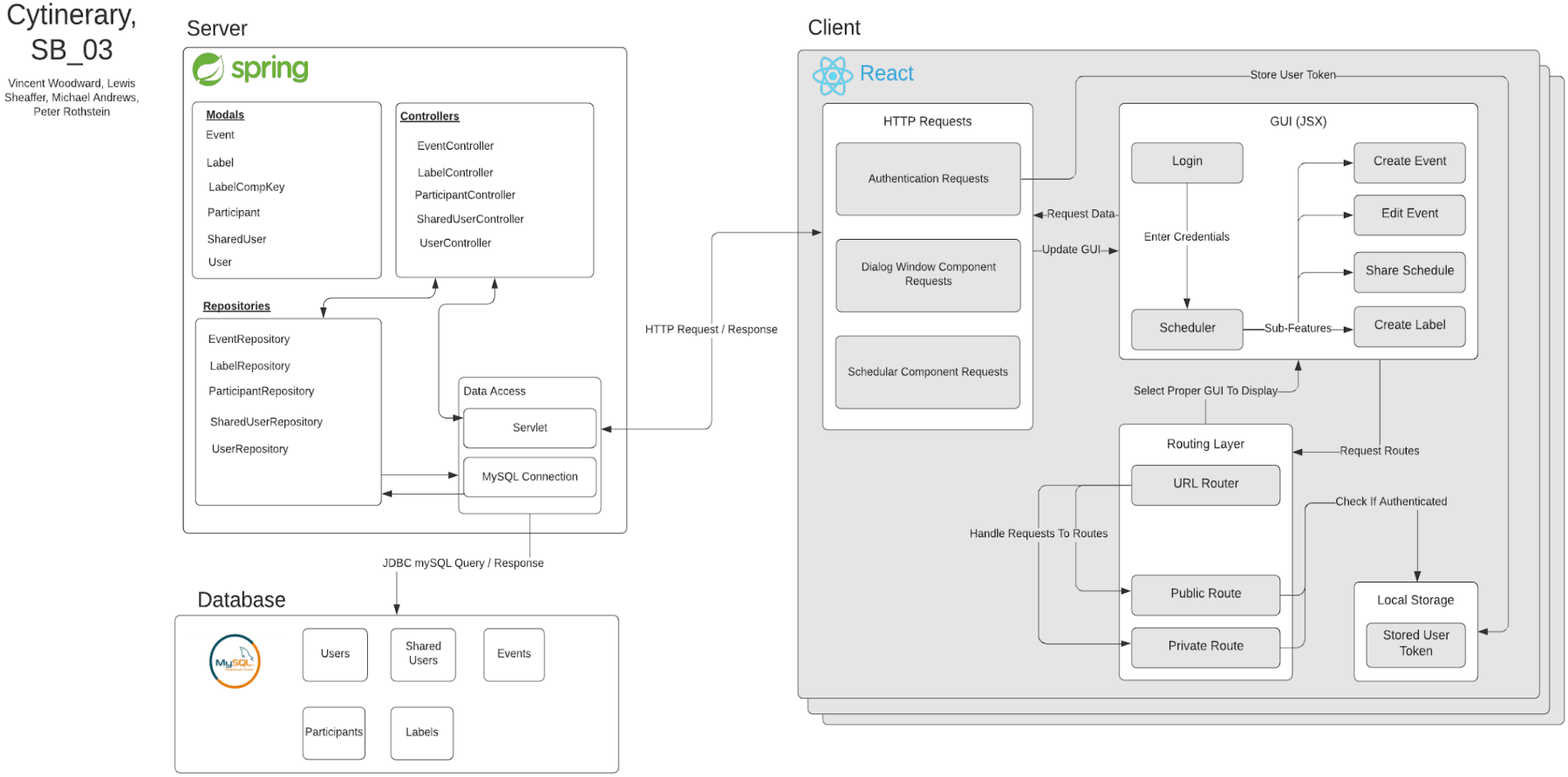
Block Diagram
SB_03

Peter Rothstein
Michael Andrews
Lewis Sheaffer
Vincent Woodward

Cytinerary, SB_03

Vincent Woodward, Lewis Sheaffer, Michael Andrews, Peter Rothstein



Design Description:

Server

Our backend is made with the Springboot framework. There are three main categories of classes for the backend: modals, controllers, and repositories. These classes interact with the MySQL database with native queries to retrieve and store data in the MySQL database. Each of these classes or "objects" represent different data stored within the database. The requests are first handled by the controllers with their respective mapping. The request then is processed by the respective repository where the native queries are then used to retrieve or store data to the MySQL database. For our project, querying data from JDBC MySQL Query that will allow the java objects to alter within the database. Any retrieved query data will be returned by the specific repository method. For controller methods with mappings that require a response (Post, Get, ...), the appropriate value will be returned.

