PAT 2022

Information Technology

ADAM FOFLONKER 11C

2022

**Task 1 – Definition and User Story**

The principal of Big Beaver High-School along with the soccer team coaches have asked me to develop an application for them to use in which they can easily create, interact with and view tournament fixtures/ladders for their upcoming inter-class soccer tournaments this season across each grade. They would like for users of their application to be able to view the current tournament fixtures in real-time while disallowing them to modify these fixtures hence, they would also require a way of only allowing tournament administrators to modify and update these fixtures and also only allowing them to submit new teams onto the roster. They require a sustainable application to be developed which they would like to eventually use for all of their other tournaments in future seasons. Their typical tournament type is a Single-Elimination tournament ladder with a total of 20 teams being able to enter.

In a meeting with the principal of the school, she stated “I would like to have an application wherein I can view the fixtures and results of the teams currently playing, and it should be capable of being updated in real-time by those in-charge of the tournaments. I think that would be something really great and effective for us.” This statement suggests that the two primary types of users of the application should be able to:

|  |  |  |
| --- | --- | --- |
| **Accessible Functionality** | **Normal Clients** | **Tournament Admins** |
| Viewing tournament fixtures and match results | *Yes* | *Yes* |
| Updating and modifying fixtures and match results (modifying database) | *No* | *Yes* |
| Creating tournaments and submitting teams | *No* | *Yes* |

To satisfy the school’s requirements of the application, it will feature a “main activity” form in which normal clients can view the tournament fixtures and match results. It will also feature an “admin login” panel that tournament administrators can use to login to the application with via username and password stored in the backend database. This will then lead to an admin form in which tournament administrators can update/modify the tournament fixtures and other related data stored in a local text file (JSON) including adding new players, teams and other administrators to the backend database.

**Task 2 – Database Design**

* The database will not store the current state of the tournament’s fixtures.

tblPlayers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Data Type** | **Field Size** | **Constraints** | **Example** |
| PlayerID | Number | 3 Digits | PRIMARY KEY | 1 |
| ForeNames | Short Text | 255 Characters | NOT NULL | Adam |
| Surname | Short Text | 255 Characters | NOT NULL | Foflonker |
| Age | Number | 2 Digits | NOT NULL | 17 |
| TeamID | Number | 2 Digits | FOREIGN KEY, NOT NULL | 3 |
| Position | Short Text | 2 Characters | NOT NULL | GK |

tblTeams

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Data Type** | **Field Size** | **Constraints** | **Example** |
| TeamID | Number | 2 Digits | PRIMARY KEY | 3 |
| TeamClassName | Short Text | 2 Characters | NOT NULL | 11C |
| Grade | Number | 2 Digits | NOT NULL | 11 |
| TeamLayout | Short Text | 5 Characters | NOT NULL | 3-4-3 |
| TeamSubmitterAdminID | Number | 2 digits | FOREIGN KEY, NOT NULL | 1 |

tblAdmins

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Field** | **Data Type** | **Field Size** | **Constraints** | **Example** |
| AdminID | Number | 2 digits | PRIMARY KEY | 1 |
| ForeNames | Short Text | 255 Characters | NOT NULL | Chad |
| Surname | Short Text | 255 Characters | NOT NULL | Emmerson |
| PasswordHash | Short Text | 255 Characters | NOT NULL | a2240212f6e8d7db337b46aab21433ad== |

**Task 3 – Navigation/Description**

**Task 4 – Data Dictionary**

The application will make use of a JSON file “Fixtures.json” to store the current state of the tournament’s fixtures. Each entry in the file will hold a TeamID from tblTeams in the database and associated with it it’s current stage on the ladder, its position within that stage and whether it has been eliminated or not.

The current stage on the ladder or “LadderStage” is an enumerated type which can equal: Disqualified (0), Qualifier (1), QuarterFinal (2), SemiFinal (3), Final (4), Champion (5). The position within stage or “StagePosition” will be an integer type and “Eliminated” will be a Boolean type.

Sample text from Fixtures.json:

{"Fixtures":

    [

        {

            "type": "PAT2022\_u.FixtureEntry",

            "id": 1,

            "fields":

            {

                "TeamID": 3,

                "LadderStage": 4,

                "StagePosition": 2,

                "Eliminated": false

            }

        },

        {

            "type": "PAT2022\_u.FixtureEntry",

            "id": 2,

            "fields":

            {

                "TeamID": 1,

                "LadderStage": 3,

                "StagePosition": 1,

                "Eliminated": true

            }

        }

    ]

}

Upon startup, the application will load this file’s contents into an array of fixture entries where each entry in this file is an entry in the array. This array will then be used to populate the visible tournament ladder on the main form.

For security reasons, admin passwords will not be stored as plain text in the backend database. Passwords will therefore be hashed when being stored in the database (signup) and when being compared to the database (logins). For this reason, a function is needed to create hashes from a password string.

*function TForm1.GenerateHash(passwd: string) : string;*

Example: Password = 12345678, Hash = a2240212f6e8d7db337b46aab21433ad==

**Task 5 – IPO and Data Validation**

**Viewing Fixtures**

|  |  |  |
| --- | --- | --- |
| **Variable** | **Data Type** | **Description** |
| arrFixtures | Array of Fixture | Array of fixtures from “Fixtures.json” where Fixture is defined as a class. |
| tFixtures | TextFile | Logical text file used for loading data into arrFixtures. |
| sLineBuf | String | Buffer used for reading in each line from tFixtures. |
| sFixtures | String | Complete string containing all text read into sLineBuf. |

**IPO**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Source** | **Data Type** | **Format of the Input** | **GUI component** |
| Plain Text | Text file “Fixtures.json” | String | JSON | None |
| **Processing** | **Action** | | | **GUI Component** |
| Create tFixtures. Populate arrFixtures from tFixtures and use arrFixtures to display tournament ladder | Assign and reset tFixtures.  Read all text from tFixtures into sFixtures.  **If** error reading file: display error.  Unmarshal sFixtures JSON text and populate arrFixtures.  **If** error unmarshalling text: display error.  **for** fixture **in** arrFixtures **do**  Populate ladder using each fixture in arrFixtures. | | | TPageControl |
| **Output** | **Format** | | | **GUI Component** |
| Show populated tournament ladder | Tournament ladder diagram | | | TPanel |

**Admin login**

|  |  |  |
| --- | --- | --- |
| **Variable** | **Data Type** | **Description** |
| sForeNames | String | Stores forenames entered. |
| sSurname | String | Stores surname entered. |
| sPassword | String | Stores password entered. |

**IPO**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Input** | **Source** | **Data Type** | **Format of the Input** | **GUI component** |
| Plain Text | Keyboard | String | Username/Password | TEdit |
| **Processing** | **Action** | | | **GUI Component** |
| Check if forenames, surname, password combination exists in database and is valid. | WHAT:  Check if names and password combination matches an entry in the tblAdmins table in the database.  HOW:  Check if forenames and surname are valid by checking length and or if they contain any digits. **If** they are not valid: display “Invalid name”  **If** forenames and surname combination is found:  Hash sPassword, **if** hashed password matches database password hash: open Admin form  **else**  display “Incorrect password”  **else**  display “Admin doesn’t exist” | | | TButton |
| **Output** | **Format** | | | **GUI Component** |
| Open admin form | Child form of main form. | | | TForm |