\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Design Document for ToDos/Skejul**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Group 2\_TZ\_1

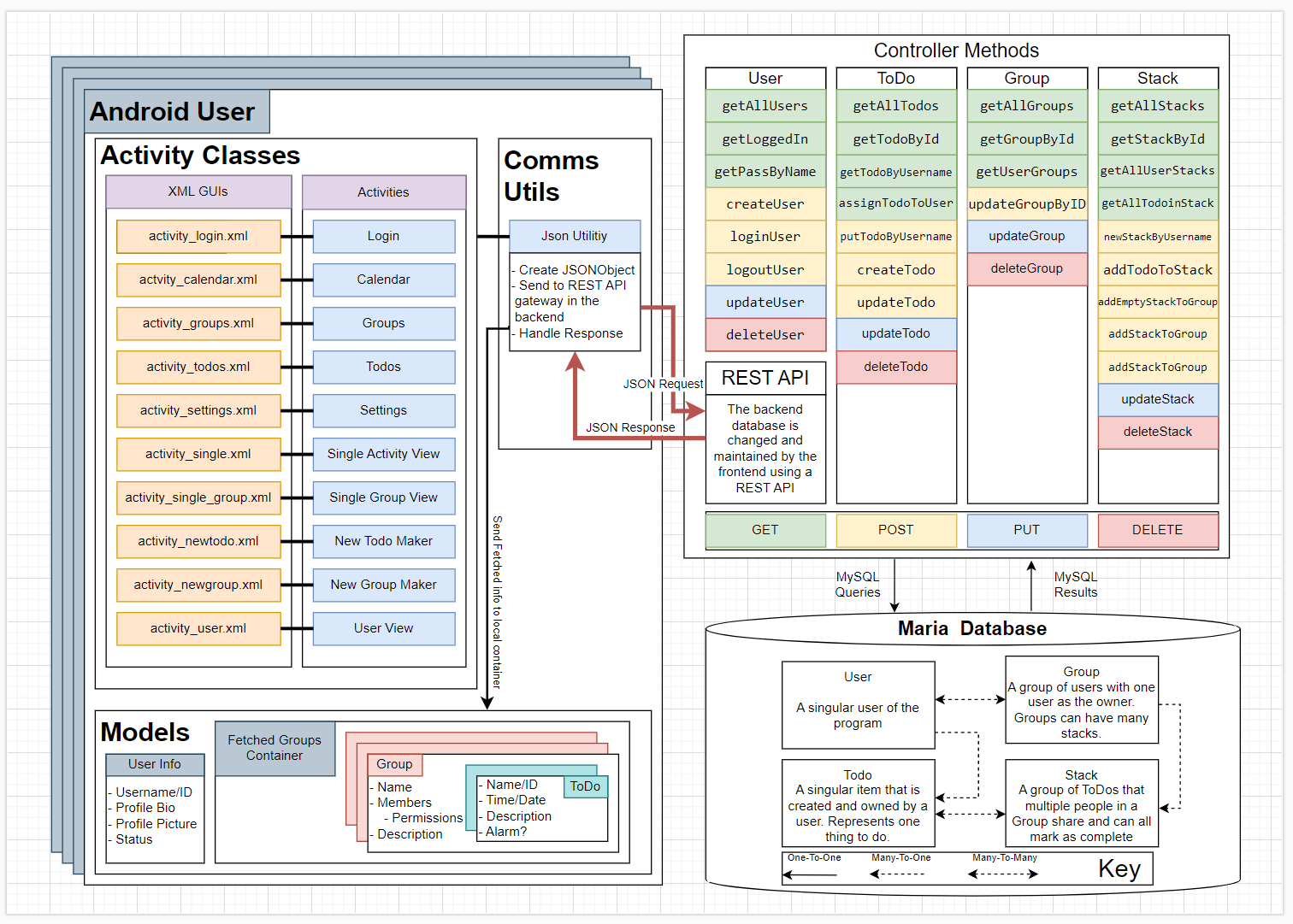
Alexander Black

Jacob Lyons

Justin Templeton

Nicholas Toothaker

**Block Diagram**



**Frontend**

*Activities*

The frontend contains an Android application with a number of activities that allow the user to view their Groups and ToDos with ease. These activities are as follows:

* Login: Opens upon opening the app. Lets the user log into the app with their username and password. Takes the user to the calendar activity upon successful login.
* Calendar: Allows the user to select a date on the calendar to focus the viewing of ToDos.
* ToDo List: Lists the user’s ToDos focused on the date selected on the calendar.
  + Single ToDo: View a single ToDo.
  + Create New ToDo: Creates a new ToDo and sends it to the backend.
* Group List:
  + Single Group: Shows group information, ToDos from the group focused on the date selected on the calendar, allows those with permission to add new members, and features a websocket chat for the group.
  + Create New Group: Creates a new group (including adding members) and sends it to the backend.
* Settings: Shows and allows edits of settings (e.g. notifications)
* Profile: Shows information about the user.

*Communication*

The JSON\_Utility class is used by all the activities that need data from the server or need to send data to the server. Information fetched from the backend is stored locally on the device (more on this under the Models subsection). This process involves:

* Creating a JSON\_Object based on what is needed from the backend.
* Sending the object to the backend (JSON request).
* Receive the JSON response from the backend and process group/ToDo information (if any is present) into the group container (under Models subsection).

*Models*

Group information that is fetched from the backend is stored locally in a group container on the device. Upon login, the user’s groups are fetched from the backend and stored here so that they may be accessed by the activities. This container is refreshed when a new group or ToDo is made and sent to the server, or when the groups or ToDos list pages are accessed. This allows the user to see the latest information.

Information about the user is stored separately upon login and can be accessed in the Profile activity

**Backend**

*Communication*

The backend database is updated using http request mappings that contain the information in the URLs. The mappings we use are the following:

* Post: we primarily use post requests to send information to the database.
* Get: get requests are used to retrieve information from the database for display on the front end.
* Put: there are a few autogenerated put requests. As a rule we use push requests in place of put requests.
* Delete: as with put there are a few autogenerated delete mappings. We tend to not delete things from the database as we want to be able to display things like completed todos.

*Controllers*

There are five controllers used for communication between the frontend and the backend database. These include the following:

* Group: uses the above mappings to create groups of users. Groups have a many-to-many relationship with users and a one-to-many relationship with stacks.
* Stack: uses the above mappings to create stacks which have a many-to-many relationship with todos and a many-to-one relationship with groups.
* ToDo: also uses the above mappings to create ToDo which have a many-to-one relationship with users and a many-to-many relationship with stacks.
* User: utilizes the above mappings to create users. Users has a one-to-many relationship with ToDo and a one-to-many relationship with Stack

**Database Schematic**

