

Collegiate Basketball Program Database for Prospective Athletes and Parents

Team 32

-

Final Report

Team Members

-

Christopher Douglas
Quang "Lance" Ngo
Sandro Sallenbach

Table of Content

Part 1: Project Requirements

Project Description	2
System Environment	2
Functional Requirements	4
Non-functional Issues	6

Part 2: Project Design

ER Diagram	7
Definition of Entities	7
Definition of Relationships	8
Schemas and Functional Dependencies	9
Tables	12

Part 3: Implementation

Implementation Overview	15
Application Setup	28

Part 4: Project Conclusion

Lessons Learned	31
Future Improvements	32

Part 1: Project Requirement

Project Description

The Collegiate Basketball Program Database Application is a database application that can assist high school students, community college transfer students, as well as their parents, to determine what school would best fit their needs in their basketball careers. Users will be able to use the convenient search tool to look up universities based on name, conference or state the school is located in.

In addition to providing students and parents with current and historical data of each university, the web application will also contain information of current NBA players from every respective school. This information will include draft year, salary outlooks as well as career lengths of each player. It is critical for students to have this information when making a decision on what school they want to attend. Since college has such a high impact on the lives of young basketball players, it is crucial for students to make an educated decision, based on facts and figures.

Our vision has been to have visitors connect to our site and have that meaningful information presented to them in a neat manner that allows them to intuitively browse through all the data and draw a fact-based conclusions about what school would best fit their needs.

Lance, Christopher, and Sandro are the internal stakeholders who could be most impacted by the success of this project. This is due to the fact that these three are putting in the most effort, and their course grades substantially rely on the quality of the project. Dr. Mike Wu also has an interest in the project as he fulfills an advisor role in this project.

High school students are going to be the most important external stakeholders of this project. This tool can help influence and guide their decision of what school to attend. Parents can use the tool when assisting their children in their decision. Community college students who haven't decided on a school to transfer to might also find the website useful.

System Environment

The Collegiate Basketball Program DB app is a 3-tier client server architecture. As it is a web app, the presentation layer is delivered through a client internet browser. We implemented the presentation layer using ReactJS/Redux. Along with the Express web app framework, ReactJS/Redux is currently the most used library to build dynamic web applications. The application layer will be hosted locally, implemented with NodeJS. This web server will contain the application logic that processes and delivers any requested data to the client side, while providing any updates to the data layer initiated by client side events. Lastly, the data layer can and should be hosted on a separate database server using MySQL server. The application and

data layers will connect using a NodeJS-MySQL api. While we anticipate the bulk of our persistent data to be relational, some of the data calls will require translation into JSON objects.

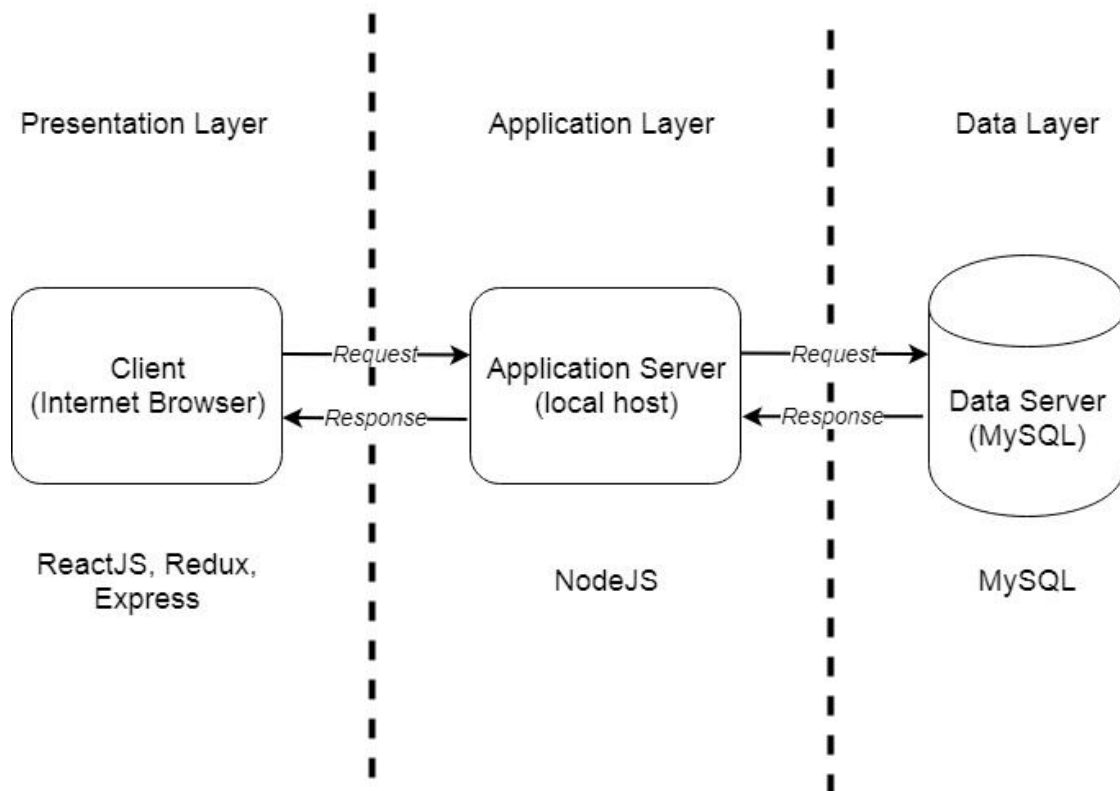


Figure 1: Three-Tier Architecture.

Framework	Languages	Utilization
React	Javascript, HTML, CSS	Front-end rendering into browser
Redux	Javascript	State Store for React
Express	Javascript	Web server in conjunction with NodeJS
NodeJS	Javascript	backend connector to database
MySQL	SQL	database server

Table 1: Technologies Used.

Functional Requirements

Users

The target audience of the Collegiate Basketball Program Database web application will consist of parents of current high school basketball players and community college players, as well as the players themselves. The application can also be used for informational purposes for all collegiate basketball fans. Users can access the application through any regular web browser and access the feature described below. The core audience will interact with the site on a read-only basis. They will not be able to modify or update the data.

There will be a login functionality for administrators only, which will grant authenticated admins read and write permissions. This includes updating the database, adding new entries, as well as deleting outdated records.

Search functionality

- The Collegiate Basketball Program Database web application will aid current highschool basketball players and their parents to search for universities that are a great academic and athletic fit for their college careers.
- All that information will be accessible through a search functionality that lets users find their preferred school.

Order Universities

- The user shall be able to view the search results and order them based on the name of the university.

Filter Universities

- The user shall be able to filter the search results based on the conference of the university.

Display Additional University Info

- The user shall be able to select a university to get further information about the organization.
- The additional information ranges from general college facts & numbers to historical basketball data.

Top Universities

- Top universities are marked with a special emblem, emphasizing their popularity and/or low tuition.

NBA Outlook

- The application will provide data on the chances of making it into the NBA. It will also provide data on the length of players' NBA careers, as well as the salary outlooks.

Positional Outlook

- As different positions in basketball require very different skill sets, it is our goal to differentiate NBA chances, career lengths and salary outlooks by position.

Administrator Login

- Administrators will be able to log in to gain access to admin-only features (as listed below)

Add, Update, Delete

- After authentication, administrators will be able to update the database. Every year, a full draft class of college players enters the NBA and needs to be added to the database.
- In order to keep a historically accurate record, deletions of records is not recommended but will be possible.
- As players progress through their careers in the National Basketball Association, team affiliations, salaries, as well as achievements need frequent updates. Our application will provide means to update and modify the database.

Collegiate Basketball Research

- The database can also provide interesting insights to fans of collegiate basketball. As a research tool, it provides various statistics about college basketball and its various teams.
- As no sign up will be required for the usage of the application, it will be accessible and available to everybody.

Non-Functional Issues

Graphical User Interface

Interface - The Collegiate Basketball Program Database's interface shall be a web application, accessible through standard web browsers.

Implementation - The frontend of the web application shall be built with the React.js framework. This includes HTML, CSS, as well as Javascript.

Non-functional Requirements

Usability - The Collegiate Basketball Program Database web application must be intuitive to navigate. All buttons and links shall be uniform and easy to understand. The search functionality shall be intuitive, even to first time users.

Performance - Latency for user operations such as search shall be less than two seconds. The web application shall also support multiple hundred concurrent users.

Security & Access Control - Basic user operations such as search shall be accessible to everybody. Database manipulation such as deletion or manipulation shall only be accessible to authenticated administrators.

Supportability - The website shall be compatible with all contemporary browsers (like Chrome, Firefox, Safari). The web application is designed for usage on desktop computers or laptops.

Reliability - Errors or crashes on the backend shall not have any effect on previous user actions. It shall only affect the current action and not interfere with other users' experience.

Part 2: Project Design

ER Diagram

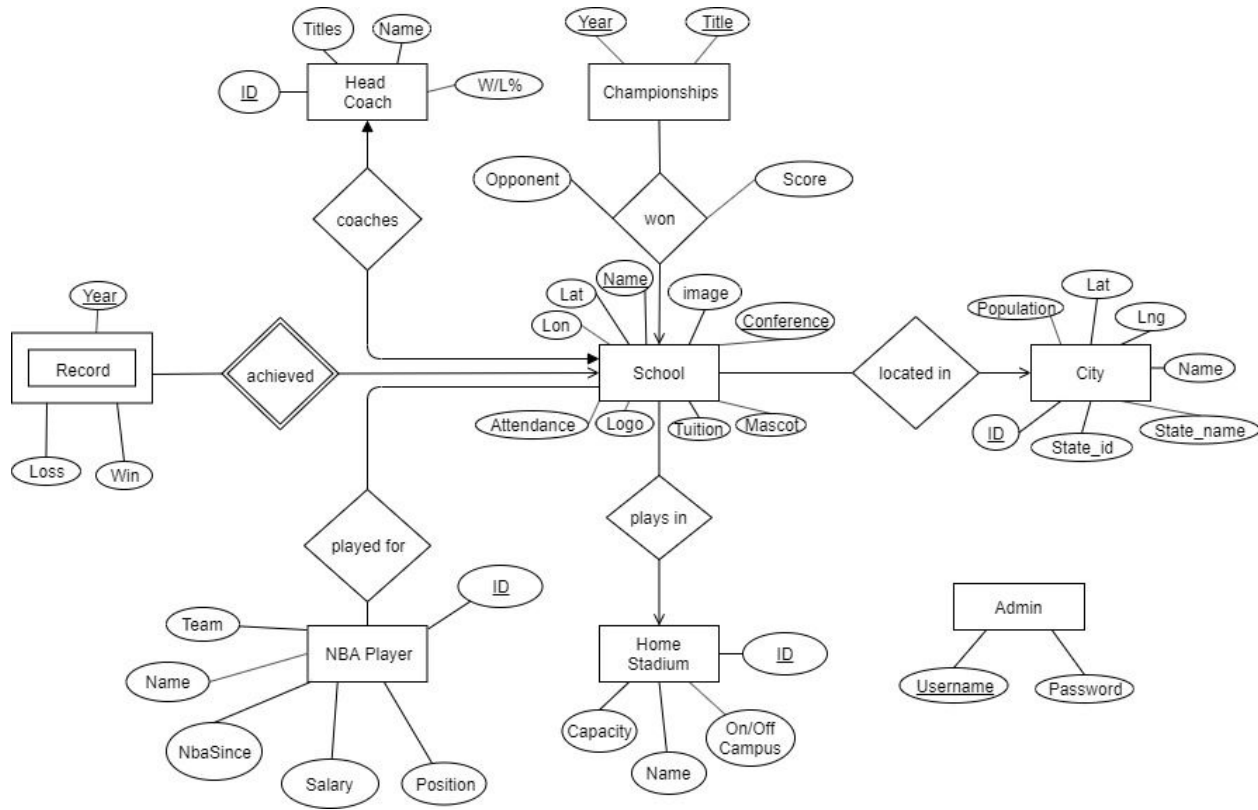


Figure 2: E/R Diagram.

