



Honors College Theses

4-29-2021

A Deep Analysis and Algorithmic Approach to Solving Complex Fitness Issues in Collegiate Student Athletes

Holly N. Puckett
Georgia Southern University

Follow this and additional works at: <https://digitalcommons.georgiasouthern.edu/honors-theses>



Part of the [Computer Engineering Commons](#), [Databases and Information Systems Commons](#), [Other Computer Sciences Commons](#), [Software Engineering Commons](#), and the [Theory and Algorithms Commons](#)

Recommended Citation

Puckett, Holly N., "A Deep Analysis and Algorithmic Approach to Solving Complex Fitness Issues in Collegiate Student Athletes" (2021). *Honors College Theses*. 629.
<https://digitalcommons.georgiasouthern.edu/honors-theses/629>

This thesis (open access) is brought to you for free and open access by Georgia Southern Commons. It has been accepted for inclusion in Honors College Theses by an authorized administrator of Georgia Southern Commons. For more information, please contact digitalcommons@georgiasouthern.edu.

A Deep Analysis and Algorithmic Approach to Solving Complex Fitness Issues in Collegiate Student Athletes

An Honors Thesis submitted in partial fulfillment of the requirements for Honors in Computer Science.

By
Holly Puckett

Under the mentorship of Dr. Andrew Allen

ABSTRACT

Sports are not simply an entertainment source. For many, it creates a sense of community, support, and trust among both fans and athletes alike. In order to continue the sense of community sports provides, athletes must be properly cared for in order to perform at the highest level possible. Thus, their fitness and health must be monitored continuously. In a professional sense, one can expect individualized attention to athletes daily due to an abundance of funding and resources. However, when looking at college communities and student athletes within them, the number of athletes per athletic trainer increases due to both limited funds and resources. Athletic trainers are responsible for athlete care but can be overwhelmed with high ratios of athletes per athletic trainer. Thus, the question comes into play, how can adequate monitoring of student athletes' health and fitness levels be implemented on a consistent basis to ensure appropriate exercise regimens are being followed to allow for maximum performance? In order to help alleviate this issue, a web application was developed to ensure student athletes are getting appropriate accommodations and exercise routines needed on an individualized basis. The algorithm used assesses the activity and fitness levels of each student athlete through user input and evaluates what type of exercise regimen is needed based on various factors discussed throughout this paper. After the deployment of this application, it was found to be effective in monitoring students' health and fitness levels.

Thesis Mentor: _____

Dr. Andrew Allen

Honors Director: _____

Dr. Steven Engel

April 2021
Department of Computer Science
Honors College
Georgia Southern University

Table of Contents

Acknowledgements	2
Introduction	3
Architectural & Algorithm Implementation	5
Methodology	7
Technologies Used	10
Database Design	11
Results and Analysis	12
Reflection	13
Conclusion	14
References	15
Appendix A: Preliminary Survey Results	16
Appendix B: Post Survey Results	17
Appendix C: Partial Source Code	18
Appendix D: Screenshots of Application	40

Acknowledgements

I would like to take a moment and thank all of those who have helped me along the way. Firstly, I would like to thank Dr. Andrew Allen for mentoring me throughout this process and taking time to help me find inspiration for this project. He answered any questions I had along the way, and I could not have asked for a better faculty mentor.

Secondly, I would like to thank a personal mentor and close friend of mine, Shayne Moore. He introduced me to the coding world and inspired me to pursue a career in Computer Science. Without his help and guidance, I would not be the programmer I am today.

Additionally, I would like to take time to thank my parents, Joe and Angela Puckett, for supporting me throughout my educational career. They have always been there to comfort me or support me in any way they could, and I am extremely grateful for their love and support.

Finally, I would like to thank both Georgia Southern University and the University Honors College for allowing me to participate and explore my thesis topic. I learned an incredible amount throughout this project, and I truly could not have asked for a better experience. The past four years have been incredible, and I am so lucky to have had the college experience I did. Go Eagles!

Introduction

From the long lines of Nathan's hotdog stands at the Braves Stadium to the roaring sound of fans in a gymnasium, almost everyone has a nostalgic memory tied to sports in some way. Sports are not simply an entertainment source. For many, it creates a sense of community, support, and trust among both fans and athletes alike. In order to continue the sense of community sports provides, athletes must be properly cared for in order to perform at the highest level possible. Thus, their fitness and health must be monitored continuously. In a professional sense, one can expect individualized attention to athletes daily due to an abundance of funding and resources. However, when looking at college communities and student athletes within them, the number of athletes per athletic trainer increases due to both limited funds and resources. Athletic trainers are responsible for athlete care but can be overwhelmed with high ratios of athletes per athletic trainer. Thus, the question comes into play, how can adequate monitoring of student athletes' health and fitness levels be implemented on a consistent basis to ensure appropriate exercise regimens are being followed to allow for maximum performance? In order to help alleviate this issue, a web application was developed to ensure student athletes are getting appropriate accommodations and exercise routines needed on an individualized basis.

While there are questionnaire-style assessment applications readily available on the market (think Google Forms), none of the applications contained the functionalities sought when researching similar applications. Thus, with the help of Dr. Andrew Allen, I was inspired to create a web-based application called FitNest to minimize the inadequate

and insufficient monitoring student athletes are receiving on college campuses across the country.

The algorithm works by having users (student-athletes) answer a series of questions regarding their health and fitness levels for that given day. For example, one of the questions may be, “How much water have you had today?”. The user then selects an appropriate response based on their water intake for that given day. If the user selects one to two glasses, the back-end side of the application scores each response based on a numerical point system. For simplicity’s sake, say one to two glasses represents one point. This is then done for each answer to each question. At the end of the questionnaire, the points are added up to be sent to the athletic trainer/coach of that specific student athlete, so they know how rigorous their workout should be for that day based on how many points the student athlete has. Coaches have their own separate portal to log in, so they are able to view the scores, as well as monitor student athletes’ individualized health and fitness levels. There is also an “admin” side of this application where coaches/athletic trainers can be deleted/added as needed, questions can be added/deleted, points are able to be adjusted for each option on a given question, and scores can be migrated to another coach if one coach decides to leave.

The main objective when creating this application was to ensure that it would make it easier for coaches and athletic trainers to track student athletes’ health and fitness levels in an equitable and affordable way. Student athletes and their coaches/athletic trainers were the target demographic for this web application.

