Design Document for Dialogue

Group SB_05

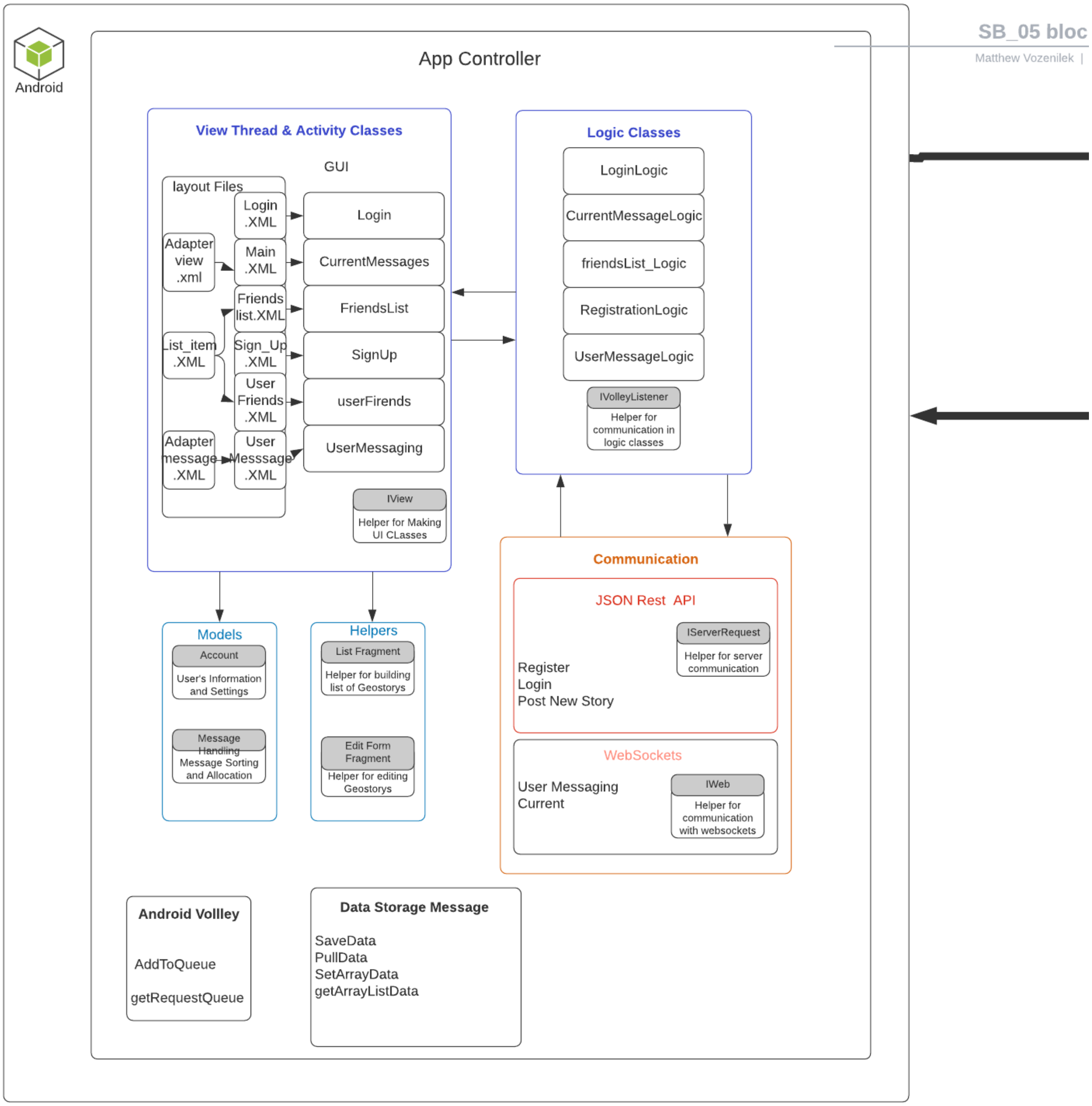
Jacob Nett: 25%

Christian William: 25%

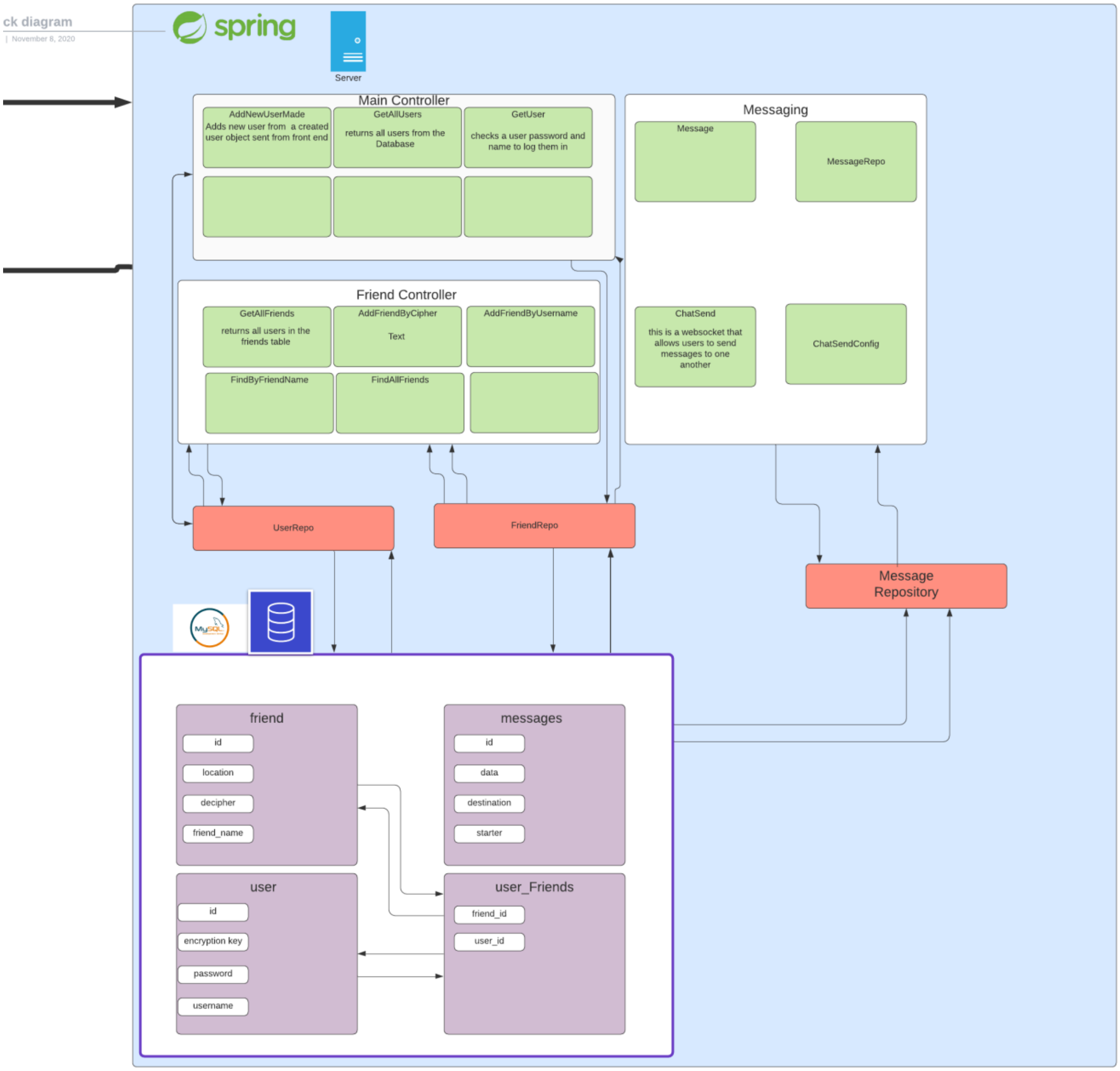
Matthew Vozenilek: 25%

Jamie McCullough: 25%

Block Diagram part 1



Block Diagram part 2



Design Descriptions

Backend Design:

Our backend server allows the front end, through HTTP requests, to access the data stored on our database. We have user information and information for the Friends of the users. Our main controller handles the following: To log in to the front end requests, our getUser method takes a username and password. On success, it returns the user requested, and on failure, it returns a null object so they can try again. It also has functionality for adding a User which takes in a JSON in the request, and if that user exists, returns a string saying the user exists and creates a Friend object from the user objects information and then saves these both to their specific repository. The Friend repository handles the following; it takes in requests to get all the friends in the database to look through the friends to add them to the user's friends list. Friends can also be searched to be added by name or by the friends decipher.

Websocket: our WebSocket adds users on logging into the app. A repository saves all the messages sent to them while offline and on logging in it sends these messages to the user so they don't miss out on what they got while offline. Once on the app, the app sends the messages normally and sorts out dm's to go to what specific user.

Frontend Design:

Android GUI:

Android GUI uses a dedicated thread that it uses to build and run activities and methods within them. In our project, we currently have 6 activity methods that follow this design pattern all of them use an xml layout file to base what gets printed to the screen around. Allow our activities are created with an iView interface to allow modular coding and expandability. Most Activity classes are associated with a similarly named logic class where a lot of activities heavy lifting is done.