Definition of Entities

School

Uses the name of the school and the conference attended as the key and holds stats such as attendance, tuition, and school mascot, a link to the school logo, a link to the image of the school and the coordinates of the school.

City

The city, the university is located in. This includes a unique ID, Name, State_id, State_name, Population, Latitude and Longitude of the location.

Championships

A championship is a title that a school can win at the end of each season. Championships are identified using the year it was won and the name of the title. Currently, we are considering only NCAA championships.

Head Coach

Head coaches are stored with their name, win/loss percentage, number of titles won as well as a unique ID.

Record

A record tracks the wins and losses recorded by a team in a given year. Record is a weak entity and uses the year, as well as name of school and name of school's conference to be identified.

NBA Player

We track current NBA players that played for their respective colleges as a measure of how successful a college basketball program is. NBA Players are identified by a unique ID with attributes of name, team, position, salary, and an NBA-since attribute.

Home Stadium

Home stadium identifies where a school's basketball team plays its home games. Home stadium is stored with the stadium name, has a number of people that it can hold, and if it's on or off campus, as well as an ID.

Admin

Controls who's allowed to access to add/edit data on the site. Uses the string Username as the key and has a password

Definition of Relationships

Located In

This is the relationship connecting City and School. It will display which school is located in which city, and include the following attributes City_id, School_name, and School_conference.

Won

A relation to connect a school with the championship titles it has won. The won relation uses title and year from the Championships' table, as well as the school's name and school's conference to be uniquely identified. The relation also has the opponent, and score as attributes.

Coaches

The relationship between head coach and school will help us identify who is currently coaching which team. The attributes will include Coach_id, School_name, and School_conference.

Achieved

The Achieved relationship will not be converted into a relation, as it is a weak relationship and would only introduce duplicate knowledge already covered in the weak entity Record.

Played for

A relation to connect an NBA player with the school or schools that he played for. It borrows and ID from the NBA Player table and school name and conference from the School relation.

Plays in

Plays in is the relationship between a School and a Home Stadium. Every school can have exactly one Home Stadium. It borrows stadium ID and school name and school conference from the two other relations.

Schemas and Functional Dependencies

All relations are in BCNF.

School (

Name: string,
Conference: string,
Mascot: string,
Attendance: integer,
Tuition: integer
Logo: string
Image: string
Lat: float
Lon: float
)

FD: Name, Conference → Mascot, Attendance, Tuition, Logo, Image, Lat, Lon

City (

ID: integer,
Name: string,
State_id: string,
State_name: string,
Population: integer
Lat: double,
Lng: double,
)

FD: ID → Name, State_id, State_name, Population, Lat, Lng

Championships (

Year: integer,
Title: string
)

FD: all attributes are part of the key. None can be derived from another.

Head Coach (

ID: integer,
Name: string,
WLP: double,
Titles: integer
)

FD: ID → Name, WLP, Titles

Record (

Year: integer,
School: string,
Conference: string,
Win: integer,
Loss: integer
)

FD: Year, School, Conference → Win, Loss

NBA Player (

ID: integer,
Name: string,
Team: string,
Position: string,
NbaSince: integer,
Salary: integer
)

FD: ID → Name, Team, Position, NbaSince, Salary

Home Stadium (

ID: integer
Name: string,
Capacity: integer,
onOffCampus: boolean
)

FD: ID → Name, Capacity, onOffCampus

Admin (

Username: string,
Password: string
)

FD: Username → Password

Located In (

City_id: integer,
School_name: string,
School_conference: string
)

FD: School_name, School_conference → City_id

Won (

SchoolName: string,
Conference: string,
Title: string,
Year: integer,
Opponent: string,
Score: string
)

FD: Title, Year → SchoolName, Conference, Opponent, Score

Coaches (

Coach_id: integer,
School_name: string,
School_conference: string
)

FD: Coach_id → School_name, School_conference

Played For (

PlayerID: integer,
SchoolName: string,
Conference: string
)

FD: all three attributes are part of the key. None can be derived from another.

Plays In (

StadiumID: integer,
SchoolName: string,
Conference: string
)

FD: SchoolName,Conference → StadiumID

Tables

The complete tables can be found on the project's GitHub repository in the form of a SQL Script.

	SchoolName	conf	attendance	tuition	mascot	logo	image
▶	San Jose State University	Mountain West	32828	7418	Sammy	https://www.ncaa.com/sites/default/files/images/log...	http://newdev.sjsu.edu/_images/
	San Diego State University	Mountain West	33778	7510	Montezuma the Aztec Warrior	https://www.ncaa.com/sites/default/files/images/log...	https://www.cappex.com/sites/
	California State University, Fresno	Mountain West	22071	6313	Victor E. Bulldog	https://www.ncaa.com/sites/default/files/images/log...	https://media.licdn.com/dms/ima
	University of Nevada, Las Vegas	Mountain West	30471	6197	Hey Rebl	https://www.ncaa.com/sites/default/files/images/log...	https://library.unr.edu/libraries/
	University of Nevada	Mountain West	20194	7142	Alphie, Wolfie Jr and Luna	https://www.ncaa.com/sites/default/files/images/log...	https://upload.wikimedia.org/wik
	University of New Mexico	Mountain West	27353	7350	Lobo	https://www.ncaa.com/sites/default/files/images/log...	https://d3el53au0d7w62.cloudfr
	University of Wyoming	Mountain West	12397	5055	Pistol Pete and Cowboy Joe	https://www.ncaa.com/sites/default/files/images/log...	http://www.uwyo.edu/foundatic
	Colorado State University	Mountain West	33413	11052	CAM the Ram	https://www.ncaa.com/sites/default/files/images/log...	https://admissions.colostate.ed
	Boise State University	Mountain West	25540	7080	Buster Bronco	https://www.ncaa.com/sites/default/files/images/log...	https://www.cappex.com/sites/

Figure 3: Schools Table.

	ID	name	State_id	State_name	Population	Lat	Lng
▶	1	San Jose	CA	California	1821899	37.3021	-121.8489
	2	San Diego	CA	California	3210314	32.8312	-117.1225
	3	Fresno	CA	California	698021	36.7831	-119.7941
	4	Las Vegas	NV	Nevada	2073045	36.2333	-115.2654
	5	Reno	NV	Nevada	433271	39.5497	-119.8483
	6	Albuquerque	NM	New Mexico	758523	35.1053	-106.6464
	7	Laramie	WY	Wyoming	33508	41.3099	-105.6085
	8	Fort Collins	CO	Colorado	303184	40.5475	-105.0651
	9	Boise	ID	Idaho	385218	43.6007	-116.2312
	10	Logan	UT	Utah	100774	41.74	-111.8419

Figure 4: City Table.

	year	title
▶	2003	NCAA Champs
	2004	NCAA Champs
	2005	NCAA Champs
	2006	NCAA Champs
	2007	NCAA Champs
	2008	NCAA Champs
	2009	NCAA Champs
	2010	NCAA Champs
	2011	NCAA Champs
	2012	NCAA Champs

Figure 5: Championships Table.

	ID	Name	WLP	Titles
▶	1	Brian Dutcher	0.642	1
	2	Allen Edwards	0.495	0
	3	Justin Hutson	0.719	0
	4	Niko Medved	0.375	0
	5	Marvin Menzies	0.5	0
	6	Eric Musselman	0.764	4
	7	Dave Pilipovich	0.428	0
	8	Jean Prioleau	0.131	0
	9	Leon Rice	0.602	1
	10	Craig Smith	0.8	2

Figure 6: Head Coach Table.

	school	conference	year	wins	losses
▶	Arizona State University	Pacific 12	2019	12	6
	Boise State University	Mountain West	2019	7	11
	California State University, Fresno	Mountain West	2019	13	5
	Colorado State University	Mountain West	2019	7	11
	Oregon State University	Pacific 12	2019	10	8
	San Diego State University	Mountain West	2019	11	7
	San Jose State University	Mountain West	2019	1	17
	Stanford University	Pacific 12	2019	8	10
	U.S. Air Force Academy	Mountain West	2019	8	10
	University of Arizona	Pacific 12	2019	8	10

Figure 7: Record Table.

	id	name	college	position	nbasince	team	salary
▶	1	Kyle Anderson	University of California, Los Angeles	PF	2014	Grizzlies	458725
	2	Ryan Anderson	University of California, Berkeley	PF	2008	Rockets	456258
	3	Trevor Ariza	University of California, Los Angeles	SF	2004	Kings	326584
	4	Deandre Ayton	University of Arizona	C	2018	Suns	547896
	5	Lonzo Ball	University of California, Los Angeles	PG	2017	Pelicans	214587
	6	Aron Baynes	Washington State University	C	2009	Suns	698745
	7	Jordan Bell	University of Oregon	PF	2017	Timberwolves	452369
	8	Jonah Bolden	University of California, Los Angeles	SF	2017	Sixers	858523
	9	Chris Boucher	University of Oregon	PF	2017	Raptors	456879
	10	Dillon Brooks	University of Oregon	F	2017	Grizzlies	214639

Figure 8: NBA Player Table.

	ID	StadiumName	onOffCampus	capacity
▶	0	Event Center at San Jose State University	1	5000
	1	Viejas Arena	1	12411
	2	Save Mart Center	1	15544
	3	Thomas & Mack Center	1	17932
	4	Lawlor Events Center	1	11536
	5	Dreamstyle Arena	0	15411
	6	Arena0Auditorium	1	15028
	7	Moby Arena	1	8745
	8	ExtraMile Arena	1	12644
	9	Dee Glen Smith Spectrum	1	10270

Figure 9: Home Stadium Table.

	username	password
▶	AmitGarg	\$2a\$10\$VinYAeqKAsgR6L5qLRMuOQYucfPo3of...
	AvinavTyagi	\$2a\$10\$VinYAeqKAsgR6L5qLRMuOQYucfPo3of...
	BarackObama	\$2a\$10\$VinYAeqKAsgR6L5qLRMuOQYucfPo3of...
	Batman	\$2a\$10\$VinYAeqKAsgR6L5qLRMuOQYucfPo3of...
	BillGates	\$2a\$10\$VinYAeqKAsgR6L5qLRMuOQYucfPo3of...
	BruceWayne	\$2a\$10\$VinYAeqKAsgR6L5qLRMuOQYucfPo3of...
	ChrisDouglas	\$2a\$10\$VinYAeqKAsgR6L5qLRMuOQYucfPo3of...
	ElonMusk	\$2a\$10\$VinYAeqKAsgR6L5qLRMuOQYucfPo3of...
	HenryFord	\$2a\$10\$VinYAeqKAsgR6L5qLRMuOQYucfPo3of...
	JuliusCaesar	\$2a\$10\$VinYAeqKAsgR6L5qLRMuOQYucfPo3of...