Architectural & Algorithm Implementation

A simple algorithm was written and implemented to execute the functions of this application. The algorithm used assesses the activity and fitness levels of each student athlete through user input and evaluates what type of exercise regimen is needed based on various factors discussed throughout this paper. When designing the algorithm for this application, focus was placed on devising an efficient plan for each test case and option within the web application. The algorithm works by running scripts in the model portion of the application and executing them appropriately. Figure I below illustrates the flowchart and thought process of designing and implementing this algorithmic approach.

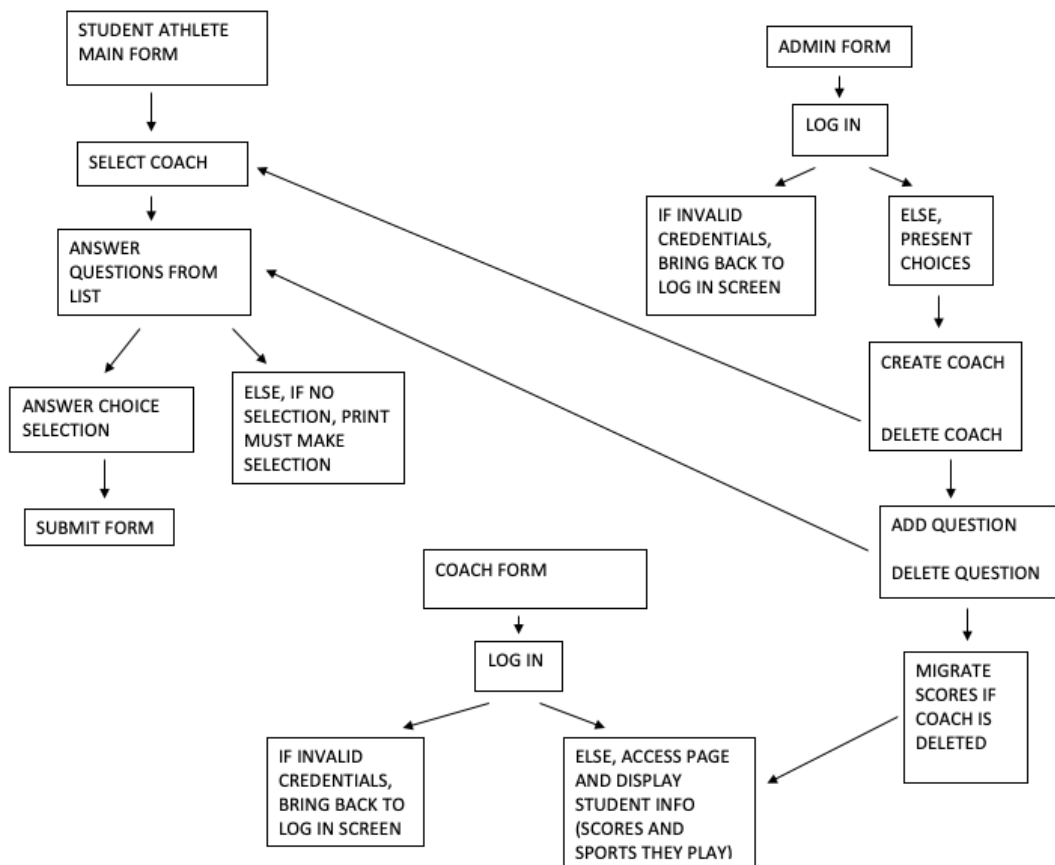


Figure I: Algorithmic Approach Flowchart

This application was developed using a Model View Controller (MVC) design pattern. An MVC design pattern consists of an application containing a data model, presentation information, and control information. With this pattern, each of these attributes are separated into different objects. Further expanding upon this pattern, the data model portion of this design pattern contains pure data. Thus, this is where the “meat” of the application resides. Every function and method exist within this portion of the application. The view portion of this application is what is displayed to the end user. It exists to present the model’s data to the actual user. However, it is important to note, it does not know the logic behind the data or what it does. It simply presents the data to the user in a user-friendly way. Finally, the controller portion of this design pattern coexists between both the view and the model. Essentially, it listens to events that are triggered by the view. Then, it will execute the appropriate function or reaction to these events. Usually, this reaction is done to call a certain method existing within the model itself. Because the view and model are interconnected, automatic reflection exists within the view portion of this design pattern.

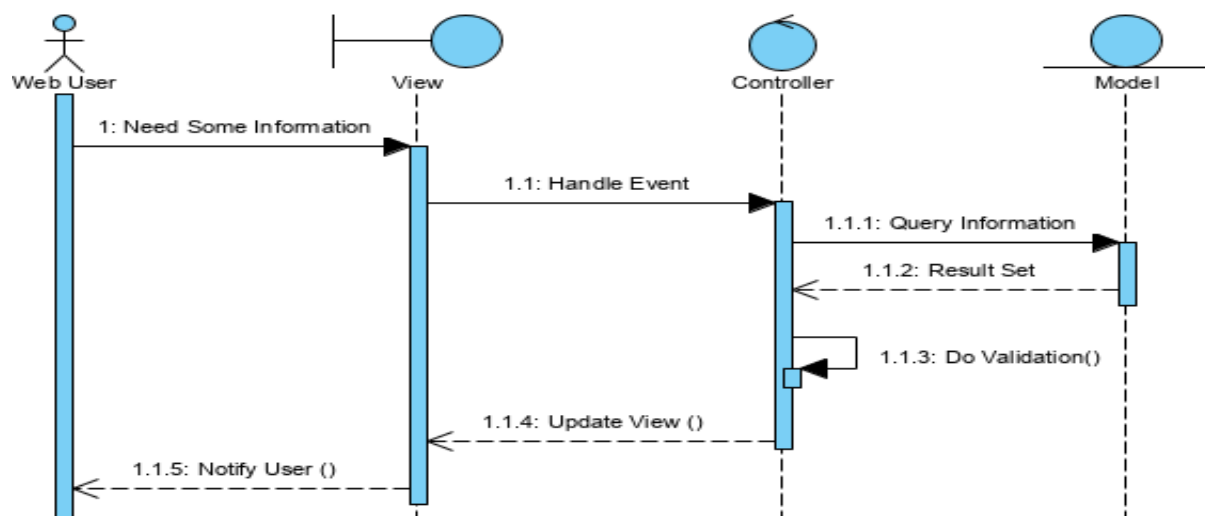


Figure II: MVC Sequence Diagram (Visual Paradigm, 2020)

Figure II above illustrates a sequence diagram showing the MVC design pattern. The sequence diagram shows the interactions between the model, view, and controller. It simply shows how the view portion is responsible for user input and output, how the controllers are able to implement the actual logic behind the transactions of the model, and the model itself containing the logic and data of the application.