Logic Class:

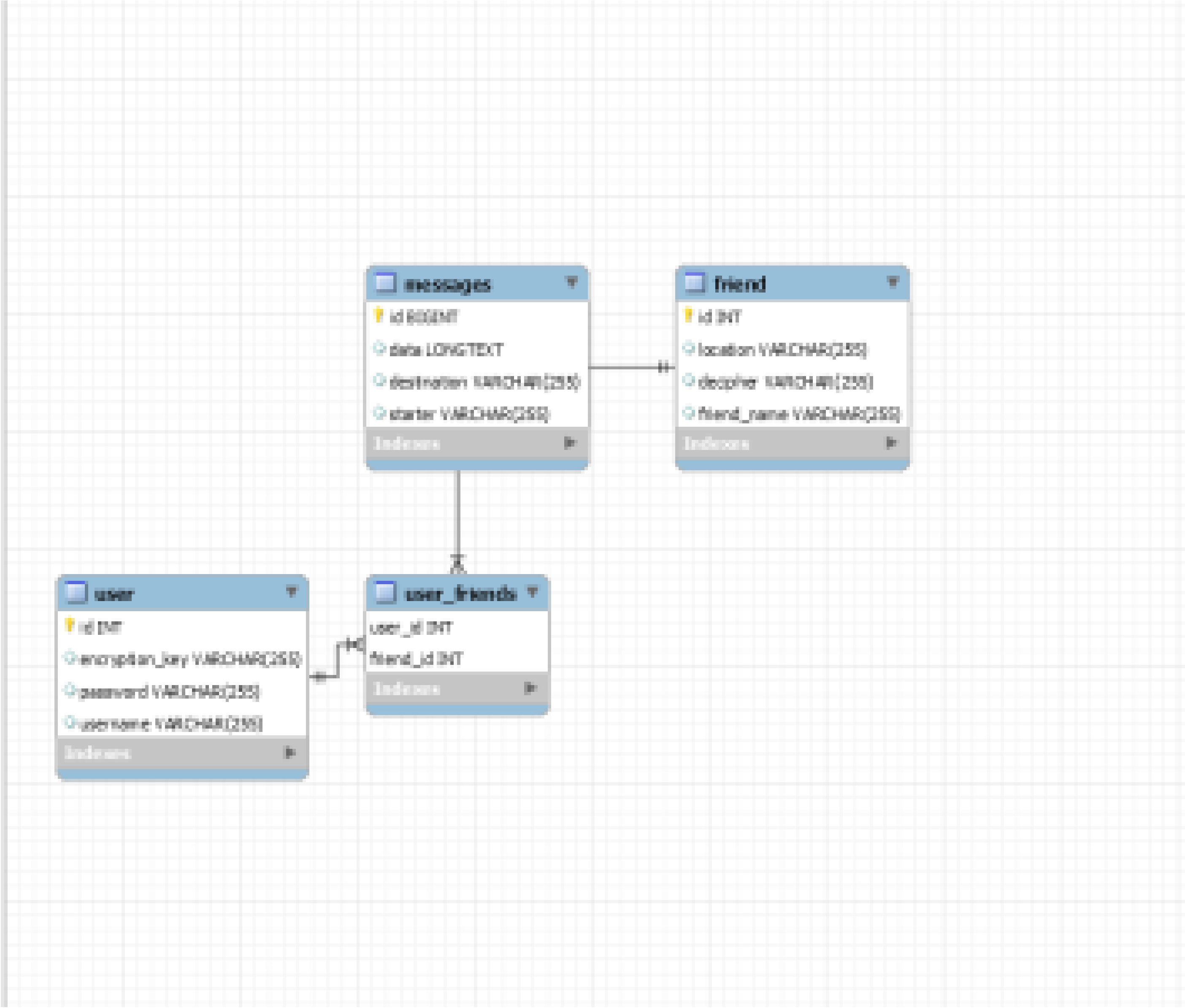
Logic class does the actual working in the app manipulating table and input sending them where they need to go and checking for validity. Logic classes are used in most of the data handling and storage for the app.

Data Storage:

For data storage create two java objects that make messages and the storage of messages. These gets stored by our app controller to be accessed by the entire application. Message storage is handled using the SharedPreferences library by converting objects to strings using the GSon library and Convert back to objects later using a similar method. As the goal of the app is took keep message data secure all of the data is stored client side by the device

Communications:

For communication between the Server and the Client a JSON Rest API is used in registering, logging in, and getting user information like the user's list of friends. For communication between each user, WebSockets are used. Websockets send a string from one user to another and when a message is received, then it will be added into the data storage. Android Volley methods are used within a few of the logic classes, we use request queue in a singleton pattern in the app controller as to make calls to the server without conflicting.


messages
id BIGINT
data LONGTEXT
destination VARCHAR(255)
starter VARCHAR(255)
Indexes
friend
id INT
location VARCHAR(255)
decipher VARCHAR(255)
friend_name VARCHAR(255)
Indexes
user
id INT
encryption_key VARCHAR(255)
password VARCHAR(255)
username VARCHAR(255)
Indexes
user_friends
user_id INT
friend_id INT
Indexes