Design Document for PlanIT

Group KM_205

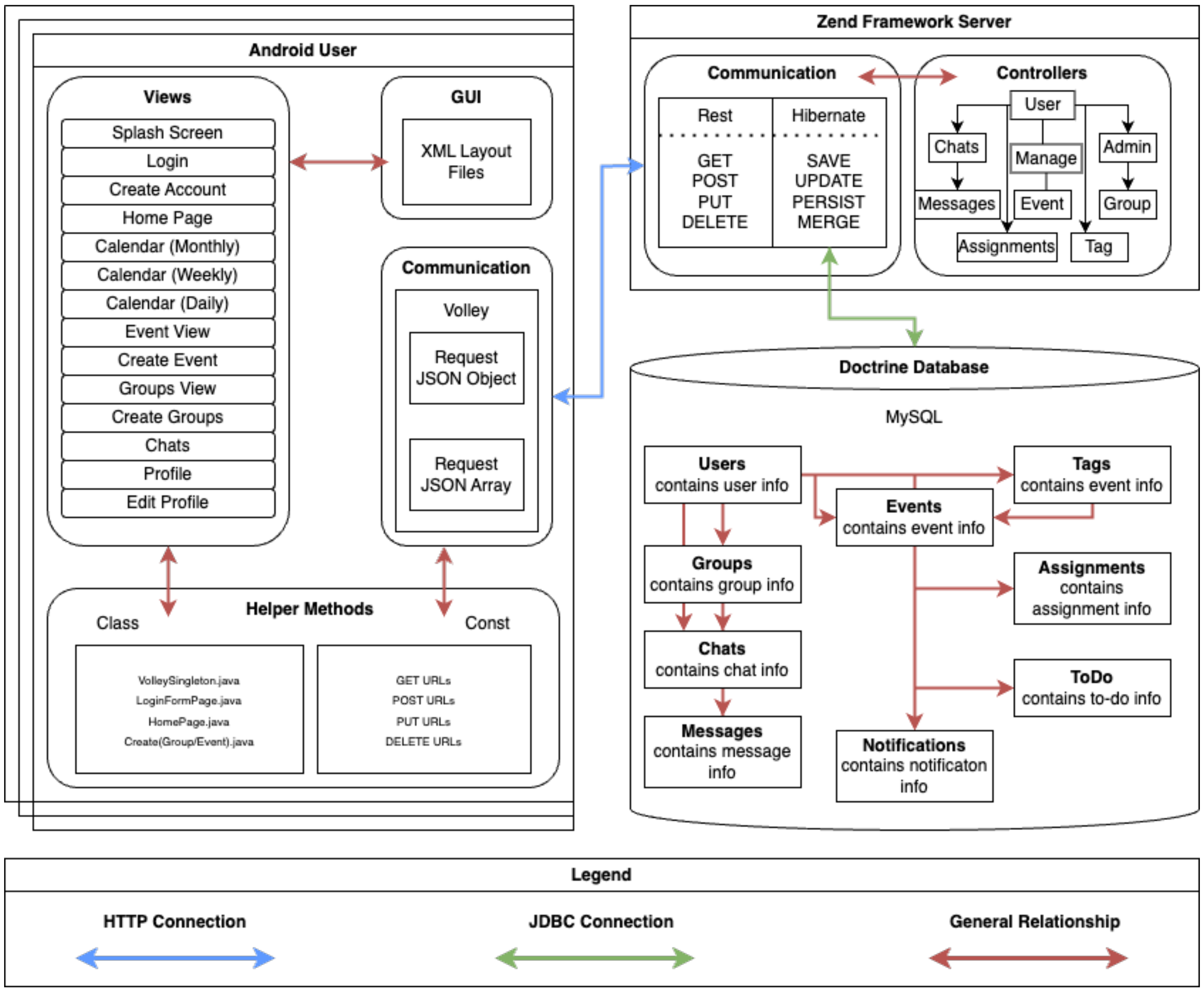
Joshua Gutierrez: 25%

Melani Hodge: 35%

Tristan Nono: 30%

Chris Smith: 10%

Block Diagram



Core Features

Frontend

-Calendar monthly view mainly gets the data so shows the right events for the right day and displays that all in the correct order making the earliest starts up top and then the event goes down. Also, the events that are past due get archived and the more recent event shows up.

-Chat view in terms of the users seeing their own chat bubbles is in a different color and the text (for that user) should be on the right side and their members or other people are in the left side with a grey text bubble.

-Incorporating the Canvas API mainly the assignments trying to fetch the data and fitting it within the events of our calendars. How can we format the assignment data in terms of ordering it?

-Home page displays the upcoming events and our priority events/task. The events in the priorities will have to show the events that have the priorities so the more important ones and depending on the data & time it will show first and similar with the upcoming events it will need to show which event is coming up (if it's the earliest coming up).