Methodology

In order to ensure there was an actual interest in the application, a meeting was set up with an athletic trainer at Georgia Southern University. In addition to this, a Google Form survey was created to be distributed to student athletes and athletic trainers/coaches at Georgia Southern University. While the pool size of participants was small ($n = 12$), adequate data was collected. The survey created contained four questions where participants were asked questions regarding the interest level, potential usage of the application, and how they would recommend the application to be used and expanded upon. The graph below summarizes the results comparing student-athletes and athletic trainers' interest in the application.

Interest Levels

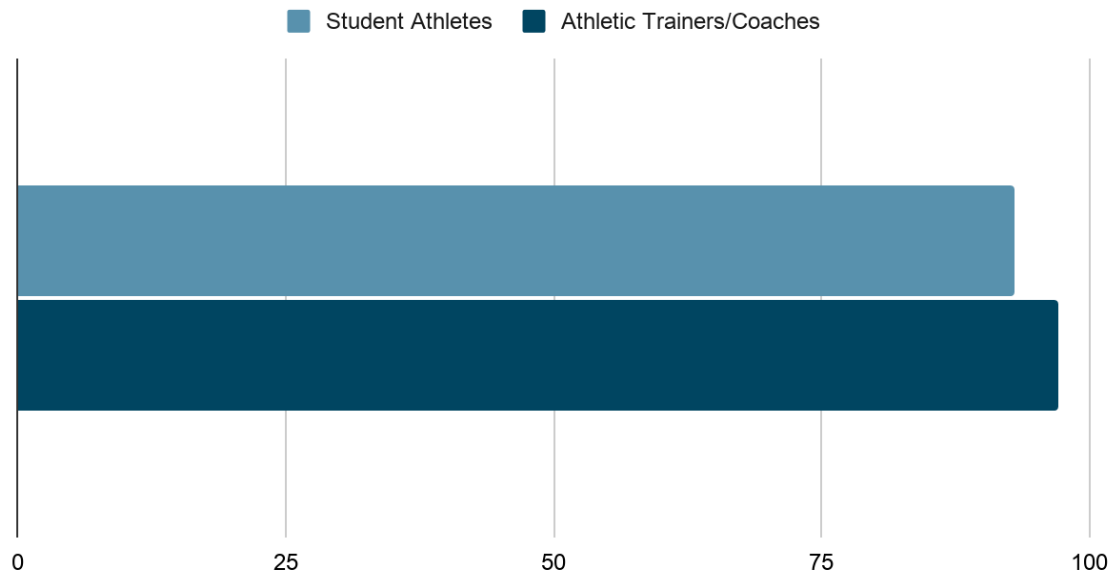


Figure III: Interest Levels Bar Graph

As seen in figure three, athletic trainers/coaches and student athletes had a strong overall interest in this application. While the interest levels are very close, athletic trainers and coaches had slightly more interest in the application. Because there was an interest from both sides in this application, the next step in the development and implementation of this application was determining which features to implement.

In order to determine which features were being sought in this application, a plan was devised based on input from athletic trainers and student athletes alike. By receiving input from both sides of the targeted users, adequate data was collected to ensure appropriate features would be implemented for all users of this application.

After determining which features to implement, the next step was to determine what algorithm and design pattern was needed to obtain the desired results. Through extensive research, a Model View Controller (MVC) pattern was chosen. The MVC design pattern was chosen because the MVC pattern is able to enable specific logical grouping of actions related to each other on a controller itself. The views portion for a specific model containing data is also able to be grouped together. This provides a high level of cohesion and usability when developing applications. In addition to this, one study found that “because of the separation of responsibilities [in the MVC pattern], further development or modifications is easier, and scalability of the product is increased” [1]. After deciding to implement this approach, the next step was to decide on a server, so there was a place to host the database.

Azure was used to host the backend of this web application. After a server was decided on, the next step consisted of the installation process of various technologies needed to create and deploy this web application.

Azure feeds off of an application called Azure Data Studio. It is simply “a cross-platform database tool for data professionals using on-premises and cloud data platforms” [2], readily available on Windows, MAC, and Linux operating systems. After this was installed, MySQL statements were needed to populate and manage the database itself. Thus, MySQL statements were written from scratch and later tested to ensure all queries ran effectively and efficiently.

When connecting the server to the database itself, it consisted of various steps. The first step was to configure the database to ensure it was populated and set up

correctly. Then, in order to surpass the firewall, a device must be connected to the database itself. This was done by defining a specific IP range that is able to access the database directly. Then, once the specific IP address needed was saved, the server's firewall was set. Because of this, the database is now able to connect to the server itself on that specific IP address. If the database needs to be accessed on a different IP address, the same steps would be performed as above.

After connecting the server to the database, Node.js is then set up in the command prompt, so that a new project can be created. Creating a directory is critical, so the Node project is able to be stored somewhere. By running a series of simple commands in the command line, (i.e. "npm init -y"), a package.json file now exists within the created directory. Thus, the SQL Server in Azure is now connected with Node.js. The next step involved utilizing the MVC design pattern to layout the foundational components and architectural design of this web application to begin the coding process. Appendix C contains the partial source code for this application, while Appendix D contains screenshots of the fully-functionally web application, FitNest.

Technologies Used

Through extensive research, Azure was decided on to host the backend of this web application. Azure was chosen over AWS (Amazon Web Services) as the hosting platform for this application because of its high availability and ease of use.

The frontend of this application consists of a web page consisting of a user interface built using Embedded JavaScript (EJS). EJS was chosen because it serves as a

simple template engine that allows for user-generated HTML combined with JavaScript. Simply put, it helps embed JavaScript to HTML pages themselves. By implementing and using EJS in web applications, it has been found to “reduce not only errors within software development but increase productivity within the workplace” [3]. Thus, EJS was proven to be a solid and reliable choice within the development phase of this process.

In order for the server and web application to communicate with each other, HTTP Request/Response was utilized between the client and server. The client must first establish a TCP connection (or another appropriate connection if the transport layer itself is not TCP). Then, the client is able to send a request and wait for an answer. Finally, the server is able to process the request, send back its answer, and eventually display the appropriate data and status code, if applicable.

