Central Connecticut State University

S.C.A.D

Salary Cap and Dynasty Fantasy Football:

A Case Study on Building a Microservice-based Software System

CS-595 Capstone Final Report

Team Members:

Ramesh Kappera

Ryan Lauzon

Philip Murray

Advisor:

Dr. Neli Zlatareva

Table of Contents:

Acknowledgements

We are grateful to Dr. Neli Zlatareva for providing our team with guidance and support, and for being kind and patient enough to listen to more fantasy football musings than she probably ever imagined she would in a lifetime.

List of Tables and Figures

Abstract

The fantasy football industry produces \$15 billion in yearly revenue with 50 million participants worldwide[1]. The top two dominant fantasy football companies are Yahoo® and ESPN®, respectively, accounting for over 50% of market share.[2]

Today, neither Yahoo nor ESPN possesses a *salary cap* feature. A salary cap is a dollar amount indicating the maximum sum of all individual players' salaries for a specific team. It is a mechanism used in professional sports for promoting parity among teams in differing economic markets, preventing the creation of "super teams" that become great by simply outbidding all competitors for top athletes.

The heretofore uncaptured entertainment value that a salary cap feature can provide to the fantasy football world is the impetus for SCAD - Salary Cap and Dynasty Fantasy Football Software Application, the subject of this paper.

SCAD provides a new format that allows users to retain their teams from season to season. Such team retainment comes with salary cap restrictions. Each player has an associated salary, and users construct their teams in such a way that the aggregate salaries of the players is less than a predetermined value, the salary cap.

It is worth noting that SCAD's salary cap feature, although nominally concerned with "money", does not involve real money at all, nor are the players' salaries in SCAD reflective of their actual salaries. The money aspect of SCAD is akin to Monopoly® money.

SCAD adds realism and sophistication to the game, increases entertainment value, necessitates new strategies for success, and fulfills a significant market demand in the fantasy football sphere. Simply stated, SCAD makes fantasy football more fun.

[1] Goff, Brian. "The \$70 Billion Fantasy Football Market" *https://www.forbes.com/sites/briangoff/2013/08/20/the-70-billion-fantasy-football-market/#4a1cbc6755c4.* Forbes Inc. Aug 20, 2013 web.
[2] Blacker, Adam, "Fantasy Football Players Declining in US" *https://blog.apptopia.com/fantasy-football-players-declining-in-u.s* Aptopia, October 22, 2018 web.

1. Introduction

This report details the development approach, project management methodology, software architecture, and technical details of SCAD. It constitutes one of three deliverables of the final Capstone project, the other two being a presentation, and the application itself.

The first quarter of the report chronicles the development process and describes the methodology used by our team to build SCAD. It includes the rationale for choosing (and in some cases changing) our team's communication methods, versioning system, and agile techniques. It also discusses a fundamental design decision our team is presented with regarding whether SCAD is best served as a platform-independent web application or one that is inexorably linked to Yahoo.

The second quarter of the report is dedicated to SCAD's software architecture; its single page design, its back-end services, and its microservice-based architectural pattern. Here, monolithic and microservice-based architectures are compared, and the "componentization" of SCAD is discussed as the deciding factor for choosing one over the other.

The final half of the report details SCAD's implementation details. It begins with the tedious yet impactful process of configuring development environments. Next is a deep dive into SCAD's security components, including how and why SCAD uses OAuth2 and OpenIDConnect for authorization and authentication, how JSON Web Tokens are decoded, and how Basic Authorization, configured in the JEE server, supplies an additional layer of security.

Following the security segment is a description of the code base itself (languages, frameworks, directory structure, etc.) and a detailed look at the data models employed, including how mappings are constructed between relational tables, entity objects, data transfer objects, and serializable JSON strings. Then, the implementation details of the REST endpoints themselves are explained, including the nomenclature and structure of each endpoint and the sequence of events that transpire upon serving various types of web requests, including exception handling.

Finally, the user interface is detailed with a description of the purpose, functionality, and pertinent data of each of SCAD's front-end pages.

2. Development Methodology

2.1 Division of Labor

2.1.1 Overview

Building SCAD in five months is both a technical undertaking and a project management challenge. We immediately see how the proper division of labor between SCAD's three developers is key in determining the success of the project.

2.1.2 The SCAD Team

Ramesh Kappera is veteran software developer with over a decade of professional experience working on enterprise systems. He assumes the role of "technical lead", and the SCAD team relies heavily on his expertise to write code and make determinations about whether something is technically feasible.

Philip Murray is a software developer with a penchant for technical writing. He plays a supporting role to Ramesh in writing the back-end code for SCAD, and he plays a leading role in writing the technical documentation for the system.

Ryan Lauzon is a software developer who is also the domain expert in fantasy football. He takes the lead in writing the front-end/UI code, and plays the role of product owner. The SCAD team relies exclusively on Ryan to determine whether SCAD's functional requirements have been satisfied.

2.2 Communication Methods

2.2.1 Technologies and Timeline

Slack® serves as the primary communication platform for SCAD's team members. Skype® and BlueJeans® are used for video conferencing. In-person meetings between all three developers occur once per week at CCSU or online.

The preliminary project timeline is shown below.

Week	Activity
1	Set up a development environment (IntelliJ, Eclipse, Git Repository, etc.)
2	Set up project management and db maintenance technologies (Liquibase, Trello, etc.)
3	Refine object models using domain driven design (UML, ER charts etc.)
4 - 7	Application development

8	Complete a minimum viable product
9 - 12	Application development
13	Complete a finalized product ready for testing
14 - 17	Finalise report and presentation

Table 2.2.1
Basic project timeline for the development of SCAD

2.2.2 Initial Discussions

Initial discussions are dedicated to fleshing out the data models and conveying an understanding of the appropriate verbiage used in fantasy football. What exactly is a league? A team? Is there a difference between a user and a player? Etcetera.

These discussions last nearly one month, and the SCAD team relies heavily on Ramesh's experience with building custom systems to guide the timeline. Ramesh emphasizes the dual purpose served by these meetings. First, they help Philip and Ramesh gain an understanding of the fantasy football domain. Second, they compel Ryan, the domain expert, to remain internally consistent in his explanations.

From these meetings emerges what is essentially a miniature DSL (domain specific language) where terms like "roster", "team", "league", "player", "game", and "season" now connote SCAD-specific details immediately understood by all three developers.

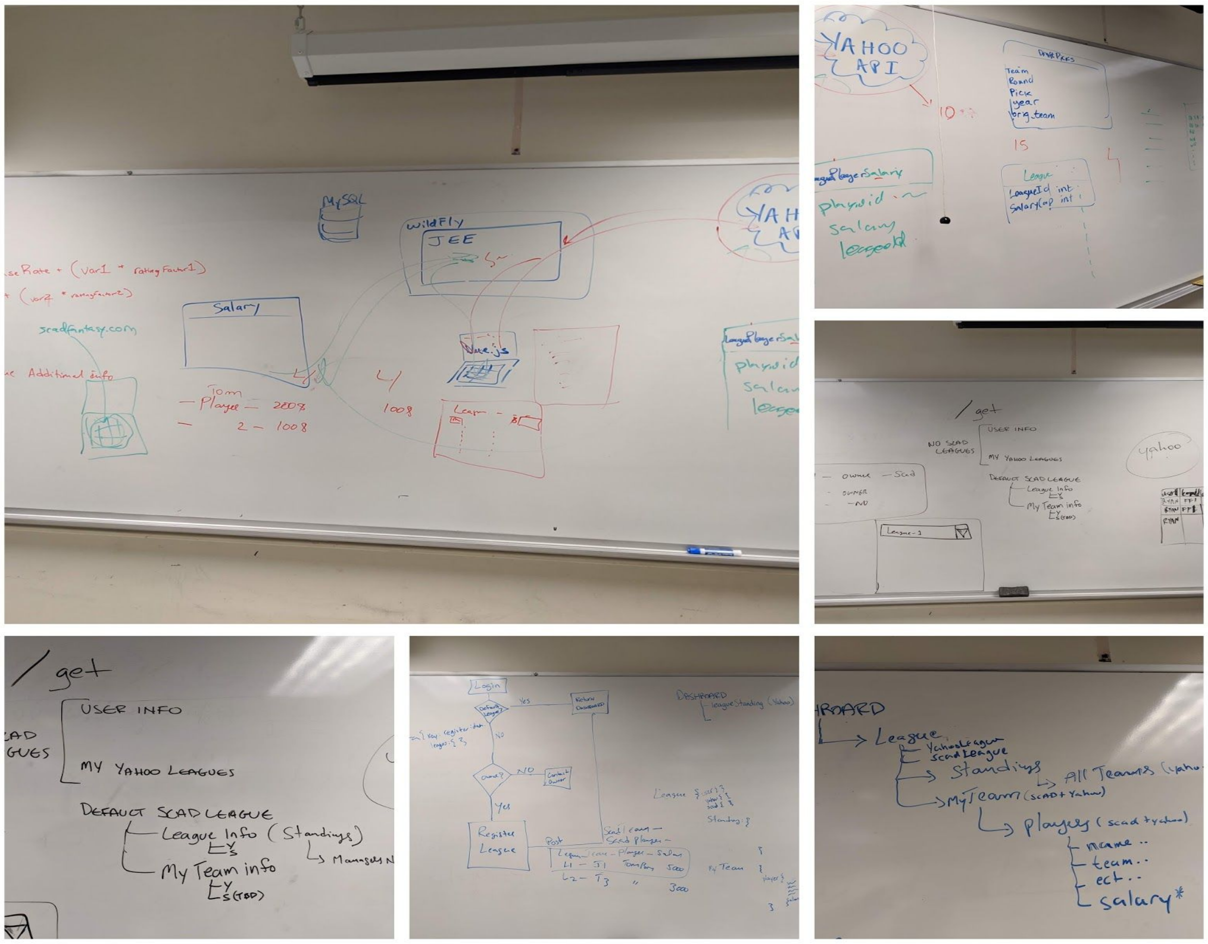


Figure 2.2.2
Collage of whiteboard photos of initial meetings of SCAD developers.

2.3 Technology Stack

2.3.1 Vue.js for the Front-End/UI

Vue.js is a modern JavaScript framework (compiled with Webpack) developed as a light-weight alternative to Angular.js and React.js. Ryan chooses Vue.js to build SCAD's front-end/UI, because of his familiarity with the technology and its conduciveness to building single-page web applications (SPA).

2.3.2 Enterprise Java and Node.js for the Back-End Services

Enterprise Java (JEE) in the Thorntail runtime environment is chosen as the primary back-end technology. JEE is familiar to both Ramesh and Philip, and although the capabilities of the Java Spring framework is considered more appropriate to build SCAD (because of easier dependency management), we opt for JEE, because Philip is unfamiliar with Spring and the estimated time required for Philip to become proficient with Spring does not justify the efficiency gains of using Spring over JEE.

JavaScript in a Node.js runtime environment is chosen as a secondary back-end technology. Node.js is familiar to Ryan and integrates readily with MongoDB. It also carries the advantage of having the front-end/UI code and back-end code written in the same language (JavaScript).

2.4 Version Control Systems

2.4.1 Git and Github

To keep track of SCAD's source code, we use Git VCS and store the repositories on GitHub®, primarily because of the teams' familiarity with the technologies as well as Git's native integration to each team member's IDE.

2.4.2 Multiple Repositories

Three separate GitHub repositories are used, one for the user interface, and the other two for the back-end microservices. Their locations are as follows:

1. User Interface: https://github.com/ccsu-grad-capstone/scad-ui.git
2. JEE Back-End: https://github.com/ccsu-grad-capstone/scadservices.git
3. Node.js Back-End: https://github.com/ccsu-grad-capstone/scad-node.git

We choose to separate source code repositories to maintain the proper microservice architecture, discussed in detail in chapter 3.

2.4.3 Using the Feature Branch Pattern

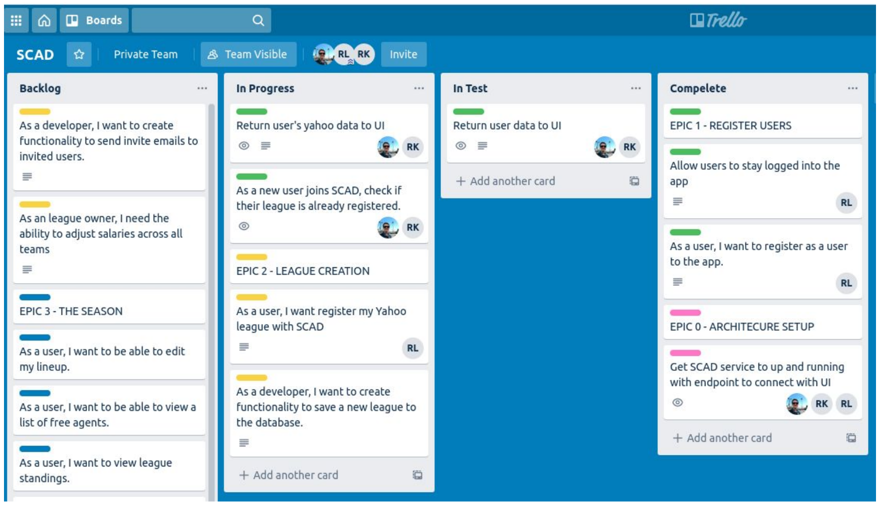
SCAD's repositories maintain a "master" branch into which a "development" branch is merged approximately once per month by Ramesh. The "development" branch is considered the branch of record, from which we create feature branches and to which we merge completed features.

Our motivation for using the feature branch pattern is familiarity and simplicity. Working off of the "development" branch directly could lead to complications from merge conflicts, but anything more sophisticated than the feature branch pattern is unnecessary for a project of SCAD's size and scope.

2.5 Agile Development

2.5.1 User Stories

To develop SCAD using agile techniques, we use a Trello® board with four columns to track a backlog of user stories and their implementations.



Backlog

- As a developer, I want to create functionality to send invite emails to invited users.
- As an league owner, I need the ability to adjust salaries across all teams
- EPIC 3 - THE SEASON
- As a user, I want to be able to edit my lineup.
- As a user, I want to be able to view a list of free agents.
- As a user, I want to view league standings.

In Progress

- Return user's yahoo data to UI (RK)
- As a new user joins SCAD, check if their league is already registered. (RK)
- EPIC 2 - LEAGUE CREATION
- As a user, I want register my Yahoo league with SCAD (RL)
- As a developer, I want to create functionality to save a new league to the database.
- + Add another card

In Test

- Return user data to UI (RK)
- + Add another card

Compelete

- EPIC 1 - REGISTER USERS
- Allow users to stay logged into the app (RL)
- As a user, I want to register as a user to the app. (RL)
- EPIC 0 - ARCHITECURE SETUP
- Get SCAD service to up and running with endpoint to connect with UI (RK) (RL)
- + Add another card

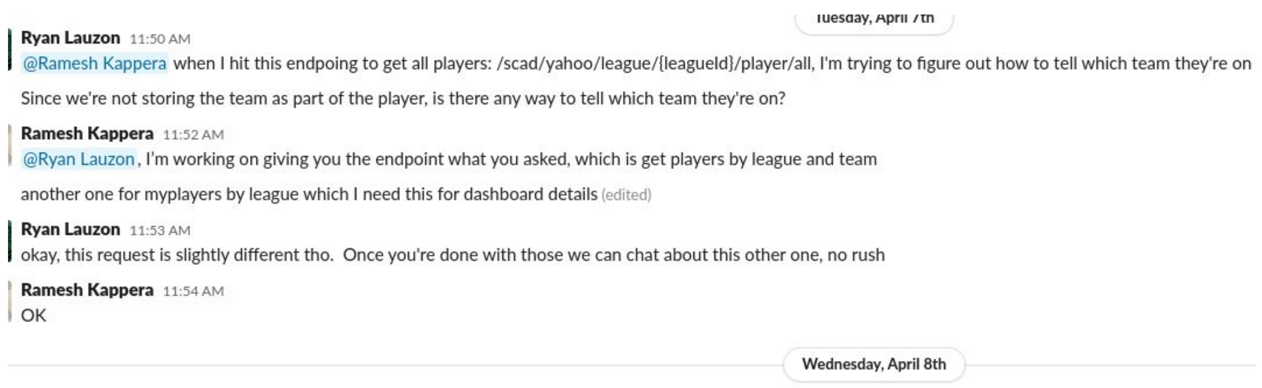
Figure 2.5.1
A screenshot of the Trello® board used to facilitate agile techniques used in building SCAD.

2.5.2 The Developer as the Product Owner

Since Ryan is both SCAD's product owner and one of SCAD's developers, and since the team consists of only three people, we find many of the conventional benefits of using agile approaches to be irrelevant. For example, as we develop the REST endpoints to return JSON data in a desired format for Ryan to consume on the front-end, it is almost comically inefficient to keep adjusting the Trello® board, rather than simply having Philip and Ramesh converse with Ryan on an as-needed basis.

2.5.3 Being Agile with Agile

As the project progresses, we keep agile techniques that are working, and we abandon those that are not. We find that the most pertinent aspect of the agile manifesto for SCAD is the importance of keeping in constant contact with the product owner.



Tuesday, April 7th

Ryan Lauzon 11:50 AM
@Ramesh Kappera when I hit this endpoing to get all players: /scad/yahoo/league/{leagueId}/player/all, I'm trying to figure out how to tell which team they're on
Since we're not storing the team as part of the player, is there any way to tell which team they're on?

Ramesh Kappera 11:52 AM
@Ryan Lauzon, I'm working on giving you the endpoint what you asked, which is get players by league and team
another one for myplayers by league which I need this for dashboard details (edited)

Ryan Lauzon 11:53 AM
okay, this request is slightly different tho. Once you're done with those we can chat about this other one, no rush

Ramesh Kappera 11:54 AM
OK

Wednesday, April 8th

Figure 2.5.3
A short conversation between a SCAD developer and product owner, illustrating the immediate feedback loop of communication made possible by SCAD's small team size. Such conversations obviate the need to use more sophisticated agile tools.

2.6 Building on Top of Yahoo

2.6.1 Pursuing Platform Independence

A lofty initial goal for SCAD is platform independence. That is, SCAD should not be dependent on any *particular* fantasy football vendor (e.g. Yahoo or ESPN). Although, *in general,* SCAD must depend on a pre-existing fantasy football vendor (to gather player data, statistics, etc.), the choice of vendor should be a configurable component of SCAD.

As we begin designing SCAD's database schema, the magnitude of the challenge of pursuing platform independence becomes clear.

To achieve platform independence, SCAD can either fully integrate with other vendors, or, more realistically, can simply gather the fantasy football *data* from another platform (e.g. Yahoo or ESPN®), and incorporate the data into SCAD's own fantasy football workflow.

As we pursue the latter course, we find ourselves spending virtually all of our development time building routine fantasy football features *that already exist in every other platform* (creating leagues, determining game days, etc.). It dawns on us that we are doing little more than "reinventing Yahoo's wheel".

2.6.2 Changing Course

At a weekly meeting, Ryan suggests a major change, that we abandon the pursuit of platform independence and build SCAD on top of Yahoo's APIs. He argues that 1) Yahoo has an existing API infrastructure created specifically for developers such as ourselves, and building SCAD on top of these APIs will prevent any "wheel reinvention", 2) Creating a platform-independent SCAD in our timeline is not doable.

Ramesh and Philip agree.

2.6.3 Yahoo as a Hard Dependency of SCAD

SCAD can now be considered an enhanced version of Yahoo fantasy football, and SCAD's developers now have a more tangible paradigm upon which to make decisions. There are numerous benefits to building SCAD in this manner, three of which are described below.

1. **Yahoo stores all of the data:** SCAD no longer has to maintain a database with dozens of tables of user data. Now, in lieu of creating proper SQL queries (or Hibernate queries) and concerning ourselves with joins, views, and the complications of maintaining relational integrity, we simply "query" Yahoo's REST endpoints to obtain JSON representations of the data, and let Yahoo handle all of the storage intricacies.

2. **Domain modeling has already been done:** Yahoo has a tried-and-tested domain model of fantasy football resources (e.g. league, game, player, team, transaction, etc.). Without using these, the nuances of creating a domain model from scratch are surprisingly difficult. For example, the relationship between a "team" and its "players" is not trivial. It requires an intermediary data structure called a "roster" which has a 5-field unique key {league, game, team, season, week}.

3. **Yahoo has solved and documented the most common problems:** APIs for basic web-application plumbing such as authenticating users and authorizing their access to data are readily available. We tap into these to secure SCAD using OAuth2 and OpenIDConnect, two protocols discussed in detail in chapter 4.

