Project Design Proposal

Jeff Johnson

CST-451 Capstone Project Final Architecture & Design

Grand Canyon University

Instructor: Professor Mark Reha

Revision: 3

Date: 11/16/2024

Last updated: 4/18/2025

ABSTRACT

The project is a mobile application for high school baseball players who seek recruitment attention from college scouts. In today's state of high school recruitment (especially baseball), many players miss out on college recruitment due to the expenses of travel baseball teams, recruitment camps, and private events hosted by colleges. Scoutbase was created for athletes who may not be able to afford the expensive process of college recruitment, with the ability to connect with college coaches seeking athletes that fit their skillset.

There are three options when creating your account, users may sign up as an athlete, a coach, or a scout. All options will require a secure email and password to prevent spam on the app. Athletes have the option to list their height and weight, high school, the position(s) they play, and upload media for scouts to watch. Coaches have the ability to search for players with certain skill sets, archetypes, locations, whether they are left-handed or right-handed, among others. Additionally, coaches will be able to make a profile of their own where they can list their team needs, their school name, their position within the organization, and a short bio. The scout option is for those who want the ability to search for and connect with athletes, but will not have the option to create a public profile for players to view.

History and Signoff Sheet

Change Record

Date	Author	Revision Notes
1/10/25	Jeff	First update to reflect significant code progress
4/18/2025	Jeff	Minor adjustments to update to reflect final code drop.

Overall Instructor Feedback/Comments

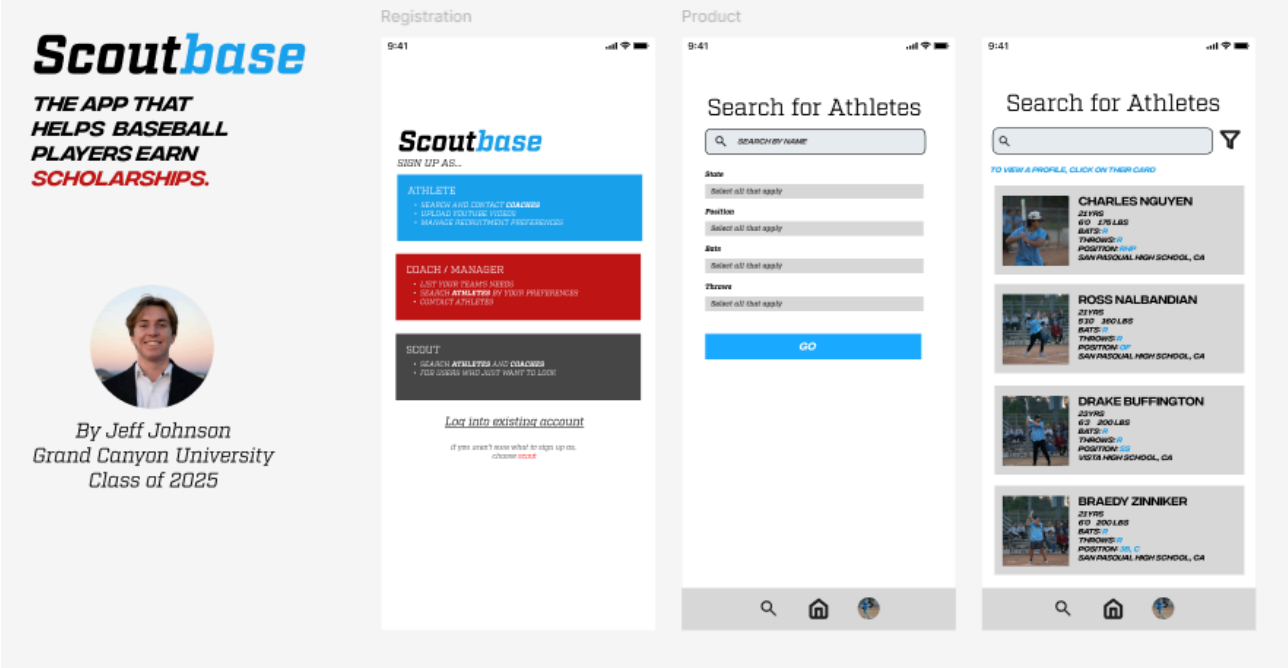
Overall Instructor Feedback/Comments

Integrated Instructor Feedback into Project Documentation

☐ Yes ☐ No

TABLE OF CONTENTS

Design Introduction



Scoutbase is a mobile application that specializes in connecting High School baseball players to University coaches and staff, to further their playing career and potentially earn athletic scholarships. The technical design approach is the following:

