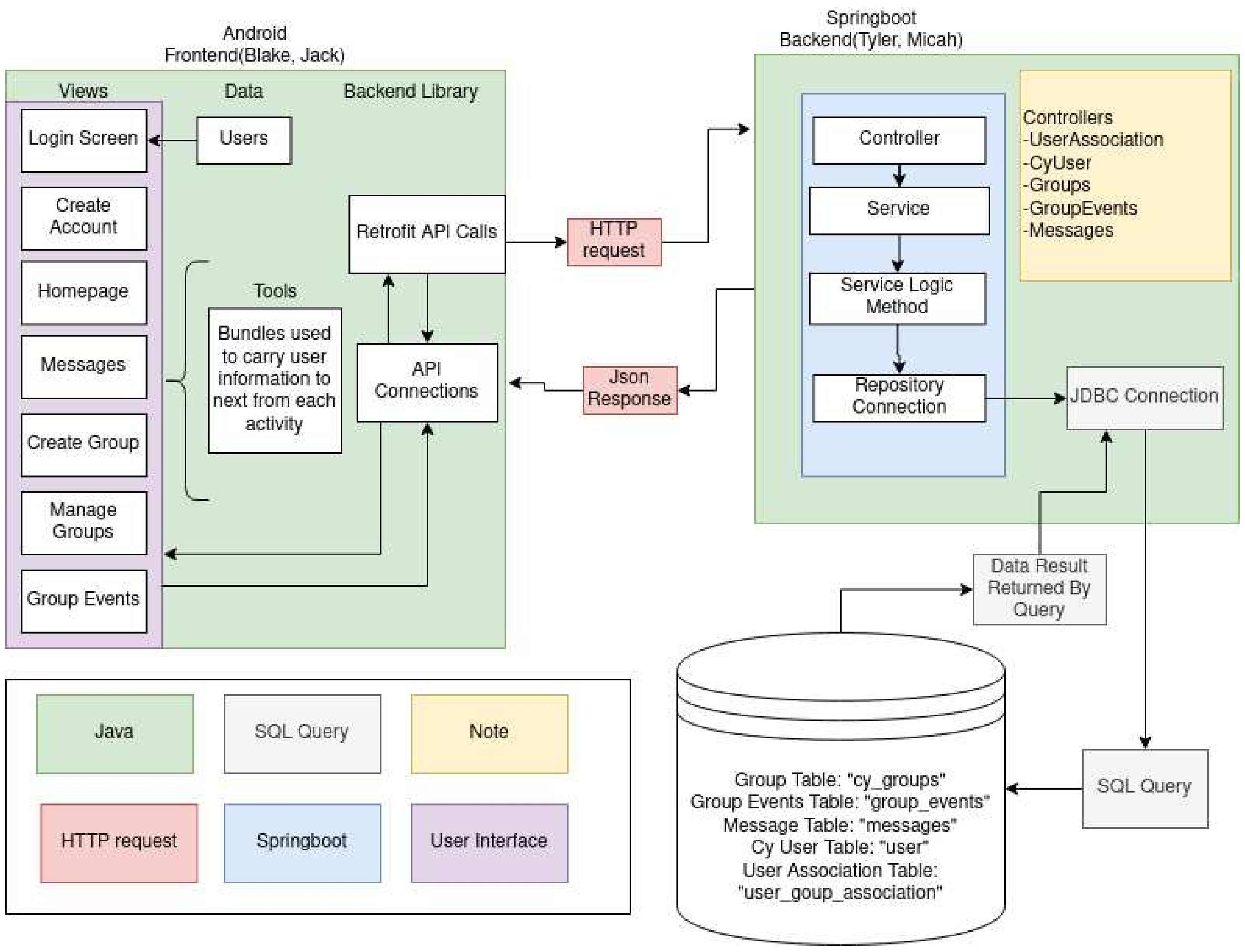
Design Document for CyMessage

Group 3_mc_6
Tyler Miller: 25%
Blake Robson: 25%
Jack Tah: 25%
Micah Gwin: 25%


Android Frontend(Blake, Jack)
Views
Data
Backend Library
Login Screen
Users
Create Account
Homepage
Messages
Create Group
Manage Groups
Group Events
Tools
Bundles used to carry user information to next from each activity
Retrofit API Calls
API Connections
HTTP request
Json Response
Springboot Backend(Tyler, Micah)
Controller
Service
Service Logic Method
Repository Connection
Controllers
-UserAssociation
-CyUser
-Groups
-GroupEvents
-Messages
JDBC Connection
Data Result Returned By Query
SQL Query
Group Table: "cy_groups"
Group Events Table: "group_events"
Message Table: "messages"
Cy User Table: "user"
User Association Table: "user_goup_association"
Java
SQL Query
Note
HTTP request
Springboot
User Interface

Design Description(Tyler and Blake)

Messaging:

Our app will use web sockets in order to allow concurrent messaging between members using the application without the need for refreshing the page to see new messages or having a background loop making post requests. We have not implemented this functionality yet, but will have this for our demo 4. The messaging will follow the following steps.