Upon the completion of each new feature, our decision to abandon platform independence and build on top of Yahoo is further validated. In fact, we begin to recognize our naivete in thinking we could do it any other way.

We strive to release version 1.0.0 of SCAD with a complete disjointness of SCAD-specific and Yahoo-specific features, achieving a level of modularity in the code that can easily be built upon in years to come.

3. Software Architecture

3.1 Overview

3.1.1 SCAD as a Single Page Web Application

SCAD is a single page web application (SPA) that communicates with back-end REST services. It employs a microservice architecture, a variant of service-oriented architecture (SOA). Microservices have gained popularity in recent years, and their ascendancy can largely be attributed to shortcomings in the monolithic applications which preceded them.

3.2 Monolithic vs. Microservice-based Architectures

3.2.1 At a High Level

At a very high level, a monolithic web application is one whose capabilities are contained in a single logical unit. In the Enterprise Java landscape, this single unit is typically a .war (web archive) file containing every capability of the application. The .war file is tagged, distributed, and deployed as a single entity and is thought of as the legal "version of reference" for a particular build of the application. For example, were we to use a monolithic architecture for SCAD, our first deliverable, SCAD v 1.0.0 would be a single .war file containing all of SCAD's functionality. Any change, however small, would necessitate a new build, a new version (e.g. SCAD v 1.0.1), and a new deployment of the entire application.

3.2.2 Componentization of SCAD

While monolithic applications contain components, their componentization is largely invisible to the user and includes such things as third-party library .jar files and database drivers.

On the contrary, a microservice-based web application is one whose capabilities are componentized at the user level. By "user level", we are referring to components that can be easily conceptualized from the end-user's perspective. For SCAD, our user-level componentization breaks down the application into three microservices, two that *serve* fantasy football data, and one that *consumes and processes* fantasy football data. Clearly, this componentizing is occurring at the user level (after all, SCAD's users are fantasy football players).

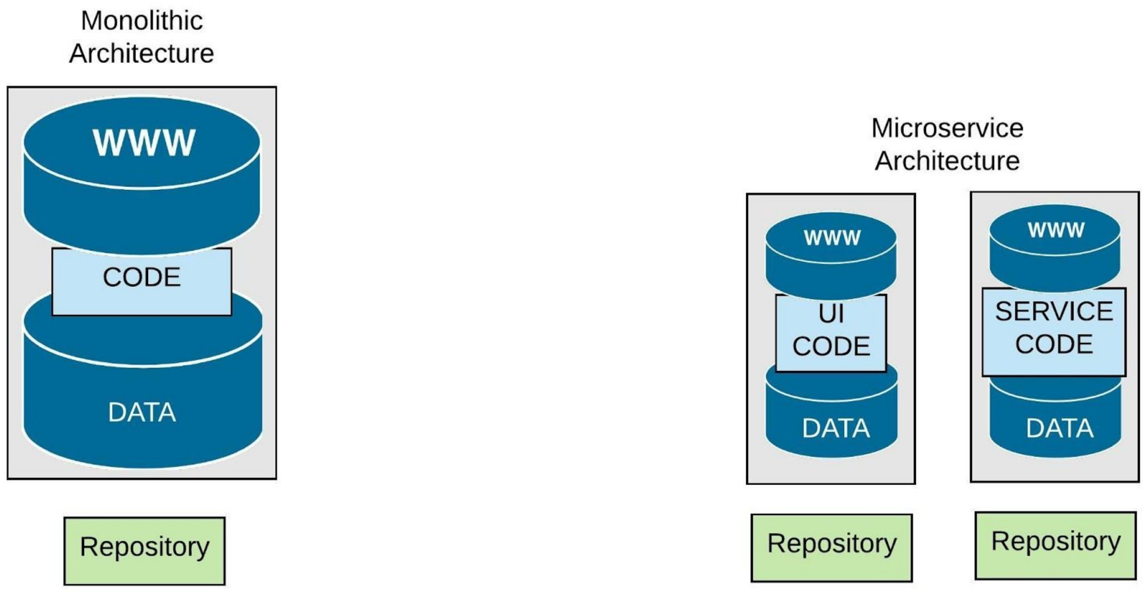


Figure 3.2.2
Schematic diagram of a monolithic vs. microservice architecture illustrating the domains (grey boxes), web servers ("www"), data sources ("DATA"), application software (light blue boxes), and source code repositories (green boxes).

3.2.3 Benefits and Drawbacks

The benefits and drawbacks of using a microservice-based architecture over a monolithic architecture are numerous, and whether one is more desirable than the other depends in large part on three factors.

3.2.3.1 Organization of teams around business functions rather than infrastructure: In companies where teams are organized around technologies and technology layers (e.g. UI/UX, data, middleware), a monolithic architecture is warranted. If a company is more effective when organizing its teams around particular business functions (customer service, product delivery, shipping etc.), microservices are likely more appropriate.

In building SCAD, we realize this benefit to a certain extent by placing Ryan, our main domain expert, in charge of the front-end microservice that contains the code requiring the most amount of domain-specific knowledge (i.e. expertise in fantasy football).

Although building the back-end microservices requires a modicum of fantasy football know-how, the "business function" around which they are built is predominantly data-retrieval and data-transfer of fantasy football data as opposed to the intricacies of the fantasy football game itself. As such, placing Ramesh and Phil, who are not domain experts, in charge of the back-end microservices is appropriate.

It is worth noting that while we certainly see how microservices *can be* conducive to organizing teams around business functions rather than technologies, our team of three developers is too small to attribute any of SCAD's successes or failures to this aspect.

3.2.3.2 Need for centralized data stores: Each microservice in a microservice-based application possesses its own independent datastore, whereas in a monolithic application, there is one datastore with which all internal components communicate. An application is conducive to using a microservice-based architecture if it can be logically broken down into business-relevant components that do not require access to common data (e.g. a specific table in a relational database). If a system's components work best by accessing a common data source, then a monolithic architecture is preferable.

In SCAD, there are two traditional and one non-traditional data stores. The traditional data stores are a relational database (MySQL) located in the JEE back-end microservice and a document database (MongoDB) in the Node.js back-end microservice. They contain SCAD-specific league configurations as well as salary-related player information.

The non-traditional data source is Yahoo itself. Yahoo is a hard dependency of SCAD, and because all SCAD users are Yahoo users, careful consideration has to be taken when designing SCAD's database schema to ensure that any changes a SCAD user makes via Yahoo do not have to be duplicated in SCAD. A guiding motto of our team is "Don't reinvent Yahoo's wheel."

Two points regarding SCAD's use of microservices should be made. First, because of how tightly coupled our system is to Yahoo, our design efforts are less about splitting SCAD's functionality into proper microservices and more about *retrofitting* our data models around Yahoo's data store to *become* microservices. Second, our front-end microservice requires no datastore at all, making it a prime candidate for a microservice.

3.2.3.3 Relevance of rapid development and deployment between components: Deployments of different microservices in a microservice-based application are completely independent of one another. The microservices have different source code repositories, different continuous integration pipelines, and the frequency with which one microservice is upgraded has no bearing on that of the others. This characteristic is beneficial if certain components of an application are expected to require more upgrades and maintenance than others.

For example, consider an application where one component is sufficiently stable that it requires upgrades only once per year, while another requires daily maintenance and enhancements. Under a monolithic architecture, the stable component will be redeployed every day alongside the other, slowing down build times. This can be particularly inefficient if new builds undergo extensive testing routines prior to deployment. If the stable component was self-contained in its own microservice, no such waste would exist, and delivery speed would increase.

In SCAD, both the UI/front-end microservice and the back-end microservices are built from scratch, so the time-to-deploy aspect is not relevant. However, we do get a glimpse of this efficiency gain when we consider how much simpler it is to handle the source control versioning (Git) of the back-end microservices by themselves compared to what it would be if the other repositories' code was also present. The commit histories of SCAD's back-end services are smaller and easier to process for the developers. This is because, by the very nature of microservices, all changes committed to source control by a developer are relevant to the component being worked on.

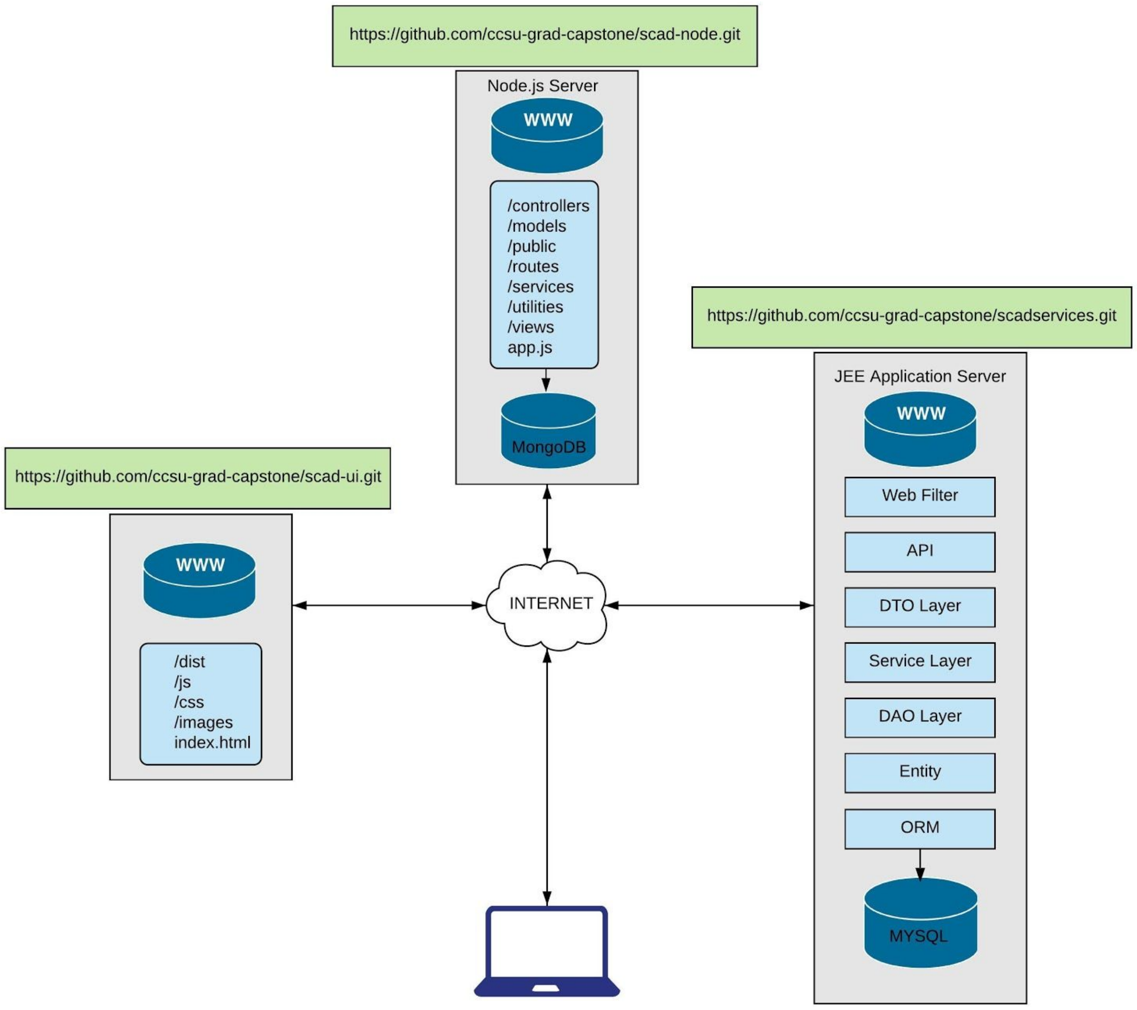


Figure 3.2.3
Schematic diagram of the three microservices that comprise SCAD. Notice that, other than the microservices' *business relationship* to SCAD, they are completely independent pieces of software.

4. Implementation

4.1 Development Environments

4.1.1 Efficiencies of Using Microservices

A major benefit of using a microservice-based architecture to build a system is the ease with which the system's development teams can be organized. Contrary to a monolithic architecture, team members working on a microservice-based system require virtually no information about system components other than the one they are working on. In fact, it is common for different microservices of the same system to be written in different languages, hosted in different environments, or even developed by different organizations altogether.

Our team of three developers realize some of these efficiencies, although numerous unsuspected inefficiencies also present themselves. Table 4.1 details the technologies we use to develop SCAD's three microservices; the front-end/UI, the JEE back-end, and the Node.js back-end.

	Front-end/UI	**Back-end/REST Services**
Developers	Ryan	Ryan, Ramesh, Phil
Language/Framework	JavaScript/Vue.js	Java/JEE, JavaScript
Development App server	Webpack/Node.js	Thorntail/WildFly Swarm, Node.js
Build tool/package manager	npm	maven, npm
Source code repository	https://github.com/ccsu-grad-capstone/scad-ui.git	https://github.com/ccsu-grad-capstone/scadservices.git, https://github.com/ccsu-grad-capstone/scad-node.git
IDE	Visual Studio (Ryan)	IntelliJ (Phil), Eclipse (Ramesh), Visual Studio (Ryan)

Table 4.1.1
Technology stacks in development for SCAD's front-end/UI and back-end/REST services.

Decoupling the front-end/UI and back-end/REST services allows SCAD's developers to select their preferred choice of language and framework, version control system, and other development-related tooling technologies.

4.1.2 Inefficiencies of Using Microservices

We find it easy to imagine how using a microservice-based architecture facilitates the development process *for large projects.* However, we are not convinced that going the microservice route results in a net gain in any development efficiencies for SCAD. This is because we find practical complications that are not obvious prima facie.

As an example, for Ryan, our front-end developer, mocking the JSON results of REST calls made to SCAD services greatly simplifies his development workflow. Ramesh and Phil, our back-end developers, use a tool (Swagger API) that, with a simple annotation, can provide Ryan with precisely these mocks. However, since Ryan's component is completely decoupled from Ramesh's and Phil's, getting Swagger up and running requires Ryan to install and run virtually all of the development software that runs on Ramesh and Phil's machine, nullifying any development efficiencies that supposedly come from using microservices.

While solutions do exist for resolving these development inefficiencies (e.g. deploy Swagger on a shared environment and use Github hooks to redeploy it when new code is pushed), their implementation does not appear to be worth the effort.

Ultimately, we find that the most efficient approach is to simply set up all environments (UI and REST services) on all developer machines.

4.1.3 Logging

When beginning to physically write the code for SCAD, we make the up-front investment in time to surround the functional code with logging code that will output to the server console. To do this we use the org.slf4j package as shown below.

```
package edu.ccsu.cs595.capstone.scadservices.util;

import org.slf4j.Logger;
import org slf4j.LoggerFactory;
public class HeaderHelper {

    private static final Logger LOG = LoggerFactory.getLogger(DashboardService.class);

    pubic Response isHeaderAccessValid(SCADSecurityManager sm) throws RuntimeException {
        if ((Objects.isNull(sm.getIDTOKEN()))) || (Objects.isNull(sm.getACCESSTOKEN()))) {
            return Response.status(Response.Status.UNAUTHORIZED).entity("Missing SCAD token");
        }

        LOG.info("Passed valid ID and Access Tokens in URL header");
        return null;
    }
}
```

Figure 4.1.3
Code snippet from SCAD's HeaderHelper class illustrating the use of the org.slf4j package for logging (highlighted in green).

Logging SCAD’s behavior in this manner becomes increasingly important as the code base grows larger. It makes debugging significantly easier and provides insight into the sequence of events occurring during the execution of SCAD’s code.

4.2 Security

4.2.1 Challenges of Securing Microservices

SCAD’s microservice-based architecture, while simplifying certain aspects of our system, complicates others. Case in points are identity and access control (e.g. authentication and authorization). Both are integral aspects of SCAD, and in monolithic applications, they are typically implemented with industry-standard, straightforward approaches (e.g. server-side session storage, browser cookies, etc.). However, in the realm of microservice-based applications, no such approach exists, and implementing identity and access control becomes surprisingly complex.

To illustrate this complexity, consider our team’s initial proposal to authenticate users with a standard form-based login page and a “user” table in our database. SCAD’s core functionality is built on top of Yahoo’s API, so the following questions immediately present themselves:

1. Can a user create a SCAD account without a Yahoo account?
2. Can a user be authenticated with SCAD but not Yahoo? If so, will the application work?
3. What if the user changes their Yahoo credentials? Will they also have to change SCAD’s?
4. Will SCAD have inappropriate permissions to a user’s Yahoo account (e.g. their stock picks etc.).

We find that traditional, form-based authentication and authorization solutions do not address the questions above in a manner that is not prohibitively cumbersome to SCAD’s end users.

4.2.2 OAuth2

Consistent with our overarching theme of “not reinventing Yahoo’s wheel” SCAD solves the access control problem with OAuth2, a modern protocol that allows applications like SCAD to access a restricted portion of a Yahoo user’s data (e.g. their fantasy football information) without ever having the Yahoo user’s login credentials pass through SCAD’s system (see Figure 4.2.2).

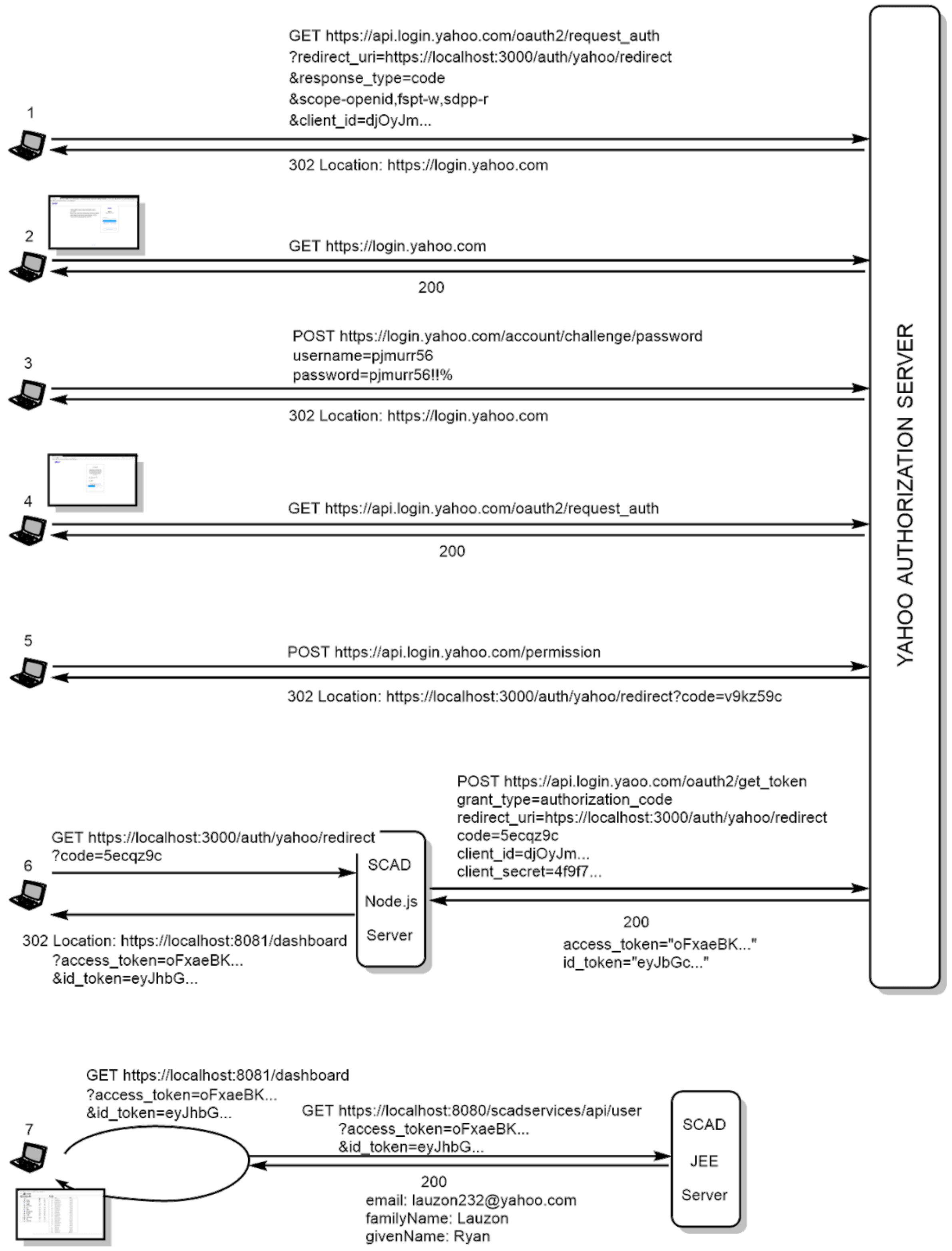

GET https://api.login.yahoo.com/oauth2/request_auth
?redirect_uri=https://localhost:3000/auth/yahoo/redirect
&response_type=code
&scope-openid,fspt-w,sdpp-r
&client_id=djOyJm...
1
302 Location: https://login.yahoo.com
2
GET https://login.yahoo.com
200
POST https://login.yahoo.com/account/challenge/password
username=pjmurr56
password=pjmurr56!!%
3
302 Location: https://login.yahoo.com
4
GET https://api.login.yahoo.com/oauth2/request_auth
200
5
POST https://api.login.yahoo.com/permission
302 Location: https://localhost:3000/auth/yahoo/redirect?code=v9kz59c
YAHOO AUTHORIZATION SERVER
POST https://api.login.yaoo.com/oauth2/get_token
grant_type=authorization_code
redirect_uri=htps://localhost:3000/auth/yahoo/redirect
code=5ecqz9c
client_id=djOyJm...
client_secret=4f9f7...
GET https://localhost:3000/auth/yahoo/redirect
?code=5ecqz9c
6
SCAD
Node.js
Server
302 Location: https://localhost:8081/dashboard
?access_token=oFxaeBK...
&id_token=eyJhbG...
200
access_token="oFxaeBK..."
id_token="eyJbGc..."
GET https://localhost:8081/dashboard
?access_token=oFxaeBK...
&id_token=eyJhbG...
7
GET https://localhost:8080/scadservices/api/user
?access_token=oFxaeBK...
&id_token=eyJhbG...
SCAD
JEE
Server
200
email: lauzon232@yahoo.com
familyName: Lauzon
givenName: Ryan

Figure 4.2.2
SCAD's OAuth2 implementation

Figure 4.2.2 details the relevant aspects of the HTTP requests and responses that occur during SCAD's implementation of OAuth2. The end user initiates the OAuth2 protocol by clicking "Login" (not shown). This is followed by a series of requests and responses (steps 1 - 5) between the user's browser and Yahoo's authorization server where the user authenticates with Yahoo. Note that in steps 1 through 5, no traffic passes through any of SCAD's servers, and except for the initial request, no SCAD code is executed. Consequently, from SCAD's point of view, the user is "authentic" only insofar as Yahoo deems them to be so.