The Scoutbase mobile application employs an extensive and scalable technical design to deliver a feature-rich and secure user experience. The back-end, built with Django, provides RESTful APIs for core features like user authentication, profile management, and custom searches, all of which are secured using JSON Web Tokens (JWT). The React Native-based front-end leverages React libraries for transitions and styling, while Expo ensures cross-device compatibility with an efficient build process. This design emphasizes scalability, user-focused functionality, and access control to meet the needs of High School athletes, University coaches, and scouts.

Deliverable Acceptance Log				
ID	Deliverable Description	Comments	Evaluator (internal or external as applicable)	Status
1	Swagger API Documentation	The provided link will direct you to the SwaggerHub for my API endpoints.	ScoutbaseAPIDoc	Completed
2	Logical Solution Design	The Logical Solution Design diagram is located in the Scoutbase Project Design Files zipped folder in the assignment submission.	LogicalSolutionDiagram - Scoutbase - Jeff Johnson	Completed
3	Physical Solution Design	The Physical Solution Design diagram is located in the Scoutbase Project Design Files zipped folder in the assignment submission.	PhysicalSolutionDiagram - Scoutbase - Jeff Johnson	Completed
4	Database DDL Scripts	The Database DDL Script is located in the Scoutbase Project Design Files zipped folder in the assignment submission.	DDLScript - Scoutbase - Jeff Johnson	Completed
5	Data Dictionary	The Data Dictionary is located in the Scoutbase Project Design Files zipped folder in the assignment submission.	DataDictionary - Scoutbase - Jeff Johnson	Completed
6	Sitemap Diagram	The Sitemap Diagram is located in the Scoutbase Project Design Files zipped folder in the assignment submission.	Sitemaps - Scoutbase - Jeff Johnson	Completed
7	User Interface Diagrams	The User Interface Diagrams are located in the Scoutbase Project Design Files zipped folder in the assignment submission.	Wireframe and User Interface Diagrams - Scoutbase - Jeff Johnson	Completed
8	UML Diagrams	The UML Diagrams and Class design are located in the Scoutbase Project Design Files zipped folder in the assignment submission.	UML and Component Diagrams - Scoutbase - Jeff Johnson	Completed

Detailed High-Level Solution Design:

Proof of Concepts			
Problem	**Description of Problem**	**Outcome of P.O.C.**	**Decisions Made from P.O.C.**
1. React Native course - Codecademy	To familiarize the team with framework principles and architecture, and to make an informed decision on whether to move forward with this framework.	React Native will be the mobile development framework moving forward for the application	Chose React Native for framework, due to greater years of support in comparison to other frameworks, i.e. Flutter. Additionally, its compatibility with both Android and iOS devices provides an advantage to system-specific languages such as Swift (iOS) or Dart (Android).
2. Django Research and Official Django Documentation	To familiarize the team with capabilities in REST API support, and to make a decision to move forward with Django as the chosen framework to build the API.	Django will be the chosen framework in API development and backend processes for this application.	Implementing a REST API in Django requires significantly fewer lines of code opposed to a Java Spring Boot application or a C# .NET MVC application. Fewer lines of code will allow for simpler debugging, and an overall faster development process.
3. Anima - Figma to React	Converting Figma components diagrams to React Code can be a lengthy process when done manually.	Anima, a free Figma community library, will be used to convert Wireframe diagrams drawn in Figma to React code to reference in development.	Anima will be the software used to reference the HTML code to match the diagrams drawn in Figma to create the most accurate interpretation for the application itself.

