Portal Chess Strike Team Java

xGame Intro

- Server running from Spring Boot
 - Java framework that allows a Client to communicate with a Java server with API Calls
 - This server is then automatically configured to communicate with a specific SQL database, and can query to update or obtain information.
- Client- running on React.js
 - Easy to build and use UI components, communicates with server with API request/response. Render changes with state components.
- Database- MySQL
 - Currently have an accounts, notifications, and matches table to store data. Getting and setting data is all done with Spring's data access object classes.

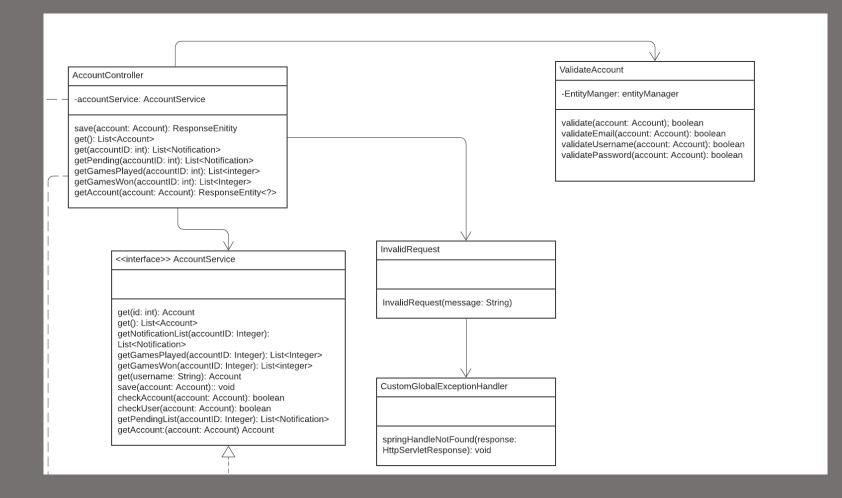
Design decisions

- Root directory is login page
- Upon logging in, user is taken to a dashboard
- User search updates in real time
 - User's profile is rendered over the page in a modal
 - future game-related statistics and match history will be displayed here
- Each page has a header with links
 - One link back to home page
 - Logged in: user's account dropdown
 - Logged out: Login/Register buttons

Design decisions

- Invitations will be sent exclusively from User Search page
- Inbox displays whether sent invitations are rejected or accepted
- User's active games and pending invitations will be displayed on Dashboard

Updated Class Diagram



AccountServiceImplementation

-accountDAO: AccountDAO

get(id: int): Account get(): List<Account>

getNotificationList(accountID: Integer): List<Notification> getGamesPlayed(accountID: Integer): List<Integer> getGamesWon(accountID: Integer): List<integer>

get(username: String): Account save(account: Account):: void

checkAccount(account: Account): boolean checkUser(account: Account): boolean

getPendingList(accountID: Integer): List<Notification>

getAccount:(account: Account) Account

<<interface>> AccountDAO

get(id: int): Account get(): List<Account>

getNotificationList(accountID: Integer): List<Notification> getGamesPlayed(accountID: Integer): List<Integer> getGamesWon(accountID: Integer): List<integer>

get(username: String): Account save(account: Account):: void

checkAccount(account: Account): boolean checkUser(account: Account): boolean

getPendingList(accountID: Integer): List<Notification>

getAccount:(account: Account) Account

AccountDAOImplementation

- entityManger: EntityManager

get(id: int): Account get(): List<Account>

getNotificationList(accountID: Integer): List<Notification> getGamesPlayed(accountID: Integer): List<Integer> getGamesWon(accountID: Integer): List<integer>

get(username: String): Account

save(account: Account):: void

checkAccount(account: Account): boolean checkUser(account: Account): boolean

getPendingList(accountID: Integer): List<Notification>

getAccount:(account: Account) Account

App

- loggedInStatus: boolean
- user: JSON
- <<constuctor>>App(props)
- getLoggedInStatus():
- handleLogIn(JSON):
- handleLogOut():
- render(): App

ChessBoard

- squaresStateArr: boolean∏
- fromPosition: string
- fromPositionIndex: int
- toPosition: string
- toPositionIndex: int
- pieceArr: ReactComponent[]
- <<constuctor>>ChessBoard(props)
- selectSquare():
- clearSelections():
- isSelected():
- render(): ChessBoard

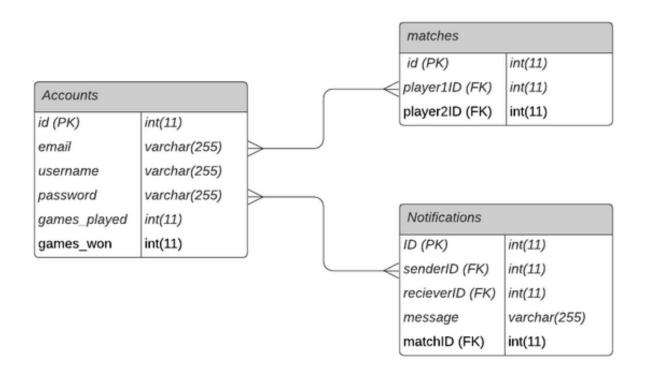
Strike Team Java Frontend

BoardSquare

- selected: boolean
 squareStyle: JSON
 blackSquareStyle: JSON
 whiteSquareStyle: JSON
 selectedSquareStyle: JSON
- <<constuctor>>BoardSquare(props)
- getDefaultSquareStyle(): JSON
- getSquareStyle(): JSONrenderSquare(): Box
- handleClick():
- render(): BoardSquare

Frontend Diagram

Strike Team Java Database ER Diagram



Database Diagram



- First 5 meetings were spent establishing acceptance criteria and formulating questions for the product owner in preparation for the official Sprint Planning meeting.
- Sprint Planning Meeting: met with Product Owner, reviewed User Stories and Acceptance Criteria, established goal to have ~40% of the project completed at the end of the Sprint.
- Scrum meetings 7-11 were conducted as normal Scrum meetings with each member updating the group on progress of assigned tasks.
- Meeting 12 was the Sprint Retrospective. We felt we struggled getting started initially, but once we had an initial setup working our progress moved quickly.
- Meeting 13 was the Sprint Review. We met the Product Owner, provided a demo of our application, received feedback from the PO and made a small number of minor changes based on that feedback.

Completed User Stories

- The User would like to be able to:
 - Create an account on the system by providing an email, username, and password.
 - Login to an account once it is present in the system.
 - Be able to log out of system.

Demo