Database Design

Azure’s SQL database was implemented and used within the web application. It consists of custom defined tables handwritten and later tested. These tables consist of Administrators, Coaches, HealthScores, QuestionOptions, and Questions.

```
CreateAdministratorsTable.sql x
1 CREATE TABLE Administrators (
2   ID INT NOT NULL IDENTITY(1,1),
3   Username VARCHAR(255) UNIQUE NOT NULL,
4   Password VARCHAR(255) NOT NULL,
5   CreatedDate DATETIME2(3) NOT NULL DEFAULT (GETDATE()),
6   PRIMARY KEY (ID)
7 )
```

```
CreateQuestionOptions.sql x
1 CREATE TABLE QuestionOptions(
2   ID INT NOT NULL IDENTITY(1,1),
3   QuestionID INT NOT NULL,
4   DisplayText VARCHAR(255) NOT NULL,
5   Score INT NOT NULL,
6   PRIMARY KEY (ID),
7   FOREIGN KEY (QuestionID) REFERENCES Questions(ID)
8 )
```

```
CreateHealthScoresTable.sql x
1 CREATE TABLE HealthScores(
2   ID INT NOT NULL IDENTITY(1,1),
3   CoachID INT NOT NULL,
4   FirstName VARCHAR(255) NOT NULL,
5   LastName VARCHAR(255) NOT NULL,
6   Score INT NOT NULL,
7   DateOfScore DATETIME NOT NULL DEFAULT (GETDATE()),
8   PRIMARY KEY (ID),
9   FOREIGN KEY (CoachID) REFERENCES Coaches(ID)
10 )
```

```
CreateCoachesTable.sql x
1 CREATE TABLE Coaches (
2   ID INT NOT NULL IDENTITY(1,1),
3   Username VARCHAR(255) UNIQUE NOT NULL,
4   Password VARCHAR(255) NOT NULL,
5   LastName VARCHAR(255),
6   FirstName VARCHAR(255),
7   Sport VARCHAR(255),
8   CreatedDate DATETIME2(3) NOT NULL DEFAULT (GETDATE()),
9   PRIMARY KEY (ID)
10 )
```

```
CreateQuestions.sql x
1 CREATE TABLE Questions (
2   ID INT NOT NULL IDENTITY(1,1),
3   Label VARCHAR(255) NOT NULL,
4   PRIMARY KEY (ID)
5 )
```

Figure IV: Database Tables

Figure IV above shows the tables used throughout the database.

Results and Analysis

The algorithm implemented was evaluated and tested by having various users fill out a post-survey following the deployment of this web application. The post-survey consisted of one question evaluating the effectiveness of this web application and its impact it had on both the student athletes and athletic trainers/coaches themselves. As shown in the pie chart below, a post-survey found FitNest to be highly effective at monitoring students' health and fitness levels at the intramural level. Thus, it can be concluded at the collegiate level, results are expected to be similar.

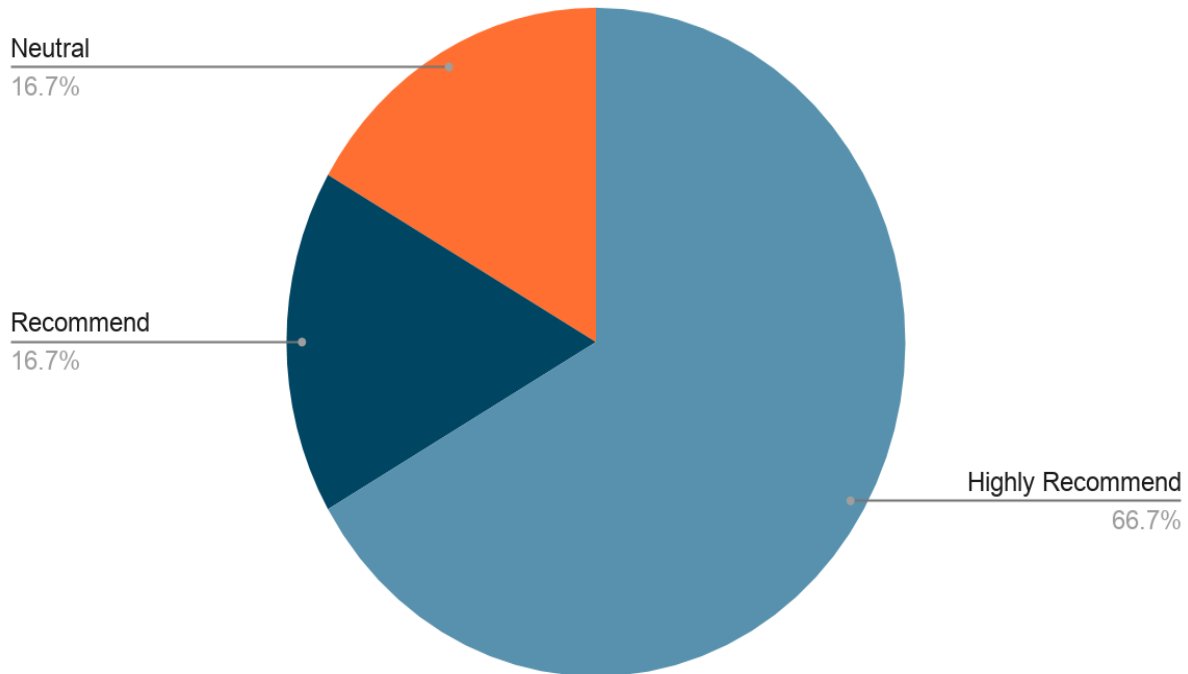


Figure V: Post-Survey Results

Reflection

Throughout this project, I have learned an incredible amount, and I could not have asked for a better experience. By completing this thesis, I was able to become exposed to various technologies, learn how they work, and even implement them in my web application. Without this opportunity, I would not have been exposed to these various technologies nor understand how they work.

By being able to participate and complete my thesis, I was able to see how a web application is designed and deployed from start to finish. I was able to learn hands-on how the entire process of building a web application from scratch exists. Because I was continuously learning about these various technologies, I was able to apply what I had learned here into both the classroom and job interviews.

Words cannot describe how invaluable this experience truly was, and I am very grateful for being able to have this opportunity. I will remember this thesis as a big milestone in my undergraduate career, and it has left a lasting impression on me. I look forward to sharing my work with others for years to come.

Conclusion

This paper describes and evaluates a web-based application built for student athletes and their athletic trainers/coaches to effectively monitor student athletes' health and fitness levels. The algorithm implemented assesses the activity and fitness levels of each student athlete through user input and evaluates what type of exercise routine is needed based on various factors about the student athletes themselves. Following the deployment of this application, it was found to be effective in monitoring students' health and fitness levels.