Figure 10: Admin Table.

	City_id	School_name	School_conference
▶	1	San Jose State University	Mountain West
	2	San Diego State University	Mountain West
	3	California State University, Fresno	Mountain West
	4	University of Nevada, Las Vegas	Mountain West
	5	University of Nevada	Mountain West
	6	University of New Mexico	Mountain West
	7	University of Wyoming	Mountain West
	8	Colorado State University	Mountain West
	9	Boise State University	Mountain West
	10	Utah State University	Mountain West

Figure 11: Located in Table.

	schoolName	conference	year	title	opponent	score
▶	Duke	ACC	2001	NCAA Champs	University of Arizona	82-72
	Maryland	Big 10	2002	NCAA Champs	Indiana	64-52
	Syracuse	ACC	2003	NCAA Champs	Kansas	81-78
	Connecticut	American	2004	NCAA Champs	Georgia Tech	82-73
	North Carolina	ACC	2005	NCAA Champs	Illinois	75-70
	Florida	SEC	2006	NCAA Champs	University of California, Los Angeles	73-57
	Florida	SEC	2007	NCAA Champs	Ohio State	84-75
	Kansas	Big 12	2008	NCAA Champs	Memphis	75-68
	North Carolina	ACC	2009	NCAA Champs	Michigan State	89-72
	Duke	ACC	2010	NCAA Champs	Butler	61-59

Figure 12: Won Table.

	Coach_id	School_name	School_conference
▶	1	San Diego State University	Mountain West
	2	University of Wyoming	Mountain West
	3	California State University, Fresno	Mountain West
	4	Colorado State University	Mountain West
	5	University of Nevada, Las Vegas	Mountain West
	6	University of Nevada	Mountain West
	7	U.S. Air Force Academy	Mountain West
	8	San Jose State University	Mountain West
	9	Boise State University	Mountain West
	10	Utah State University	Mountain West

Figure 13: Coaches Table.

	pid	schoolname	conference
▶	1	University of California, Los Angeles	Pacific 12
	2	University of California, Berkeley	Pacific 12
	3	University of California, Los Angeles	Pacific 12
	4	University of Arizona	Pacific 12
	5	University of California, Los Angeles	Pacific 12
	6	Washington State University	Pacific 12
	7	University of Oregon	Pacific 12
	8	University of California, Los Angeles	Pacific 12
	9	University of Oregon	Pacific 12
	10	University of Oregon	Pacific 12

Figure 14: Played For Table.

	SchoolName	conf	id
▶	San Jose State University	Mountain West	0
	San Diego State University	Mountain West	1
	California State University, Fresno	Mountain West	2
	University of Nevada, Las Vegas	Mountain West	3
	University of Nevada	Mountain West	4
	University of New Mexico	Mountain West	5
	University of Wyoming	Mountain West	6
	Colorado State University	Mountain West	7
	Boise State University	Mountain West	8
	Utah State University	Mountain West	9

Figure 15: Plays In Table.

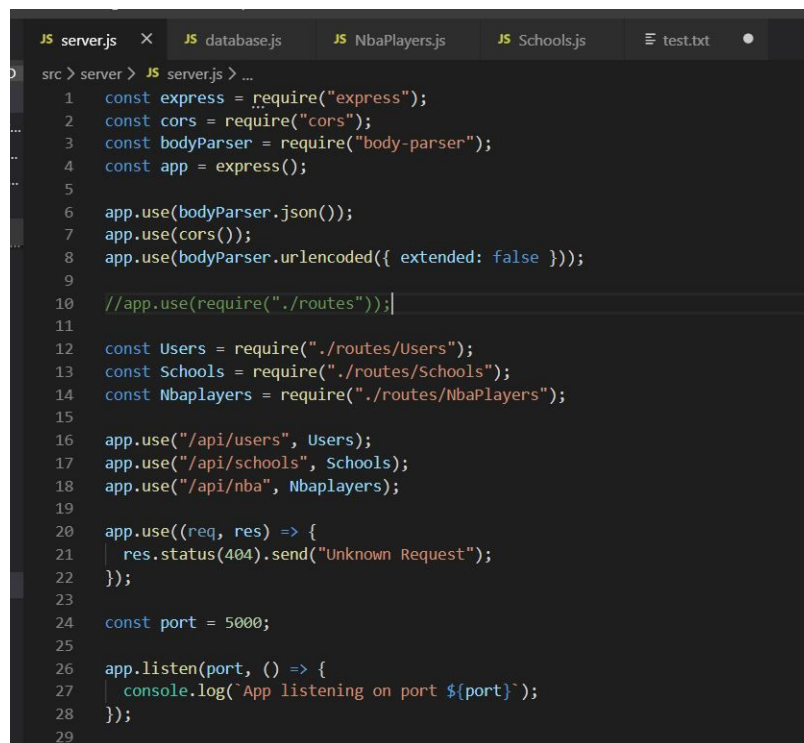
Part 3: Implementation

Implementation Overview

Backend

The backend of the application was implemented in Javascript on the NodeJS framework with appropriate MySQL module imports to serve as the database connector.

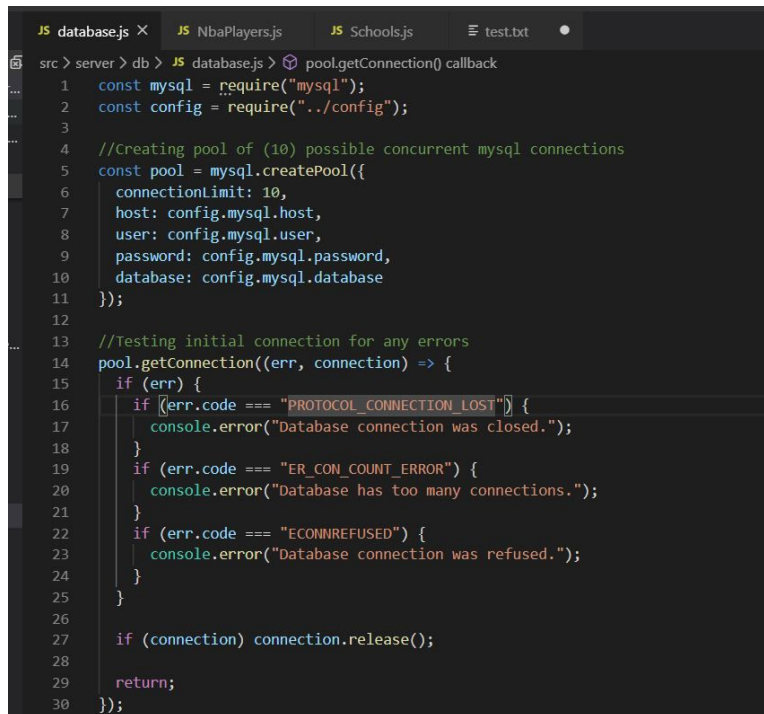
The primary application file for the server is “server.js”. It defines the various endpoints required to access backend functionality and data, and imports the necessary modules as middleware. The “server.js” file ultimately initiates the server application to listen for requests on a specified port.



```
src > server > JS server.js > ...
1  const express = require("express");
2  const cors = require("cors");
3  const bodyParser = require("body-parser");
4  const app = express();
5
6  app.use(bodyParser.json());
7  app.use(cors());
8  app.use(bodyParser.urlencoded({ extended: false }));
9
10 //app.use(require("./routes"));
11
12 const Users = require("./routes/Users");
13 const Schools = require("./routes/Schools");
14 const NbaPlayers = require("./routes/NbaPlayers");
15
16 app.use("/api/users", Users);
17 app.use("/api/schools", Schools);
18 app.use("/api/nba", NbaPlayers);
19
20 app.use((req, res) => {
21 |   res.status(404).send("Unknown Request");
22 | });
23
24 const port = 5000;
25
26 app.listen(port, () => {
27 |   console.log(`App listening on port ${port}`);
28 | });
29
```

[Figure 16: Server.js file]

A second critical file is the “database.js” file which creates and exports a pool of MySQL connections that other files will use to query the database. The MySQL connections are created using credentials which are imported from a config file so that it is hidden from GitHub.



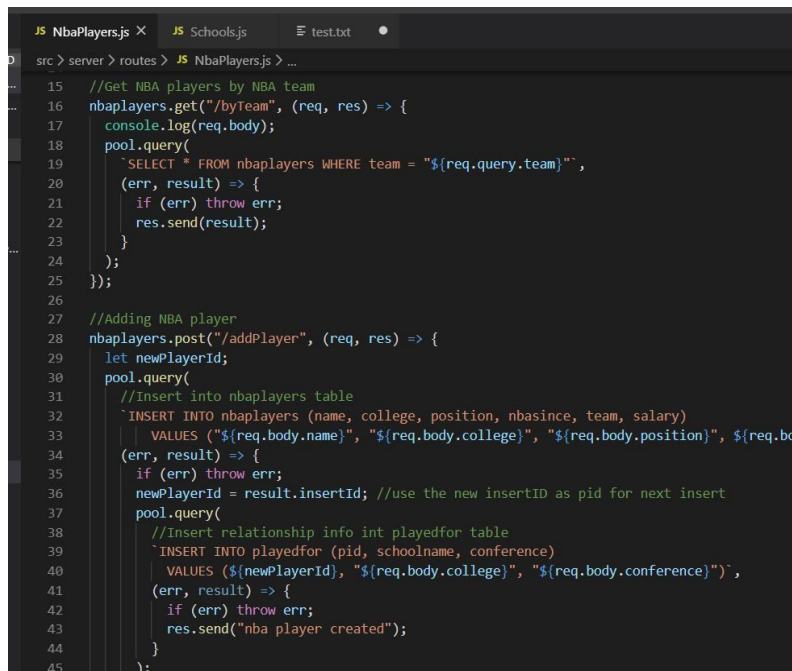
```

JS database.js X JS NbaPlayers.js JS Schools.js test.txt
src > server > db > JS database.js > pool.getConnection() callback
1  const mysql = require("mysql");
2  const config = require("../config");
3
4  //Creating pool of (10) possible concurrent mysql connections
5  const pool = mysql.createPool({
6    connectionLimit: 10,
7    host: config.mysql.host,
8    user: config.mysql.user,
9    password: config.mysql.password,
10   database: config.mysql.database
11  });
12
13  //Testing initial connection for any errors
14  pool.getConnection((err, connection) => {
15    if (err) {
16      if (err.code === "PROTOCOL_CONNECTION_LOST") {
17        console.error("Database connection was closed.");
18      }
19      if (err.code === "ER_CON_COUNT_ERROR") {
20        console.error("Database has too many connections.");
21      }
22      if (err.code === "ECONNREFUSED") {
23        console.error("Database connection was refused.");
24      }
25    }
26
27    if (connection) connection.release();
28
29    return;
30  });