4. VSCode or PyCharm for Django Development	To familiarize the team with advantages and disadvantages between Python compilers.	PyCharm will be used as the Code IDE for building the Django REST API.	While the team is more familiar with VSCode as a code compiler, PyCharm is significantly more intuitive for Python development, with tailored interfaces, ease in navigation, and credibility among the community.

Hardware and Software Technologies
1. Visual Studio Code IDE v1.94
2. React Native v0.75
3. React v18
4. Django v5.1.2
5. Swagger v3.1.0
6. PyCharm 2024.3
7. MySQL Workbench 8.0.5
8. Python 3
9. Lenovo Laptop Intel(R) Core(TM) i5-1035G1 CPU @ 1.00GHz 1.20 GHz 12 GB RAM
10. JawsDB (no version)
11. Heroku (no version)
12. Postman (no version)

Logical Solution Design:

My solution diagram represents the logical processes throughout the application, ranging from the Presentation Layer to the Data Access Layer.

Django Portion:
The process begins at the MySQL database, where key user data for the application is stored. Next, the data flows through the Data Access layer, responsible for establishing the connection between the business logic and the database. In the data access layer, the four models are defined: Athlete Model, Coach Model, Scout Model, and User Model, each containing key class data required to support the functionality of the application. The Repositories layer contains the SQL queries in the application to support features such as searching a user, creating a new user, logging in, among others. It serves as the bridge between the Data Access layer and the Business Services layer, handling database interactions and ensuring efficient data retrieval and operations. The Business Services layer contains the core logic of the application, where role-specific operations and application workflows are executed, such as validating user credentials, filtering search results, and processing profile updates. This layer ensures that business rules are consistently applied before passing data to the Controller Views layer. For the final portion of the Django code, the Controller API Views layer acts as the intermediary between the front-end and back-end, processing user requests, applying necessary logic, and returning structured data through secure and well-defined API endpoints.

React Native Portion:
Following the Controller API Views layer, the React Native block is responsible for the application's front-end functionality. This block is divided into two layers: the API Calls layer and the Presentation layer. The API Calls layer handles communication with the Django REST API, sending user requests such as login credentials, search parameters, or profile updates and receiving structured JSON responses. This layer ensures connectivity between the front-end and back-end, with error handling and response validation in place. The Presentation layer translates the data received into an intuitive user interface. It is responsible for rendering dynamic components, applying consistent styling, and navigation through features like user registration, and profile searches. Together, these layers ensure that users experience a responsive and visually appealing application tailored to their roles as athletes, coaches, or scouts.

The Logical Solution Diagram can be found in the ZIP File contained in the assignment submission, titled "LogicalSolutionDiagram - Scoutbase - Jeff Johnson"