References

- [1] Adam Altar and Dragos-Paul Pop, “Designing an MVC Model for Rapid Web Application Development,” *Procedia Engineering*, vol 69, no. 1, p 1172-1179, March 2014
- [2] Rizik Al-Sayyed, et al, “An Investigation of Microsoft Azure and Amazon Web Services from Users’ Perspectives,” *International Journal of Emerging Technologies in Learning*, vol 14, no 10, p 24-321, November 2019
- [3] Ruben Heradio, et al, “Making EJS applications at the OSP Digital Library Available from Moodle,” *International Conference on Remote Engineering and Virtual Instrumentation*, vol 1, no 2, p 5, February 2014
- [4] Visual Paradigm. (2019, May 14). *MVC Sequence Pattern* [Photograph]. Visual Paradigm. <https://www.visual-paradigm.com/servlet/editor-content/guide/uml-unified-modeling-language/how-to-model-mvc-with-uml-sequence-diagram/sites/7/2019/09/mvc-sequence-diagram-example-2.png>

Appendix A: Preliminary Survey Results

Key:

SA = Student Athlete

ATC = Athletic Trainer/Coach

Participant Number	Enter a numerical value 1-5 based on interest level, with 5 being highly interested	Enter a numerical value 1-5 based on potential usage with 5 being you would use it everyday	Enter a numerical value 1-5 based on if you would recommend the application, with 5 being highly recommend	Additional Comments
P1 SA	5	5	5	NA
P2 SA	5	5	5	NA
P3 SA	5	5	5	NA
P4 SA	5	4	5	Automatic reminders to ask us to do it every day?
P5 SA	4	5	5	NA
P6 SA	4	3	4	UI is lacking color
P7 ATC	5	4	4	NA
P8 ATC	5	4	5	NA
P9 ATC	5	4	5	NA
P10 ATC	5	5	5	NA
P11 ATC	5	5	4	NA
P12 ATC	4	5	4	NA

Appendix B: Post Survey Results

Key:

SA = Student Athlete

ATC = Athletic Trainer/Coach

Participant Number	Type if HR (highly recommend), R (recommend), N (neutral), DNR (do not recommend), SDNR (strongly do not recommend) with how you feel about recommending this application to someone
P1 SA	HR
P2 SA	HR
P3 SA	HR
P4 SA	HR
P5 SA	R
P6 SA	N
P7 ATC	R
P8 ATC	HR
P9 ATC	HR
P10 ATC	HR
P11 ATC	HR
P12 ATC	N

Appendix C: Partial Source Code

Below is a portion of the source code written for this web application. A model, view, and controller class are below. The complete source code is available at:

<https://github.com/hp01674/FitNest>

```
//Administrators.js

const {
  Connection,
  Request
} = require('tedious');

const DatabaseConfiguration =
require('./DatabaseConfiguration.json');

const bcrypt = require('bcrypt');

const config = {
  authentication: {
    options: {
      userName: DatabaseConfiguration.userName,
      password: DatabaseConfiguration.password
    },
    type: 'default'
  },
  server: DatabaseConfiguration.server,
  options: {
```

```

        database: DatabaseConfiguration.database,
        encrypt: true
    }
};

const q = require('q');

class Administrators {
    constructor() {}

    static login(admin) {
        let result;

        const deferred = q.defer();

        const connection = new Connection(config);

        connection.on('connect', function (err) {
            const query = `SELECT ID, Username, Password FROM
Administrators WHERE Username='${admin.username}'`;

            const request = new Request(query, function (err,
rowCount) {

                if (err || rowCount === 0) {
                    deferred.resolve(result);
                }
            });
        });
    }
};

```

```

request.on('row', async (columns) => {
    const {password} = admin;
    const hash = columns[2].value;
    const match = await bcrypt.compare(password,
hash);

    if(match) {
        result = {
            id: columns[0].value,
            username: columns[1].value
        };
    }

    deferred.resolve(result);
});

connection.execSQL(request);
});

connection.on('error', (error) => {
    console.log(error);
    deferred.resolve(result);
});

```

```

        return deferred.promise;
    }

    static create(username, password) {
        const deferred = q.defer();
        const connection = new Connection(config);

        connection.on('connect', function (err) {
            bcrypt.hash(password, 10, function (bcryptError,
hash) {
                const query = `INSERT INTO Administrators
(Username, Password) VALUES ('${username}', '${hash}')`;
                const request = new Request(query, function
(err, rowCount) {
                    if (err) {
                        console.log(err);
                        deferred.resolve(false);
                    }
                });

                request.on('doneProc', () => {
                    deferred.resolve(true);
                });
            });
        });
    }

```

```
        connection.execSQL(request);
    });

});

connection.on('error', (error) => {
    console.log(error);
    deferred.resolve(false);
});

return deferred.promise;
}
}
```

```
module.exports = Administrators;
```

```
//login.html (view)
```

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
    <meta charset="utf-8">
```



```

    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta1/dist/css/bootstrap.min.css" rel="stylesheet"
        integrity="sha384-
giJF6kkoqNQ00vy+HMDP7azOuL0xtbfIcaT9wjKHr8RbDVddVHyTfAAsrekwKmP1
" crossorigin="anonymous">
    <title>GSU Athlete Form</title>
</head>

<body>
    <div class="container-fluid pt-5">
        <div class="row">
            <div class="col-2 d-sm-none d-md-block"></div>
            <div class="col-sm-8 col-12 px-2">
                <h3>Admin Login</h3>
                <form method="POST" action="/admin/login">
                    <div class="form-group">
                        <label
for="usernameInput">Username</label>
                            <input type="username" name="username"
class="form-control" id="usernameInput" aria-
describedby="emailHelp"
                                placeholder="Username" required>
                        </div>
                    <div class="form-group">

```

```

        <label
for="passwordInput">Password</label>
        <input type="password" name="password"
class="form-control" id="passwordInput" placeholder="Password"
required>
    </div>
    <div class="text-danger"><%= message %>
</div>
    <button type="submit" class="mt-4 btn btn-
primary">Submit</button>
    </form>
</div>
<div class="col-2 d-sm-none d-md-block"></div>
</div>
</div>
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
beta1/dist/js/bootstrap.bundle.min.js"
    integrity="sha384-
ygbV9kiqUc6oa4msXn9868pTtWMgiQaeYH7/t7LECLbyPA2x65Kgf800JFdroafW
"
    crossorigin="anonymous"></script>
</body>

```

```
</html>
```

```
//show.html (view)
```

```
<!DOCTYPE html>
```

```
<html lang="en">
```

```
<head>
```

```
  <meta charset="utf-8">
```

```
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-  
beta1/dist/css/bootstrap.min.css" rel="stylesheet"  
        integrity="sha384-  
giJF6kkoqNQ00vy+HMDP7azOuL0xtbfIcaT9wjKHr8RbDVddVHyTfAAsrekwKmP1  
" crossorigin="anonymous">
```