```

[Figure 17: database.js file]

Lastly, various middleware files are implemented as needed for all the required functionality such as authentication and database query. For example, the “NbaPlayers.js” file employs the Express framework to define all the GET and POST functions for the requests that come from the front end. The defined functions incorporate MySQL queries if data retrieval, update, or deletion is involved.



```

JS NbaPlayers.js X JS Schools.js test.txt
src > server > routes > JS NbaPlayers.js > ...
15  //Get NBA players by NBA team
16  nbaplayers.get("/byTeam", (req, res) => {
17    console.log(req.body);
18    pool.query(
19      `SELECT * FROM nbaplayers WHERE team = "${req.query.team}"`,
20      (err, result) => {
21        if (err) throw err;
22        res.send(result);
23      }
24    );
25  });
26
27  //Adding NBA player
28  nbaplayers.post("/addPlayer", (req, res) => {
29    let newPlayerId;
30    pool.query(
31      //Insert into nbaplayers table
32      `INSERT INTO nbaplayers (name, college, position, nbasince, team, salary)
33      VALUES ("${req.body.name}", "${req.body.college}", "${req.body.position}", "${req.body.nbasince}", "${req.body.team}", "${req.body.salary}")`,
34      (err, result) => {
35        if (err) throw err;
36        newPlayerId = result.insertId; //use the new insertID as pid for next insert
37        pool.query(
38          //Insert relationship info into playedfor table
39          `INSERT INTO playedfor (pid, schoolname, conference)
40          VALUES (${newPlayerId}, "${req.body.college}", "${req.body.conference}")`,
41          (err, result) => {
42            if (err) throw err;
43            res.send("nba player created");
44          }
45        );
46      }
47    );
48  });

```

[Figure 18: NbaPlayers.js file]

Frontend

The frontend of the application was built using HTML, CSS and Javascript, which are all part of the React framework. React is one of the most popular front-end frameworks available. It is developed and maintained by Facebook and offers an easy to approach way to create a frontend for any application. All front-end components were built with the Material-UI user interface framework. Most components were modified and customized to our needs with CSS and HTML modifications. Redux was another frontend tool used in this application. It helps to manage states and functions as the connector between the GUI and the backend of the application.

The application was built page by page, starting with the landing page and the about us page. Next, components were incrementally added to the Search Overview Page and the University Details Page. Three additional pages, showing school championship games, school records and NBA Players were created after that. The Administrator Panel was implemented last, as it relied on the rest of the application already being completed first.

GUI Screenshots

The following is a demo walkthrough of the entire application. Each screenshot is accompanied by a list of entities and attributes involved. Each screenshot will also have a description of what functional requirements it is associated with. Since all tables are in BCNF and each relation only has one functional dependency (based on the table's keys), please refer to *Schemas and Functional Dependencies* for that information. Relationships are treated as regular relations as described in *Definition of Relationships*. Screenshots of all relations can be found in *Tables*.

The Landing page prompts the user to enter a search prompt. The user can search by University name, Conference name or State.

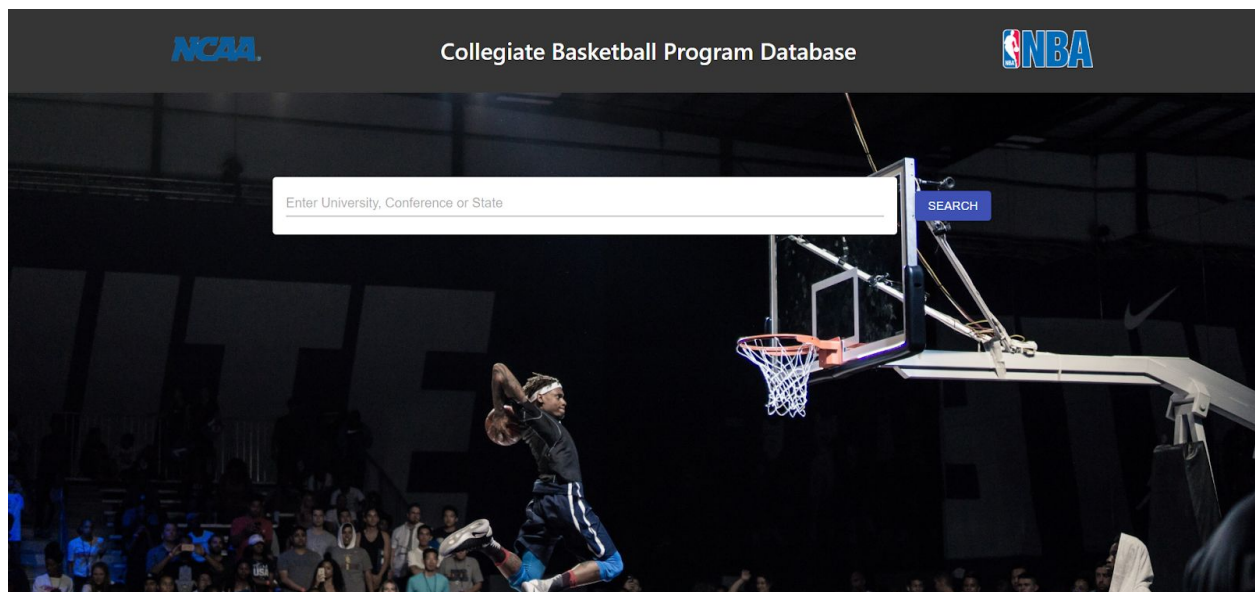


Figure 19: Landing Page.

After entering a search query, the backend of the application tries to match the search string with the name of a university, the name of a conference or the state, a university is located in. The backend will inspect the Schools and City tables. The information presented is based on the Schools table.

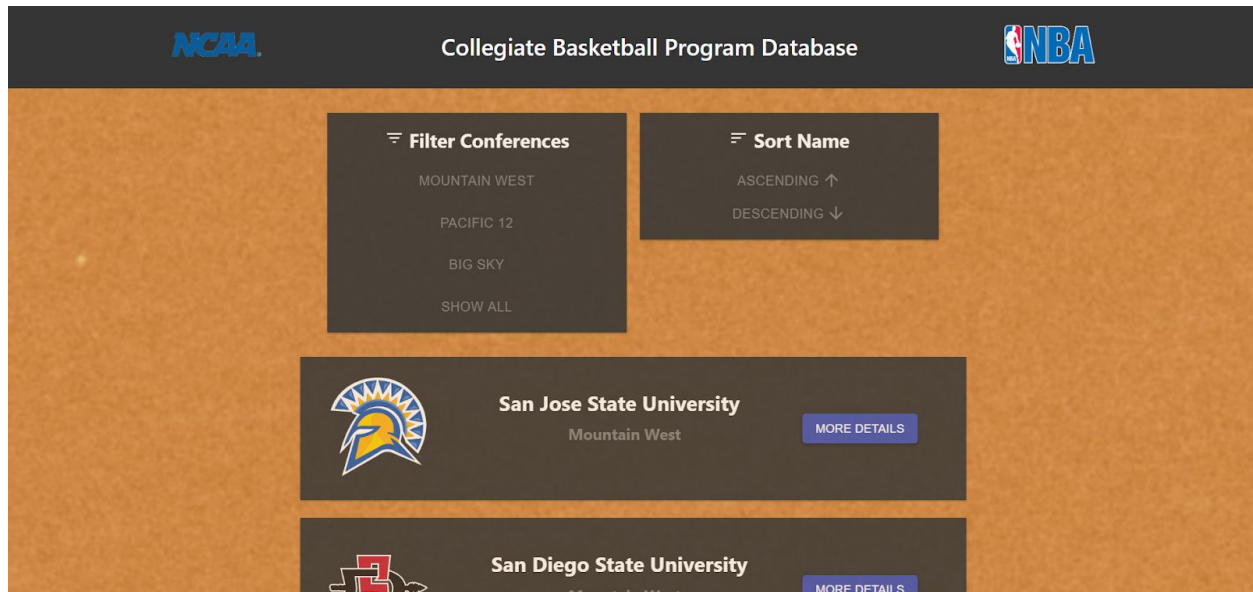


Figure 20: Search Result Overview.

On the Search Overview Page, a user can search all the results by name in ascending or descending order. The screenshot also shows a star next to Arizona State University, which is part of the Top Universities functional requirement.

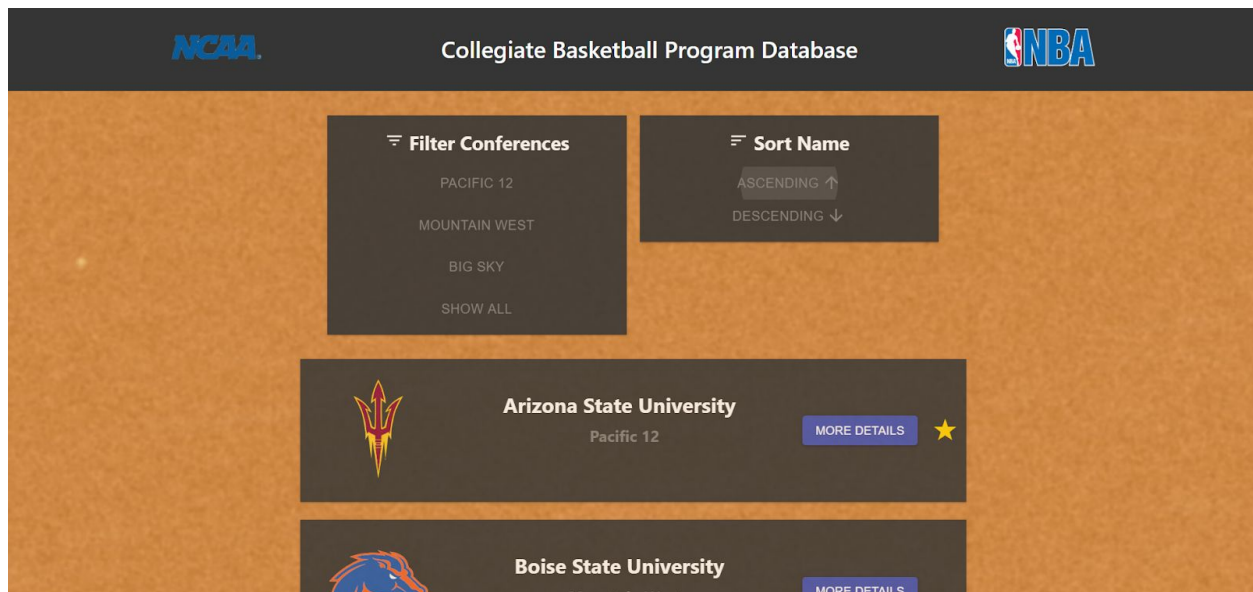


Figure 21: Sort by Name, Ascending.

The user is also able to filter the search result based on the Conferences represented in the search result. Clicking on 'SHOW ALL' will return all results that initially matched the search query.