In step 6, following the 302 redirect response, the browser sends a GET request to SCAD's Node.js server along with a "code" query parameter (obtained from Yahoo in step 5). A JavaScript routine running on SCAD's Node.js server combines the value of this "code" with SCAD's "client_id" and "client_secret" into a POST request sent to Yahoo's /get_token endpoint, which returns the access token and id token at the heart of the OAuth2 protocol.

A question immediately arises. Why is it necessary to pass through the Node.js server?

The answer is that without going through a back-end channel (e.g. SCAD's Node.js server), implementing OAuth would require storage of SCAD's client id and client secret on the user's browser. Doing so would allow anyone to impersonate SCAD from the perspective of Yahoo, a significant security vulnerability. To Yahoo, API requests containing SCAD's client id and client secret *are* SCAD requests.

4.2.3 OpenIDConnect

SCAD solves the identity problem with OpenIDConnect, a protocol built on top of OAuth2 that, at its essence, provides SCAD with user information in the form of a JSON Web Token (JWT). A verified JWT, while not necessarily encrypted, is guaranteed to be authentic. In other words, a verified JWT is a guarantee to SCAD that, according to Yahoo, the user is who they claim to be.

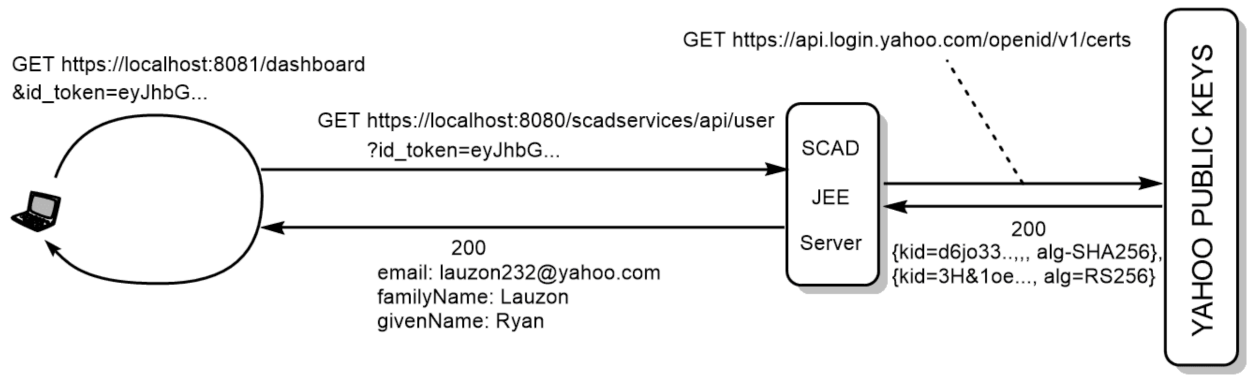

GET https://localhost:8081/dashboard
&id_token=eyJhbG...
GET https://localhost:8080/scadservices/api/user
?id_token=eyJhbG...
200
email: lauzon232@yahoo.com
familyName: Lauzon
givenName: Ryan
SCAD
JEE
Server
GET https://api.login.yahoo.com/openid/v1/certs
200
{kid=d6jo33..,,, alg-SHA256},
{kid=3H&1oe..., alg=RS256}
YAHOO PUBLIC KEYS

Figure 4.2.3
Execution flow for SCAD's "dashboard" homepage, which involves a REST call to SCAD's user endpoint and the verification of a JWT.

Figure 4.2.3 illustrates the REST call made to SCAD's "/user" endpoint that occurs as part of rendering the dashboard landing page. The value of the "id_token" query parameter is the JWT, which we parse and verify prior to using the data in its payload portion.

4.2.4 Decoding JSON Web Tokens

A JWT is a Base64-encoded character string with three, dot-separated sections: 1) a header section, 2) a payload section, 3) a signature. To verify a Yahoo JWT, we first decode the header and retrieve the algorithm and public key id (kID). Then, an HTTP request is sent to Yahoo's /openId/v1/certs endpoint to retrieve a list of Yahoo's public keys. The public key with matching ("alg", "kID") values is selected, and x and y values are retrieved from the public key.

These x and y values, along with a particular type of elliptic curve (P_256), a particular algorithm (ECDSA256), and the kID are used to generate an asymmetric, elliptic-curve based key with help from a third-party cryptography library (Nimbus JOSE/JWT). which is used to verify the authenticity of the signature section of the JWT. The relevant Java code from SCAD's UserApiImpl class is shown in figure 4.2.4.

```
ECKey key = new ECKey.Builder(Curve.P_256, x, y)
                            .algorithm(JWSAlgorithm.ES256)
                            .keyID(kId)
                            .build();

JWSVerifier verifier = new ECDSAVerifier(key);
SignedJWT signedJWT = SignedJWT.parse(idToken);
Boolean verified = signedJWT.verify(verifier);

if (!verified) {
       return Response.status(Response.Status.UNAUTHORIZED).entity("Your
       SCD id token's signature could not be verified.").build();
}
```

Figure 4.2.4
Java code snippet for JWT verification.

A verified JWT indicates that; 1) the token was issued by Yahoo, and 2) the token has never been modified. Once the token is verified, the user data contained in the payload section can be trusted as authentic and used.

4.2.5 Basic Authentication

In addition to securing SCAD at the application level with OAuth2 and OpenIDConnect, SCAD contains an additional, more general layer of security for its REST services. Although SCAD has a microservice architecture, its current version is not designed to expose any REST service to a consuming client other than SCAD's front end. Therefore, all requests made to a SCAD service from a non-SCAD client should be rejected.

We implement this feature with basic authentication, a technique whereby a username/password combination is sent from SCAD's front-end on every REST call (as a Base64 encoded header). Although it is possible to implement basic authentication on the server side with code, the JEE application server that we use (Thorntail) can be configured to provide basic authentication. We use this configuration-based approach by including the following snippet in our project-local.yml configuration file.

```
security:
 security-domains:
  classic-authentication:
   login-modules:
    scad-ba-rs:
     code: org.jboss.security.auth.spi.UsersRolesLoginModule
     flag: required
     module-options:
      userProperties: ${env.SCAD_USERS_PATH}
      rolesProperties: ${env.SCAD_ROLES_PATH}
      hashAlgorithm: SHA-256
      hashEncoding: base64
```

Figure 4.2.5
Thorntail configuration for basic authentication.

We also include the following in our jboss-web.xml file.

```
<jboss-web>
        <context-root>scadservices</context-root>
        <security-domain>scad-basicauth</security-domain>
</jboss-web>
```

Figure 4.2.5a
JEE web application server configuration for basic authentication.

It is worth noting that this basic authentication, combined with SCAD's exclusive use of HTTPS, prevents non-SCAD users (e.g. spammers, bots, etc.) from infiltrating our system, but it does not provide any protection against a malicious actor who is also a SCAD user. A SCAD user can access the basic auth username/password combination directly from their browser.

4.3 Code Base

4.3.1 Front-end/User Interface

SCAD’s front-end/UI microservice repository consists of approximately 10,000 lines of code contained in 89 files organized into 17 directories. The breakdown by file type is shown in the figure below.

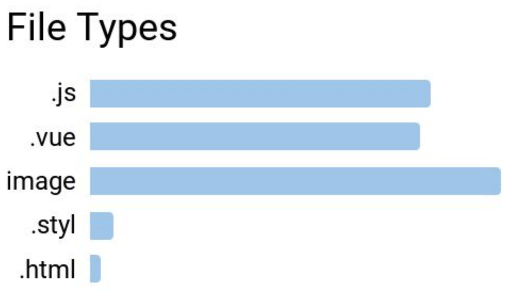


Figure 4.3.1
Breakdown of SCAD’s front-end/UI code base by file type. Image files are .png, .svg, .ico, or .jpeg.

SCAD’s front-end/UI project directory structure is shown in figure 4.3.1b. It is worth noting that prior to deployment, the Vue.js framework is compiled and packaged by WebPack, which ultimately converts the .vue and .styl files into standard JavaScript (.js) files and cascading style sheets (.css). Hence, in production, only a lightweight web server is required because code is never *executed* on the server side. It is simply returned as a static asset.

Also note the presence of a single .html file, the hallmark of a single page application (SPA). SCAD’s only .html file is index.html, which has only 17 lines of markup.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width,initial-scale=1.0">
    <link rel="icon" href="<%= BASE_URL %>favicon.ico">
    <title>scad-ui</title>
  </head>
  <body>
    <noscript>
      <strong>We're sorry but scad-ui doesn't work properly without JavaScript enabled. Please
enable it to continue.</strong>
    </noscript>
    <div id="app"></div> <!-- built files will be auto injected -->
  </body>
</html>
```

Figure 4.3.1a
Content of index.html, SCAD’s only .html file.

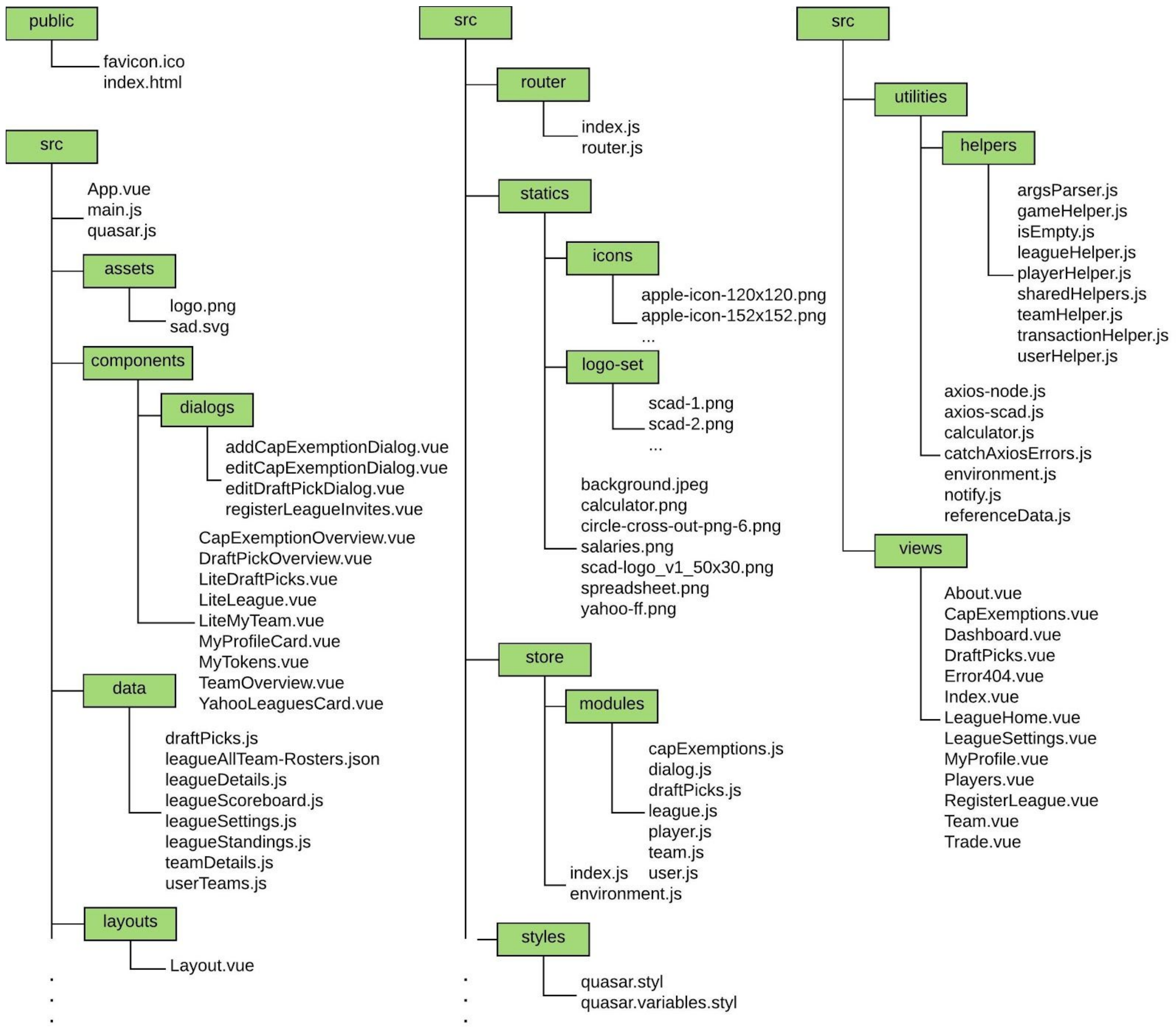


Figure 4.3.1b
Directory structure of SCAD's front-end/UI microservice. Note: the three boxes denoted "src" refer to the same directory.

4.3.2 Node.js Microservice

SCAD's Node.js back-end microservice repository consists of approximately 500 lines of code contained in 15 files organized into 10 directories. The breakdown by file type and directory structure is shown in figures 4.3.2 and 4.3.2a below.

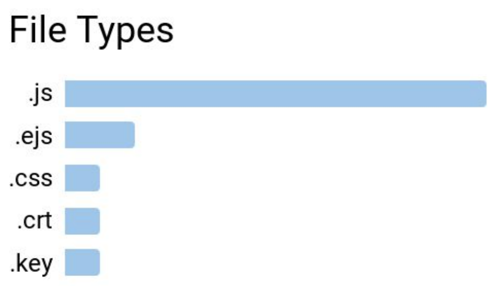


Figure 4.3.2
Breakdown of SCAD’s Node.js code base repository by file type.

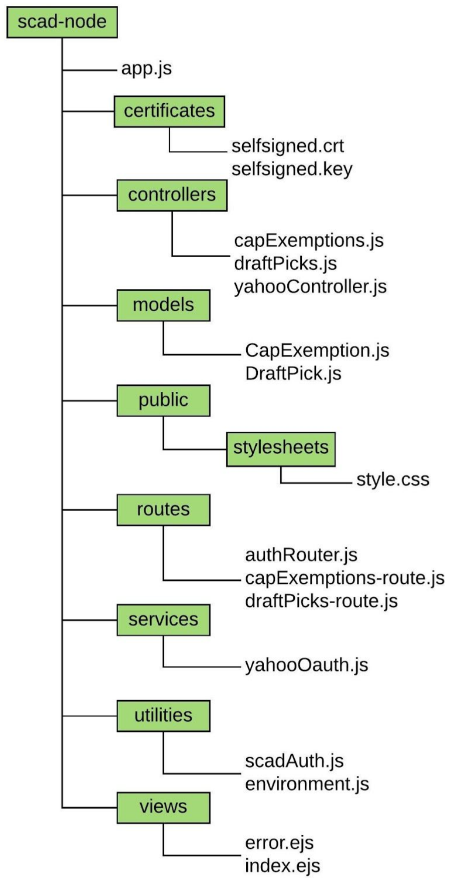


Figure 4.3.2a
Directory structure of SCAD’s Node.js microservice.

4.3.3 JEE Microservice

SCAD's JEE microservice repository consists of approximately 15,000 lines of source code contained in 93 files and organized into 24 directories. The breakdown by file type is as follows:

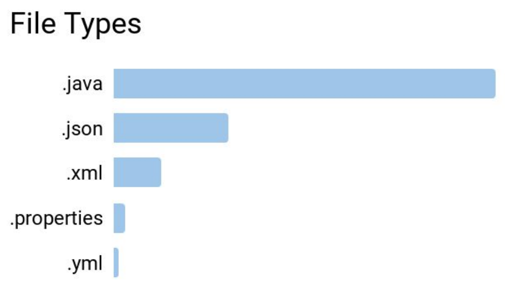


Figure 4.3.3
Breakdown of SCAD JEE back-end code base by file type.

SCAD's maven-style project directory structure is shown in figures 4.3.3a and 4.3.3b below.

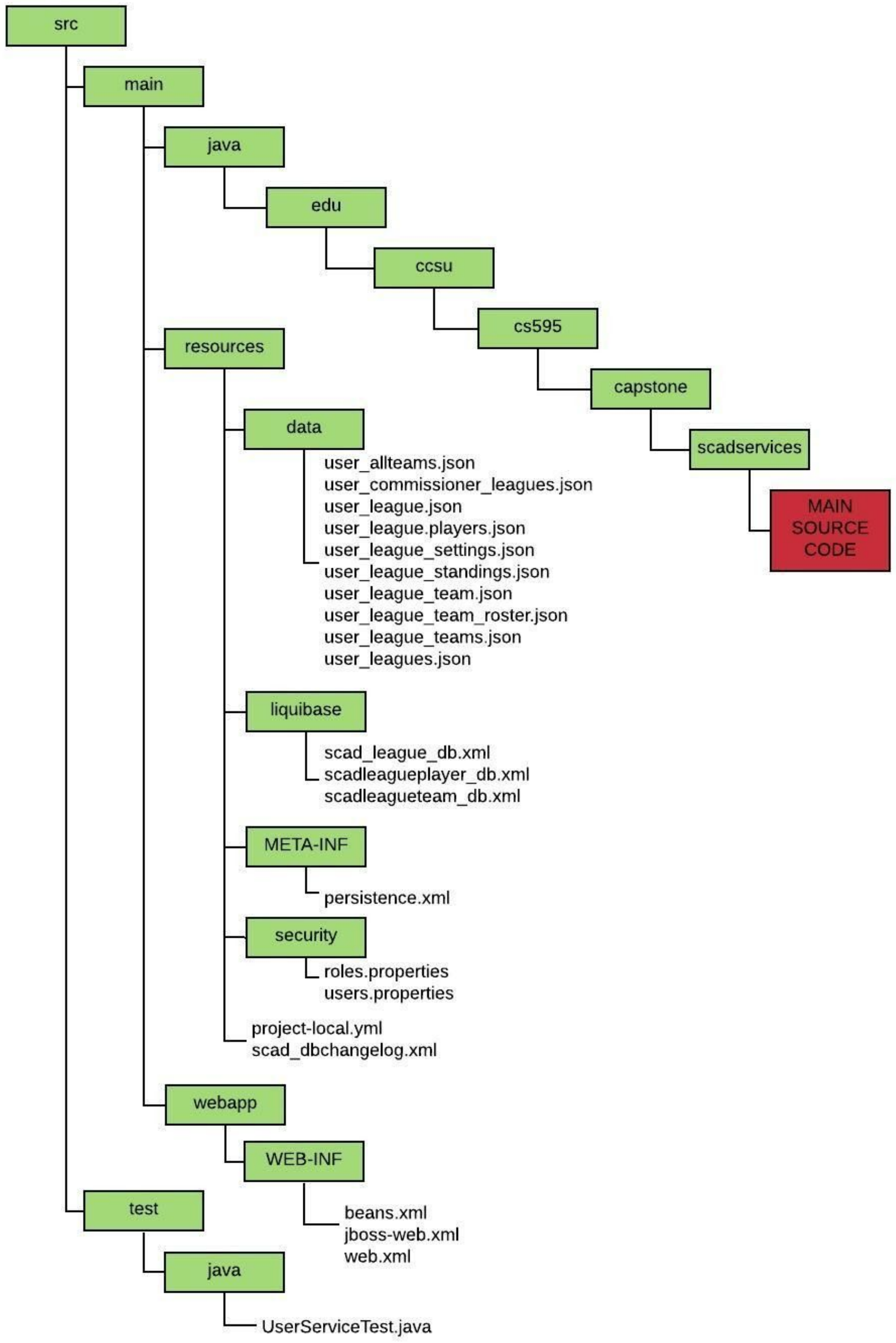


Figure 4.3.3a
Directory structure of SCAD's JEE back-end microservice. The directory structure of the main source files (red box) is shown in figure 4.3.3b.

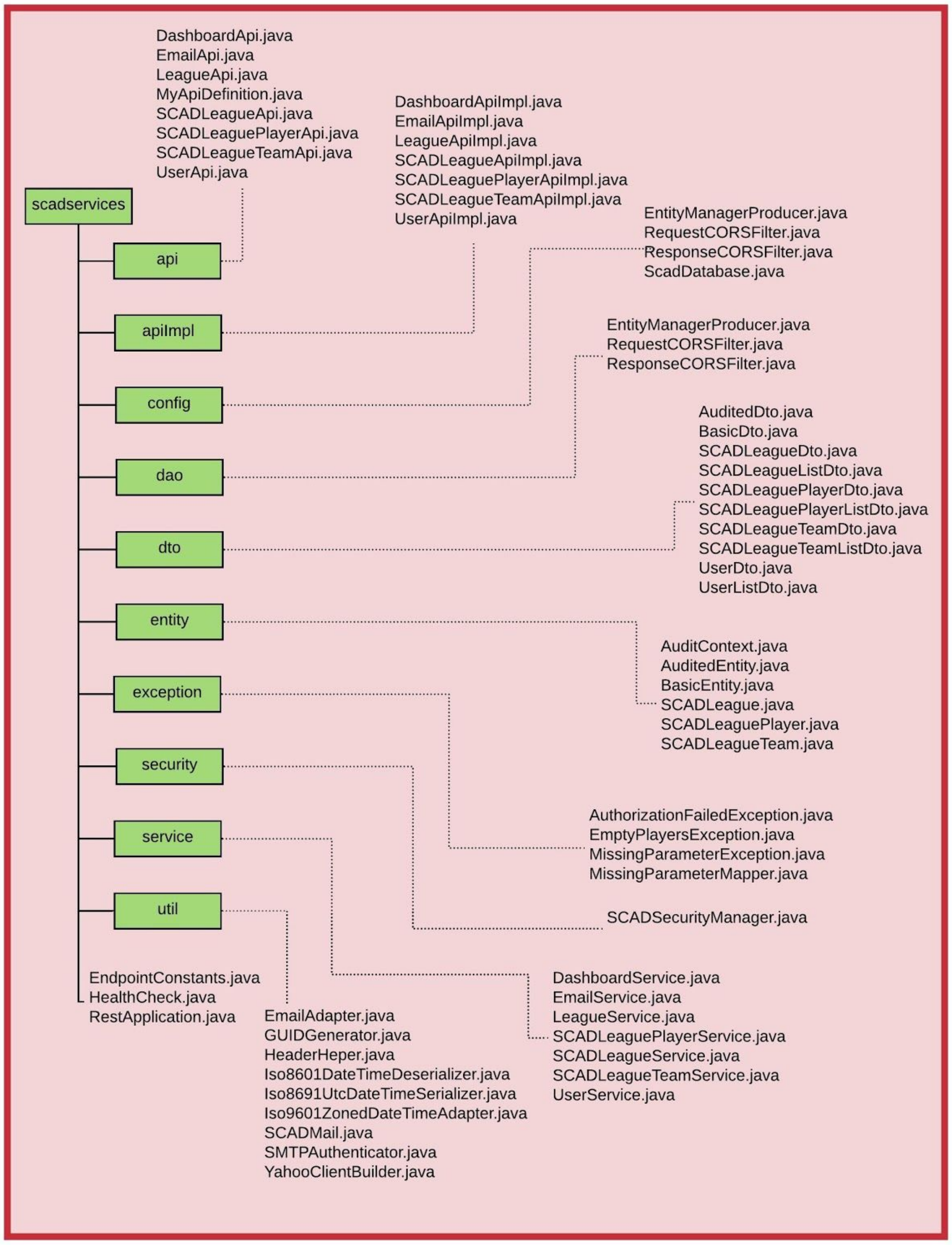


Figure 4.3.3b
Directory structure of Java classes in SCAD’s edu.ccsu.cs595.capstone.scadservices package, the main source code of the JEE back-end.

4.3.4 Swagger

SCAD documents its REST endpoints with an extremely useful API documentation tool called Swagger. To use this tool, back-end REST endpoints are annotated with @Api, @ApiOperation, and @ApiParam and the Swagger .jar file is included as a runtime dependency of Thorntail (SCAD's JEE application server).

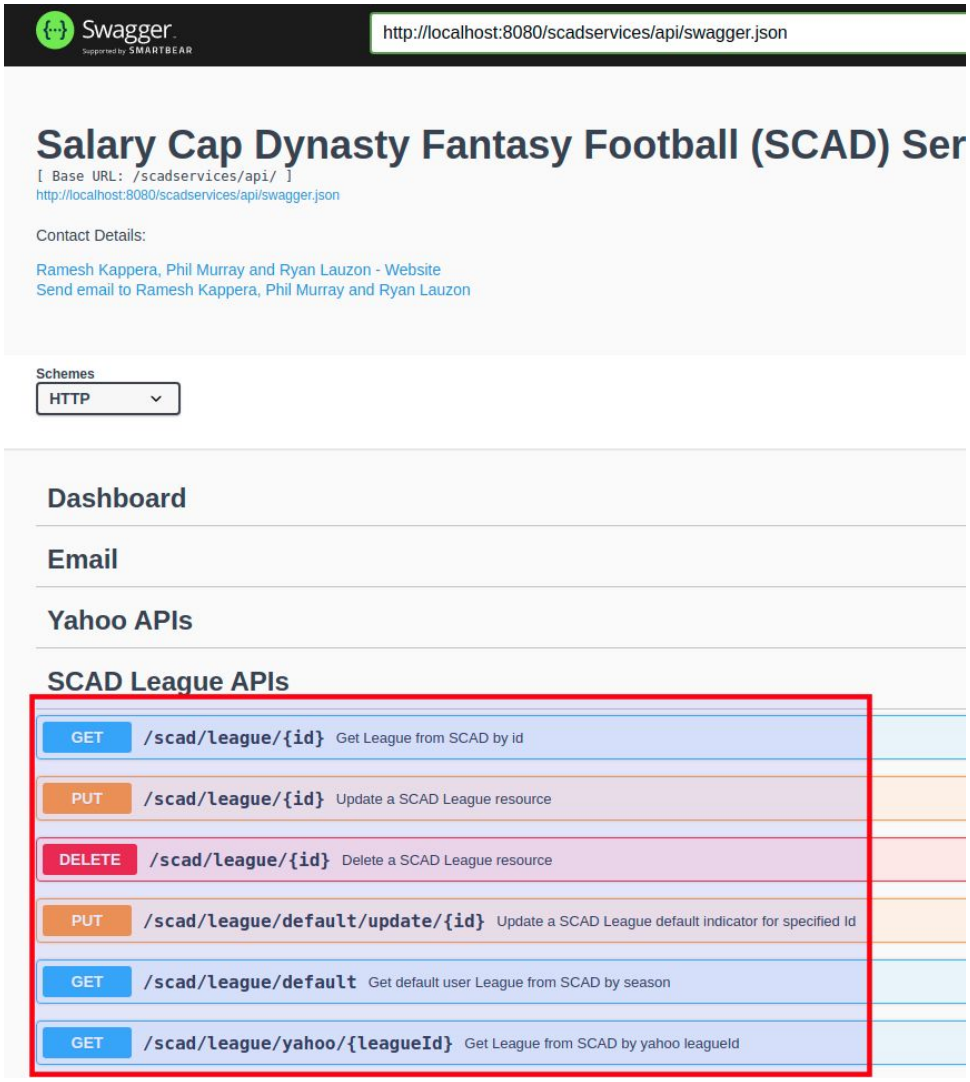
Consider the code of SCADLeaugeApi.java interface shown below. Highlighted in green are the Swagger annotations.

