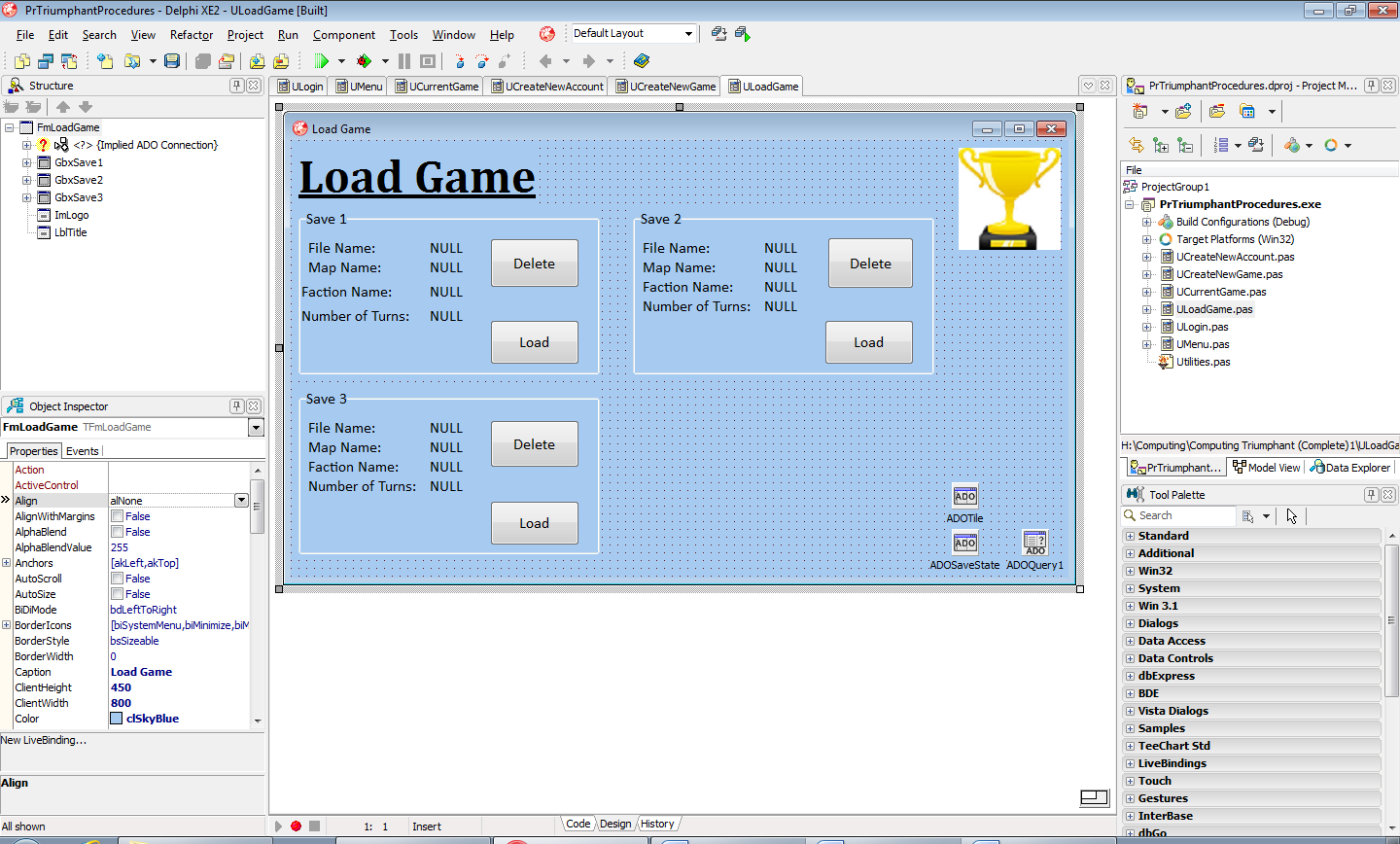
**User Interface Design**

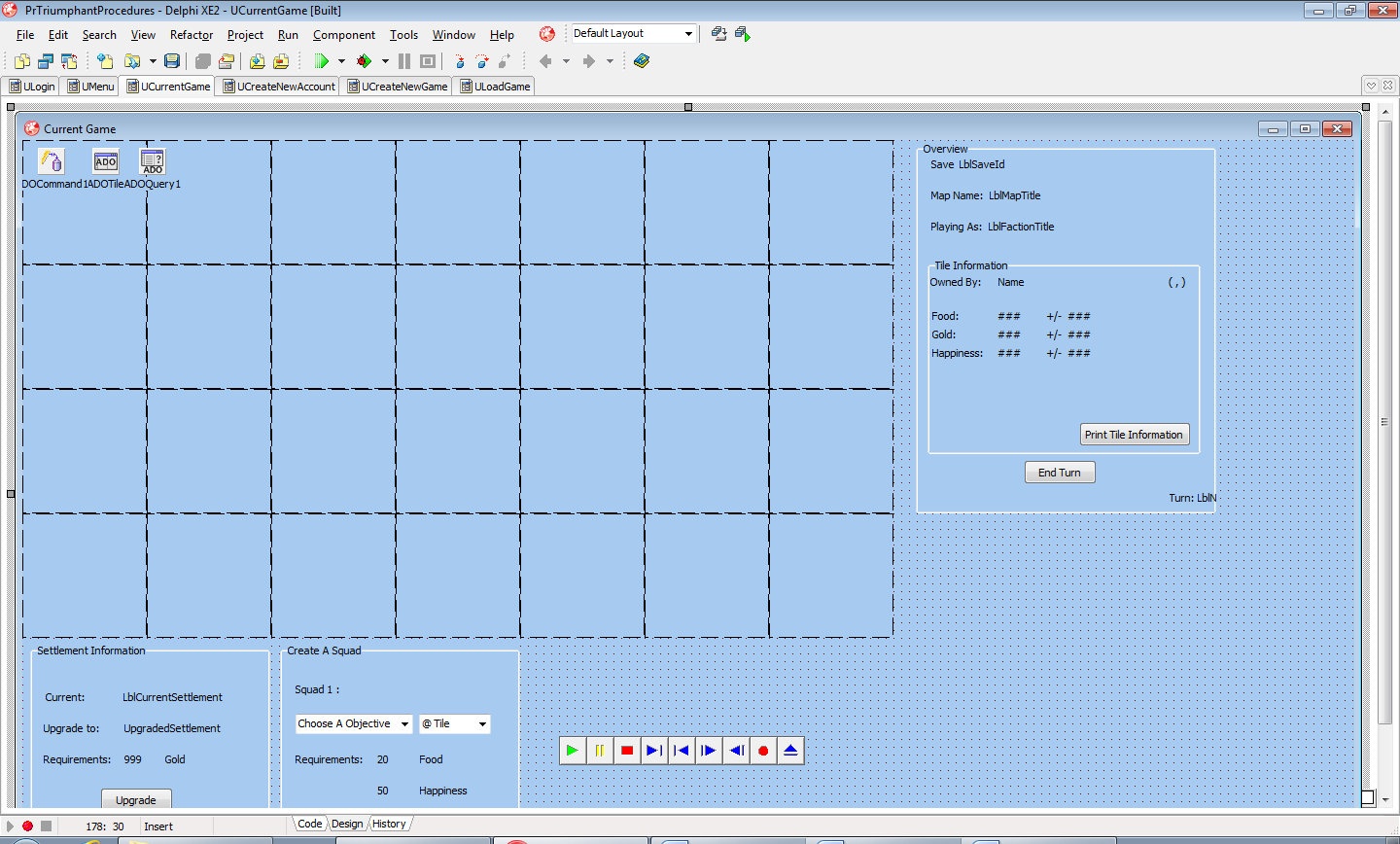












**System Security and the Integrity of Data**

Within the project, certain precautions have been taken to ensure that player usernames and passwords are kept secure and accurate. The project includes a doubly entry check of the user’s initial account creation to make sure their password is correct. Additionally, visual help will be included to give helpful messages to the user so that they don’t understand why they cannot yet enter the project.

To ensure referential integrity, if the player would like to delete their account they must first delete all their save states. If they attempt to delete their account with any save states still present they will be notified of why they cannot proceed.

As for data security, a login will be needed for the player to have access to the system, making sure that their save states remain intact.