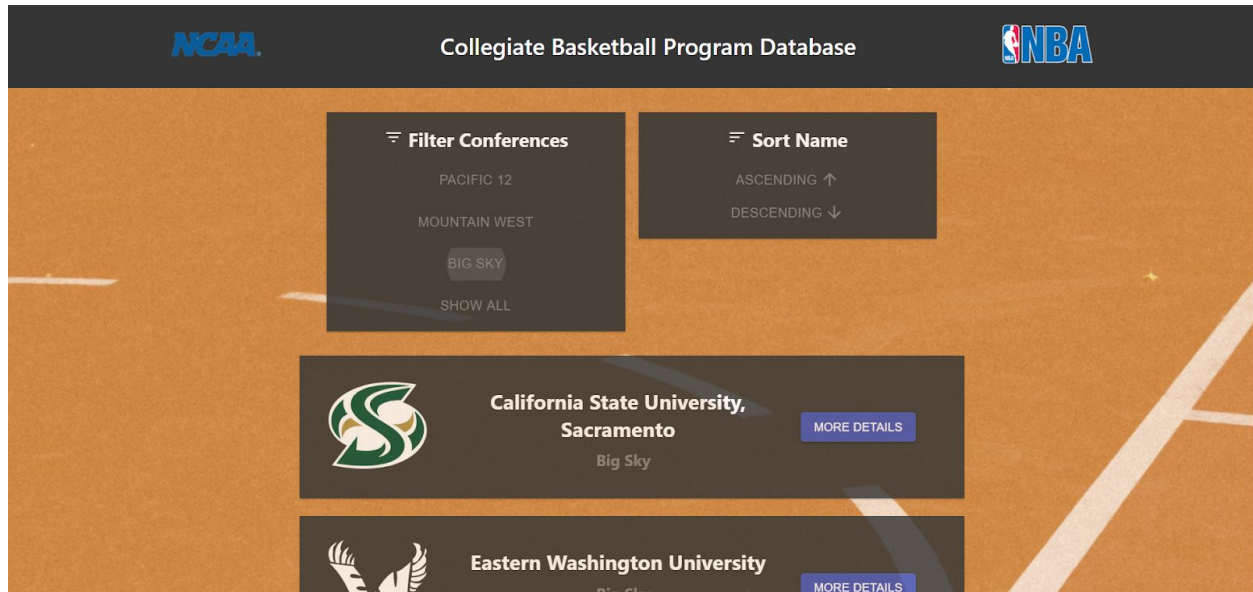


Figure 22: Filter by Conference.

By clicking on 'MORE DETAILS' on the Search Overview Page, a user is presented with more information about the university. It draws information from the Schools, Home Stadium, Head Schools, and City tables.

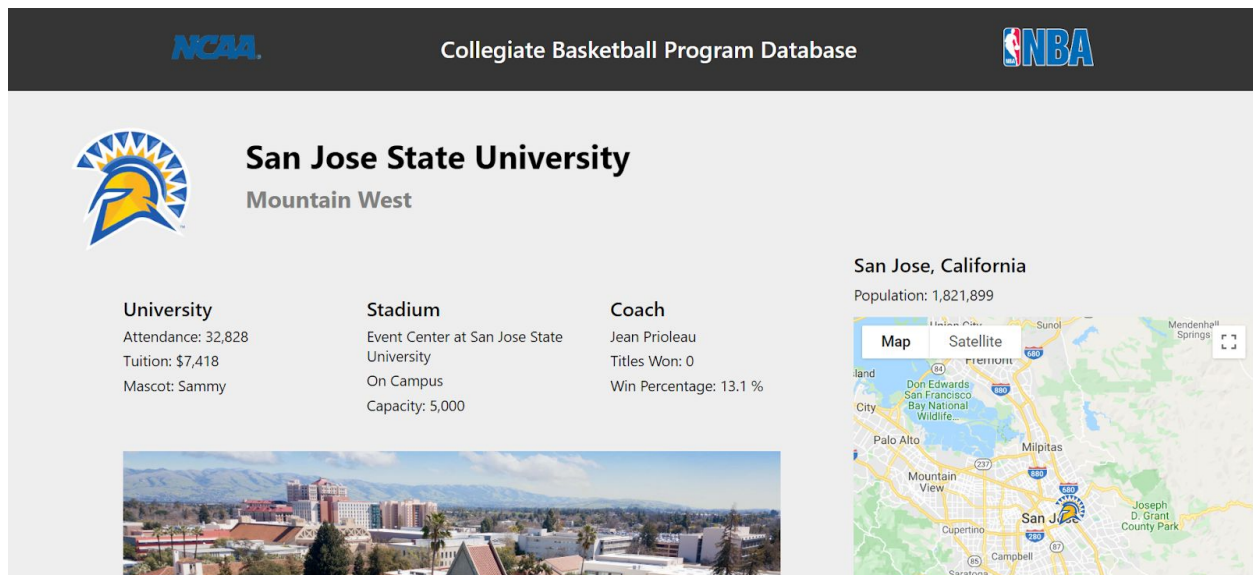
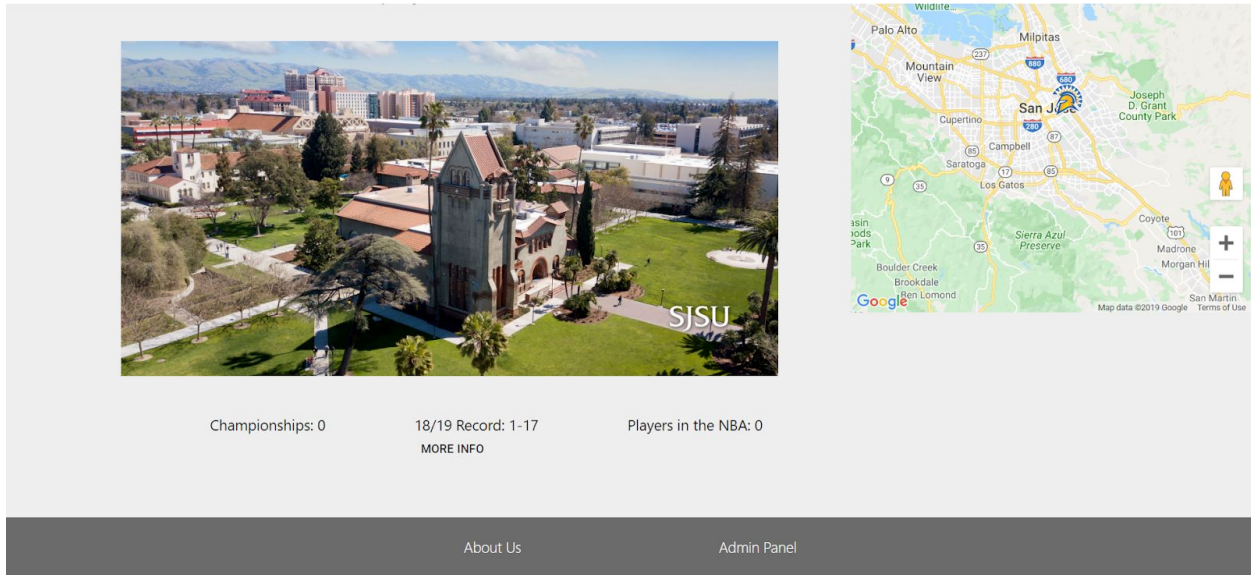


Figure 23: University Details Page 1.

This screenshot shows the second half of the Individual University Page. It shows a quick overview of Championships, Record and Players in the NBA and draws from the three corresponding tables.

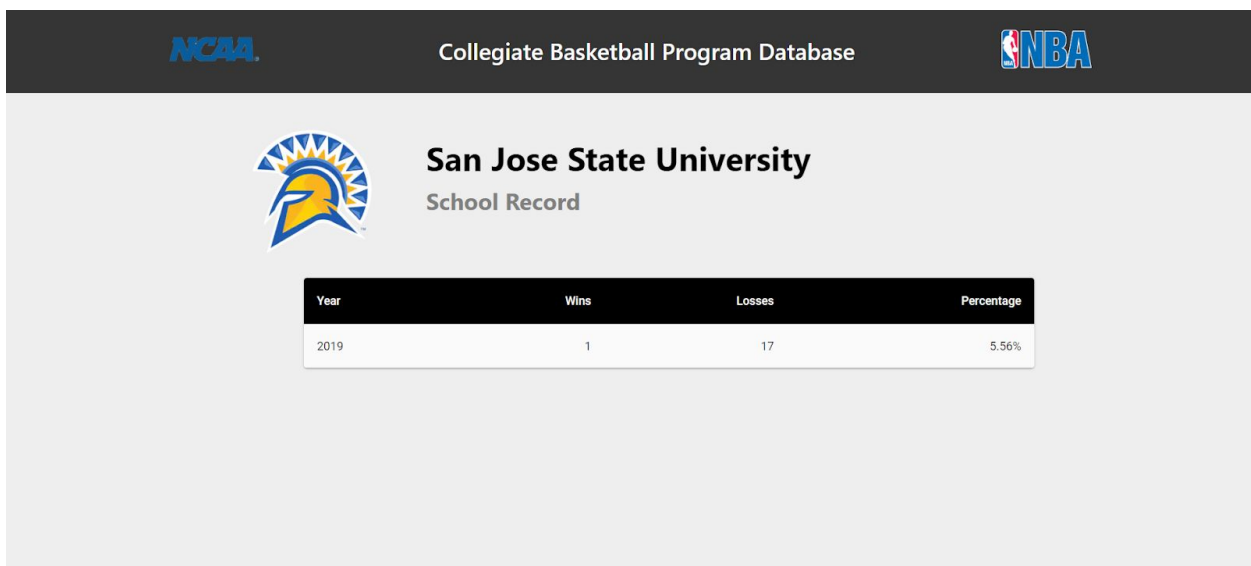


Championships: 0 18/19 Record: 1-17 [MORE INFO](#) Players in the NBA: 0

[About Us](#) [Admin Panel](#)

Figure 24: University Details Page 2.

This screenshot shows a more detailed overview of the school record and draws from the Record relation.

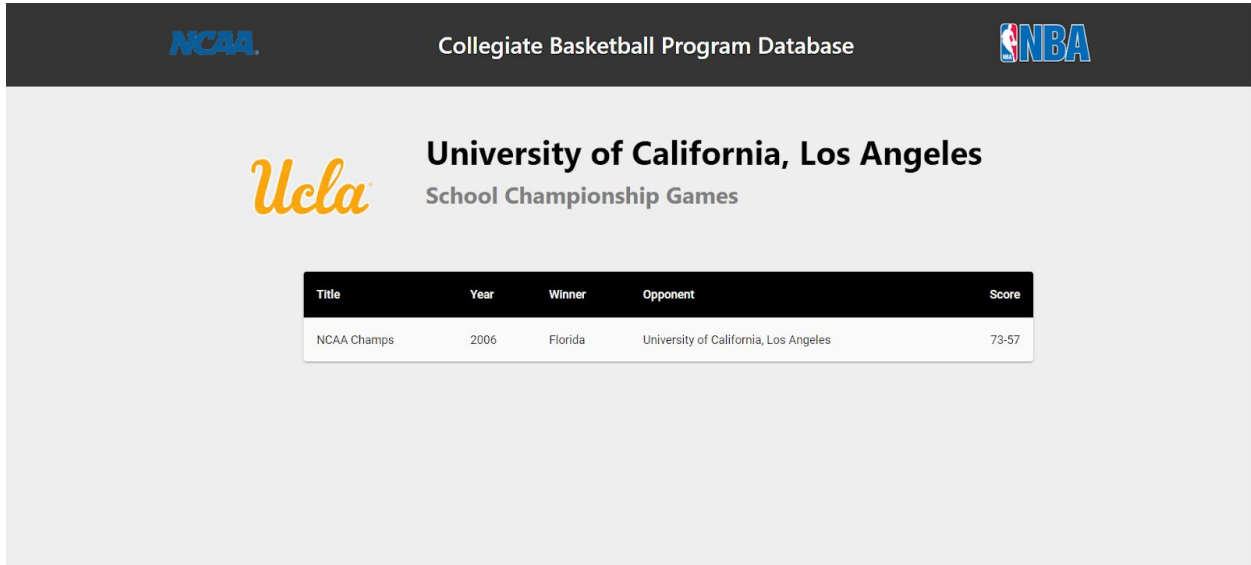


San Jose State University
School Record

Year	Wins	Losses	Percentage
2019	1	17	5.56%

Figure 25: University Game Record.