```
  <title>Admin Functions</title>
```

```
</head>
```

```
<body>
```

```
  <div class="container-fluid pt-5">
```

```
    <div class="row">
```

```
      <div class="col-2 d-sm-none d-md-block"></div>
```

```
      <div class="pt-4 col-md-8 col-12 px-2">
```

```
        <a href="/logout">Logout</a>
```

```
        <h3>Admin Functions</h3>
```

```

<hr>
<div class="container-fluid">
  <div class="row">
    <div class="pt-4 col-md-4 col-12 px-2">
      <div class="card">
        <div class="card-body">
          <h5 class="card-
title">Create a Coach</h5>
          <hr>
          <p class="card-text">Creates
a coach with a given username and password. The user
          can go back and update
their profile once they've logged in.</p>
          <form method="POST"
action="/coach/new">
            <div class="form-group">
              <label
for="usernameInput">Coach Username</label>
              <input
type="username" name="username" class="form-control"
id="usernameInput" aria-describedby="emailHelp"
placeholder="Username"
              required>

```

```

        </div>
        <div class="form-group">
            <label
for="passwordInput">Coach Password</label>
            <input type="text"
name="password" class="form-control" id="passwordInput"
placeholder="Password" required>
            </div>
            <button type="submit"
class="mt-4 btn btn-success">Create Coach</button>
        </form>
    </div>
</div>
</div>
<div class="pt-4 col-md-4 col-12 px-2">
    <div class="card">
        <div class="card-body">
            <h5 class="card-
title">Create an Admin</h5>
            <hr>
            <p class="card-text">Creates
an Administrator with a given username and password.

```

The admin can go back
and update their password.</p>

```
<form method="POST"
action="/admin/new">
    <div class="form-group">
        <label
for="usernameInput">Admin Username</label>
        <input
type="username" name="username" class="form-control"
id="usernameInput" aria-describedby="emailHelp"
placeholder="Username"
required>
    </div>
    <div class="form-group">
        <label
for="passwordInput">Admin Password</label>
        <input type="text"
name="password" class="form-control" id="passwordInput"
placeholder="Password" required>
    </div>
    <button type="submit"
class="mt-4 btn btn-success">Create Admin</button>
```

```

        </form>
    </div>
</div>
</div>
</div>
<div class="pt-4 col-md-4 col-12 px-2">
    <div class="card">
        <div class="card-body">
            <h5 class="card-
title">Delete a Coach</h5>
            <hr>
            <p class="card-text">Deletes
a Coach with a given username, and <b>will also delete Health
Scores assigned to this coach</b></p>
            <form method="POST"
action="/coach/delete">
                <div class="form-group">
                    <label
for="usernameInput">Coach Username</label>
                    <input
type="username" name="username" class="form-control"
id="usernameInput" aria-describedby="emailHelp"
placeholder="Username"
                    required>

```

```

        </div>
        <button type="submit"
class="mt-4 btn btn-danger">Delete Coach</button>
    </form>
</div>
</div>
</div>
<div class="pt-4 col-md-4 col-12 px-2">
    <div class="card">
        <div class="card-body">
            <h5 class="card-
title">Migrate Health Scores</h5>
            <hr>
            <p class="card-
text">Migrates Health Scores from one Coach to another.</p>
            <form method="POST"
action="/healthtracking/migrate">
                <div class="form-group">
                    <label
for="usernameInput">From</label>
                    <input
type="username" name="from" class="form-control"
id="usernameInput" placeholder="Username"

```



```

                required>
            </div>
            <div class="form-group">
                <label
for="usernameInput">To</label>
                <input
type="username" name="to" class="form-control"
id="usernameInput" placeholder="Username"
                required>
            </div>
            <button type="submit"
class="mt-4 btn btn-primary">Migrate</button>
        </form>
    </div>
</div>
<div class="pt-4 col-md-4 col-12 px-2">
    <div class="card">
        <div class="card-body">
            <h5 class="card-title">Add
Question</h5>
            <hr>

```

```

        <p class="card-text">Adds a
new question to the Student Form</p>
        <form method="POST"
action="/healthtracking/question">
            <div class="form-group">
                <label
for="question">Question Label:</label>
                <input type="text"
name="questionLabel" class="form-control"
id="usernameInput" placeholder="Label"
required>
            </div>
            <button type="submit"
class="mt-4 btn btn-success">Add</button>
        </form>
    </div>
</div>
<div class="pt-4 col-md-4 col-12 px-2">
    <div class="card">
        <div class="card-body">
            <h5 class="card-title">Add
Option</h5>

```

```

        <hr>
        <p class="card-text">Adds an
option to the specified Question</p>
        <form method="POST"
action="/healthtracking/option">
            <div class="form-group">
<label>Question</label>
                <select class="form-
control mb-4" name="selectedQuestion">
                    <%
questions.forEach(function(question) { %>
                        <option
value='<%= question.id %>''>
                            <%=
question.label %>
                                </option>
                            <% }) %>
                    </select>
                </div>
            <div class="form-group">
                <label
for="question">Option Display Text:</label>

```

```

        <input type="text"
name="optionDisplayText" class="form-control"

id="usernameInput" placeholder="Label"

        required>
    </div>
    <div class="form-group">
        <label
for="question">Option Value:</label>
        <input type="number"
name="optionValue" class="form-control"

id="usernameInput" placeholder="Label"

        required>
    </div>
    <button type="submit"
class="mt-4 btn btn-success">Add</button>
    </form>
</div>
</div>
</div>
<div class="pt-4 col-md-4 col-12 px-2">
    <div class="card">
        <div class="card-body">

```

```

        <h5 class="card-
title">Delete a question</h5>

        <hr>

        <p class="card-text">Adds an
option to the specified Question</p>

        <form method="POST"
action="/healthtracking/question/delete">

            <div class="form-group">

                <label>Question</label>

                    <select class="form-
control mb-4" name="selectedQuestion">

                        <%
questions.forEach(function(question) { %>

                            <option
value='<%= question.id %>''>

                                <%=
question.label %>

                                    </option>

                                        <% }) %>

                                            </select>

                                                </div>

                                                    <button type="submit"
class="mt-4 btn btn-danger">Delete</button>

```

```

                                </form>
                                </div>
                                </div>
                                </div>
                                </div>
                                </div>
                                </div>
                                </div>
                                </div>
                                <div class="col-2 d-sm-none d-md-block"></div>
                                </div>
                                </div>

                                <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-
                                beta1/dist/js/bootstrap.bundle.min.js"
                                integrity="sha384-
                                ygbV9kiqUc6oa4msXn9868pTtWMMgiQaeYH7/t7LECLbyPA2x65Kgf80OJFdroafW
                                "
                                crossorigin="anonymous"></script>
                                </body>

