Eyevory - An Application Monitor

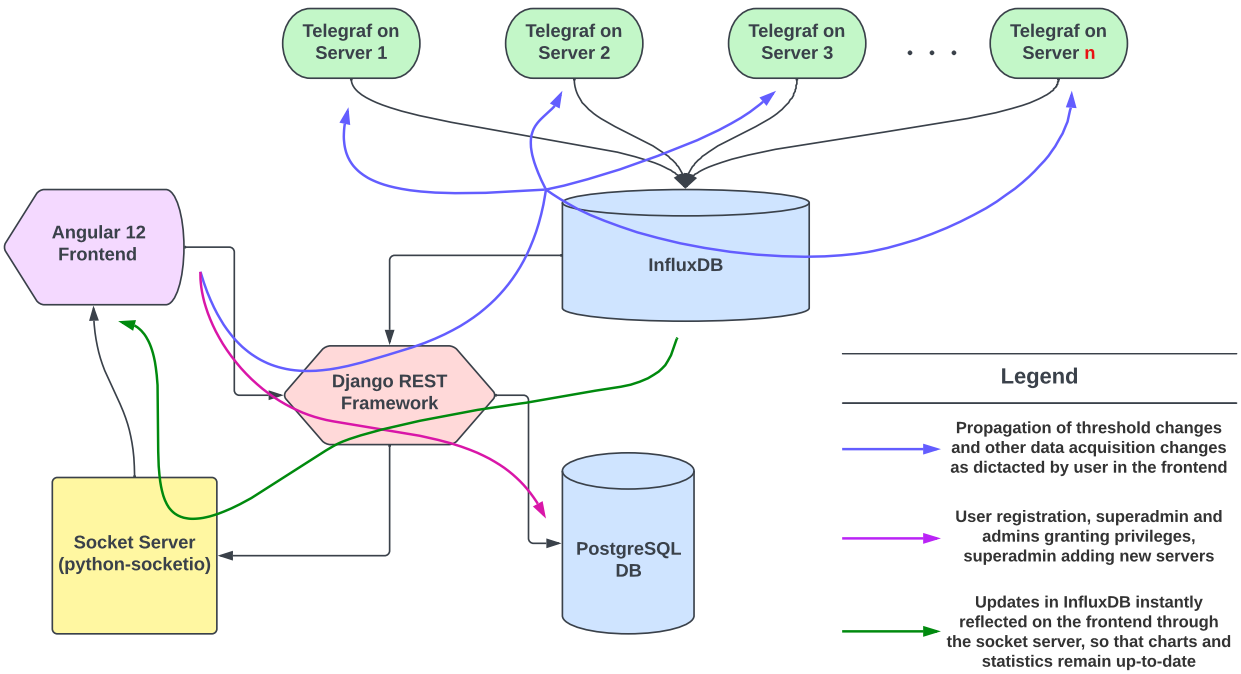
Project Design

CS 387 - Spring 2022

Group Members

Name	Roll Number	Contact
Ankit Kumar Misra	190050020	ankitkmisra@cse.iitb.ac.in
Richeek Das	190260036	richeek@cse.iitb.ac.in
Rahul Prajapat	190050095	rahulprajapat@cse.iitb.ac.in
Raja Gond	190050096	rajagond@cse.iitb.ac.in

Overall Design



Framework Setup and Interactions in our application

Design of Data flow

In this section we describe the data flow pipelines between Telegraf, InfluxDB, PostgreSQL, Django REST Framework, python-socketio, and Angular 12.