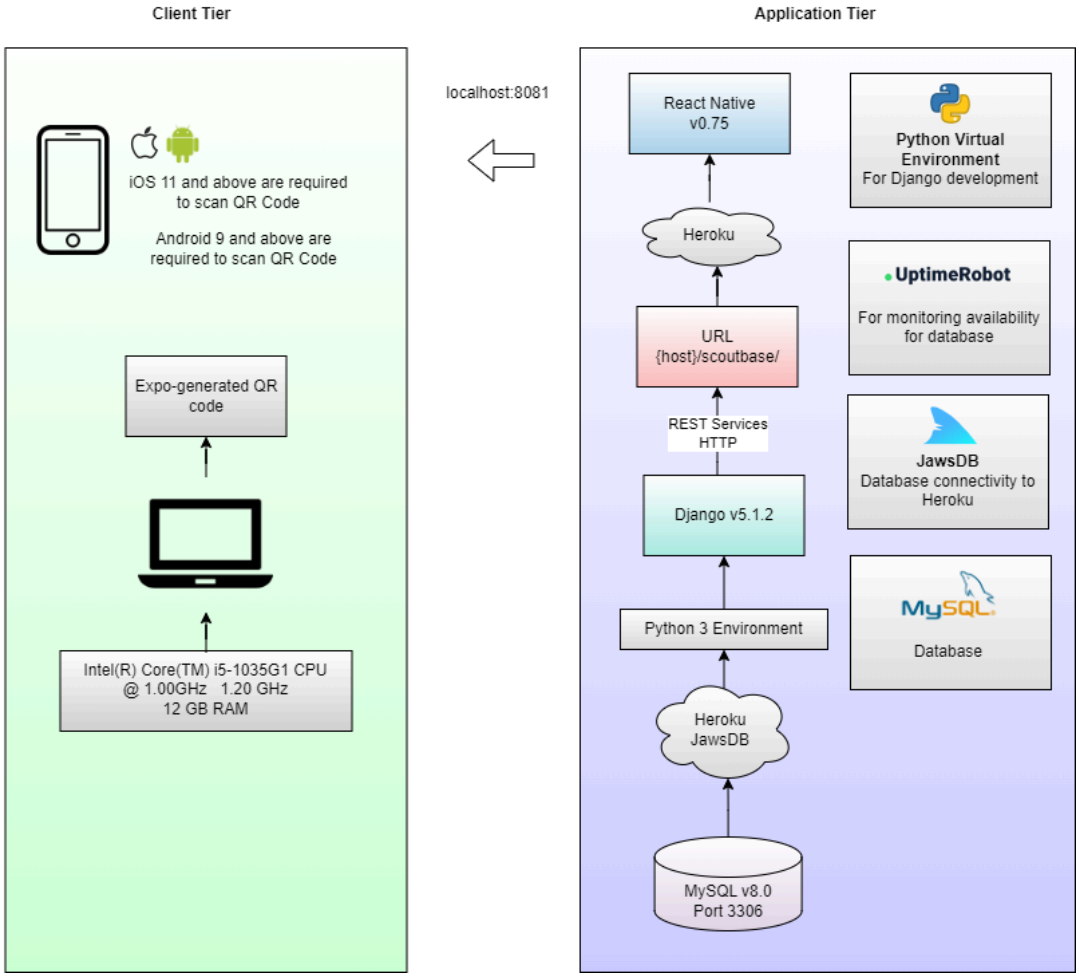
Physical Solution Design:

The physical solution diagram demonstrates two tiers to the application's architecture: Application and Client.
The Client Tier contains the REST Services and back-end support deployed to Heroku, the front-end utilizing React Native, and the MySQL Database.

- The REST Service block handles back-end processes, such as user login and registration, searching for profiles, editing profiles, etc..
- The React Native block serves as the application's functionality and front-end, with connectivity to API endpoints created in the previous tier.
- The Client Tier represents the application from the user's perspective, with access through the web and android mobile devices.
- The MySQL database is deployed in the cloud through Heroku and JawsDB, and is utilized in the REST Services tier.

Notable technologies utilized in the application's physical layer include MySQL, Heroku, Python, React Native, and Expo.

- MySQL is used as the application's database, hosted in Heroku connecting directly to the Django project.
- Django is a Python framework, specializing in web applications and API creations. Django serves as the application's back-end, hosting the creation of the various REST API endpoints for the application.
- React Native is the technology used for the application's front-end services, responsible for connecting to the API endpoints and building an intuitive front-end.



Detailed Technical Design

General Technical Approach:

The general technical approach for the Scoutbase mobile application is the following;

The technical approach for the Scoutbase mobile application involves an extensive back-end and front-end design to ensure a positive user experience and a secure foundation. The back-end is developed using Django, a Python framework, to provide RESTful API services for core functions such as user authentication, profile registration, and profile searches based on user-defined criteria. JSON Web Tokens (JWT) secure the endpoints, enforcing role-specific access control across the application. The front-end is implemented with the React Native framework, incorporating the React libraries for design, transitions, and styling. Additionally, Expo serves as the application compiler as it provides a QR code that will adapt the application's components to fit the device of the user.