                                </html>
                                //AdminController.js
                                const Administrators = require('../models/Administrators');
                                const Questions = require('../models/Questions');
```

```

class AdminController {
  static checkSignedInAsAdmin(req, res, next) {
    if (req.session.admin) {
      next(); // If session exists, proceed to page
    } else {
      const err = new Error('Not logged in!');
      next(err); // Error, trying to access unauthorized
page!
    }
  }
}

static async show(req, res) {
  const questions = await Questions.getAllQuestions();
  res.render('admin/show', {
    questions: questions
  });
}

static showLogin(req, res) {
  res.render('admin/login', {
    message: ''
  });
}
}

```

```

static async login (req, res) {
  if (!req.body.username || !req.body.password) {
    res.render('admin/login', {
      message: 'Please enter both id and password'
    });
  } else {

    const user = {
      username: req.body.username,
      password: req.body.password
    };

    const admin = await Administrators.login(user);
    console.log(admin);
    if (admin !== undefined) {
      req.session.admin = admin;
      res.redirect('/admin');
    } else {
      res.render('admin/login', {
        message: 'Invalid credentials!'
      });
    }
  }
}

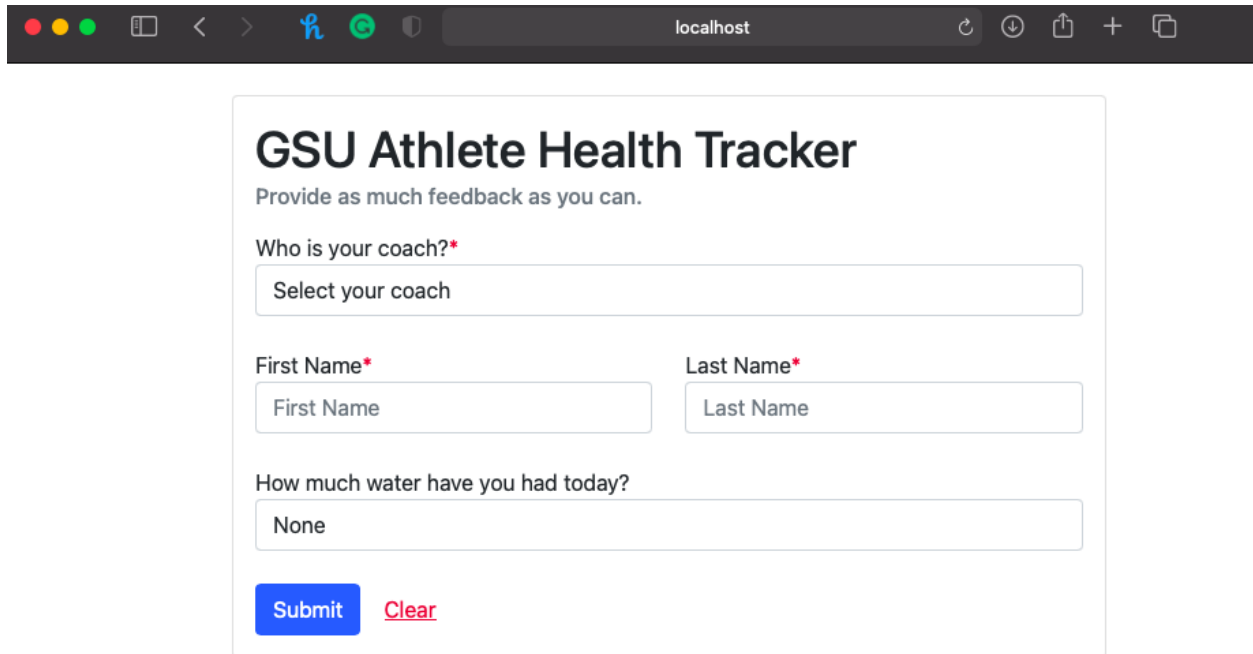
```



```
static async new (req, res) {  
    await Administrators.create(req.body.username,  
req.body.password);  
    res.redirect('/admin');  
  
}  
  
static failedLoginRedirect (err, req, res, next) {  
    res.redirect('/admin/login');  
}  
};  
  