This screenshot shows a more detailed overview of the school championship games and draws form the Championships and Won relations.



Title	Year	Winner	Opponent	Score
NCAA Champs	2006	Florida	University of California, Los Angeles	73-57

Figure 26: University Championship Games.

This screenshot shows a more detailed overview of the university's graduates in the NBA and draws form the NBA Player and Played For tables.



Player Name	Position	Draft Year	Current NBA Team	Salary
Kyle Anderson	PF	2014	Grizzlies	\$ 458,725
Trevor Ariza	SF	2004	Kings	\$ 326,584
Lonzo Ball	PG	2017	Pelicans	\$ 214,587
Jonah Bolden	SF	2017	Sixers	\$ 858,523
Aaron Holiday	PG	2018	Pacers	\$ 245,786

Figure 27: University Graduates in the NBA.

Administrators have to authenticate themselves in order to gain access to add, update and delete functions. Username and password are compared to the Admin table to verify authentication of the administrator.

The screenshot shows the 'Administrator Login' page of the 'Collegiate Basketball Program Database'. The page has a dark header with the NCAA and NBA logos. The main content area is light gray and features a white login form. The form has two input fields: 'Username' with the value 'SandroSallenbach' and 'Password' with masked characters. Below the fields is a blue 'LOGIN' button. At the bottom of the page, there is a dark footer with links for 'About Us' and 'Admin Panel'.

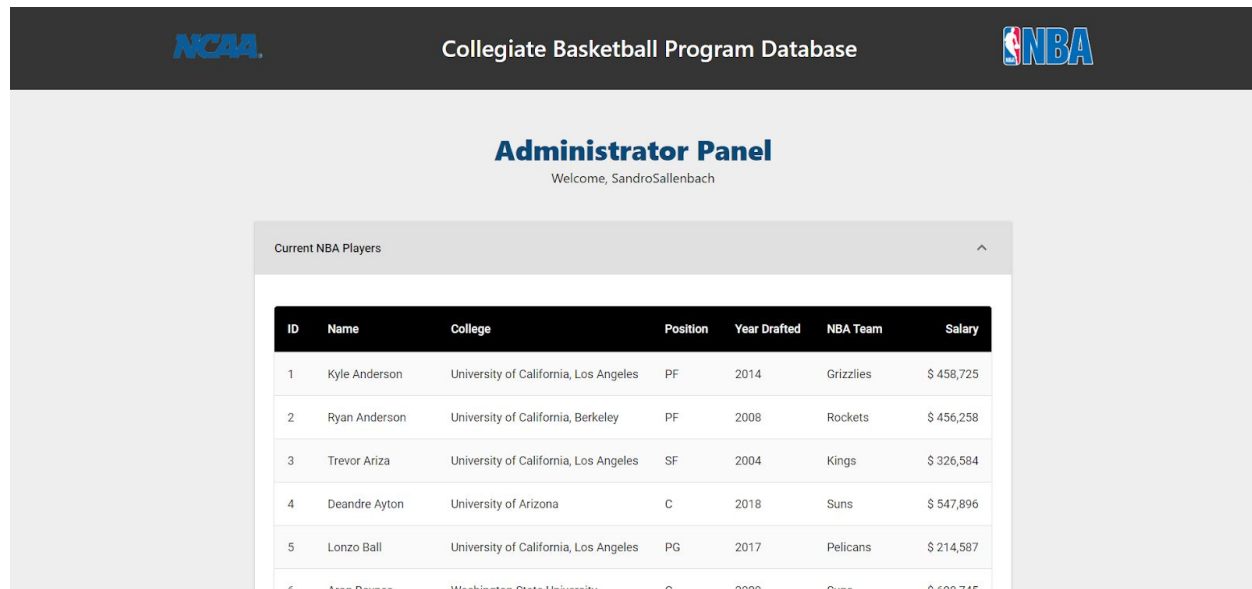
Figure 28: Administrator Login.

After logging in, administrators are presented with an admin panel that features all current NBA players (drop down) as well as options to add, edit and delete a player from the database. This page draws from the Admin and NBA Player tables.

The screenshot shows the 'Administrator Panel' page. The header is identical to Figure 28. The main content area is light gray and features a dark blue title 'Administrator Panel' with a subtitle 'Welcome, SandroSallenbach'. Below the title is a dropdown menu labeled 'Current NBA Players'. The page is divided into three columns for player management: 'Add a new Player', 'Edit a Player', and 'Delete a Player'. The 'Add a new Player' column has four input fields: 'Name *', 'College *', 'Conference *', and 'Position *'. The 'Edit a Player' column has three input fields: 'Player ID *', 'New NBA Team *', and 'New Salary *'. The 'Delete a Player' column has one input field: 'Player ID *'. Each column has a 'SUBMIT' button at the bottom.

Figure 29: Administrator Panel.

When expanding the drop down table, the administrator is presented with all current NBA players in the database. As mentioned above, this page draws from the NBA Players table to gather this data.

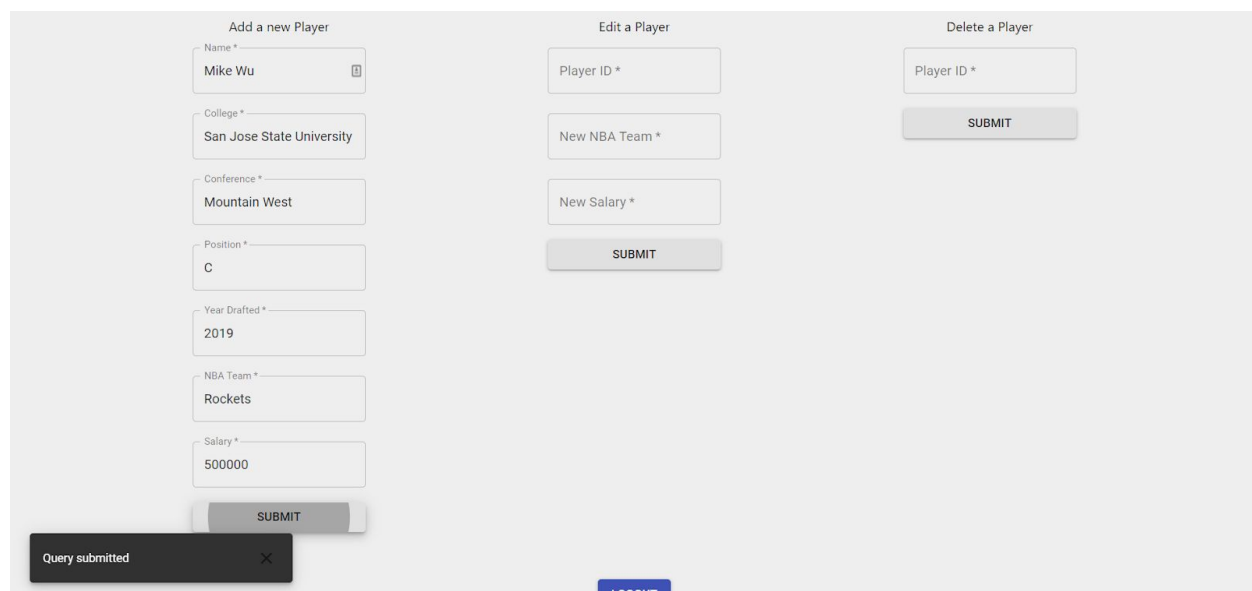


The screenshot shows the 'Administrator Panel' with a welcome message 'Welcome, SandroSallenbach'. Below this is a section titled 'Current NBA Players' which contains a table with the following data:

ID	Name	College	Position	Year Drafted	NBA Team	Salary
1	Kyle Anderson	University of California, Los Angeles	PF	2014	Grizzlies	\$ 458,725
2	Ryan Anderson	University of California, Berkeley	PF	2008	Rockets	\$ 456,258
3	Trevor Ariza	University of California, Los Angeles	SF	2004	Kings	\$ 326,584
4	Deandre Ayton	University of Arizona	C	2018	Suns	\$ 547,896
5	Lonzo Ball	University of California, Los Angeles	PG	2017	Pelicans	\$ 214,587
6	Aron Barnes	Washington State University	C	2009	Suns	\$ 698,745

Figure 30: Administrator Panel with Current NBA Players.

An administrator can add a new player to the database by entering all the mandatory data. The form checks for the completion of the text fields before submitting to the backend. Adding a new player will update the NBA Player and Played For relations.



The screenshot shows the 'Add a new Player' form with the following fields and values:

- Name *: Mike Wu
- College *: San Jose State University
- Conference *: Mountain West
- Position *: C
- Year Drafted *: 2019
- NBA Team *: Rockets
- Salary *: 500000

A 'SUBMIT' button is located at the bottom of the form. A 'Query submitted' message is displayed at the bottom left. To the right, there are sections for 'Edit a Player' and 'Delete a Player', each with a 'SUBMIT' button.

Figure 31: Adding a new Player.

The following screenshot shows the updated NBA Player relation after a new player has been added to the database.

	id	name	college	position	nbasince	team	salary
	17	Damyean Dotson	Houston	SG	2017	Knicks	568262
	18	Drew Eubanks	Oregon State University	C	2018	Spurs	216612
	19	Markelle Fultz	University of Washington	PG	2017	Magic	226463
	20	Taj Gibson	University of Southern California	F	2009	Knicks	264221
	21	Aaron Gordon	University of Arizona	F	2014	Magic	646482
	22	James Harden	Arizona State University	SG	2009	Rockets	264653
	23	Solomon Hill	University of Arizona	SF	2013	Grizzlies	264218
	24	Aaron Holiday	University of California, Los Angeles	PG	2018	Pacers	245786
	36	Mike Wu	San Jose State University	C	2019	Rockets	500000

Figure 32: Updated NBA Player Table 1.