Key Technical Design Decisions:

Key technical design decisions for this application include the use of Django for creating the RESTful API, chosen for its ability to deliver strong functionality with fewer lines of code compared to frameworks like Spring Boot or .NET. The development was streamlined using PyCharm as the IDE,

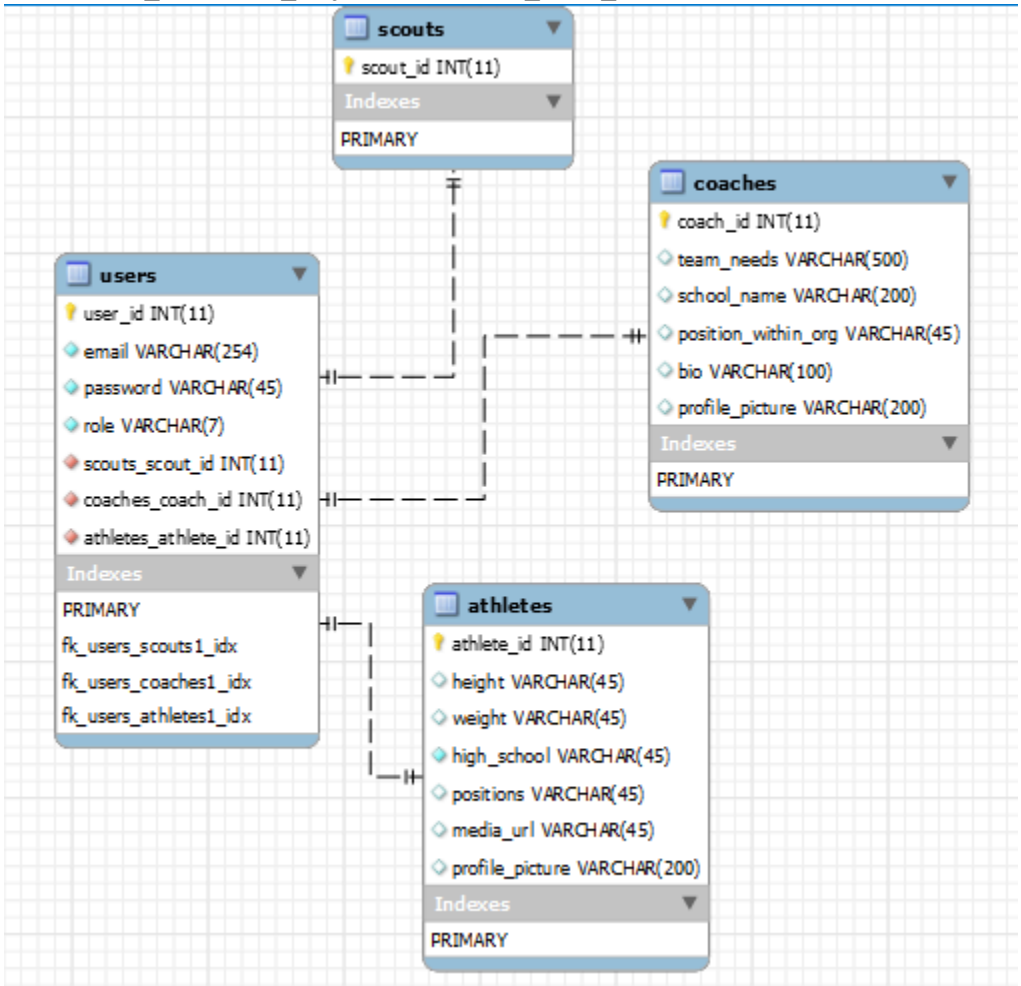
which provided efficient tools for managing Django's features. React Native was selected as the mobile framework for the front-end due to its long-standing credibility, native component support, and ability to translate components for both Android and iOS platforms. To secure user data, JSON Web Tokens (JWT) were implemented for API authentication, ensuring a reliable and scalable solution to protect sensitive information while maintaining role-based access control through the application.

Database ER Diagram:

The image below represents the database tables in the Scoutbase MySQL database. The users table will have a one-to-one relationship with all user types (Athletes, Coaches, Scouts), every account created entered into the user table and added to their desired role when selected during the registration process.