module.exports = AdminController;
```

Appendix D: Screenshots of Application

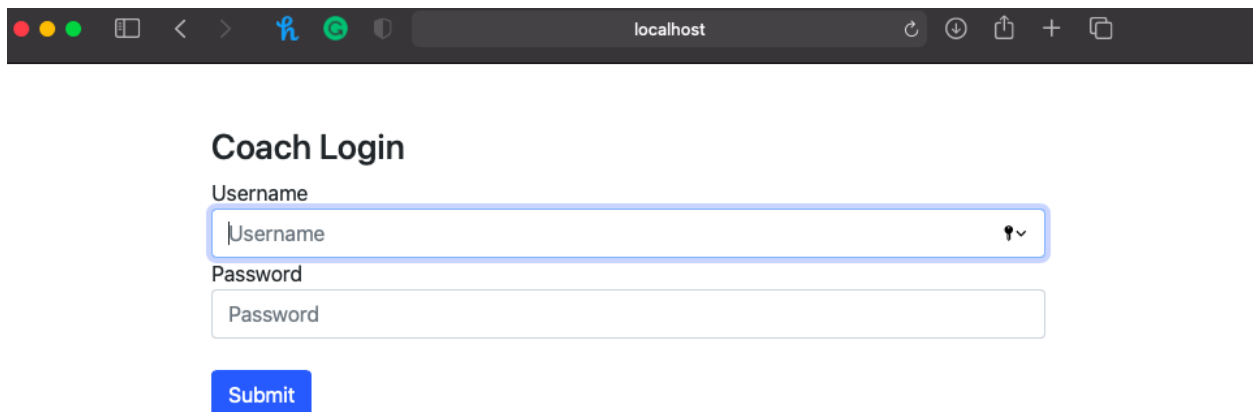
// Home screen student athletes see



A browser window showing the 'GSU Athlete Health Tracker' application. The browser address bar shows 'localhost'. The application title is 'GSU Athlete Health Tracker' with the subtitle 'Provide as much feedback as you can.' Below the title is a form with the following elements:

- A label 'Who is your coach?*' followed by a dropdown menu containing the text 'Select your coach'.
- Two input fields: 'First Name*' and 'Last Name*', each containing the placeholder text 'First Name' and 'Last Name' respectively.
- A label 'How much water have you had today?' followed by a dropdown menu containing the text 'None'.
- Two buttons at the bottom: a blue 'Submit' button and a red 'Clear' link.

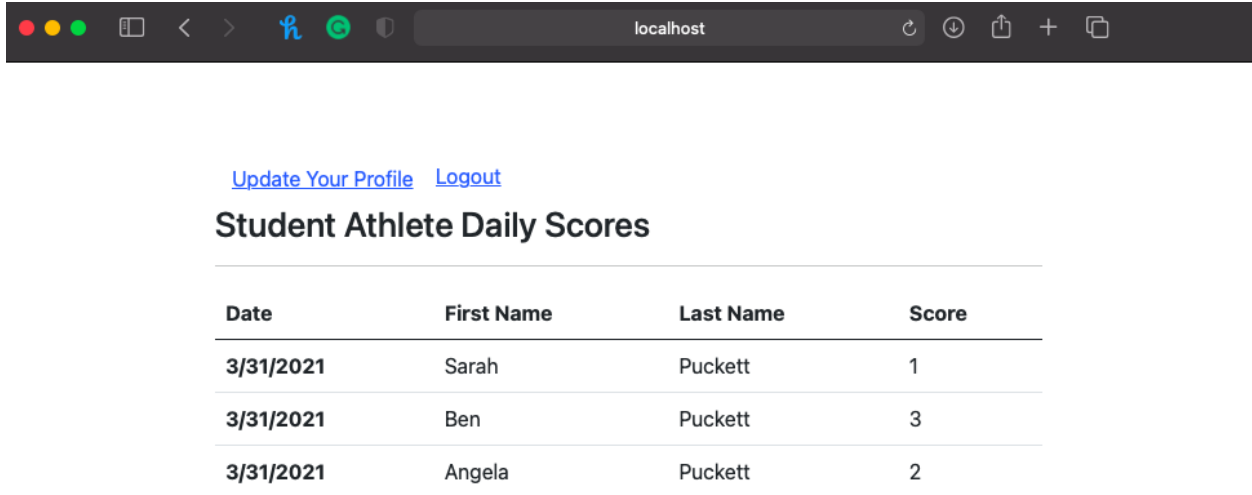
// Coach Log In Screen



A browser window showing the 'Coach Login' application. The browser address bar shows 'localhost'. The application title is 'Coach Login'. Below the title is a form with the following elements:

- A label 'Username' followed by an input field containing the placeholder text 'Username' and a dropdown arrow icon.
- A label 'Password' followed by an input field containing the placeholder text 'Password'.
- A blue 'Submit' button.

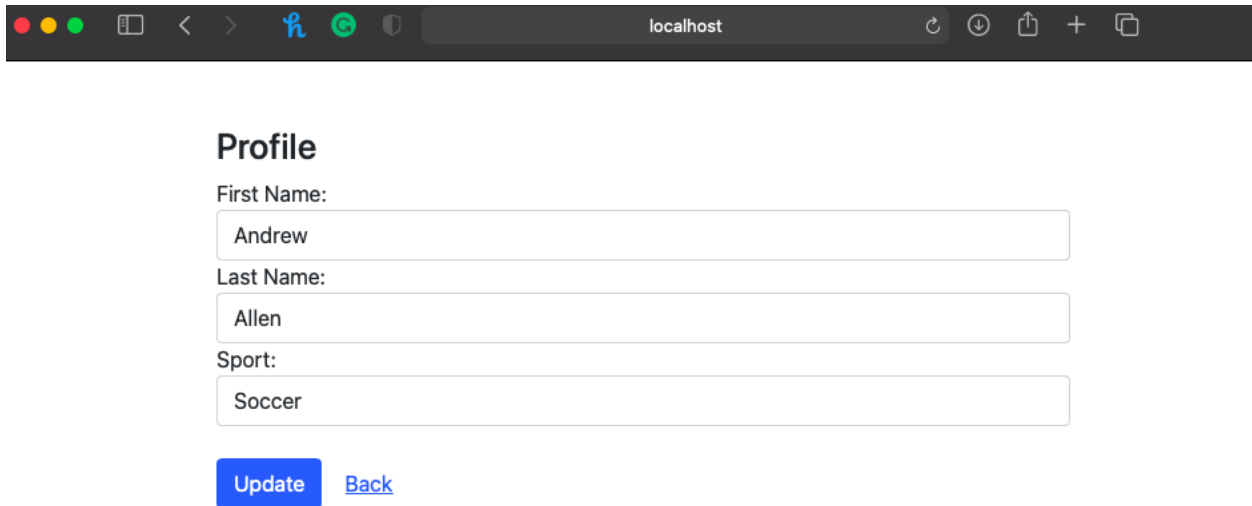
// What coaches see when they log in with valid credentials



The screenshot shows a web browser window with the address bar set to localhost. The page content includes two links at the top: [Update Your Profile](#) and [Logout](#). Below these links is a section header **Student Athlete Daily Scores**. Underneath the header is a table with the following data:

Date	First Name	Last Name	Score
3/31/2021	Sarah	Puckett	1
3/31/2021	Ben	Puckett	3
3/31/2021	Angela	Puckett	2

// Coaches have the option to update their profiles



The screenshot shows a web browser window with the address bar set to localhost. The page content includes a section header **Profile**. Below the header are three form fields: **First Name:** with the value "Andrew", **Last Name:** with the value "Allen", and **Sport:** with the value "Soccer". At the bottom of the form are two buttons: a blue **Update** button and a [Back](#) link.

// Admin Log In

localhost

Admin Login

Username
Username

Password
Password

Submit

//What Admin sees when they log in with valid credentials

localhost

[Logout](#)

Admin Functions

Create a Coach

Creates a coach with a given username and password. The user can go back and update their profile once they've logged in.

Coach Username
Username

Coach Password
Password

Create Coach

Create an Admin

Creates an Administrator with a given username and password. The admin can go back and update their password.

Admin Username
Username

Admin Password
Password

Create Admin

Delete a Coach

Deletes a Coach with a given username, and **will also delete Health Scores assigned to this coach**

Coach Username
Username

Delete Coach

Migrate Health Scores

Migrates Health Scores from one Coach to another.

From
Username

To
Username

Add Question

Adds a new question to the Student Form

Question Label:
Label

Add

Add Option

Adds an option to the specified Question

Question
How much water have you had t

Option Display Text:

Create Coach

Create Admin

Migrate Health Scores

Migrates Health Scores from one Coach to another.

From

Username

To

Username

Migrate

Add Question

Adds a new question to the Student Form

Question Label:

Label

Add

Add Option

Adds an option to the Question

Question

How much water ha

Option Display Text:

Label

Option Value:

Label

Add

Delete a question

Adds an option to the specified Question

Question

How much water have you had t