The following screenshot shows the updated Played For relation after a new player has been added to the database.

	pid	schoolname	conference
	15	University of Southern California	Pacific 12
	16	University of Southern California	Pacific 12
	17	University of Colorado, Boulder	Pacific 12
	18	Oregon State University	Pacific 12
	19	University of Washington	Pacific 12
	20	University of Southern California	Pacific 12
	21	University of Arizona	Pacific 12
	22	Arizona State University	Pacific 12
	23	University of Arizona	Pacific 12
	24	University of California, Los Angeles	Pacific 12
▶	36	San Jose State University	Mountain West

Figure 33: Updated Played For Table.

Editing an NBA player will require the player's ID, which can be found in the drop down list. Updating a player's team or salary will update both the NBA Player and Played For relations.

Administrator Panel
Welcome, SandroSallenbach

Current NBA Players

Add a new Player

Name *

College *

Conference *

Position *

Query submitted

Edit a Player

Player ID *
36

New NBA Team *
Warriors

New Salary *
1500000

SUBMIT

Delete a Player

Player ID *

SUBMIT

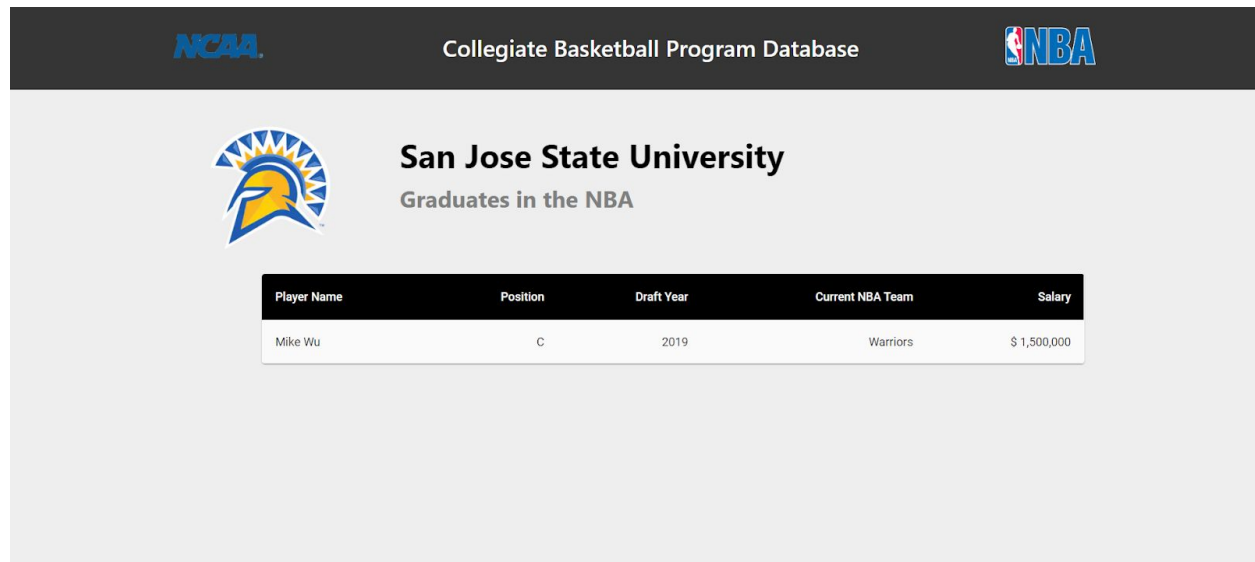
Figure 34: Editing an NBA Player.

The following screenshot shows the updated NBA Player relation. The Played For relation was updated as well but is omitted here as it would be redundant information.

	id	name	college	position	nbasince	team	salary
	17	Damyean Dotson	Houston	SG	2017	Knicks	568262
	18	Drew Eubanks	Oregon State University	C	2018	Spurs	216612
	19	Markelle Fultz	University of Washington	PG	2017	Magic	226463
	20	Taj Gibson	University of Southern California	F	2009	Knicks	264221
	21	Aaron Gordon	University of Arizona	F	2014	Magic	646482
	22	James Harden	Arizona State University	SG	2009	Rockets	264653
	23	Solomon Hill	University of Arizona	SF	2013	Grizzlies	264218
	24	Aaron Holiday	University of California, Los Angeles	PG	2018	Pacers	245786
	36	Mike Wu	San Jose State University	C	2019	Warriors	1500000

Figure 35: Updated NBA Player Table 2.

Of course, the database change is also displayed in the application itself, giving San Jose State University its first NBA player.

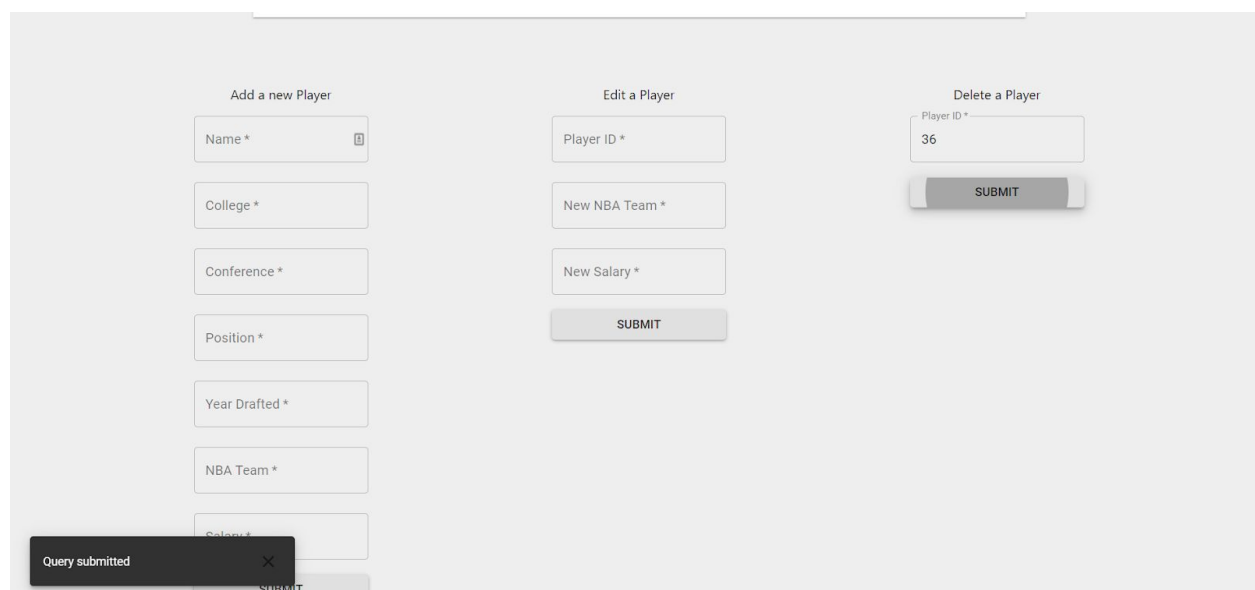


The screenshot shows the 'Collegiate Basketball Program Database' header with NCAA and NBA logos. Below is the San Jose State University logo and the title 'San Jose State University Graduates in the NBA'. A table lists the graduates.

Player Name	Position	Draft Year	Current NBA Team	Salary
Mike Wu	C	2019	Warriors	\$ 1,500,000

Figure 36: Updated University Graduates in the NBA.

Deleting an NBA player will require the player's ID, which can be found in the drop down list. Deletions will remove the player from the NBA Player and Played For relations.



The screenshot displays three forms: 'Add a new Player', 'Edit a Player', and 'Delete a Player'. The 'Delete a Player' form has 'Player ID *' set to 36 and a 'SUBMIT' button. A 'Query submitted' notification is visible at the bottom left.

Add a new Player

Name *

College *

Conference *

Position *

Year Drafted *

NBA Team *

Select

SUBMIT

Edit a Player

Player ID *

New NBA Team *

New Salary *

SUBMIT

Delete a Player

Player ID *

36

SUBMIT

Query submitted



Figure 37: Deleting a player.

This screenshot shows the updated NBA Player relation. The player was also deleted from the Played For table.

	id	name	college	position	nbasince	team	salary
	14	DeMar DeRozan	University of Southern California	GF	2009	Spurs	458711
	15	Dewayne Ded...	University of Southern California	PF	2013	Kings	232456
	16	Spencer Dinwi...	University of Colorado, Boulder	PG	2014	Nets	545315
	17	Damyean Dotson	Houston	SG	2017	Knicks	568262
	18	Drew Eubanks	Oregon State University	C	2018	Spurs	216612
	19	Markelle Fultz	University of Washington	PG	2017	Magic	226463
	20	Taj Gibson	University of Southern California	F	2009	Knicks	264221
	21	Aaron Gordon	University of Arizona	F	2014	Magic	646482
	22	James Harden	Arizona State University	SG	2009	Rockets	264653
	23	Solomon Hill	University of Arizona	SF	2013	Grizzlies	264218
	24	Aaron Holiday	University of California, Los Angeles	PG	2018	Pacers	245786

Figure 38: Updated NBA Player Table 3.

Again, the database change is also displayed in the application itself. By deleting the newly added player, San Jose State loses its sole NBA player.

Championships: 0
18/19 Record: 1-17
Players in the NBA: 0

[MORE INFO](#)

[About Us](#)
[Admin Panel](#)

Figure 39: Updated University Details Page.

Application Setup

System Details

Our test systems consisted of Windows 10 machines. Just about any off the shelf laptop will do. We used MySQL as our RDBMS, Visual Studio Code as our development environment, and NodeJS as our JavaScript runtime environment. Administrator rights are required on the device since you will have to install at least MySQL and NodeJS. It's strongly encouraged that Visual Studio Code is used when setting up and testing the application but any text editor will do since the application can be run from the command line.

Step-by-Step Instructions

1. Download and install MySQL from <https://www.mysql.com/downloads/>
2. Open Workbench and connect to your database.
3. Copy the script into workbench but don't run it until the changes in step 4 are made
 - a. <https://github.com/CS157A-32/CS157A-32/blob/master/SQL%20Script%20to%20create%20entire%20Schema.sql>
4. By default the schema is called "new_schema". We recommend you change this but it will work if you leave it as is.
 - a. We suggest you name it 'collegeball'
 - b. To change the schema name, replace 'new_schema' on lines 1 and 2 with your preferred name
5. After those changes run the script. Your schema should now look like this:

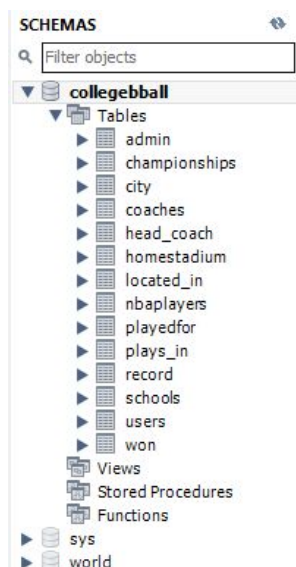


Figure 40: Tables.