Relations:

- user_id to coach_id: joined on coaches_coach_id
- user_id to athlete_id: joined on athletes_athlete_id
- user_id to scout_id: joined on scouts_scout_id



Database DDL Scripts:
The database DDL Script contains the following processes:

- Creates a Scoutbase schema
- Creates an Athletes table, including columns and primary key assignment
- Creates a Coaches table, including columns and primary key assignment
- Creates a Scouts table, including columns and primary key assignment
- Creates a Users table, including columns and primary key assignment

The database DDL Script is included in the ZIP file in the assignment submission, and is titled "DDLScript - Scoutbase - Jeff Johnson"

Data Dictionary:
The data dictionary includes 19 fields, ranging from the Athletes, Coaches, Scouts, and Users table. Notable information:

- Every VARCHAR data type contains appropriate character limits for each field, respectively.
    - Biographies are limited to 1,000 characters, Emails are limited to 254 characters, etc.
- Every INT data type contains appropriate digit limits for each field.
    - Id columns (User_id, Scout_id, Athlete_id, Coach_id) are all limited to 11 digits, but can be adjusted as needed.
    - Id columns are all auto-incrementing from the previous value and cannot be NULL.
- Foreign Key relationships:
    - user_id to coach_id: joined on coaches_coach_id
    - user_id to athlete_id: joined on athletes_athlete_id
    - user_id to scout_id: joined on scouts_scout_id

The Data Dictionary is located in the ZIP file in the assignment submission, and is titled "DataDictionary - Scoutbase - Jeff Johnson"

Sitemap Diagram:

The Sitemap Images represent the navigation between pages, and the typical process flow between pages. The flow is as follows:

- User is prompted to register an account
- Once user has registered an account, they will be directed to the login page
- Once they are logged in, they will be directed to the home page, where they can access the search page or view profile page.
- From the view profile page, you can navigate back to the home page, or navigate to the search page via the navigation bar at the bottom of the screen. The three primary pages (View profile, search, and home) are all accessible via the navigation page once logged in.
- In the search page, if the user elects to filter their search, they will be prompted with filters to narrow athletes or coaches based on search criteria.
- The initial search will begin with querying all coaches or athletes (opposite role from user). If the user filters their search, they will be redirected to the search results page, with their filter applied.

Attached in the document are two diagrams, a high fidelity sitemap and a low-fidelity sitemap. The high-fidelity sitemap can provide greater context into the specific buttons and flows that connect one process to the other, but can be more difficult to interpret. The low-fidelity sitemap lacks the context of a high-fidelity sitemap, but is much simpler to interpret and understand.
The Sitemap Diagrams are located in the ZIP file with assignment submission, and are both located in the Word Document titled:

"Sitemaps - Scoutbase - Jeff Johnson"

User Interface Diagrams:
The document contains all four general processes for the application: registration, log-in, searching for a profile, and viewing/editing a profile.

The initial page that every new user will encounter is the role selection page, where they have the option of signing up as an athlete, coach or scout. The fields will be different depending on the role selected, with Coaches prompted to enter information such as team needs and their role within the organization, and Athletes prompted to enter information such as positions, high school, etc..
Additionally, the search page for each role will look different, as the criteria to search for will be different. Coaches and Scouts will search for athletes, and they may search for their name, their position, their high school, and Athletes and Scouts will search for Coaches which may include their school name, their team needs, etc.. The search form will be different depending on whether the user is searching for a Coach or Athlete, and these differences are depicted in the wireframes.

The User Interface and Wireframe Diagrams are located in the ZIP file in the assignment submission, and is in the Word Document titled
"Wireframe and User Interface Diagrams".

UML Diagrams:
The UML diagram below represents the relationship between different components in the application, beginning with the User Registration process. The relationships are explained in the diagram, and in text below:

Summary of Relationships: Django
- UserModel extends AthleteModel, ScoutModel, and CoachModel
- Each Model(Athlete, Scout, User, Coach) uses its corresponding Repository, respectively.
- Each Repository(Athlete, Scout, User, Coach) uses its corresponding Service.
- Each Service(Athlete, Scout, User, Coach) uses its corresponding APIView (or controller)

Summary of Relationships: React Native
The React Native project begins at the Index.js file, with each Screen given a separate file and later referenced in the Index.js file.