-Calendar daily view the event blocks are going to need to be on their own corresponding date but, not only that they also must be on their own day if a certain event is on Tuesday, it will be on Tuesday. Also, for the event blocks they have to cover up to their times like 4:00-5:00 covering up that time slot as well. Plus, the Canvas aspect such as the TODO assignments will have to be a line and still visible on the daily view.

-Chats, making sure that the user profile picture is shown and making sure that each chat is tied to their own group/team while making sure other user interaction is working like embedding the images for when users want to send in images. Chat messages should also be saved.

-When in the calendar views like daily and weekly the data of the date needs to updated as the week changes and day changes making that it corresponds to the actual date itself and making sure it's only pulling up the right data and not just staying on there.

-The group search is a popup where you can search for the group reason being is with the drag and drop and making sure that the drag and drop saves the order for that person without affecting anything in the database so keeping that on the client side. Group search does function the same as the event search just with a popup.

-Event search functions as a search for searching up the events and also there is a filter for the events for the type of event and the dates of when that event happens.

Backend

Communication

The backend uses mappings to update and add user information to the database based on information sent from the frontend to the path denoted in the mapping. There are four different types of communication implemented for each table (some tables differ depending on use). The four types of communication are:

1. **GET** – This method requests information from the database with user specific information, retrieves that information, and displays it for the user to. The user specific information allows a user to only see his/her own data.
2. **POST** – This method creates a new object or element and sends it to the database to be saved. This method allows for users to create their own data or their own personalized experience within the application.
3. **PUT** – This method allows for a user to update data that has already been posted to the database. Only certain entities allow a user to PUT (ie. notifications).
4. **DELETE** – This method allows for a user to delete any information they no longer need in the application. Most entities allow for this as to allow the user to keep their data in the app organized and to their liking.

Controller and Service Classes

The controller class contains the mappings for communication between frontend and the database. The controllers create paths for the data in the database to be retrieved, added, updated, or deleted. The service class contains methods used by the controller class to POST, PUT, and DELETE data in the database. These include:

- **User** – Contains methods for adding a user to the database, updating a user's information, and deleting a user if they no longer want their account.
- **Assignment** – Contains methods for adding an assignment to a user's list of assignments, updating information on an assignment, and deleting an assignment or marking the assignment as done.
- **Chat** – Contains methods for creating chats for users to communicate with one another in the app, updating a chat name, and deleting a chat.
- **Event** – Contains methods for adding an event to a user's list of events, updating an event's information, and deleting an event off a user's list. When there is more than one user in an event, if a user is the manager of the event and they delete the event, the event will be removed from every user's calendar. If the user is not an admin, the event will just be removed from their own calendar. Only the manager of the event will have access to edit information in an event.
- **Message** – Contains methods for creating messages tied to a chat and deleting messages once they are sent. A user will not have the ability to update a message as once it is sent, the user only can delete the message.
- **Notification** – Contains methods for creating notifications tied to a user in the database and deleting notifications from a user's notifications list. A user will not have the ability

to update notifications as it is not necessary since it is information sent from the backend of the application.