```
package edu.ccsu.cs595.capstone.scadservices.api;

import edu.ccsu.cs595.capstone.scadservices.EndpointConstants;
import edu.ccsu.cs595.capstone.scadservices.dto.SCADLeagueDto;
import edu.ccsu.cs595.capstone.scadservices.dto.SCADLeagueListDto;
import edu.ccsu.cs595.capstone.scadservices.exception.AuthorizationFailedException;
import edu.ccsu.cs595.capstone.scadservices.exception.MissingParameterException;
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiOperation;
import io.swagger.annotations.ApiParam;


@ApplicationScoped
@Path(EndpointConstants.SCAD)
@Api(value = EndpointConstants.LEAGUE, tags = "SCAD League APIs", description = "Get League settings from
SCAD system")
public interface SCADLeagueApi {

        @GET
        @Path(EndpointConstants.LEAGUE + "/default")
        @ApiOperation(value = "Get default user League from SCAD by season", notes = "Returns default user
        League from SCAD by season", response = SCADLeagueDto.class)
        @Produces({ MediaType.APPLICATION_JSON })
        public Response getDefaultUserSCADLeagueBySeason() throws AuthorizationFailedException,
        RuntimeException;



        @PUT
        @Path(EndpointConstants.LEAGUE + "/default/update/{id}")
        @ApiOperation(value = "Update a SCAD League default indicator for specified Id", notes = "Update
        League default indicator in SCAD system", response = SCADLeagueDto.class)
        @Produces({ MediaType.APPLICATION_JSON })
        public Response updateSCADLeagueDefaultIndicator(
                @ApiParam(value = EndpointConstants.RESOURCE_ID, required = true)
                @PathParam(value = EndpointConstants.ID) Long id)
                throws AuthorizationFailedException, RuntimeException;



        @GET
        @Path(EndpointConstants.LEAGUE+ "/{id}")
        @ApiOperation(value = "Get League from SCAD by id", notes = "Returns League from SCAD by id",
        response = SCADLeagueDto.class)
        @Produces({ MediaType.APPLICATION_JSON })
        public Response getSCADLeague(
                @ApiParam(value = EndpointConstants.RESOURCE_ID, required = true)
                @PathParam(EndpointConstants.ID) Long id)
                throws AuthorizationFailedException, RuntimeException;



        @PUT
        @Path(EndpointConstants.LEAGUE + "/{id}")
        @ApiOperation(value = "Update a SCAD League resource", notes = "Update League additional settings in
        SCAD system", response = SCADLeagueDto.class)
        @Produces({ MediaType.APPLICATION_JSON })
        @Consumes({ MediaType.APPLICATION_JSON })
        public Response updateSCADLeague(
                @ApiParam(value = EndpointConstants.RESOURCE_ID, required = true)
                @PathParam(value = EndpointConstants.ID) Long id,
                @ApiParam(value = EndpointConstants.PUTPROPOSED, required = true)
                SCADLeagueDto proposed)
                throws MissingParameterException, AuthorizationFailedException, RuntimeException;



        @DELETE
        @Path(EndpointConstants.LEAGUE + "/{id}")
        @ApiOperation(value = "Delete a SCAD League resource", notes = "Delete League additional settings
        from SCAD system", response = Boolean.class)
        @Produces({ MediaType.APPLICATION_JSON })
        public Response deleteSCADLeague(
                @ApiParam(value = EndpointConstants.RESOURCE_ID, required = true)
                @PathParam(value = EndpointConstants.ID) Long id)
                throws MissingParameterException, AuthorizationFailedException, RuntimeException;



        @GET
        @Path(EndpointConstants.LEAGUE + "/yahoo/{leagueId}")
        @ApiOperation(value = "Get League from SCAD by yahoo leagueId", notes = "Returns League from SCAD by
        yahoo leagueId", response = SCADLeagueDto.class)
        @Produces({ MediaType.APPLICATION_JSON })
        public Response getSCADLeagueByYahooLeague(
                @ApiParam(value = EndpointConstants.LEAGUE_RESOURCE_ID, required = true)
                @PathParam(EndpointConstants.LEAGUEID) Long leagueId)
                throws AuthorizationFailedException, RuntimeException;

}
```

Figure 4.3.4

The Java code in SCAD's SCADLeagueApi.java file. Highlighted in green are the Swagger annotations.

When the development server is started, the Swagger annotations in the SCADLeagueApi.java file enable the runtime environment to create a GUI tool where those endpoints can be easily accessed from a web browser.


Swagger
Supported by SMARTBEAR

http://localhost:8080/scadservices/api/swagger.json

Salary Cap Dynasty Fantasy Football (SCAD) Ser

[Base URL: /scadservices/api/]

http://localhost:8080/scadservices/api/swagger.json

Contact Details:

Ramesh Kappera, Phil Murray and Ryan Lauzon - Website

Send email to Ramesh Kappera, Phil Murray and Ryan Lauzon

Schemes

HTTP

Dashboard

Email

Yahoo APIs

SCAD League APIs

GET /scad/league/{id} Get League from SCAD by id

PUT /scad/league/{id} Update a SCAD League resource

DELETE /scad/league/{id} Delete a SCAD League resource

PUT /scad/league/default/update/{id} Update a SCAD League default indicator for specified Id

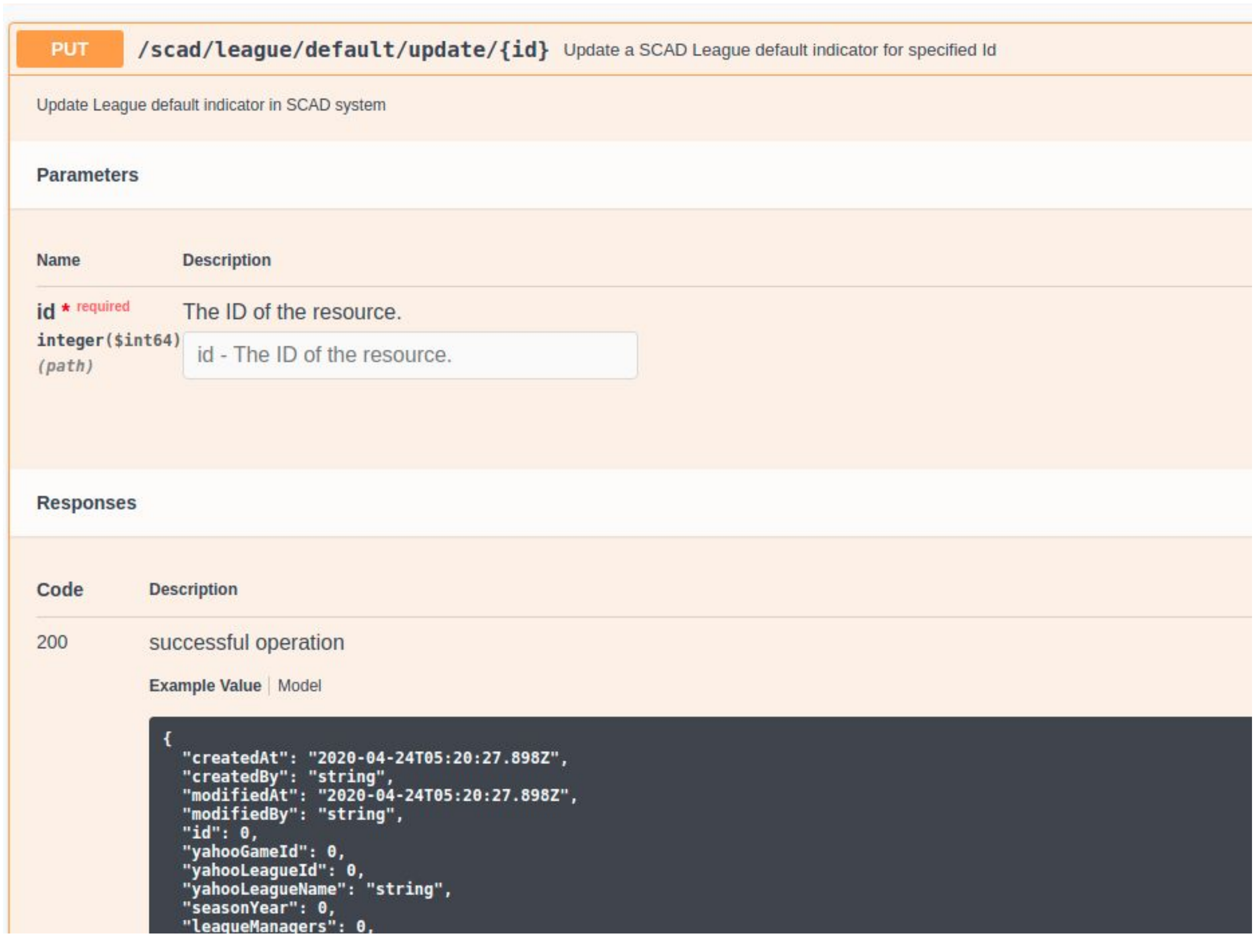
GET /scad/league/default Get default user League from SCAD by season

GET /scad/league/yahoo/{leagueId} Get League from SCAD by yahoo leagueId

Figure 4.3.4a

GUI tool created by Swagger as a result of including Swagger's annotations in the SCADLeagueApi.java interface file. The red box surrounds the six endpoints that correspond to the six interface methods in the code.

The Swagger tool not only documents the REST endpoints, but also allows the developer to test the endpoints with various parameters and headers.


PUT **/scad/league/default/update/{id}** Update a SCAD League default indicator for specified Id

Update League default indicator in SCAD system

Parameters

Name	Description
id * required integer($int64) *(path)*	The ID of the resource. id - The ID of the resource.

Responses

Code	Description
200	successful operation **Example Value** | Model

```
{
  "createdAt": "2020-04-24T05:20:27.898Z",
  "createdBy": "string",
  "modifiedAt": "2020-04-24T05:20:27.898Z",
  "modifiedBy": "string",
  "id": 0,
  "yahooGameId": 0,
  "yahooLeagueId": 0,
  "yahooLeagueName": "string",
  "seasonYear": 0,
  "leagueManagers": 0,
```

Figure 4.3.4b
A screenshot of Swagger's sample response when accessing SCAD's /scad/league/default/update/{id} endpoint. In this case, the HTTP verb is "PUT", and its function is to set a certain league as the default.

Swagger is a particularly valuable tool for SCAD developers, because SCAD's front-end/UI is a separate microservice than SCAD's back-end REST services, so having clearly documented REST endpoints (with sample return payloads) allows the front-end/UI developers to work independently of the back-end team. This prevents "hold and wait" situations in development and significantly increases productivity.

The Swagger .jar files are included in SCAD via a project dependency configured in SCAD's pom.xml as shown below:

```
<dependency>
    <groupId>io.thorntail</groupId>
    <artifactId>swagger</artifactId>
</dependency>
<dependency>
    <groupId>io.thorntail</groupId>
    <artifactId>swagger-webapp</artifactId>
</dependency>
```

Figure 4.3.4c
Snippet of SCAD's pom.xml file showing how SCAD includes the Swagger tool.

4.4 Data Models

4.4.1 Disjointness of SCAD's Models and Yahoo's Models.

Yahoo's domain models contain seven primary resources: game, league, team, roster, player, transaction, and user. Yahoo's API enables access to these resources by making REST calls to its urls. For example, if we call https://fantasysports.yahooapis.com/fantasy/v2/league/390.l.22351?format=json and include the appropriate security headers, we get the following JSON payload of a Yahoo league resource.

