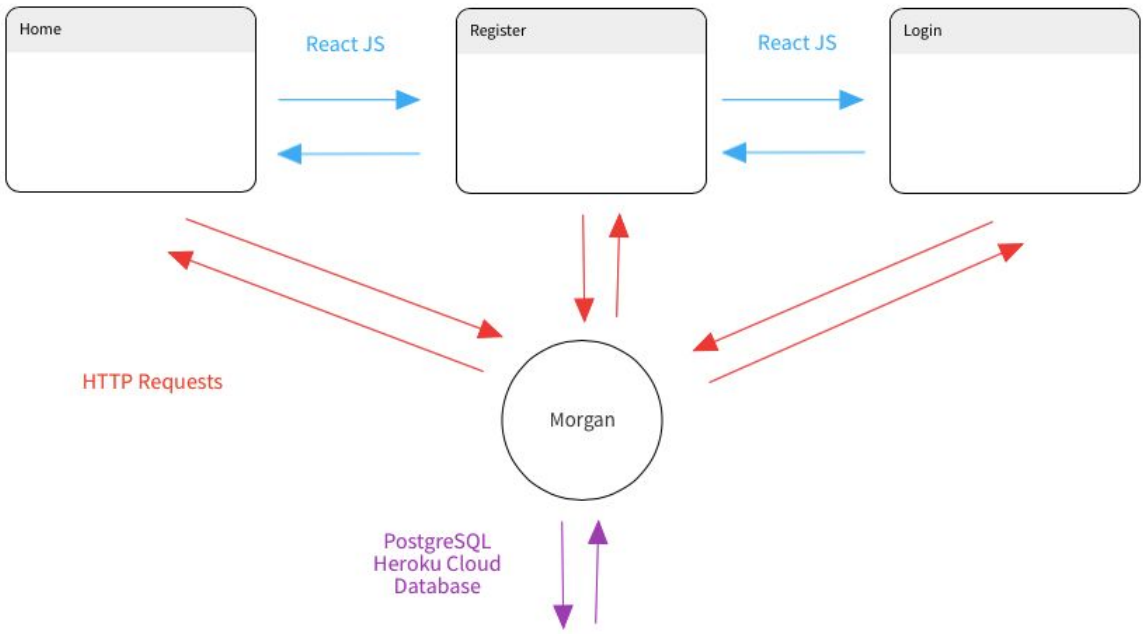
Team Number - 203-1 (Yash Sapra)
Names - Jia, Justin, Kyran, Qiuyang, Sahil, Tahmina

Milestone 4

- **Revised List of Features**
    - Login to existing personal account for users, signup for an account if not already created
    - Landing/home page: view existing chats, create new chats, recent activity in chats, and indication of unread messages.
    - Notifications of incoming unread messages
- **Architecture Diagram**



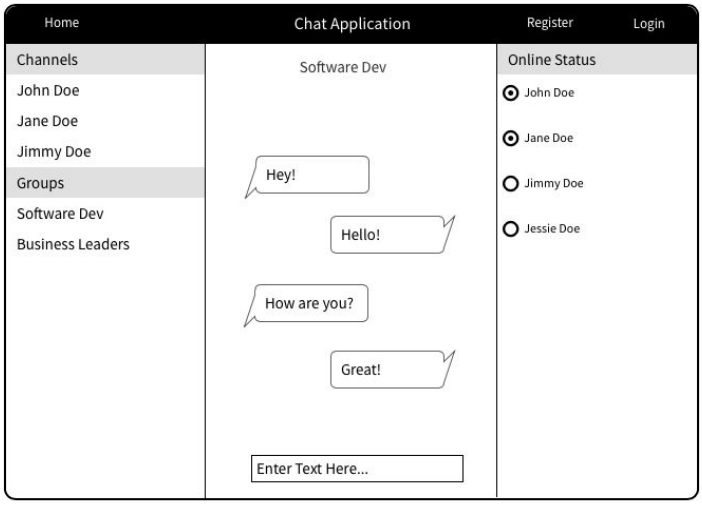
Username	Password
Joe	******
Jane	******
James	******

(Morgan is just the logger, but Express and Node happens on purple).

