Design Document for Palo

Group TZ-02
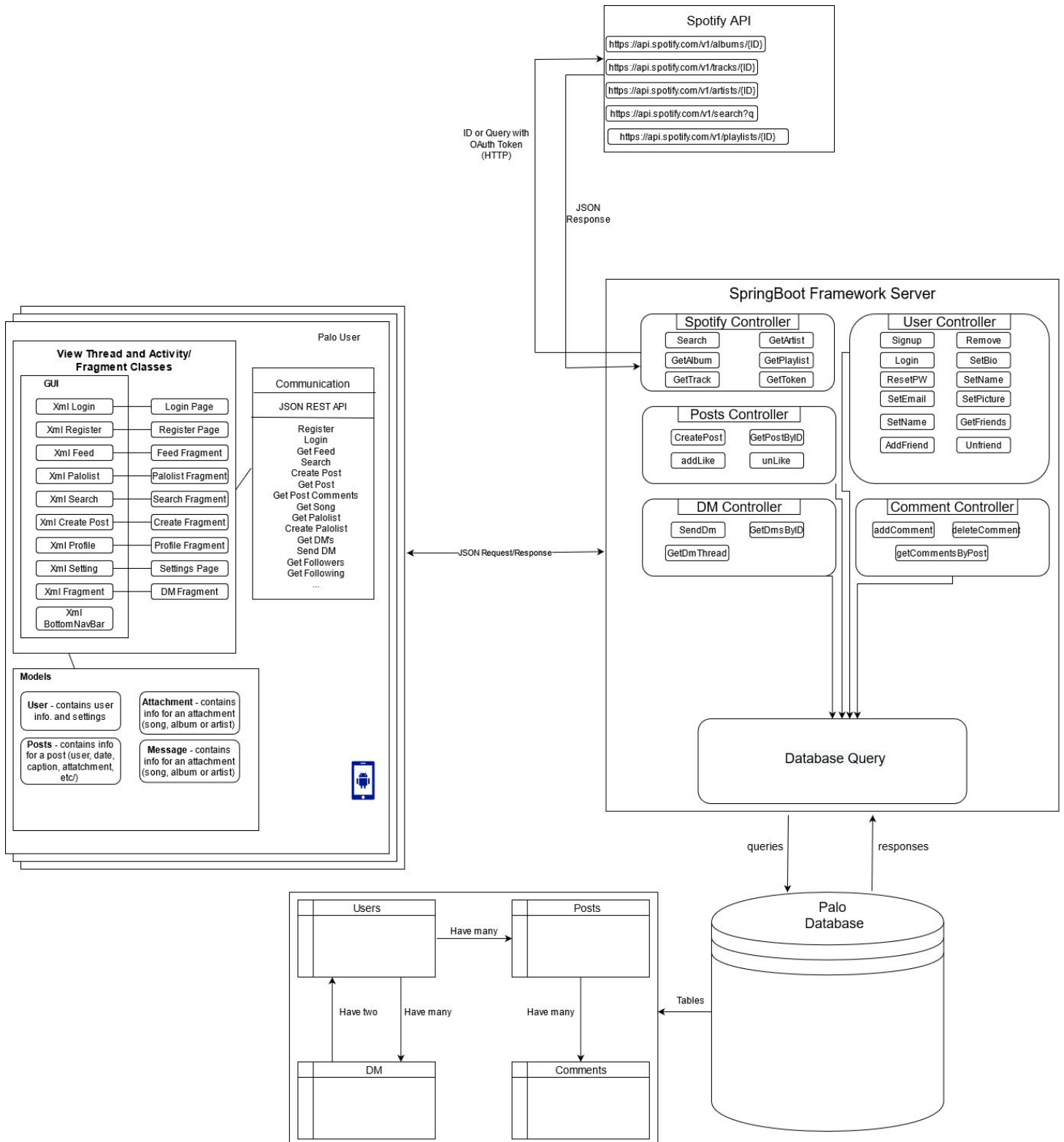
Tom: 25%

Marshall: 25%

David: 25%

Tiffany: 25%

Block Diagram

Spotify API

https://api.spotify.com/v1/albums/{ID}

https://api.spotify.com/v1/tracks/{ID}

https://api.spotify.com/v1/artists/{ID}

https://api.spotify.com/v1/search?q

https://api.spotify.com/v1/playlists/{ID}

ID or Query with
OAuth Token
(HTTP)

JSON
Response

SpringBoot Framework Server

Spotify Controller

Search	GetArtist
GetAlbum	GetPlaylist
GetTrack	GetToken

User Controller

Signup	Remove
Login	SetBio
ResetPW	SetName
SetEmail	SetPicture
SetName	GetFriends
AddFriend	Unfriend

Posts Controller

CreatePost	GetPostByID
addLike	unLike

DM Controller

SendDm	GetDmsByID
GetDmThread	

Comment Controller

addComment	deleteComment
getCommentsByPost	

Database Query

Palo User

View Thread and Activity/ Fragment Classes

GUI

Xml Login	Login Page
Xml Register	Register Page
Xml Feed	Feed Fragment
Xml Palolist	Palolist Fragment
Xml Search	Search Fragment
Xml Create Post	Create Fragment
Xml Profile	Profile Fragment
Xml Setting	Settings Page
Xml Fragment	DM Fragment
Xml BottomNavBar	

Communication

JSON REST API

Register
Login
Get Feed
Search
Create Post
Get Post
Get Post Comments
Get Song
Get Palolist
Create Palolist
Get DMs
Send DM
Get Followers
Get Following
...

Models

User - contains user info. and settings

Attachment - contains info for an attachment (song, album or artist)

Posts - contains info for a post (user, date, caption, attatchment, etc/)

Message - contains info for an attachment (song, album or artist)

JSON Request/Response

queries

responses

Palo Database

Tables

Users		Posts
	Have many	

Have two

Have many

Have many

DM	Comments

Design Description

Android User GUI

Our Palo Application will consist of different fragments that make up the bulk of the app. On startup the first time, the user will be presented with a login page. If the user doesn't have an account already, they can register themselves to the app. This will create an account on our database. When they login, they will be put on the feed fragment. These fragments are designed for easy access to other parts of the app. All of the fragments and pages are located on the "View Thread and Activity/Fragment Classes" section. They all have a corresponding xml page that will include their design. The activity and fragment classes will hold their functionality and what we want the pages to do.

Android Models

We have four planned models: User, Posts, Attachment, and Message. These models allow us to keep related data together and allows us to retrieve/send data back and forth between the client and the server more cleanly.

Android Communication

To communicate with our backend, we will use different JSON requests to get information for our users. Different activities and fragments will use different requests to our server to organize the information properly and store it in our database.

Spotify API

For our application we want to use the Spotify API to be able to create a user experience that will include songs, artists, and albums right on the app. This will make it easier for users to share their stories without having to switch back and forth between Spotify and our app.

Framework/Server

For our server we're using springboot to host our java based api. This api will consist of a number of different url endpoints that will take in data in json format from our frontend application. The data will then be processed differently based on the endpoint it is tied to. Any endpoint that is using the Spotify API will not touch the database, but instead ping the Spotify API using the given information, parse the response and return a json string back to the requesting application. We are accessing our sql database using the hibernate as a framework for managing our relational database.

Table Relationships Diagram

PUT THE TABLE RELATIONSHIPS DIAGRAM on this fourth page! (Create the picture using MySQLWorkbench)