```
{
  "fantasy_content": {
        "xml:lang": "en-US",
        "yahoo:uri": "/fantasy/v2/league/390.l.22351",
        "league": [
               {
               "league_key": "390.l.22351",
               "league_id": "22351",
               "name": "Salary Cap Dynasty 2019",
               "url": "https://football.fantasysports.yahoo.com/archive/nfl/2019/22351",
               "logo_url": false,
               "draft_status": "postdraft",
               "num_teams": 12,
               "edit_key": "16",
               "weekly_deadline": "",
               "league_update_timestamp": "1577871056",
               "scoring_type": "head",
               "league_type": "private",
               "renew": "380_34068",
               "renewed": "399_13088",
               "iris_group_chat_id": "EWFTYULT45AEXPDOCQXVOC55XQ",
               "allow_add_to_dl_extra_pos": 0,
               "is_pro_league": "0",
               "is_cash_league": "0",
               "current_week": "16",
               "start_week": "1",
               "start_date": "2019-09-05",
               "end_week": "16",
               "end_date": "2019-12-23",
               "is_finished": 1,
               "game_code": "nfl",
               "season": "2019"
              }
         ],
        "time": "18.115997314453ms",
        "copyright": "Data provided by Yahoo! and STATS, LLC",
        "refresh_rate": "60"
    }
}
```

Figure 4.4.1
Raw JSON returned from a direct REST call made to Yahoo's league API. Of course, the payload is missing any salary cap data, because it comes directly from Yahoo.

By looking at the JSON payload, we can see the properties and structure of Yahoo's "league" resource.

SCAD *also* has a league resource, persisted in a MySQL database and designed specifically to have one linking field (league_id ↔ yahooLeagueId), but otherwise be completely disjoint from Yahoo's league resource. This *SCAD* league resource is accessed by SCAD's front-end/UI at the url https://<scad_top_level_domain>/scadservices/api/scad/league/yahoo/22351. It returns a JSON payload like the one shown below.

```
{
    "yahooLeagueId": 22351,
    "yahooGameId": 390,
    "seasonYear": 2019,
    "leagueManagers": 12,
    "rookieDraftRds": 3,
    "rookieDraftStrategy": "Message Board",
    "rookieWageScale": "Standard",
    "teamSalaryCap": 250,
    "leagueSalaryCap": 3000,
    "salaryCapExemptionLimit": 25,
    "irReliefPerc": 50,
    "franchiseTagDiscount": 50,
    "franchiseTagSpots": 1,
    "tradingDraftPickYears": 5,
    "qbMin": 2,
    "qbMax": 5,
    "rbMin": 4,
    "rbMax": 7,
    "wrMin": 5,
    "wrMax": 8,
    "teMin": 2,
    "teMax": 4,
    "kMin": 0,
    "kMax": 2,
    "defMin": 2,
    "defMax": 4,
    "isDefault": true,
    "ownerGuid": "2OMLCT3C2A42Z3FCGWJZCIDYLU",
    "renewSCADLeagueId": 0,
    "rosterSpotLimit": 0,
    "isCurrentlyLoggedInUserACommissioner": true,
    "id": 1,
    "createdBy": "2OMLCT3C2A42Z3FCGWJZCIDYLU",
    "createdAt": "2020-04-10T14:54:51Z",
    "modifiedBy": "2OMLCT3C2A42Z3FCGWJZCIDYLU",
    "modifiedAt": "2020-04-10T14:54:51Z"
}
```

Figure 4.4.1a

JSON payload returned from a REST call made to SCAD's league API. Although this SCAD league and the Yahoo league in figure 4.4.1 refer to the same resource, notice how their fields are disjoint.

SCAD is built on top of Yahoo, so the primary design objective for SCAD's persistent entities is to create domain models that 1) refer to Yahoo resources, and 2) have property sets that are disjoint from those resources. SCAD's persisted entities are detailed in the next section.

4.4.2 Relational Model

Remarkably, SCAD version 1.0.0 only requires three tables in its relational database. The entities are illustrated in figure 4.4.2 below.

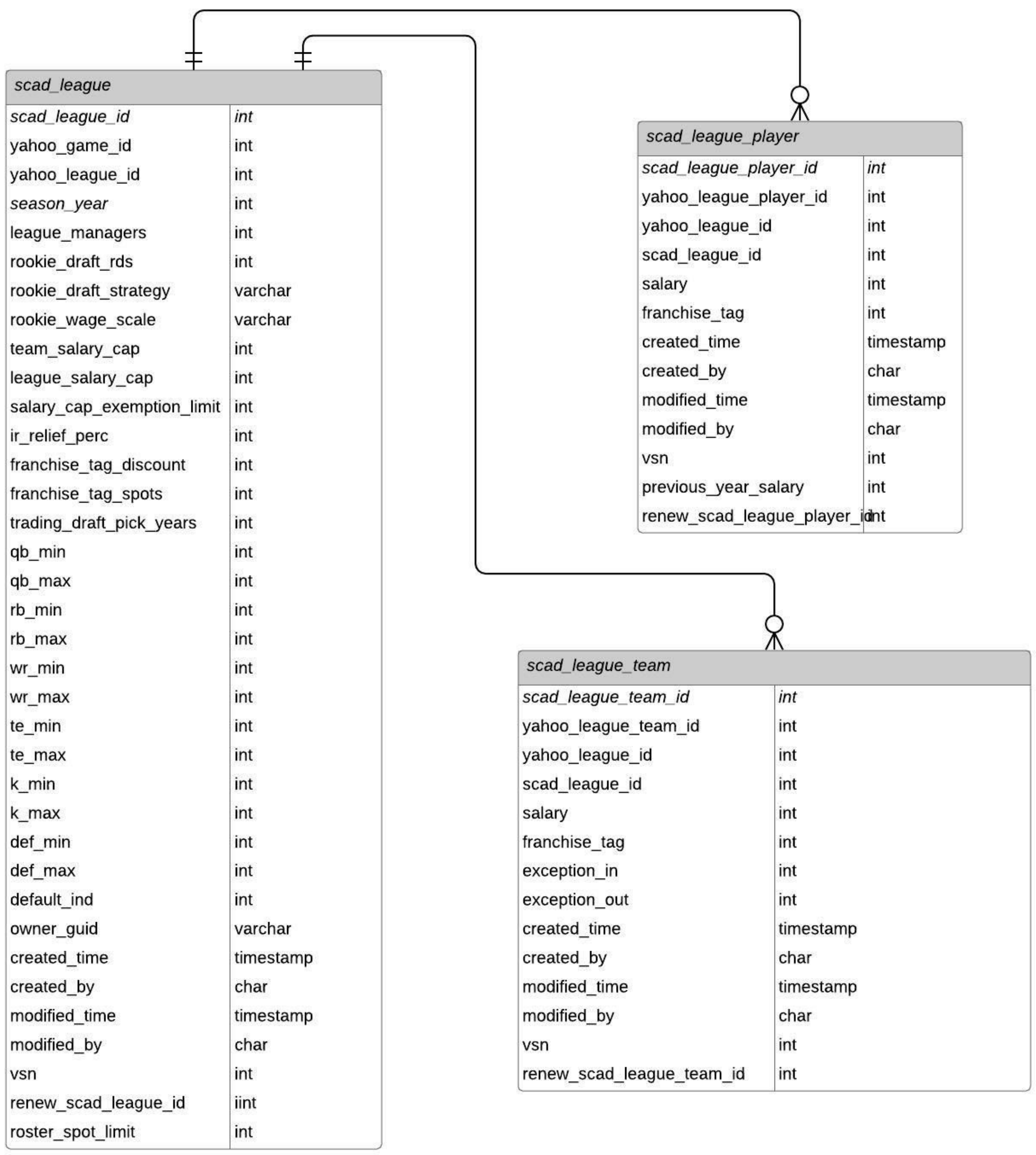


Figure 4.4.2

Entity-relationship diagram of SCAD's relational database. Note the lack of any relationship between scad_league_team and scad_league_player. In fact, the relationship does exist, but the linking entity is a "roster", which SCAD does not persist, because it would duplicate data and violate SCAD's design principle of maintaining disjointness from Yahoo.

4.4.3 Hibernate

SCAD uses the popular Hibernate framework for its object-relational mapper (ORM), obviating the need for direct SQL queries to the database. Since Thorntail includes fractions for Hibernate, implementing the ORM is accomplished in SCAD by including the Thorntail dependency in SCAD’s pom.xml, registering the persistence-unit and entity classes in the persistence.xml file, and creating an EntityManagerProducer class that is empty except for an EntityManager member variable annotated with @Produces, and @PersistenceContext.

```
<dependency>
    <groupId>io.thorntail</groupId>
    <artifactId>jpa</artifactId>
</dependency>
```

Figure 4.4.3
Snippet from SCAD’s pom.xml illustrating how SCAD includes Hibernate ORM.

```
<persistence-unit name=”scadSvcPu” transaction-type=”JTA”>
    <provider>org.hibernate.ejb.HibernatePersistence</provider>
    <jta-data-source>java:/jdbc/ccsu/ScadDS</jta-data-source>
    <class>edu.ccsu.cs595.capstone.scadservices.entity.SCADLeague</class>
    <class>edu.ccsu.cs595.capstone.scadservices.entity.SCADLeagueTeam</class>
    <class>edu.ccsu.cs595.capstone.scadservices.entity.SCADLeaguePlayer</class>
</persistence unit>
```

Figure 4.4.3a
Snippet from SCAD’s persistence.xml illustrating how Hibernate, MySQL, and the entity classes are linked.

```
package edu.ccsu.cs595.capstone.scadservices.config;

import javax.enterprise.inject.Produces;
import javax.persistence.EntityManager;
import javax.persistence.PersistenceContext;

public class EntityManagerProducer {

    @Produces
    @PersistenceContext(name = “scadSvcPu”, unitName = “scadSvcPu”)
    @ScadDatabase
    private EntityManager scadEm;
}
```

Figure 4.4.3b
Snippet from SCAD’s EntityMangerProducer class illustrating Hibernate’s entry point into SCAD’s application code.

4.4.4 Cleaning Yahoo's Raw Data

Although SCAD's front-end/UI can call Yahoo's APIs directly, the raw JSON payloads returned from Yahoo often contain superfluous fields (e.g. refresh_rate, copyright) and are frequently returned as untidy structures with empty arrays, singleton JSON sub-objects, and unnecessary nesting.

Managing SCAD and Yahoo payloads on the front-end is much easier if the payloads have the same structure. To this end, SCAD possesses a collection of endpoints that essentially mimic Yahoo's endpoints but return cleaner payloads. For example, the Yahoo API endpoint that returns the payload shown in figure 4.4.1 can be called through SCAD at http://<scad_top_level_domain>/scadservices/api/yahoo/league/22351, returning a cleaner version of the same data.

```
{
   "league_key": "390.l.22351",
   "league_id": "22351",
   "name": "Salary Cap Dynasty 2019",
   "url": "https://football.fantasysports.yahoo.com/archive/nfl/2019/22351",
   "logo_url": false,
   "draft_status": "postdraft",
   "num_teams": 12,
   "edit_key": "16",
   "weekly_deadline": "",
   "league_update_timestamp": "1577871056",
   "scoring_type": "head",
   "league_type": "private",
   "renew": "380_34068",
   "renewed": "399_13088",
   "iris_group_chat_id": "EWFTYULT45AEXPDOCQXVOC55XQ",
   "allow_add_to_dl_extra_pos": 0,
   "is_pro_league": "0",
   "is_cash_league": "0",
   "current_week": "16",
   "start_week": "1",
   "start_date": "2019-09-05",
   "end_week": "16",
   "end_date": "2019-12-23",
   "is_finished": 1,
   "game_code": "nfl",
   "season": "2019"
}
```

Figure 4.4.4
JSON payload returned from one of SCAD's "cleaning" endpoints. The "unclean" version of this payload returned directly from Yahoo is shown in figure 4.4.1.

4.4.5 JAX-RS

SCAD heavily depends on JAX-RS, the Java API for RESTful web services. SCAD's JEE runtime environment (Thorntail) can essentially transform a standard Java class into a REST endpoint with the addition of a simple @Path annotation at the class-level, parameterized by a string indicating the relative url mapped to that class.

At the method level, the @Path annotation denotes the url relative to the class-level @Path annotation, and the @Produces annotation indicates the desired MIME type. Methods of a class annotated with @Path can essentially be thought of as the "entry points" for the incoming HTTP request, where SCAD's business logic begins. In addition to @Path, the methods must also be annotated with the appropriate HTTP verb (@GET, @PUT, @POST, @DELETE, etc.).

```
package edu.ccsu.cs595.capstone.scadservices.api

import javax.enterprise.context.ApplicationScoped;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;

import edu.ccsu.cs595.capstone.scadservices.EndpointConstants;

@ApplicationScoped
@Path(EndpointConstants.SCAD)
public interface DashboardApi {

    @GET
    @Path(EndpointConstants.DASHBOARD + "/details")
    @Produces({ MediaType.APPLICATION_JSON })
    public Response getDashboardDetails() throws RuntimeException;
}
```

Figure 4.4.5
Code snippet from SCAD's DashboardApi interface, illustrating the JAX-RS @Path annotations responsible for properly routing an HTTP request to the getDashBoardDetails() method.

In addition to request mappings, SCAD makes use of JAX-RS for request and response filtering. Every incoming HTTP request, after passing the basic-authentication security measure configured in Thorntail, will filter through SCAD's RequestCORSFilter class, which contains two JAX-RS hooks; 1) being annotated at the class level with @Provider notifies the JEE runtime environment that the class has special significance to JAX-RS; 2) implementing javax.ws.rs.container.ContainerRequestFilter tells the runtime environment that all requests should filter through the "filter()" method of the class.

```
package edu.ccsu.cs595.capstone.scadservices.config;

import java.inject.Inject;
import javax.ws.rs.container.ContainerRequestContext;
import javax.ws.rs.container.ContainerRequestFilter;
import javax.ws.rs.ext.Provider;
import edu.ccsu.cs595.capstone.scadservices.security.SCADSecurityManager;

@Provider
public class RequestCORSFilter implements ContainerRequestFilter {
    @Inject
    SCADSecurityManager sm;

    @Override
    public void filter(ContainerRequestContext requestContext) {
        sm.setIDTOKEN(requestContext.getHeaderString(sm.ID_TOKEN_HEADER);
        sm.setACCESSTOKEN(requestContext.getHeaderString(sm.ACCESS_TOKEN_HEADER);
        sm.setAUTHORIZATION(requestContext.getHeaderString(sm.AUTHORIZATION_HEADER);
    }
}
```

Figure 4.4.5a
Code snippet of RequestCORSFilter class, illustrating how SCAD uses two JAX-RS hooks (highlighted in green) to convert the class into one whose filter() method is invoked on every incoming request.

4.4.6 From Request to Response

When a user accesses SCAD's landing page, SCAD serves the index.html file from its Node.js back-end microservice along with many JavaScript, CSS, and image files. Being a single page web application, if the user subsequently clicks to visit another page, SCAD does not retrieve another .html page, but rather triggers the execution of client-side JavaScript routines that retrieve necessary static assets from the Node.js server (done automatically as part of the Vue.js framework), and also fetches data by making REST calls to SCAD's service endpoints.

A typical pattern in SCAD is shown in figure 4.4.6, where SCAD's front-end/UI makes a REST call to retrieve a SCAD resource followed by another REST call to retrieve a Yahoo resource of the same type (league). These resources are then combined and displayed on the page as needed.

It is worth noting that this "combining" of SCAD and Yahoo resources could theoretically be done on the back-end and served to the front-end/UI as a single payload. The motivation for pushing the extra logic to the front-end is simply that SCAD's front-end developer is also a product owner who understands the business logic better than the back-end developers.

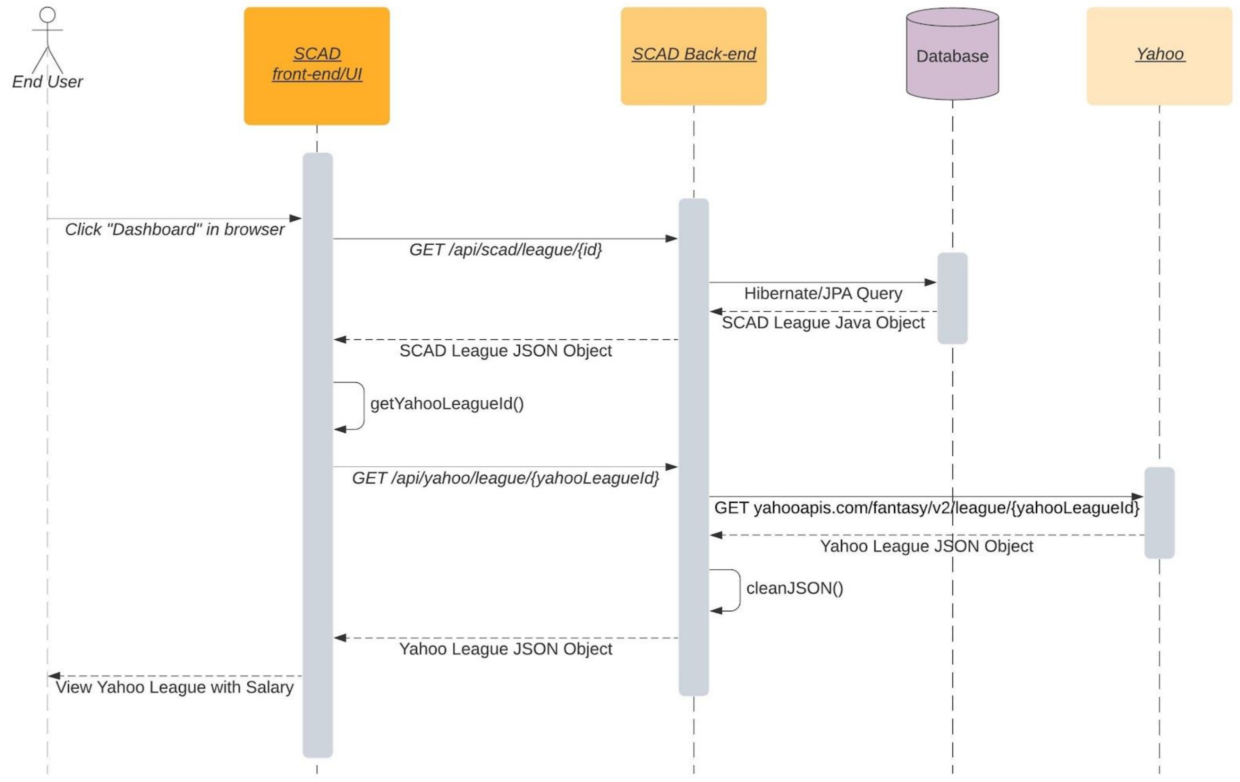


Figure 4.4.6
A high-level sequence diagram showing SCAD's rendering of the "dashboard" page.

The code executed when a SCAD endpoint creates and returns a JSON payload is organized into 5 layers.

Layer 1 - API Implementation Classes:
An incoming HTTP request is routed by the JEE application server to an appropriate method on an API implementation class (e.g. SCADLeagueApiImpl, DashboardApiImpl, etc.). These api implementation classes constitute layer 1. As shown in figure 4.4.6a, the class- and method-level @Path annotation on the interface implemented provide the mappings from the url to the method.

Layer 2 - Data Transfer Object Layer (DTO)
Frequently, JSON payloads being consumed and produced by SCAD's endpoints require some modification or decoration prior to being marshalled or unmarshalled. A suite of data transfer objects (DTOs) serves this purpose and constitutes layer 2. In the cases of PUTs and POSTs, where the endpoints must unmarshall JSON objects early on in the request lifecycle, the DTO layer is accessed before layers 3, 4, and 5. However, for GETs and DELETEs, the DTO layer will not be accessed until right before marshalling, which occurs after layers 3, 4, and 5 have been accessed.

SCADLeagueApi.java

```
@Path(EndpointConstants.SCAD)
public interface SCADLeagueApi {

        @GET
        @Path(EndpointConstants.LEAGUE+ "/{id}")
        @Produces({ MediaType.APPLICATION_JSON })
        public Response getSCADLeague(
        @PathParam(EndpointConstants.ID) Long id) throws RuntimeException;
}
```

SCADLeagueApiImpl.java

```
public class SCADLeagueApiImpl implements SCADLeagueApi {

@Inject
SCADLeagueService slSvc;