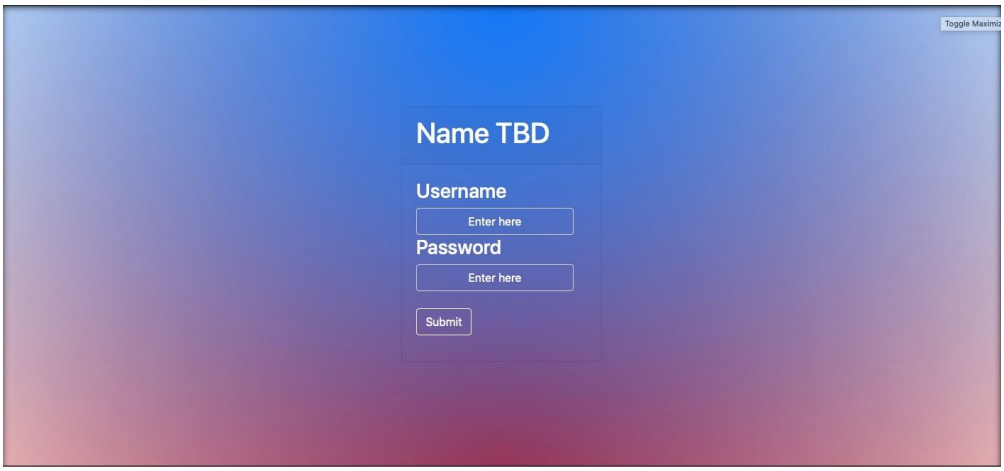
- Front-End
    - HTML/CSS for page design and overall architecture. ReactJS for page routing, user interaction with page. Login/signup/chatting.

    - ReactJS uses react-router to switch between pages
- Middle
    - Morgan prints detailed HTTP requests to the console.
    - ReactJS uses react-router to switch between pages.
    - When the user logs in, the request is sent to the /auth.js file which uses passport to authenticate the user. Passport asks bcrypt if the password matches the passwordhash, and returns login information if they are authenticated.
- Back-End
    - The back-end design comprises of a PostgreSQL database, currently locally hosted but planning on moving to Heroku. The database scheme is diagrammed below.
    - server.js is being hosted using Node.js.
    - Express is used to make simple post requests diagram to files. For example, if a page requests '/api/auth', express can take that and run './routes/api/auth.js', or whatever file needed.
    - The main authentication uses Passport, which accepts the json from the middle-end, and Passport uses bcrypt in order to determine if the username and password hash match a user within the database. If they do, the auth.js / Passport returns login information, and cookie-session logs a cookie with the current password hash.
- **Front End Design**
  Overall, the front end is written and designed using HTML and CSS, and uses features like buttons, background setting, username and password setting, as well as password authentication. ReactJS will be used to to switch between pages

- Home Page
    - An example of our home page is displayed below
    - Contains chatting mechanism in the center
    - Displays different channels on the left
    - Displays online statuses on the right
    - Toolbar at top has login link/status, registration page link, and home page link

