Software Engineering Assignment 3
CS400-01
Software Design and Development
Fall 2021

SportsLink: Developing a Software Architecture

Claire Roeder, Divya Manirajan, Cameron Herbert

Introduction:

SportsLink is an app designed to help people create and find sports pickup games and other activities in their community. The app has several features that can be implemented through various components. These components can be broken up into two categories: microservices and database functions. We decided a multi-tier architecture with a web server, application server, and database server, is the best type of architecture for this product. These servers will need to communicate with each other since the data they share must be consistent and the different servers will need to access information from all others. The technologies we will use include programming languages such as JavaScrip, CSS, and PHP, and database technology such as MySQL and Amazon RDS.

How should the system be organized as a set of architectural components, where each of these components provides a subset of the overall system functionality?

- Overall components:
    - GUI to navigate through app to access different features
    - Mobile will be touch-screen and Web will be mouse and keyboard
- Each feature will have an architectural design that most appropriately suits its purposes
    - *Create Event*
        - Purpose:
            - Form to fill out with: what event, who can join, when, and where
        - Component:
            - **Microservice -** form/survey service
            - Form that inputs data and outputs a calendar event
            - Send data to a database to keep track of different activities, dates, etc.
    - *Search*
        - Purpose:
            - Input name of activity and output a list of events related to that activity with an extra option to create if none are shown

        - Component:
            - Database query to search through events
                - Use query to show events in a calendar form
                - Use query to show a list of events which links to a calendar entry
                - If no results then there is an option to create a new one
    - *Filtering*
        - Purpose:
            - Events are narrowed to only those that satisfy filters
        - Component:
            - Checkbox + fill in the blank system for selecting filters
            - When searching for events, filters will narrow down the database with the given parameters
            - EX: age = "24-34" → query only searches for age appropriate events
    - *User Profile*
        - Purpose:
            - Profile is created where personal information is protected
        - Component:
            - **Microservice** - log-in service
            - Log-in information creates profiles which are needed to use the app
            - App can redirect user to a microservice that handles log-in credentials
    - *Roster*
        - Purpose:
            - The specified event will display a roster of all users that are signed up for the event
        - Component:
            - Database has a list of people in specific events
            - Query will return this list when users want a roster for an event
    - *Reporting*
        - Purpose:
            - An official form that details a malicious event is filled out by users and submitted to "officials" to be reviewed
        - Component:
            - **Microservice** - form/survey service
            - Form is filled by users and sent to "officials" for review
    - *Feed*
        - Purpose:

            - Events are added to the feed based on how soon they are going to take place
        - Component:
            - Database query that gives all events during a specified time range
            - Time range must be inputted by the user to narrow down the query
            - Feed button → input time range → output list
    - *Cancel/Withdrawal*
        - Purpose:
            - The user will be removed from a specified event and the roster will be updated
        - Component:
            - Connected to database so it can communicate with the roster
            - When a user cancels an event, the roster for that event is updated through the database
            - (Possible notifications through emails - user profiles may include email addresses for each person which can be stored in database)
    - *Tournament Creation*
        - Purpose:
            - A tournament can be created where other players can sign up to compete
        - Component:
            - **Microservice -** form/survey service
            - The architecture will be identical to that of the create event feature
                - Form that inputs data and outputs a calendar event
                - Send data to a database to keep track of different activities, dates, etc.
            - Difference is the output will be shown as a tournament instead of an event
    - *Ranked/Casual*
        - Purpose:
            - The events will be narrowed to either ranked or casual depending on what is selected
        - Component:
            - Similar to filter feature
            - Checkbox for ranked or casual modes
            - Narrows query to only event with the selected mode
    - *Coaching*
        - Purpose:
            - Users will be able to watch instructional videos uploaded by approved or verified users.

        - Component:
            - **Microservice -** video sharing service (ex. kaltura or youtube)
            - Only approved or verified users are allowed to upload
                - Could be a set up where approved users have certain privileges to share but other users can only watch
                - Connected to database which has information about if a user has certain “coaching” privileges
            - All users can watch videos
    - *Output Calendar*
        - Purpose:
            - A calendar that shows all upcoming events and where they take place
        - Component:
            - **Microservice -** outlook calendar service
            - Connected to database
            - Shows database events in calendar form

The system should be organized with six microservices and two database servers. The microservices will be for the create event, user profile, reporting, tournament creation, coaching, and output calendar features. These microservices will all serve different functions depending on which feature they correspond with. Some of these services may need to communicate with the database servers.

The database servers will be for the search, filter, roster, feed, ranked vs casual, and cancel/withdrawal features. Two database tables will be needed, one to keep track of user information and another to keep track of event information. The user table will include information such as: userID, name, log-in username, email, approved/verified. The event table will include information such as: what the event is, who can participate, when it takes place, and where the event will take place.

How should these architectural components be distributed and communicate with each other?

These components may work best in a client-server distribution. A multi-tier architecture of this system would include a web server, an application server, and a database server. The web server can deliver the feed as well as any real-time or social features of the app. The application server can handle the functionality of the app and include things like searching, creating events, etc. Finally, the database server can manage the two database tables for user and event information

It might also be useful to group components into modules. Some examples of this would be:

- An event module that manages components such as Event Creation, Tournament Creation.
- An information module that manages Searching, Filtering, Ranked/Casual.

All components will need to be able to communicate with the databases so that they can retrieve and update information. The components will need to communicate with each other in several ways, for example:

- Event Creation will need to communicate with the Output Calendar.
- Event Creation will need to communicate with the roster, which will also communicate with user profiles.
- Cancel/Withdrawal will need to communicate with the roster.
- Filtering Component is **part-of** the Search Component
- Feed Component **shares-data-with** the Output Calendar

What technologies should you use in building the system and what components should be reused?

- Technologies
    - Programming languages
        - Java Script (with jQuery framework)
            - Front end language
            - For all user interactions with the application such as clicking a button or searching
            - For sending information to the back end such as user profiles and events created by users
        - CSS (with Bootstrap framework)
            - Front end language
            - For the presentation of our application
        - PHP
            - Back end language
            - For retrieving and sending information to and from our database
            - For retrieving and sending information to and from any API's that might be included in the future
    - Database technology
        - MySQL
            - For storing user profile data and event data
        - Amazon RDS
            - For hosting our MySQL database in the cloud