        @Override
        public Response getSCADLeague(Long id) throws RuntimeException {
                hHlpr.isHeaderAccessValid(sm);
                SCADLeagueDto result = slSvc.getSCADLeague(id);
                if (Objects.isNull(result)) {
                        return Response.status(Response.Status.NOT_FOUND).build();
                }
                return Response.status(Response.Status.OK).entity(result).build();
        }
}
```

SCADLeagueDto.java

```
@JsonIgnoreProperties(ignoreUnknown = true)
@JsonSerialize(include=JsonSerialize.Inclusion.NON_NULL)
@JsonInclude(JsonInclude.Include.NON_EMPTY)
public class SCADLeagueDto extends AuditedDto {
        private Long id;
        private Long yahooGameId;
        private Long yahooLeagueId;
        ...
}
```

Figure 4.4.6a

Code snippets from SCADLeagueApi.java and SCADLeagueApiImpl.java illustrating layer 1 of SCAD's 5 logical layers. Also shown is a snippet from SCADLeagueDto.java illustrating level 2. Highlighted in green are the annotations that provide the necessary information for routing the HTTP request to the appropriate method. Highlighted in red is the code responsible for transitioning to layer 3. Highlighted in blue is the data transfer object (DTO) of layer 2 that is marshalled to JSON, and highlighted in orange are the annotations necessary for converting the SCADLeagueDto class from a regular Java object to a true DTO.

Layer 3 - Service Layer
The methods of layer 1 call upon classes of the service layer (e.g. SCADLeagueService) to perform business logic. These classes typically contain many injection points for other services and utilities.

SCADLeagueService.java

```
public class SCADLeagueService {

       @Inject
       YahooClientBuilder yahoo;

       @Inject
       SCADLeagueTeamService sltSvc;

       @Inject
       SCADLeaguePlayerService slpSvc;

       @Inject
       LeagueService lSvc;

       @Inject
       SCADLeagueDao slDao;

       public SCADLeagueDto getSCADLeague(Long id) throws RuntimeException {
              SCADLeagueDto result = null;
              SCADLeague slEntity = slDao.find(id);
              result = this.entityToDto(slEntity);
              return result;
       }
}
```

Figure 4.4.6b
Code snippet from SCADLeagueService.java illustrating layer 3. Highlighted in red is the code responsible for transitioning to layer 4. Note the presence of many injected services and utilities.

Layer 4 - Data Access Object Layer (DAO)
The service layer methods frequently require access to persisted SCAD entities, but rather than including direct access to the javax.persistence.EntityManager in the service layer, a suite of *data access object* (DAO) classes exists, forming layer 4. Typically, the only injected resource to a DAO is the JPA entity manager and most methods resemble SQL or HQL queries.

SCADLeagueDao.java

```
public class SCADLeagueDao {

       @Inject
       @ScadDatabase
       private EntityManager scadEm;
```

```
        public SCADLeague find(Long id) {
               return scadEm.find(SCADLeague.class, id);
        }
}
```

Figure 4.4.6c
Code snippet from SCADLeagueDao.java illustrating layer 4. Highlighted in red is the code responsible for transitioning to layer 5.

Layer 5 - Entity Layer
The DAO methods call on a javax.persistence.EntityManager instance to query SCAD's MySQL database and use its object relational mapper technology (e.g. Hibernate) to convert the relational representation of the entity to a populated Java object. This is layer 5, and the code implementing the layer 5 methods is close enough to the persisted data itself that it no longer belongs to SCAD.

EntityManager.java

```
package javax.persistence;

public interface EntityManager {
       <T> T find(Class<T> var1, Object var2)
}
```

Figure 4.4.6d
Code snippet from javax.persistence.EntityManager.java illustrating level 5, the entity level. Notice that methods at this deep level are not implemented by SCAD.

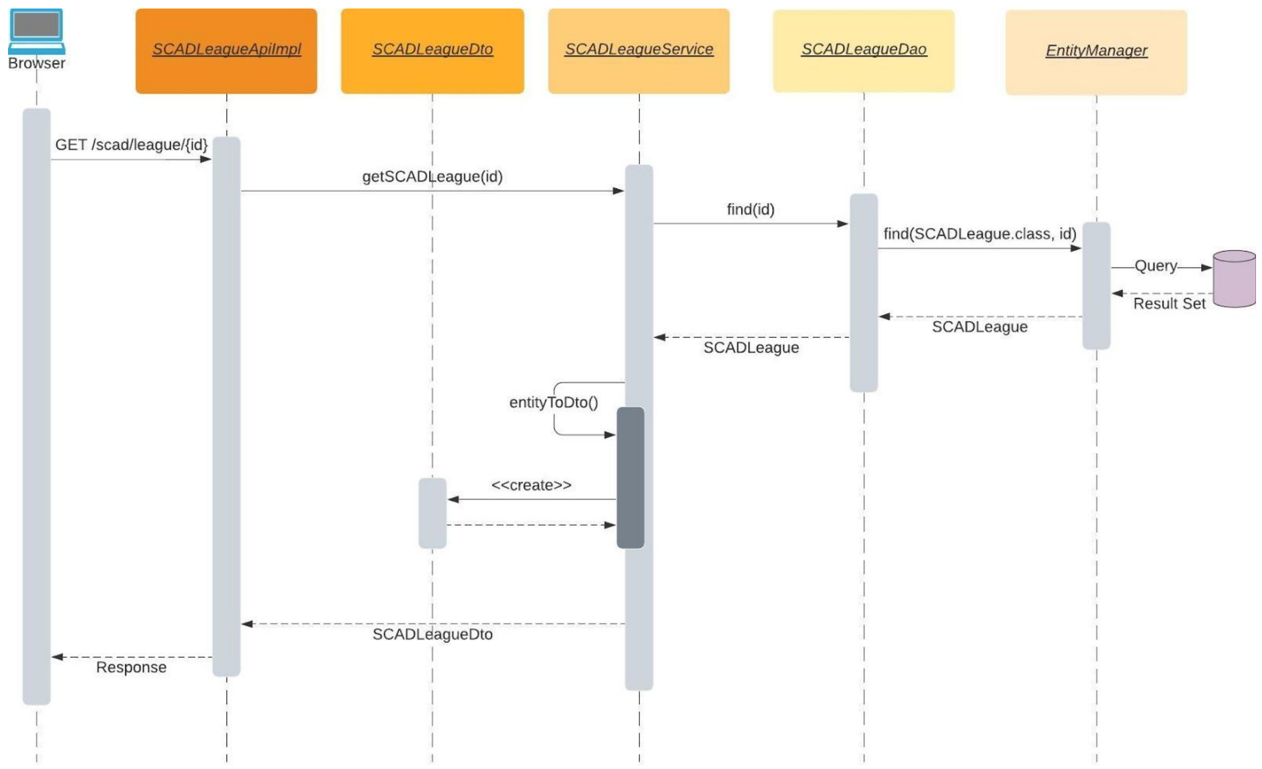


Figure 4.4.6e
A sequence diagram showing the flow of execution through the 5 logical layers of a SCAD's back-end REST service in the process of serving an HTTP request for a SCAD league object.

Although the majority of requests made from SCAD's front-end/UI to its back-end REST services traverse all five layers, this is not the case for requests made to SCAD for Yahoo-specific resources (e.g. /yahoo/league/{id}). For these requests, data is not persisted in SCAD's relational database, but rather at Yahoo. Therefore, the entity layer is replaced with a REST client layer.

Furthermore, since the JSON data is returned from Yahoo as a java.lang.String, the DAO and DTO layers are unnecessary and are replaced with JSON object "formatting" functions in the service layer that process and clean the raw data. One such function is shown below.

LeagueService.java

```
private String formatLeagueData(JsonObject leagueObj) throws RuntimeException {
  String result = null;
    if (Objects.nonNull(leagueObj)) {
       try {
           JsonElement error = leagueObj.get("error");
           if (Objects.isNull(error)) {
               JsonObject fantasyContent = leagueObj.get("fantasy_content").getAsJsonObject();
               JsonArray league = fantasyContent.get("league").getAsJsonArray();
               JsonObject ldata = league.get(0).getAsJsonObject();
               result = ldata.toString();
           } else {
               LOG.error("League object has error: {} ", error);
```

```
                result = "ERROR:" + error.toString();
            }
        } catch (Exception ex) {
            throw new RuntimeException(ex.getMessage());
        }
    }
    return result;
}
```

Figure 4.4.6f
A “formatting” method located in SCAD’s service layer used to clean raw JSON strings returned from Yahoo.

SCAD’s formatting functions make heavy use of Google’s GSON library (com.google.gson) with classes such as JsonArray, JsonElement, JsonObject, and JsonParser to manipulate the raw JSON data returned from Yahoo.

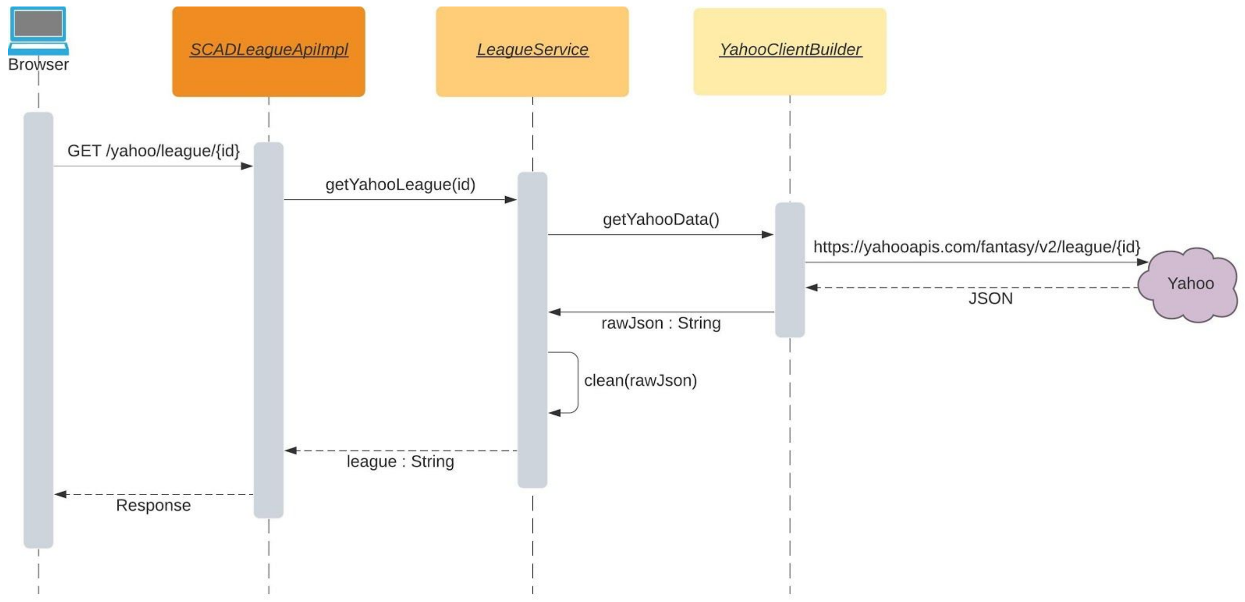


Figure 4.4.6g
A sequence diagram showing the process of serving a web request for a Yahoo league object. Note the absence of the DAO, DTO, and entity layers.

4.5 REST Services

4.5.1 Endpoints

SCAD version 1.0.0 contains a total of 45 REST endpoints. Their url mappings and descriptions are shown in the table below:

Dashboard:

Relative URL	**Description**
GET /dashboard/details	Returns dashboard details of SCAD UI.

User:

Relative URL	**Description**
GET /user	Returns the users details from

Email:

Relative URL	**Description**
POST /email/registered/{leagueId}	Sends an email to all registered users of a particular league.
POST /email/request/{leagueId}	Sends an email to the owner of a particular league.

Yahoo League:

Relative URL	**Description**
GET /yahoo/league/all	Returns all Yahoo leagues for the current season for the currently logged in user.
GET /yahoo/league/commissioner/all	Returns all Yahoo leagues of which the current user is a commissioner.
GET /yahoo/league/{leagueId}	Returns the specified Yahoo league.
GET /yahoo/league/{leagueId}/teams	Returns all teams associated with the specified league.
GET /yahoo/league/{leagueId}/myTeam	Returns the currently logged in user's team associated with the specific league.

GET /yahoo/league/{leagueId}/myPlayers	Returns all players associated with the current user for a specified league.
GET /yahoo/league/{leagueId}/settings	Returns the league settings for the specified league.
GET /yahoo/league/{leagueId}/standings	Returns the standings for the specified league.
GET /yahoo/league/{leagueId}/team/{teamId}/roster	Returns the team and roster details for the specified team and league.
GET /yahoo/league/{leagueId}/team/{teamId}/players	Returns the players of the specified team and league.
GET /yahoo/league/{leagueId}/players	Returns all players associated with the given league.

SCAD League:

Relative URL	**Description**
GET /scad/league/default	Returns the current season's default SCAD league for the current user.
PUT /scad/league/default/update/{id}	Updates the "default" indicator for a specified SCAD league.
GET /scad/league/all	Returns all current user's SCAD leagues for the present season.
GET /scad/league/{id}	Returns the specified SCAD league.
POST /scad/league/	Creates a new SCAD league resource.
PUT /scad/league/{id}	Updates the specified SCAD league.
DELETE /scad/league/{id}	Deletes the specified SCAD league.
GET /scad/league/yahoo/{leagueId}	Returns the SCAD league associated with the specified Yahoo league Id.

Players:

Relative URL	**Description**
GET /scad/player/{id}	Returns the specified SCAD player
GET /scad/league/{scadLeagueId}/player/all	Returns all SCAD players from the specified SCAD league.
GET /scad/league/{scadLeagueId}/player/myPlayers	Returns all SCAD players of current user for the specified SCAD league.
GET /scad/league/yahoo/{leagueId}/player/myPlayers	Returns all SCAD players of current user for the specified Yahoo league.
GET /scad/league/yahoo/{leagueId}/player/all	Returns all SCAD players for the specified Yahoo league.
GET /scad/league/{scadLeagueId}/player/{id}	Returns SCAD player for the specified SCAD league and player id.
GET /scad/league/yahoo/{leagueId}/player/{playerId}	Returns a SCAD player for the specified Yahoo league and player id.
GET /scad/league/{scadLeagueId}/team/{scadTeamId}/players	Returns all SCAD players for the specified SCAD league and team.

GET /scad/league/yahoo/{leagueId}/team/{teamId}/players	Returns all SCAD players for the specified Yahoo league and team.
POST /scad/player	Creates a new SCAD player.
PUT /scad/player/{id}	Updates a SCAD player.
DELETE /scad/player/{id}	Deletes a SCAD player.

Team:

Relative URL	**Description**
GET /scad/team/{id}	Returns a SCAD team for the specified Yahoo team id.
GET /scad/league/{scadLeagueId}/team/all	Returns all SCAD teams for the specified SCAD league id.
GET /scad/league/{scadLeagueId}/team/myTeam	Returns current user's SCAD team for specified SCAD league id.
GET /scad/league/yahoo/{leagueId}/team/myTeam	Returns current user's SCAD team for specified Yahoo league id.
GET /scad/league/yahoo/{leagueId}/team/all	Returns all SCAD teams for the specified Yahoo league id.
GET /scad/league/{scadLeagueId}/team/{id}	Returns a SCAD team for the specified SCAD league and team id.
GET /scad/league/yahoo/{leagueId}/team/{teamId}	Returns a SCAD team for the specified Yahoo league and team id.
POST /scad/team	Creates a SCAD team.
PUT /scad/team/{id}	Updates a SCAD team.
DELETE /scad/team/{id}	Deletes a SCAD team.

Table 4.5.1
Relative urls and descriptions of SCAD's 45 RESTful endpoints, organized by resource. The base url is https://<scad_top_level_domain>/scadservices/api.

4.5.2 Error and Exception Handling

Gracefully handling 400-level errors in a RESTful web service amounts to converting exceptions or error states into properly formatted HTTP responses. SCAD version 1.0.0 leaves much to be desired in this regard, returning 404 responses with generic error messages for most exceptional behavior.

The SCAD team's aspirational goals for error and exception handling in the next release are twofold.

First, return a 401-Unauthorized response early on in the request/response lifecycle if either the access-token or id_token header is missing. By convention, SCAD's front-end/UI includes an access_token and id_token header on every REST call, so if either header is null, this should be picked up immediately, even before the resource method is matched with the incoming HTTP request.

This can be accomplished with a class implementing javax.ws.rs.container.ContainerRequestFilter that is annotated with @PreMatching.

Second, use the standard control flow of exception handling to properly name and propagate exceptions up to the API implementation layer. At this layer, convert the exceptions into javax.ws.rs.core.Response objects with an appropriate 400-level status code (based on the exception type), and attach the error message from the exception to the JSON payload returned.

A sequence diagram for proper error and exception handling in SCAD version 1.1.0 is shown in figure 4.5.2 below.

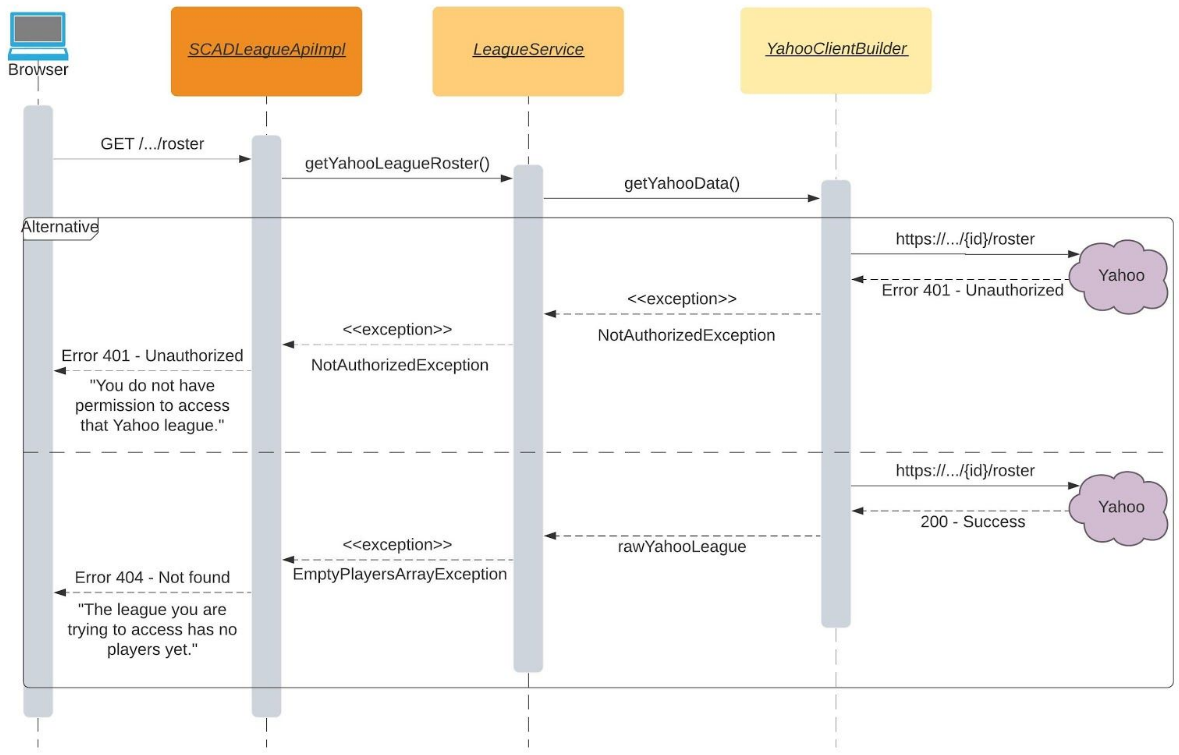


Figure 4.5.2
Sequence diagram illustrating how SCAD version 1.1.0 will convert exceptions into appropriate 400-level HTTP responses.

The JSON payload is shown in figure 4.5.2a.