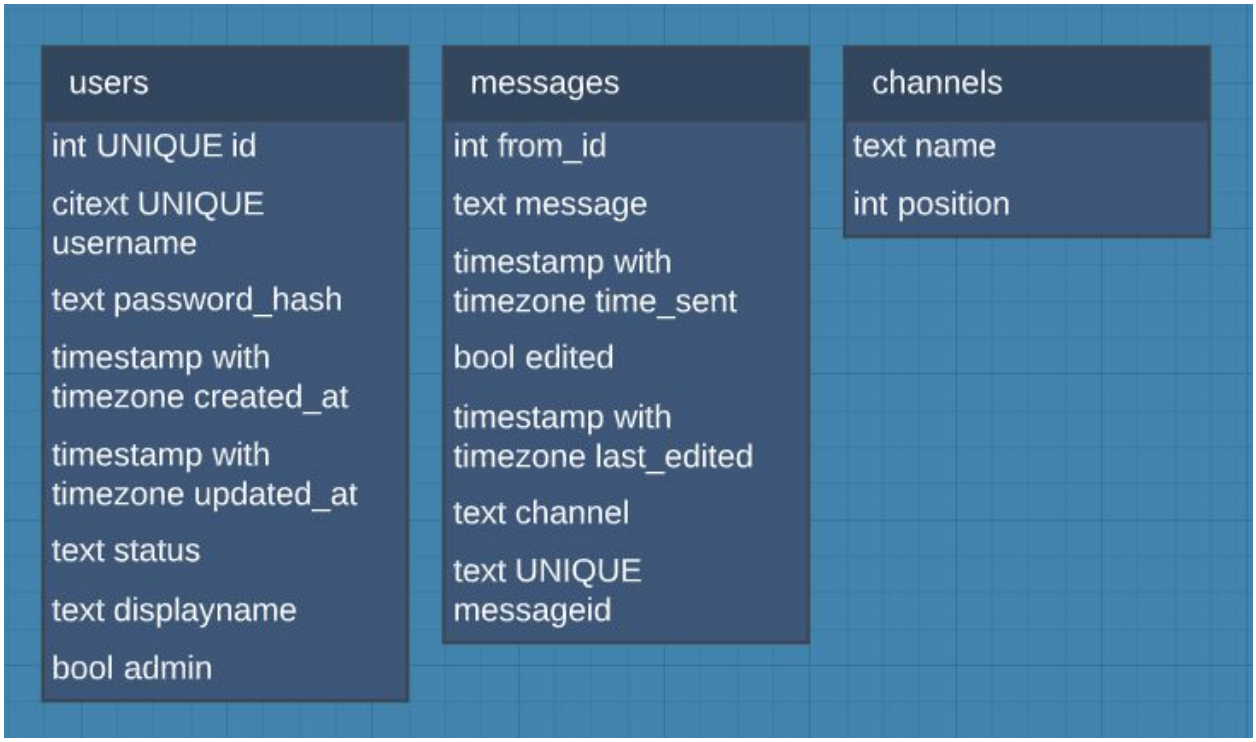


- Login Page
    - The login page is a simple HTML page that uses CSS and includes a username field, a password field, and a submit button to submit the entered information.
    - An example is shown in the image below



- Registration Page
    - The registration page, which was also written in HTML/CSS, allows a user to register for an account by creating a unique username and setting a password
    - Information for each user is stored in a database
-

- **Web Service Design**
    - Unsure of Heroku or Github would fall under this category, but no other API's are being used.
- Database Design (using PostgreSQL, Sequelize Library, and soon moving to Heroku).



- Users
    - ID: unique identifier given to each user. This is the PRIMARY KEY.
    - Username: what the user uses to login. This has to be unique. CITEXT is used so that during the comparison of login, the user can enter a case insensitive name, however it will still store as a specific case for referencing.
    - Password_hash: Self explanatory, but is calculated using bcrypt to semi-encrypt the password.
    - Created_at: it's stored as a specific time, and shows what time this user was created
    - Updated_at: possibly will remove, however shows when the user was last updated.
    - Status: the user can set a custom status that others can see, such as "Away" or "Doing the Dishes"
    - Displayname: This is the data that will be shown on the 'online users' list as well as any sent message. This can be the same as someone else's.
    - Admin: simple boolean to dictate if they're an admin of the website, for debugging purposes.
- Messages
    - From_id: shows the user id of who sent the message
    - Message: message contents
    - Time_sent: is the time sent
    - Edited: shows if the message has been edited.
    - Last_edited: shows when the message was edited

        - Channel: shows which channel the message was sent in
        - Messageid: PRIMARY KEY for sorting and identifying the messages.
- Channels (strong potential of getting changed / removed)
    - Name: channel name
    - Position: dictates the order these should be displayed within the list of channels.