The screens include the following:
- RegisterScreen, fetching the following components: Coach, Athlete, and Scout Registration Form, and the Role Selection form.
- LoginScreen, fetching the following components: LoginForm
- AthleteEditProfileScreen, fetching the following components: AthleteEditProfileCard
- CoachEditProfileScreen, fetching the following components: CoachEditProfileCard
- HomeScreen, fetching the following components: HomeComponent
- CoachSearchScreen, fetching the following components: SearchCoachForm
- AthleteSearchScreen, fetching the following components: SearchAthleteForm

For context, React Native components are reusable building blocks that define the visual and functional elements of the app, including forms, buttons, and text fields, using JavaScript and native platform code. The Components of the app include the following:
- CoachRegistrationForm, AthleteRegistrationForm, ScoutRegistrationForm

- RoleSelection
- LoginForm
- SearchCoachForm, SearchAthleteForm
- AthleteCard, CoachCard
- AthleteEditProfileCard, CoachEditProfileCard

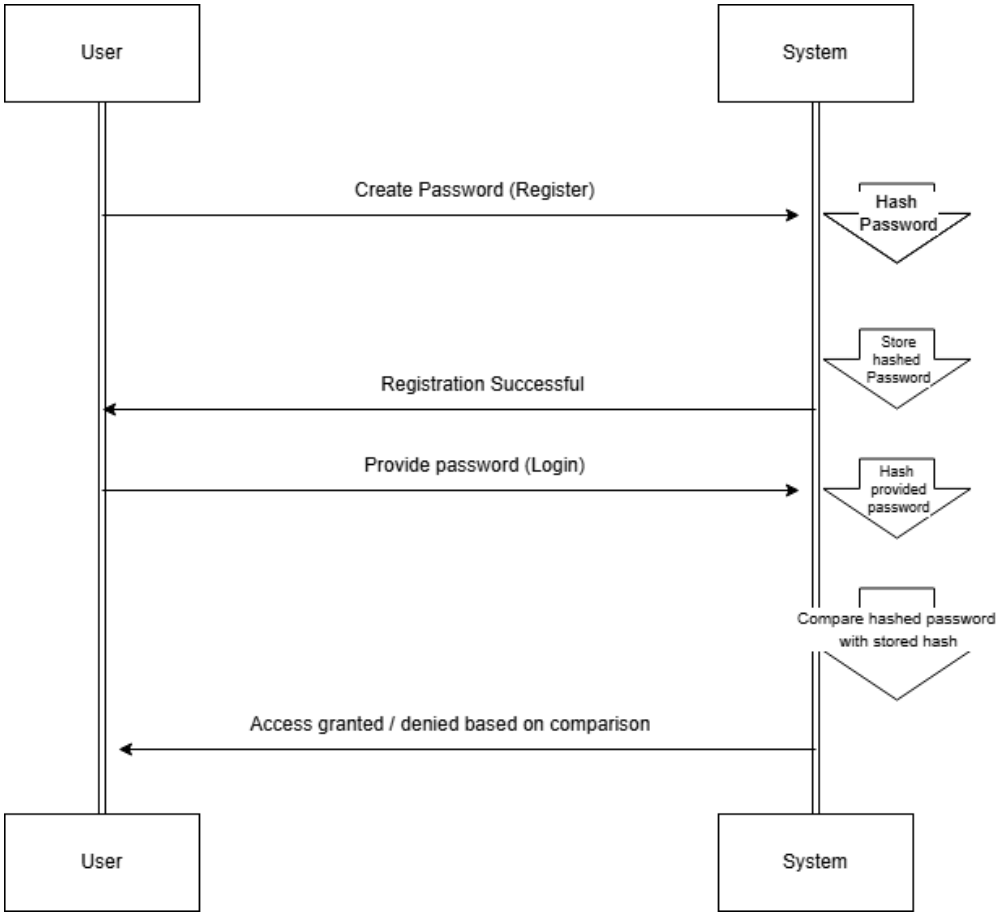
*Note that scouts will not create a “Card” in the application. They create an account, but will not have a visible profile as a coach or athlete would. Think of a logged-in user that is strictly limited to viewing.