```
{
  "error": {
    "lang": "en-US",
    "description": "The league you are trying to access has no players yet."
  }
}
```

Figure 4.5.2a
Sample JSON payload of the 404 response returned when an exception is thrown.

It is worth mentioning that malformed HTTP requests that cannot be routed to any API implementation method by the JEE runtime environment are handled by JEE. They return a 404 “Not Found” response with a simple error message such as “RESTEASY003210: Could not find resource for full path: http://localhost:8080/scadservices/api/ya” contained in the body.

4.5.3 Email

Sending emails to current and prospective SCAD users is an integral component of the application. For example, if a user wishes to register their Yahoo league with SCAD, but they are not a commissioner of their Yahoo league, they can click a button in SCAD’s UI to send an email to the commissioner requesting the registration of that particular Yahoo league with SCAD.

To support emailing, SCAD exposes two REST endpoints, “/mail/registered/{leagueId}” and “/mail/request/{leagueId}”, and implements these by including the javax.mail package and using a version of the basic adapter pattern shown below.

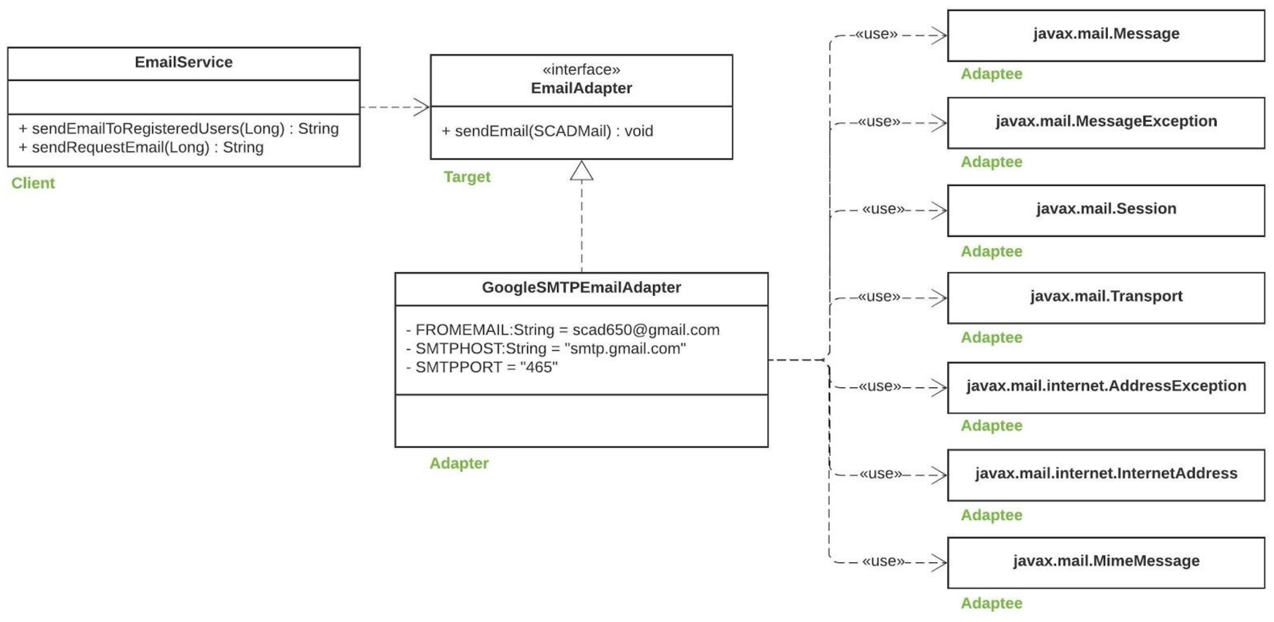

EmailService
+ sendEmailToRegisteredUsers(Long) : String
+ sendRequestEmail(Long) : String
Client
«interface»
EmailAdapter
+ sendEmail(SCADMail) : void
Target
GoogleSMTPEmailAdapter
- FROMEMAIL:String = scad650@gmail.com
- SMTPHOST:String = "smtp.gmail.com"
- SMTPPORT = "465"
Adapter
«use»
javax.mail.Message
javax.mail.MessageException
javax.mail.Session
javax.mail.Transport
javax.mail.internet.AddressException
javax.mail.internet.InternetAddress
javax.mail.MimeMessage
Adaptee

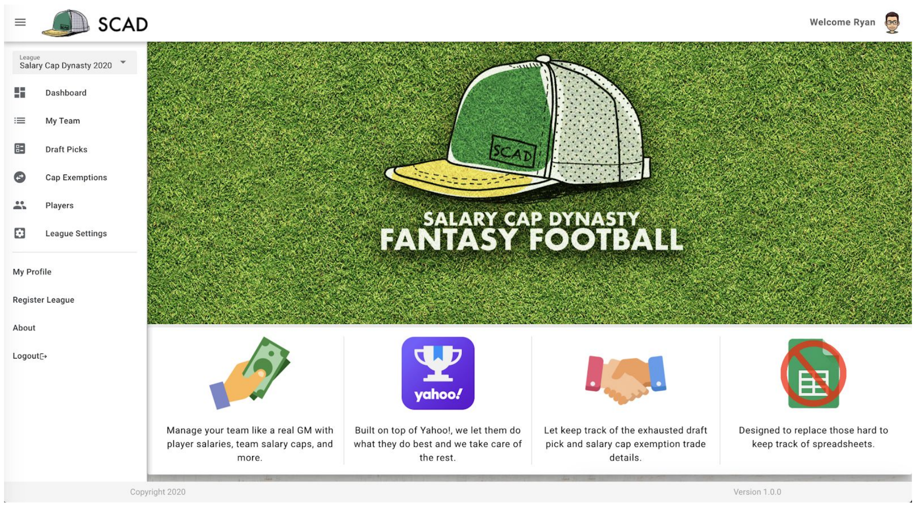
Figure 4.5.3
UML diagram showing the basic adapter pattern used to implement SCAD's email service.

We consider the adapter pattern to be warranted here, because it is likely that the component that sends emails will change in future versions. In fact, Yahoo itself contains an email API that is accessed via HTTP rather than SMTP, so using the adapter pattern will allow a seamless switchover by simply creating another appropriate adapter (e.g. YahooWebEmailAdapter) that implements the target interface EmailAdapter.

4.6 User Interface

4.6.1 Landing Page

Upon landing on SCAD's home page, a user is presented with the screen shown below. If the user is currently logged in (as shown in the figure), the drawer on the left hand side will be drawn to allow the user to navigate throughout the application. This page is also visible through the "About" tab.



Manage your team like a real GM with player salaries, team salary caps, and more.

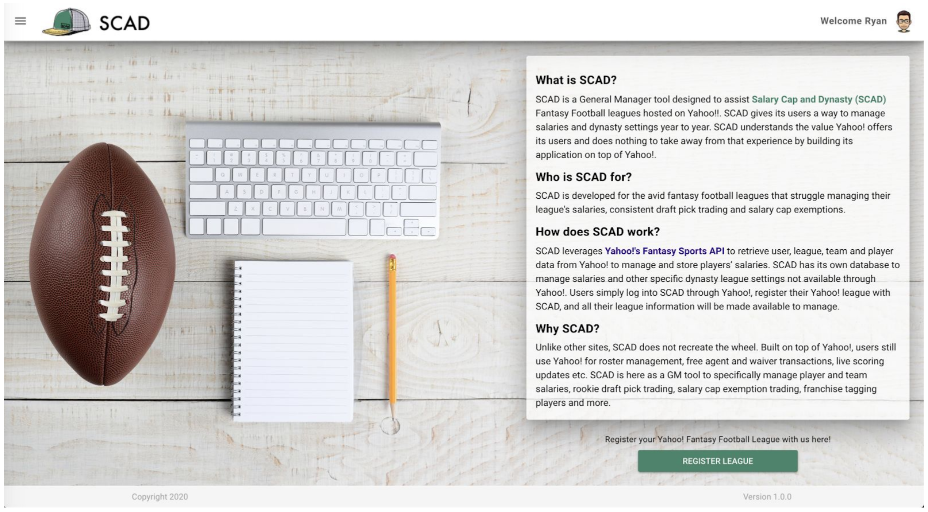
Built on top of Yahoo!, we let them do what they do best and we take care of the rest.

Let keep track of the exhausted draft pick and salary cap exemption trade details.

Designed to replace those hard to keep track of spreadsheets.

Figure 4.6.1
Screenshot of SCAD's landing page (first half).

The figure below shows the second half of the landing/about-us page. Here we have a brief description of SCAD and what it has to offer its users, along with a button that sends the user to the "Register League" page. If a user is not logged in, this button is replaced with a "Log In" button.



What is SCAD?

SCAD is a General Manager tool designed to assist Salary Cap and Dynasty (SCAD) Fantasy Football leagues hosted on Yahoo!!. SCAD gives its users a way to manage salaries and dynasty settings year to year. SCAD understands the value Yahoo! offers its users and does nothing to take away from that experience by building its application on top of Yahoo!.

Who is SCAD for?

SCAD is developed for the avid fantasy football leagues that struggle managing their league's salaries, consistent draft pick trading and salary cap exemptions.

How does SCAD work?

SCAD leverages Yahoo!'s Fantasy Sports API to retrieve user, league, team and player data from Yahoo! to manage and store players' salaries. SCAD has its own database to manage salaries and other specific dynasty league settings not available through Yahoo!. Users simply log into SCAD through Yahoo!, register their Yahoo! league with SCAD, and all their league information will be made available to manage.

Why SCAD?

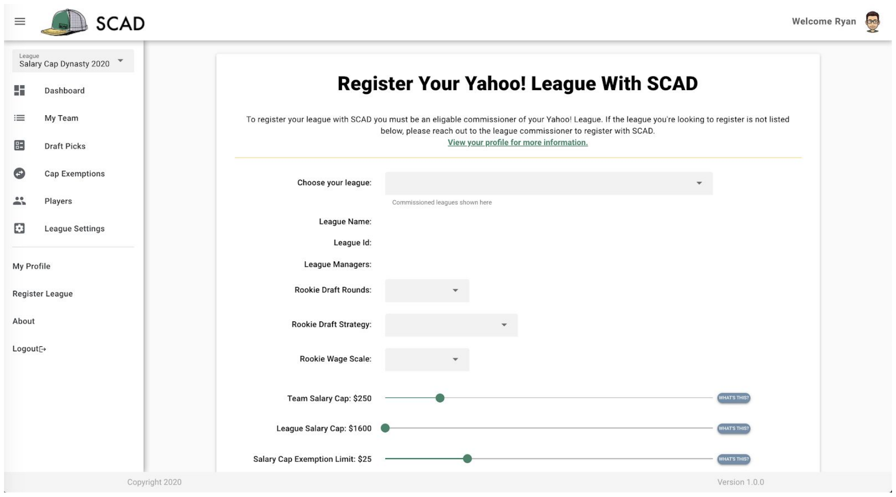
Unlike other sites, SCAD does not recreate the wheel. Built on top of Yahoo!, users still use Yahoo! for roster management, free agent and waiver transactions, live scoring updates etc. SCAD is here as a GM tool to specifically manage player and team salaries, rookie draft pick trading, salary cap exemption trading, franchise tagging players and more.

Register your Yahoo! Fantasy Football League with us here!

Figure 4.6.1a
Screenshot of SCAD's landing page (second half).

4.6.2 Register League Page

To register a league with SCAD, users must have an existing fantasy football league already registered with Yahoo. SCAD limits registration to only league commissioners of the Yahoo registered league. As the page loads, SCAD will return to the user a list of leagues they are a commissioner of. If leagues exist, this page will appear and the user can fill out the form to register with SCAD.



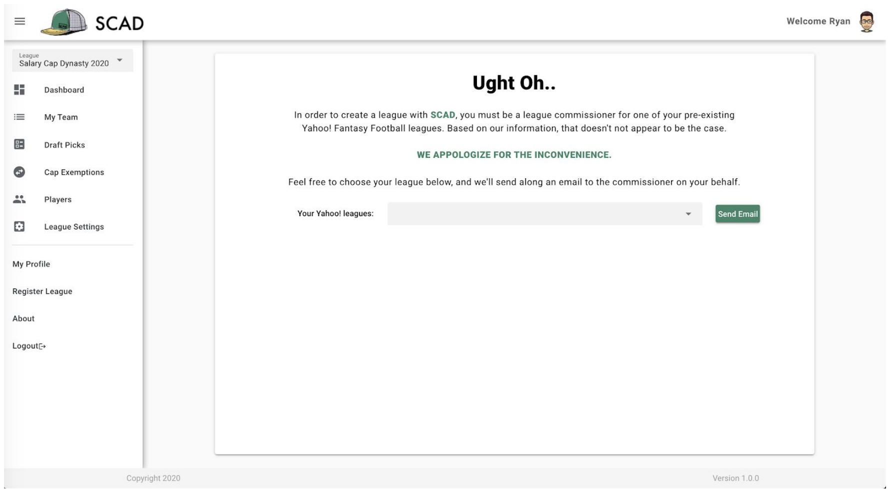
Register Your Yahoo! League With SCAD

To register your league with SCAD you must be an eligable commissioner of your Yahoo! League. If the league you're looking to register is not listed below, please reach out to the league commissioner to register with SCAD.
View your profile for more information.

Choose your league:
Commissioned leagues shown here
League Name:
League Id:
League Managers:
Rookie Draft Rounds:
Rookie Draft Strategy:
Rookie Wage Scale:
Team Salary Cap: $250
League Salary Cap: $1600
Salary Cap Exemption Limit: $25

Figure 4.6.2
Screenshot of SCAD's registration page.

If the user tries to register a league with SCAD, but they are not a commissioner of an existing Yahoo league, the page shown below is displayed. This page gives the user the ability to send an email to their league's commissioner to notify them of SCAD's services.



Ught Oh..

In order to create a league with SCAD, you must be a league commissioner for one of your pre-existing Yahoo! Fantasy Football leagues. Based on our information, that doesn't not appear to be the case.

WE APPOLOGIZE FOR THE INCONVENIENCE.

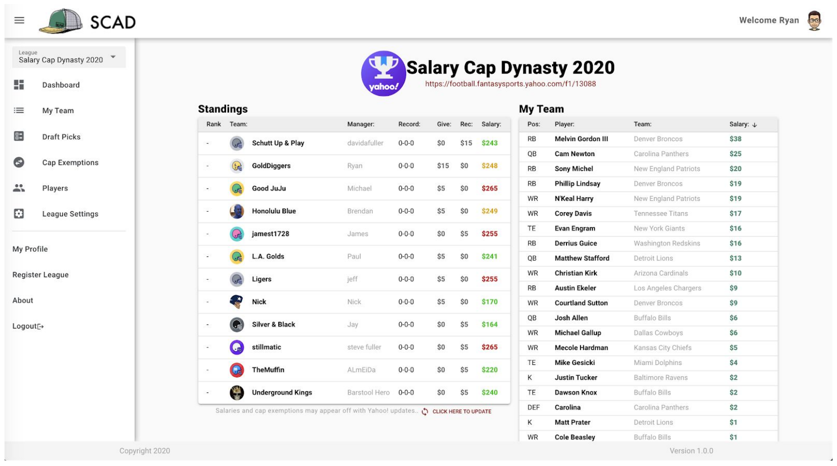
Feel free to choose your league below, and we'll send along an email to the commissioner on your behalf.

Your Yahoo! leagues:

Figure 4.6.2a
Screenshot of SCAD's registration page for non-commissioners.

4.6.3 Dashboard Page

The dashboard page is the landing spot once a league has been registered for a user. Here, SCAD retrieves Yahoo league and team details to display an overview to the user while injecting salary cap details for all teams and players. This page gives users an overview of all teams and their current stance against the salary cap.



Salary Cap Dynasty 2020
https://football.fantasysports.yahoo.com/f1/13088

Standings

Rank	Team:	Manager:	Record:	Give:	Rec:	Salary:
-	Schutt Up & Play	davidafuller	0-0-0	$0	$15	$243
-	GoldDiggers	Ryan	0-0-0	$15	$0	$248
-	Good JuJu	Michael	0-0-0	$5	$0	$265
-	Honolulu Blue	Brendan	0-0-0	$5	$0	$249
-	jamest1728	James	0-0-0	$0	$5	$255
-	L.A. Golds	Paul	0-0-0	$5	$0	$241
-	Ligers	jeff	0-0-0	$5	$0	$255
-	Nick	Nick	0-0-0	$5	$0	$170
-	Silver & Black	Jay	0-0-0	$0	$5	$164
-	stillmatic	steve fuller	0-0-0	$0	$5	$265
-	TheMuffin	ALmEiDa	0-0-0	$0	$5	$220
-	Underground Kings	Barstool Hero	0-0-0	$0	$5	$240

Salaries and cap exemptions may appear off with Yahoo! updates.. CLICK HERE TO UPDATE

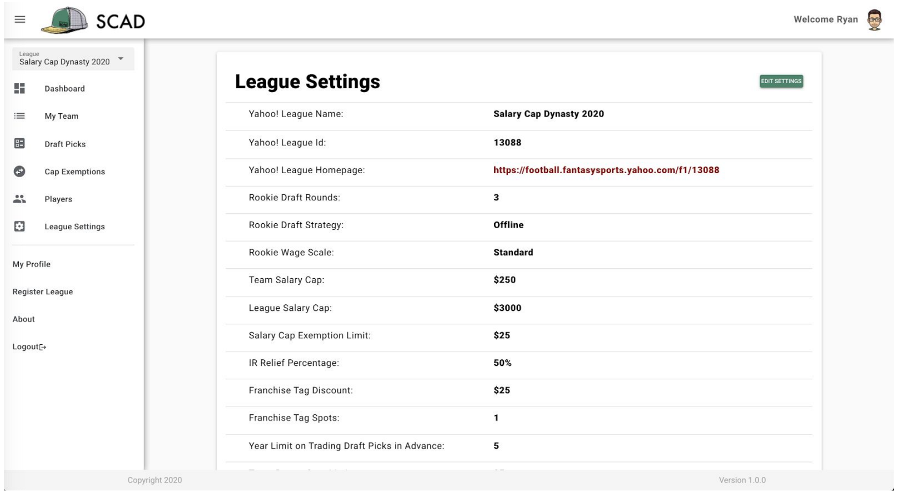
My Team

Pos:	Player:	Team:	Salary: ↓
RB	Melvin Gordon III	Denver Broncos	$38
QB	Cam Newton	Carolina Panthers	$25
RB	Sony Michel	New England Patriots	$20
RB	Phillip Lindsay	Denver Broncos	$19
WR	N'Keal Harry	New England Patriots	$19
WR	Corey Davis	Tennessee Titans	$17
TE	Evan Engram	New York Giants	$16
RB	Derrius Guice	Washington Redskins	$16
QB	Matthew Stafford	Detroit Lions	$13
WR	Christian Kirk	Arizona Cardinals	$10
RB	Austin Ekeler	Los Angeles Chargers	$9
WR	Courtland Sutton	Denver Broncos	$9
QB	Josh Allen	Buffalo Bills	$6
WR	Michael Gallup	Dallas Cowboys	$6
WR	Mecole Hardman	Kansas City Chiefs	$5
TE	Mike Gesicki	Miami Dolphins	$4
K	Justin Tucker	Baltimore Ravens	$2
TE	Dawson Knox	Buffalo Bills	$2
DEF	Carolina	Carolina Panthers	$2
K	Matt Prater	Detroit Lions	$1
WR	Cole Beasley	Buffalo Bills	$1

Figure 4.6.3
Screenshot of SCAD's dashboard page.

4.6.4 League Settings Page

Here a user can view the current SCAD settings for their league. If a user is the commissioner of the corresponding Yahoo league, they will have access to edit these settings.



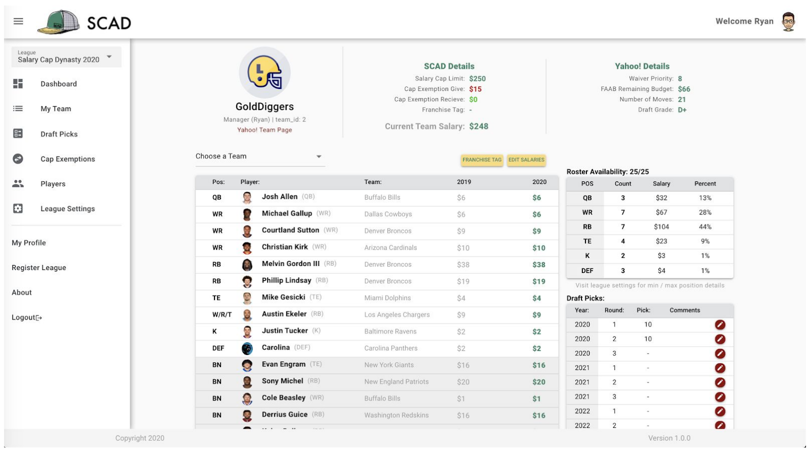
League Settings

Yahoo! League Name:	Salary Cap Dynasty 2020
Yahoo! League Id:	13088
Yahoo! League Homepage:	https://football.fantasysports.yahoo.com/f1/13088
Rookie Draft Rounds:	3
Rookie Draft Strategy:	Offline
Rookie Wage Scale:	Standard
Team Salary Cap:	$250
League Salary Cap:	$3000
Salary Cap Exemption Limit:	$25
IR Relief Percentage:	50%
Franchise Tag Discount:	$25
Franchise Tag Spots:	1
Year Limit on Trading Draft Picks in Advance:	5

Figure 4.6.4
Screenshot of SCAD's league settings page.

4.6.5 Team Page

The team page offers a complete overview of a team. SCAD retrieves Yahoo team details each time the page is called to ensure the information is up-to-date. Here, users are able to adjust a player's salary and apply a franchise tag to a player, both of which will update the overall team salary. This page also supplies a roster breakdown by position, and will notify the user if their roster is above or below roster restrictions set by SCAD settings. A list of the users upcoming draft picks and current cap exemptions are listed here as well.



GoldDiggers
Manager (Ryan) | team_id: 2
Yahoo! Team Page

SCAD Details

Salary Cap Limit:	$250
Cap Exemption Give:	$15
Cap Exemption Recieve:	$0
Franchise Tag:	-
Current Team Salary:	$248

Yahoo! Details

Waiver Priority:	8
FAAB Remaining Budget:	$66
Number of Moves:	21
Draft Grade:	D+

Pos:	Player:	Team:	2019	2020
QB	Josh Allen (QB)	Buffalo Bills	$6	$6
WR	Michael Gallup (WR)	Dallas Cowboys	$6	$6
WR	Courtland Sutton (WR)	Denver Broncos	$9	$9
WR	Christian Kirk (WR)	Arizona Cardinals	$10	$10
RB	Melvin Gordon III (RB)	Denver Broncos	$38	$38
RB	Phillip Lindsay (RB)	Denver Broncos	$19	$19
TE	Mike Gesicki (TE)	Miami Dolphins	$4	$4
W/R/T	Austin Ekeler (RB)	Los Angeles Chargers	$9	$9
K	Justin Tucker (K)	Baltimore Ravens	$2	$2