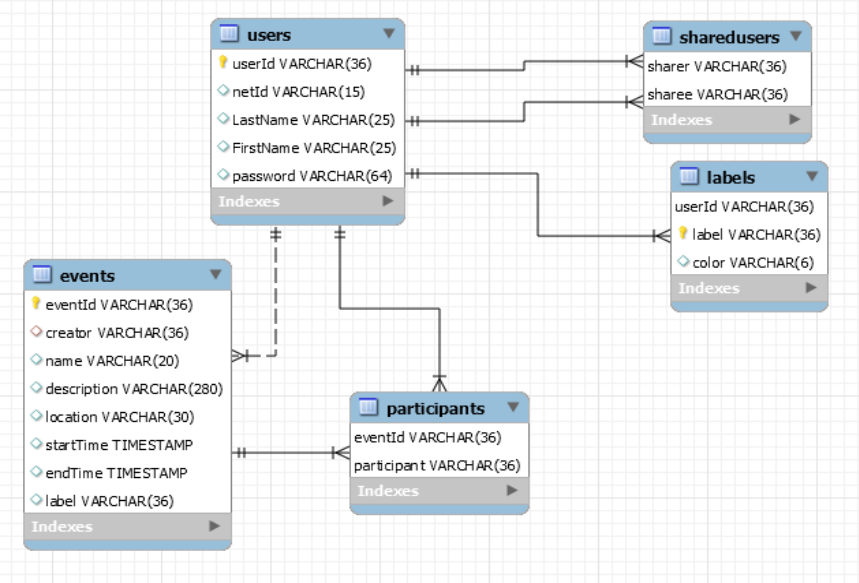
Client

Our client side consists of a TypeScript React application. The app contains multiple screen jsx components: LoginScreen, CreateAccountScreen, and the MyScheduleScreen.
In the projects app.tsx file, we use react router to correctly navigate between the screens. When logged in, the user will be routed to the private route component, MyScheduleScreen, regardless of the entered url. When not logged in, the user will only be able to navigate between the public routes components: CreateAccountScreen and the LoginScreen.
In the LoginScreen, when the login button is clicked an http post request will be sent to the server to authenticate the entered credentials. If these credentials prove to be consistent with those in the database, the user id will be returned in a response, and the user's userId credential will be stored in the browser using localstorage.
The MyScheduleScreen is the main homepage of this application. It contains the schedular component, as well as additional dialog modal components for interacting with user event data. Given the logged in userId credential, when rendered, the scheduler component will send a http post request to the Spring Boot server to retrieve all of the events specific to that userId in the form of an array of JSON objects. That information is parsed and mapped onto the scheduler in the form of timeblocks components. Information specific to the event such as position and eventId is mapped to the timeblock through its props.

Database

Our MySQL database consists of several tables primarily associated with Cytinerary users and the various pieces of personal/shared data associated with these users. The users table contains entries for user data such as user ID, ISU net-ID, name and password. Users can create events that are stored in the events table. The events table contains entries for data such as event ID, the creator of the event (a user in the user table), event name, event description, event label (a label in the label table), etc. The labels table contains entries for data such as user ID (the owner/creator of a label found in the user table), the label string, and the label color (for use in the GUI). Users can share events with other Cytinerary users. There exists a participants table that contains foreign keys that reference an event in the events table and an associated user in the user table that did not create the associated event. Users can also share their entire schedules with other users. There exists a shared users table that contains foreign keys that reference two users (the schedule sharer and the schedule sharee).

Table fields and entries



Tables:

users

- userId (Primary Key)
- netId
- LastName
- FirstName
- password

events (Primary Key: eventId)

- eventId
- creator (Foreign Key (userId) from users)
- name
- description
- location
- startTime
- endTime
- label

participants (Primary Key: eventId, participant)

- eventId (Foreign Key (eventId) from events)
- participant (Foreign Key (userId) from users)

labels (Primary Key: userId, label)

- userId (Foreign Key (userId) from users)
- label
- color

sharedusers (Primary Key: sharer, sharee)

- sharer (Foreign Key (userId) from users)
- sharee (Foreign Key (userId) from users)