The UML diagrams are located in the ZIP file in the assignment submission, and is titled “UML and Component Diagrams - Scoutbase - Jeff Johnson”

Service API Design:
As stated previously, the application hosts a back-end REST API developed in Django, a python framework. API documentation was created in Swagger, and can be found in the link below:
https://app.swaggerhub.com/apis/JJOHNSON769_1/Scoutbase/1

NFR: Security
Requirement: User passwords will not be stored in plain text.

The image below is a sequence diagram, explaining the process as follows:
The password is hashed upon creation, and is stored in the database pending a successful registration using the UserSerializer. During the login process, the system hashes the provided password, compares the entered hashed password with the stored hash password linked with the email address, and will grant or deny access based on the results from the comparison.



Operational Support Design:
The Django back-end portion of the application will utilize logging through the Python library "dictConfig". The React Native portion will not support a logging service.
The Django back-end (hosted in Heroku cloud) will be monitored with Uptime Robot.

Other Documentation:
Cloud Hosting Considerations:
The database and REST API for this application will be deployed to the cloud through Heroku and JawsDB.
The application itself, includes an out-of-scope project goal of deploying the application to the cloud.
Security: Listed above under NFR: Security

Appendix A – Technical Issue and Risk Log

Risk Management Log

Risk Management					
Event Risk	**Risk Probability (high, medium, low)**	**Risk Impact**	**Risk Mitigation**	**Contingency Plan**	**Result**
What is the risk?	What is the probability?	What is the impact if the risk occurs?	What can be done to minimize the risk?	What can be done to minimize the impact of the risk?	What is the result?
Cloud deployment credits	Low	If the Heroku free credits are not sufficient enough to deploy the database, switching to another cloud hosting platform (Azure) would be necessary.	Constant monitoring credit levels, pausing/deleting old projects, and contacting support if necessary.	Prepare to upload database to Azure	Azure will be the backup cloud provider if issues occur in Heroku.
Database Development	Low	Failure to design a proper database could lead to a magnitude of issues later on in the development process.	Allocating sufficient time to properly plan for, set-up, and maintain the database.	Switch to a different database provider, such as MongoDB.	MongoDB would be the alternate choice for the database.
API security	Low	Data points included in the applications REST API will include user data, such as passwords, emails, etc. Proper security fundamentals will need to be implemented.	Adhering to industry approved security measures.	JWT to authenticate users and secure endpoints. Constant testing to ensure API endpoints are displaying correct information, as well as monitoring logs from backend service..	API security will be implemented through JSON Web Tokens.

Appendix B – References

The document contains no in-text references to the listed sources, but knowledge was gathered by the author using these sources.

Create amazing apps that run everywhere. Expo Documentation. (n.d.). https://docs.expo.dev/

Django documentation: Django documentation. Django Project. (n.d.).

https://docs.djangoproject.com/en/5.1/

Introduction · REACT NATIVE. React Native RSS. (2024, October 23).

https://reactnative.dev/docs/getting-started

R/react on reddit: Since there seems to be no standard for drawing diagrams on react , we came up with our own for this term's project. what do you think? (n.d.-a). https://www.reddit.com/r/react/comments/utodds/since_there_seems_to_be_no_standard_for_drawing/

Yasasvi, T. (2022, September 9). *Traditional "N-layer" architecture in .NET Core web api*. LinkedIn. https://www.linkedin.com/pulse/traditional-n-layer-architecture-net-core-web-api-tiromika-yasasvi/

Course information for Learning Django and Learn React Native can be accessed through the links below:

https://www.codecademy.com/learn/paths/build-python-web-apps-with-django

https://www.codecademy.com/learn/learn-react-native

Appendix C – External Resources

GIT URL:	*https://github.com/jeffjohnson18/ScoutbaseRest-API* *https://github.com/jeffjohnson18/ScoutbaseFrontEnd*
Hosting URL:	N/A