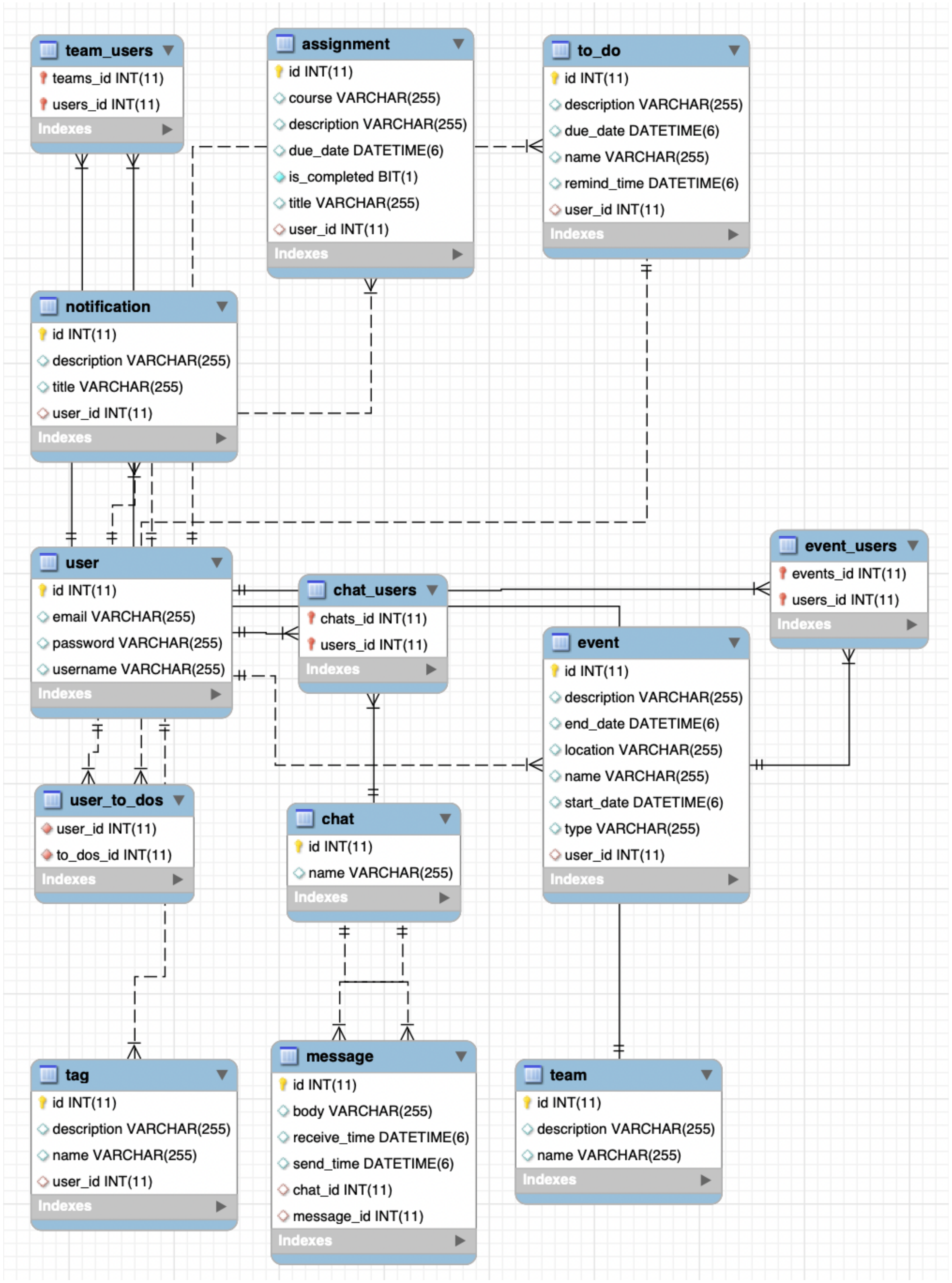
- **Tag** – Contains methods for creating tags tied to a user and deleting tags from a user's tags list. Tags are used to put labels on events or assignments that will be later used to filter the assignments, to-dos, or events list.
- **Team** – Contains methods for creating a group, updating a group, and deleting a group from the database. Only the group administrator will have the ability to update or delete a group. The administrator will have the ability to transfer ownership of the group within the application.
- **To-Do** – Contains methods for adding a to-do to a user's list of to-dos, updating information on a to-do, and deleting a to-do or marking the to-do as done.

Web Sockets

The web sockets will be used for communication between users in the application and for real-time data reflected within the application. The web sockets currently implemented in the backend of the application include:

- **Active Users** – This web socket shows what users are online whenever a user logs into the app and is on the home page. This web socket sends the number of online users to the frontend to display. The web socket updates on a user login and on a user logout.
- **Messaging** – This web socket allows for users to send messages to one another through connected chats.
- **Notifications** – This web socket sends a user a notification whenever that user is invited to an event, added to a group, or receives a message from another user.

MySQL Generated Schema



team_users
teams_id INT(11)
users_id INT(11)
Indexes
assignment
id INT(11)
course VARCHAR(255)
description VARCHAR(255)
due_date DATETIME(6)
is_completed BIT(1)
title VARCHAR(255)
user_id INT(11)
Indexes
to_do
id INT(11)
description VARCHAR(255)
due_date DATETIME(6)
name VARCHAR(255)
remind_time DATETIME(6)
user_id INT(11)
Indexes
notification
id INT(11)
description VARCHAR(255)
title VARCHAR(255)
user_id INT(11)
Indexes
user
id INT(11)
email VARCHAR(255)
password VARCHAR(255)
username VARCHAR(255)
Indexes
chat_users
chats_id INT(11)
users_id INT(11)
Indexes
event_users
events_id INT(11)
users_id INT(11)
Indexes
event
id INT(11)
description VARCHAR(255)
end_date DATETIME(6)
location VARCHAR(255)
name VARCHAR(255)
start_date DATETIME(6)
type VARCHAR(255)
user_id INT(11)
Indexes
user_to_dos
user_id INT(11)
to_dos_id INT(11)
Indexes
chat
id INT(11)
name VARCHAR(255)
Indexes
tag
id INT(11)
description VARCHAR(255)
name VARCHAR(255)
user_id INT(11)
Indexes
message
id INT(11)
body VARCHAR(255)
receive_time DATETIME(6)
send_time DATETIME(6)
chat_id INT(11)
message_id INT(11)
Indexes
team
id INT(11)
description VARCHAR(255)
name VARCHAR(255)
Indexes