DEF	Carolina (DEF)	Carolina Panthers	$2	$2
BN	Evan Engram (TE)	New York Giants	$16	$16
BN	Sony Michel (RB)	New England Patriots	$20	$20
BN	Cole Beasley (WR)	Buffalo Bills	$1	$1
BN	Derrius Guice (RB)	Washington Redskins	$16	$16

Roster Availability: 25/25

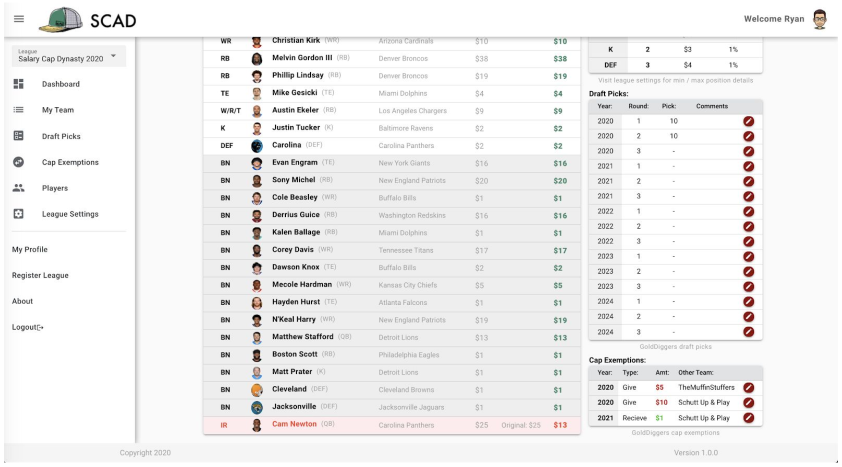
POS	Count	Salary	Percent
QB	3	$32	13%
WR	7	$67	28%
RB	7	$104	44%
TE	4	$23	9%
K	2	$3	1%
DEF	3	$4	1%

Visit league settings for min / max position details

Draft Picks:

Year:	Round:	Pick:	Comments
2020	1	10	
2020	2	10	
2020	3	-	
2021	1	-	
2021	2	-	
2021	3	-	
2022	1	-	
2022	2	-	

Figure 4.6.5
Screenshot of SCAD’s team page (top half)



WR	Christian Kirk (WR)	Arizona Cardinals	$10		$10
RB	Melvin Gordon III (RB)	Denver Broncos	$38		$38
RB	Phillip Lindsay (RB)	Denver Broncos	$19		$19
TE	Mike Gesicki (TE)	Miami Dolphins	$4		$4
W/R/T	Austin Ekeler (RB)	Los Angeles Chargers	$9		$9
K	Justin Tucker (K)	Baltimore Ravens	$2		$2
DEF	Carolina (DEF)	Carolina Panthers	$2		$2
BN	Evan Engram (TE)	New York Giants	$16		$16
BN	Sony Michel (RB)	New England Patriots	$20		$20
BN	Cole Beasley (WR)	Buffalo Bills	$1		$1
BN	Derrius Guice (RB)	Washington Redskins	$16		$16
BN	Kalen Ballage (RB)	Miami Dolphins	$1		$1
BN	Corey Davis (WR)	Tennessee Titans	$17		$17
BN	Dawson Knox (TE)	Buffalo Bills	$2		$2
BN	Mecole Hardman (WR)	Kansas City Chiefs	$5		$5
BN	Hayden Hurst (TE)	Atlanta Falcons	$1		$1
BN	N'Keal Harry (WR)	New England Patriots	$19		$19
BN	Matthew Stafford (QB)	Detroit Lions	$13		$13
BN	Boston Scott (RB)	Philadelphia Eagles	$1		$1
BN	Matt Prater (K)	Detroit Lions	$1		$1
BN	Cleveland (DEF)	Cleveland Browns	$1		$1
BN	Jacksonville (DEF)	Jacksonville Jaguars	$1		$1
IR	Cam Newton (QB)	Carolina Panthers	$25	Original: $25	$13

K	2	$3	1%
DEF	3	$4	1%

Visit league settings for min / max position details

Draft Picks:

Year:	Round:	Pick:	Comments
2020	1	10	
2020	2	10	
2020	3	-	
2021	1	-	
2021	2	-	
2021	3	-	
2022	1	-	
2022	2	-	
2022	3	-	
2023	1	-	
2023	2	-	
2023	3	-	
2024	1	-	
2024	2	-	
2024	3	-	

GoldDiggers draft picks

Cap Exemptions:

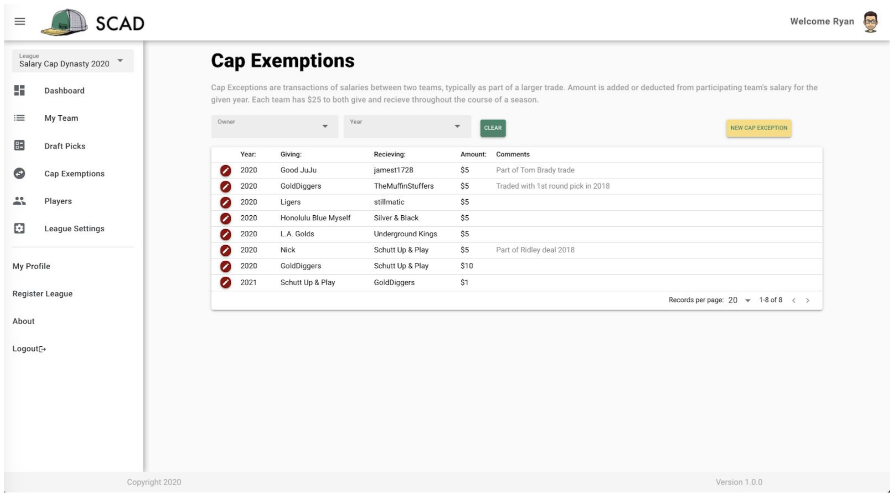
Year:	Type:	Amt:	Other Team:
2020	Give	$5	TheMuffinStuffers
2020	Give	$10	Schutt Up & Play
2021	Recieve	$1	Schutt Up & Play

GoldDiggers cap exemptions

Figure 4.6.5a
Screenshot of SCAD’s team page (bottom half).

4.6.6 Cap Exemptions Page

Cap exemptions are salary transactions between two teams, typically as part of a larger trade. These are often executed if a team's salary is above the salary cap. One team will trade another team a dollar amount, which is applied to each team's current salary. The giving team will have their overall salary increased, and the receiving team will have their overall salary decreased by the agreed upon amount.



Cap Exemptions

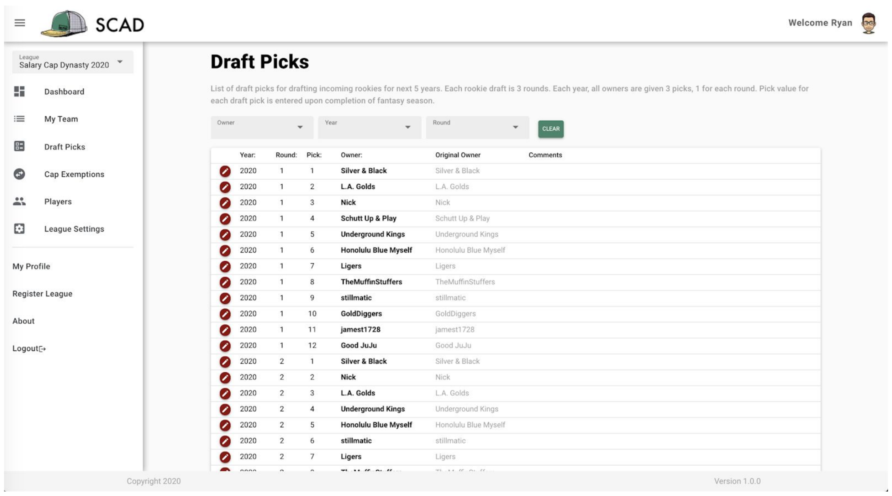
Cap Exceptions are transactions of salaries between two teams, typically as part of a larger trade. Amount is added or deducted from participating team's salary for the given year. Each team has $25 to both give and recieve throughout the course of a season.

Year:	Giving:	Recieving:	Amount:	Comments
2020	Good JuJu	jamest1728	$5	Part of Tom Brady trade
2020	GoldDiggers	TheMuffinStuffers	$5	Traded with 1st round pick in 2018
2020	Ligers	stillmatic	$5	
2020	Honolulu Blue Myself	Silver & Black	$5	
2020	L.A. Golds	Underground Kings	$5	
2020	Nick	Schutt Up & Play	$5	Part of Ridley deal 2018
2020	GoldDiggers	Schutt Up & Play	$10	
2021	Schutt Up & Play	GoldDiggers	$1	

Figure 4.6.6
Screenshot of SCAD's cap exemption page.

4.6.7 Draft Picks Page

Dynasty leagues are designed to allow users to keep their entire roster year after year. Subsequently, a draft at the beginning of each year consists of the new players entering the league. Each user gets one draft pick per round, but these picks are expendable. Teams often trade draft picks as part of a larger deal, or reposition themselves in the draft to get a player they strongly desire. This page is designed to track and manage those draft picks. The edit button in the first column of the table allows the user to edit the current owner of the draft pick if a pick is traded to another owner.



Draft Picks

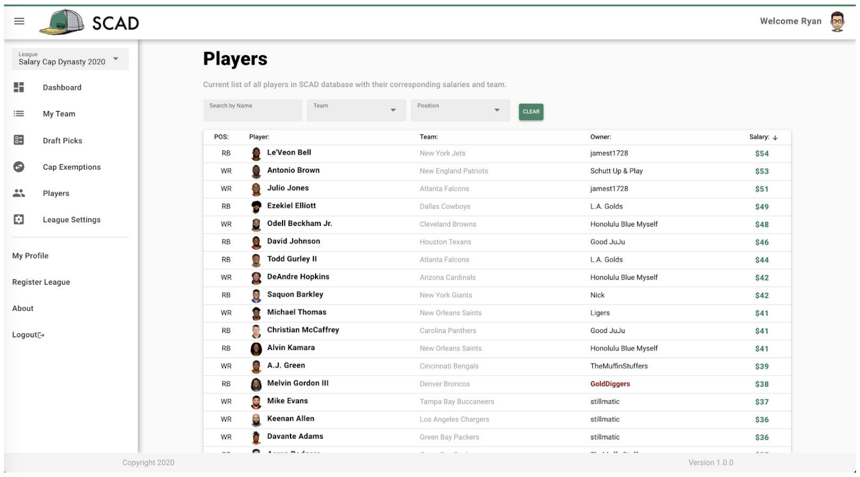
List of draft picks for drafting incoming rookies for next 5 years. Each rookie draft is 3 rounds. Each year, all owners are given 3 picks, 1 for each round. Pick value for each draft pick is entered upon completion of fantasy season.

Year:	Round:	Pick:	Owner:	Original Owner	Comments
2020	1	1	Silver & Black	Silver & Black	
2020	1	2	L.A. Golds	L.A. Golds	
2020	1	3	Nick	Nick	
2020	1	4	Schutt Up & Play	Schutt Up & Play	
2020	1	5	Underground Kings	Underground Kings	
2020	1	6	Honolulu Blue Myself	Honolulu Blue Myself	
2020	1	7	Ligers	Ligers	
2020	1	8	TheMuffinStuffers	TheMuffinStuffers	
2020	1	9	stillmatic	stillmatic	
2020	1	10	GoldDiggers	GoldDiggers	
2020	1	11	jamest1728	jamest1728	
2020	1	12	Good JuJu	Good JuJu	
2020	2	1	Silver & Black	Silver & Black	
2020	2	2	Nick	Nick	
2020	2	3	L.A. Golds	L.A. Golds	
2020	2	4	Underground Kings	Underground Kings	
2020	2	5	Honolulu Blue Myself	Honolulu Blue Myself	
2020	2	6	stillmatic	stillmatic	
2020	2	7	Ligers	Ligers	

Figure 4.6.7
Screenshot of SCAD's draft picks page.

4.6.8 Players Page

The "Players Page" displays a table of all the players in SCAD's database with their corresponding salaries. Users can not edit or make changes to any SCAD information from this page. It is simply a tool for users to view and research existing players and how their salaries stack up to others' across the league.



Players

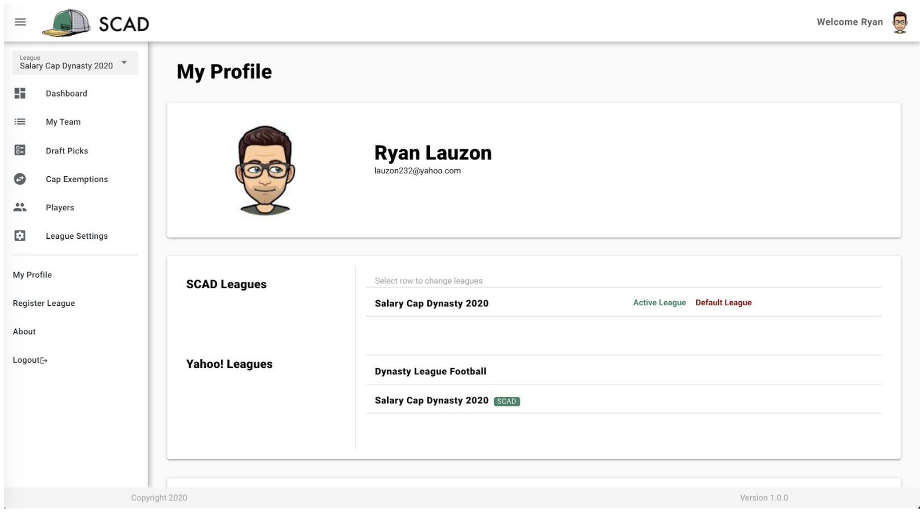
Current list of all players in SCAD database with their corresponding salaries and team.

POS:	Player:	Team:	Owner:	Salary: ↓
RB	Le'Veon Bell	New York Jets	jamest1728	$54
WR	Antonio Brown	New England Patriots	Schutt Up & Play	$53
WR	Julio Jones	Atlanta Falcons	jamest1728	$51
RB	Ezekiel Elliott	Dallas Cowboys	L.A. Golds	$49
WR	Odell Beckham Jr.	Cleveland Browns	Honolulu Blue Myself	$48
RB	David Johnson	Houston Texans	Good JuJu	$46
RB	Todd Gurley II	Atlanta Falcons	L.A. Golds	$44
WR	DeAndre Hopkins	Arizona Cardinals	Honolulu Blue Myself	$42
RB	Saquon Barkley	New York Giants	Nick	$42
WR	Michael Thomas	New Orleans Saints	Ligers	$41
RB	Christian McCaffrey	Carolina Panthers	Good JuJu	$41
RB	Alvin Kamara	New Orleans Saints	Honolulu Blue Myself	$41
WR	A.J. Green	Cincinnati Bengals	TheMuffinStuffers	$39
RB	Melvin Gordon III	Denver Broncos	GoldDiggers	$38
WR	Mike Evans	Tampa Bay Buccaneers	stillmatic	$37
WR	Keenan Allen	Los Angeles Chargers	stillmatic	$36
WR	Davante Adams	Green Bay Packers	stillmatic	$36

Figure 4.6.8
Screenshot of SCAD's players page.

4.6.9 My Profile Page

The “My Profile” page is a very basic profile page displaying the users currently registered SCAD and Yahoo! leagues. A user with multiple SCAD leagues can come to this page to switch between leagues, and to select a new default league. A default league is the league SCAD pulls first when the page is loaded. By selecting on a Yahoo! league, SCAD will redirect the user to their Yahoo! homepage. If the user is a commissioner, they will be presented with a button to register that league with SCAD.



My Profile

Ryan Lauzon
lauzon232@yahoo.com

SCAD Leagues

Select row to change leagues

Salary Cap Dynasty 2020 Active League Default League

Yahoo! Leagues

Dynasty League Football

Salary Cap Dynasty 2020 SCAD

Figure 4.6.9
Screenshot of SCAD’s profile page.

5. Conclusion

We conclude this paper with some final reflections about our journey building SCAD from concept to delivery, as well as the software development process writ large.

5.1 Technical

On the technical side, we find that using a microservice-based architecture is something to be done out of necessity rather than intrigue. We are thrilled with the experience gained in building SCAD with a microservice-based architecture but, admittedly, have our doubts about its overall utility compared to its monolithic counterpart. We find this particularly true for a team of only three developers.

We also find that OAuth2 is much more difficult to understand and implement than it appears at first glance. However, there is a certain satisfaction that now comes to us in having a basic understanding of what is going on under the hood when we visit a website and see a link saying, “Login with Google or Facebook”.

Finally, there is a reason the front-end/UI developer community is flocking to Vue.js. We find it to be an extremely powerful technology, and are delighted at our choice to use it to build SCAD over more robust alternatives such as Angular.js and React.js.

5.2 Human

On the human side, we must first acknowledge the immense challenge of building software in a team composed of developers with varying levels of proficiency. The harsh reality is that one experienced developer may be able to produce 10 to 100 times more code than a less experienced developer. We find this aspect of software development to be fundamentally different than, say, bricklaying, where an experienced bricklayer may be more productive, but not by such magnitudes.

Consequently, maintaining interest, morale, and agency among the team members is an important component of the project. We find carving out niches (e.g. front-end/UI, back-end, documentation, etc.) in SCAD for each team member to be particularly helpful in this regard. Few things are more deflating to a developer than completing a feature only to find out later that it has already been completed by a fellow developer.

As for our team, we are utterly thrilled with the human component of our experience building SCAD. We feel extremely fortunate to have had the opportunity to work with one another, and as the project progressed from the initial whiteboard meeting to the hundreds of hours of code writing, we admit that our primary motivation for keeping our noses to the grindstone slowly changed from a desire to graduate from CCSU, to a desire not to let each other down.

6. References

1. Coward, Danny Dr. *Java EE 7: The Big Picture*. Oracle Press, McGraw Hill, 2015.
2. Gupta, Arun *Java EE 7 Essentials.* O'Reilly Publications Inc., 2015.
3. Neward, Ted *Effective Enterprise Java*. Addison-Wesley, 2004.
4. Macrae, Callum *Vue.js Up & Running: Building Accessible and Performant Web Apps.* O'Reilly Media, Inc. 2018.
5. Fowler, Martin. (2014, Dec). *Microservices*. Presented at GOTO 2014. Berlin, Germany.
6. Barberttini, Nate. (2018, Feb). *OAuth 2.0 and OpenID Connect (in Plain English)*. Presented at OktaDev. San Francisco, CA.
7. Sablonniere, Hubert. (2016, Nov). *100% Stateless with JWT (JSON Web Token)*. Presented at Devoxx. Antwerp, Belgium.