6. Download and install NodeJS from <https://nodejs.org/en/>
7. Download and install Visual Studio Code from <https://code.visualstudio.com/>
8. Open Visual Studio Code and use the Add Folder button to create a folder that serves as your workspace

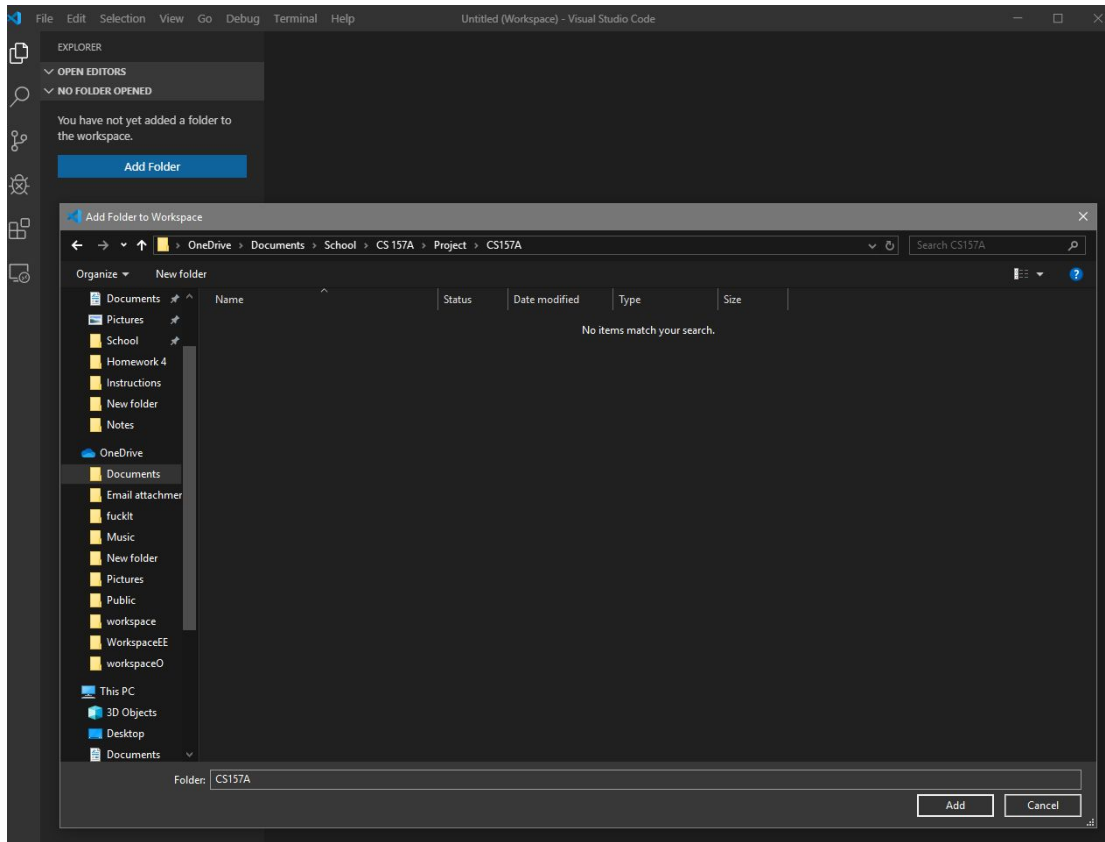


Figure 41: Making a workspace.

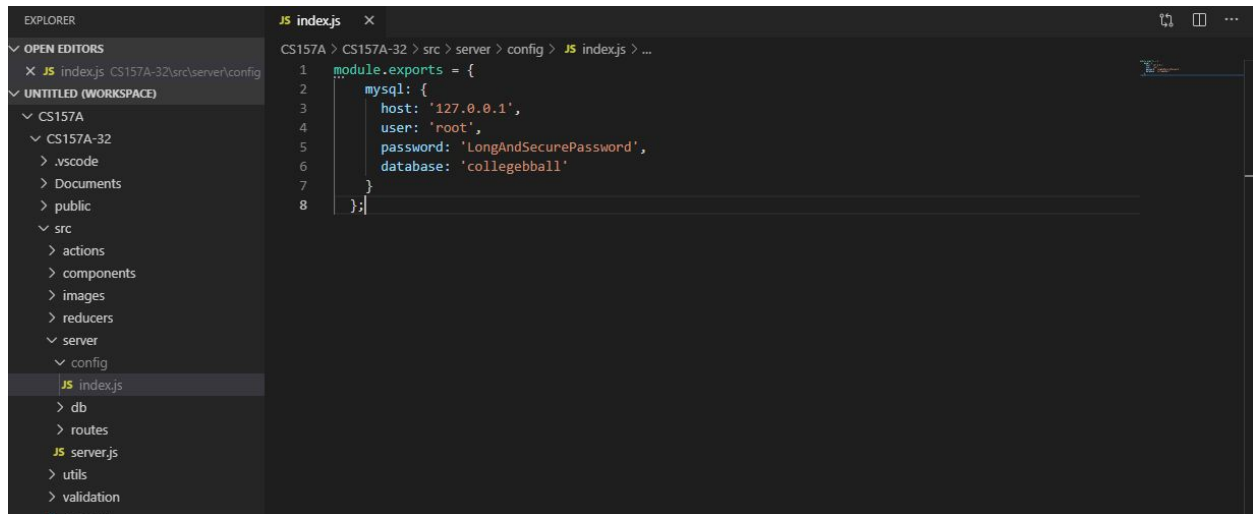
9. In the menu bar select Terminal and select New Terminal
10. use "git clone https://github.com/CS157A-32/CS157A-32.git" to clone the repo
11. On the left you should now see some folders. In src/server/ , make a new folder named config
12. Inside this new config folder make a file named index.js
13. Copy and paste the following in the index.js file

```
"module.exports = {  
  mysql: {  
    host: 'IP ADDRESS OF DATABASE',  
    user: 'DATABASE USERNAME',  
    password: 'PASSWORD',  
    database: 'DATABASE'  
  }  
};"
```


14. Replace the following:

- a. 'IP ADDRESS OF DATABASE' with your database's ipaddress. If your database is running on that same machine then 'localhost' will be fine
- b. 'DATABASE USERNAME' with your username
- c. 'PASSWORD' with your password
- d. 'DATABASE' with the name of the schema you chose.

Here's an example of what it should now look like with the new config folder, index.js file, and the copy and pasted code:

A screenshot of the Visual Studio Code editor interface. The Explorer sidebar on the left shows a project structure with folders like 'src', 'server', and 'config'. The 'config' folder is expanded, showing a file named 'index.js'. The main editor window displays the content of 'index.js', which contains a JavaScript configuration object for MySQL. The code is as follows:

```
1 module.exports = {  
2   mysql: {  
3     host: '127.0.0.1',  
4     user: 'root',  
5     password: 'LongAndSecurePassword',  
6     database: 'collegebball'  
7   }  
8 };
```

Figure 42: Making the index.js file.

15. In your terminal, execute the following commands:

- a. "cd .\CS157A-32"
- b. "npm install"
- c. "npm run dev"

Part 4: Project Conclusion

Lessons Learned

Chris Douglas

Planning and preparation go a long way. In a project like this having the documentation completed makes a significant difference during implementation. Producing documentation, such as the ER diagram, improved our ability to work together and put us all on the same page. If there was any kind of doubt about who was doing what or what we agreed on it became easy to pull the relative documentation or conversation and determine what to do from there.

Communication is also a valuable skill to have when working on a project like this. Being able to effectively communicate what you're doing to teammates saves lots of time down the road. Effective communication helps to prevent a team member from implementing something wrong or discover something that they may have missed (for example, forgetting to include the stakeholders in the final report). Our team had great communication skills and each team member was more than eager to help one another.

Lance Ngo

In a project with minimum complement of team members, 3 in our case, it can be efficient to divide the workload into independent modules that will ultimately integrate. We split our workload into front end and back end development, with the third team member spot programming where help was needed. While this method of division of labor worked out well for this project, it is conceivable that it will pose integration problems for more complex projects. Team members should be cognisant of implementation strategies beyond their assigned scope so that final integration of modules can be efficient.

In the same vein, project scheduling is very important especially when dependencies are involved. We were fortunate that our project did not have many critical dependencies among the different functionalities of our application. In a bigger project, however, the order and timeline of function implementations will have a determinant effect on efficiencies in time management. Consequently, larger projects require concrete project management strategies and methodologies to help ensure the entire team is working well together.

Sandro Sallenbach

Working with other engineers in a team always poses some challenges. Planning and communication become a critical factor for success. Planning a project, defining functional and non-functional requirements early on is crucial to have a guideline later on. It also helps when splitting up work and assigning responsibilities to the individual team members. Being open to change and adapting accordingly is essential for a positive outcome as well. Communication channels should be defined early and frequent exchanges are beneficial to all members and the

project overall. Planning and communication are not two newly learned lessons from this project, but applying them in this project have proven their tremendous benefit.

When volunteering to present in class during the first code review as well as for the final project has made three things very clear to me: presentation skills are enormously important, having a predefined script for a demonstration enables a smooth flow and a GUI can make or break an entire project. All three are crucial to elevate one's project above the competition, and I think we have executed all of them very well.

Future Improvements

While making the application several topics about future changes were discussed. These topics covered tools, naming schemes, input validation, security in general, additional data to add to the database, and monetization. Though we completed all of the requirements we set out to do we know there's still a lot more work we would have liked to do had time permit us to do so.

One thing we would like to implement if we had more time was additional sorting tools. For example, the ability to sort universities by tuition and other related factors. Another item to consider is that early in the project it was noted that schools aren't always referred to by their turner name and are often referred to by their abbreviations. For example San Jose State is our school's turner name and SJSU is how most people refer to it. Having the ability to search for schools via this app would improve ease of use for visitors. Both of these changes can be made at roughly the same time since we need to update the school table with out of state tuition as well. Implementing these wouldn't be difficult, however, it would take time.

Input validation is another topic that was touched on during one of our discussions. A concern that Sandro had was that what would happen if input was improperly formatted or was too big for the DB to handle. For example, if Mike Wu was given a salary of \$999,999,999 to play center position then we already know this would crash the application after we saw it during the presentation on December 3rd. So input validation and sanitization needs to be implemented.

As it is right now, the site does not currently use https. Though we do not currently handle any information that is highly sensitive it's still something we should use regardless. There is a benefit for us, however, Google's page rank algorithm now rewards sites that practice better security by increasing their rank. This would increase our visibility and help us reach more users.

The database has a very small set of data that we used for the sake of testing and demoing the product. This needs to be expanded if it were to become a viable product. The team got a lot of the information from public sources. Kaggle was a great tool for quickly collecting information about the teams, players, and schools. More information regarding other

conferences could be gathered from the source. Things like pictures of stadiums, on campus training facilities, and team photographs can be scraped from various online sources. To simplify the process, custom applications, such as web scrapers, can be created for gathering these pictures and then inserted into the database. Additional data about NBA players would need to be included as well. There was brief talk about enabling universities to update their own data if the web application became popular enough. We would benefit in the way that information is up to date and we don't have to spend the time and money doing it ourselves.

Of course, all of this cost money, there would be several expenses we need to account for and to make a profit off of this we would need to monetize the app in some way. The first way would be with AdSense. Implementing a simple banner on the site at the top or on the side would be the easiest way to start generating money. Ad blockers would create an issue here but that's something we don't have the power to avoid. To generate more revenue through AdSense we could work on implementing a mobile version of the site down the road. After the site gain's enough traffic we can look into sponsoring universities on our landing page for a small fee.