1) User sends a message
2) The message is received on the backend where it is processed and stored in the database
3) The message is returned to the client via web-sockets and is displayed in the messaging group

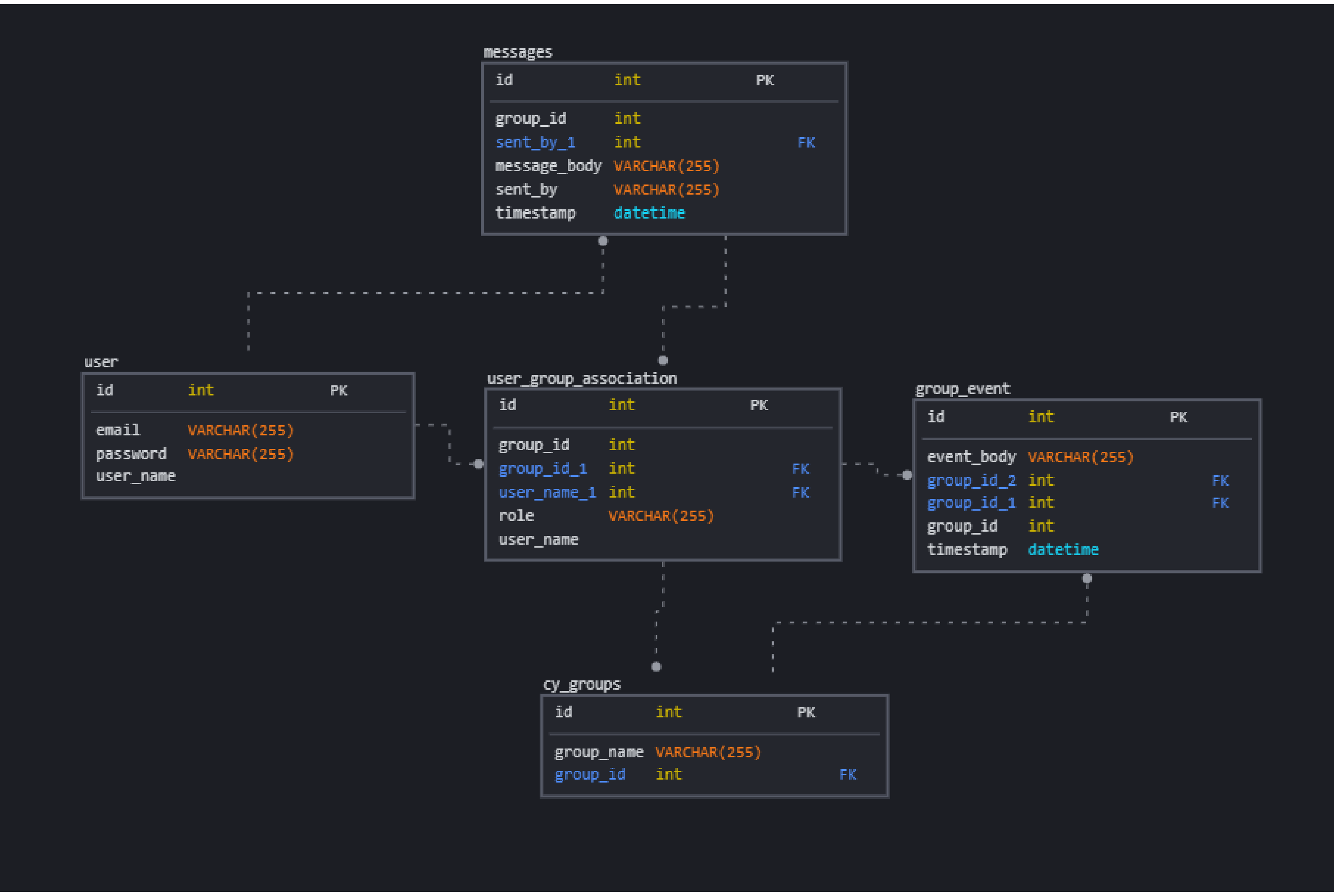
This will allow us to not only have current session messaging, but the messages will be saved outside of the current application session.

Controllers:

All of our controllers feature the same POST, PUT, and DELETE requests, but each controller has its own unique GET request methods depending on the class. Our most complicated Control is for the UserAssociation table which is a controller used by all aspects of the site. The User Controller holds the messaging groups the user is in as well as their user_name and role which are used for granting permissions on the front-end. The User Controller serves as a central location to get needed information on the front end to grab groups the user is in to display for messaging as well as for forum messages. We use SpringBoot RESTful APIS to achieve this since they are easy to use with Springboot.

Visuals:

Our front end consists of xml files and java classes to create the UI for our app. The app opens on a login page, which then takes the user to the homepage where they can do a variety of things. The primary feature of the app is messaging in groups, so the user has the option to create a group as well as select a group to view the messages of. Both of these features require API calls to the backend to get Group, User, UserAssociation, and Message information. The app also has a feature for group events which will display events from the groups the user is in. Also, groups have user roles, which determine the level of access a user has to a group. The messaging screen itself dynamically adds messages to the screen, and it gets those messages by retrieving messages in a group from the backend. A user can also add messages to the group, and refresh to get new messages.


messages
id int PK
group_id int
sent_by_1 int FK
message_body VARCHAR(255)
sent_by VARCHAR(255)
timestamp datetime
user
id int PK
email VARCHAR(255)
password VARCHAR(255)
user_name
user_group_association
id int PK
group_id int
group_id_1 int FK
user_name_1 int FK
role VARCHAR(255)
user_name
group_event
id int PK
event_body VARCHAR(255)
group_id_2 int FK
group_id_1 int FK
group_id int
timestamp datetime
cy_groups
id int PK
group_name VARCHAR(255)
group_id int FK