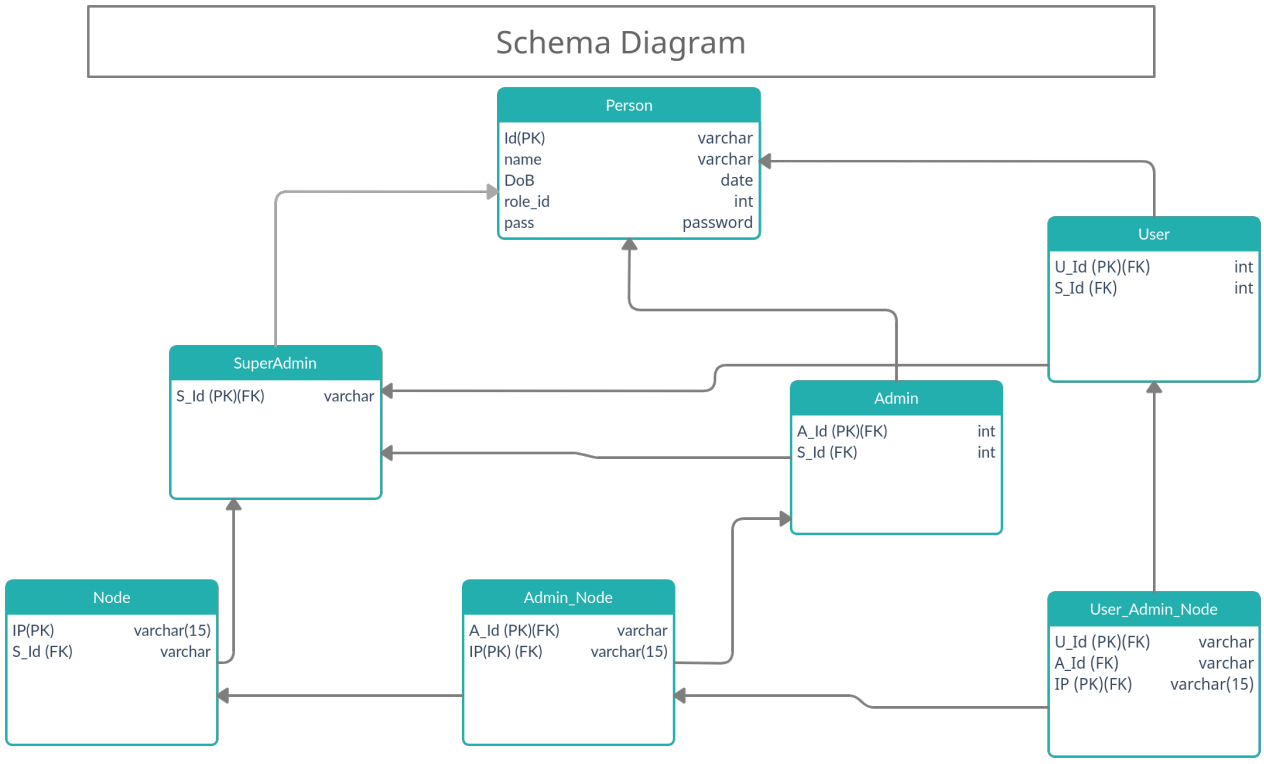
Data Acquisition Flow : There are running Telegraf processes on the target servers. Each of these Telegraf processes will stream data into the central InfluxDB. The servers will be distinguished by their IP.

Angular→DRF→InfluxDB→Telegraf Pipeline : Users can change the frequency of data acquisition, processes to monitor, safety thresholds and add new servers. All of these changes will be reflected by this pipeline. User responses from the frontend will be taken by DRF and it in turn will make the changes in the InfluxDB config file and Telegraf acquisition parameters.

Angular→DRF→PostgreSQL DB Pipeline : User registration/login, granting of server privileges by superadmins to admins/users and by admins to users. Every user specific change made in the frontend is propagated through this pipeline and updated in the persistent PostgreSQL DB.

InfluxDB→DRF→Socket Server (python-socketio)→Angular Pipeline : As soon as new data is supplied by Telegraf into InfluxDB, we will ideally want to update the charts and statistics displayed on the frontend. This is taken care of by this pipeline. As soon as data in InfluxDB is updated, the DRF sends the updated data to the Socket Server which in turn forwards it to Angular and enables live dynamic charts.

PostgreSQL DB Design & Schema



Schema Diagram DB Design

Integrity Constraints

- For creating an account it is checked if an acount is already present with the same ID.
- S_Id attribute of User and Admin cannot be NULL as all the Users and Admins are required to associate with a Super Admin.
- S_Id attribute of Node cannot be NULL as a node can only be added by a Super Admin, hence S_Id should refer to some Super Admin.
- role_id of a Person should never be NULL as it should be Super Admin, Admin, or User.
- As Person when deleted should delete the respective entries in the role (Super Admin, Admin, User) table.
- When a Super Admin is deleted, everyone (Users and Admins) associated with it should also be deleted.
- When a Super Admin deletes a node, the corresponding entries in Admin_Node and User_Admin_Node tables should also be deleted.
- When an entry in Admin_Node is deleted the corresponding entry in the User_Admin_Node should also be deleted as the Admin who granted the permission to that user no longer has permission itself, so the permission to user is also invoked.

Transactions

- Account Creation : Populate the Person table and a respective entry in the role associated.

```
decision = IF EXISTS (SELECT * FROM Person WHERE Id = Id)
if decision:
    throw error
INSERT INTO PERSON VALUES (Id, pass, name, email, DoB, role_id);
switch(role_id):
case 2: INSERT INTO ADMIN VALUES (Id, S_Id);
case 3: INSERT INTO USER VALUES (Id, S_Id);
```

  Super Admin can only be created by Super User of Django.

- Login : Check if the credentials match

```
decision = IF EXISTS (SELECT * FROM PERSON WHERE Id = Id AND pass = pass)
if decision:
   all good login
else:
   throw error
```

- Adding a Node: Populate the Node table with an entry

```
INSERT INTO NODE VALUES (IP, Id);
```

- Deleting a Node: Normal Deletion as cascading is already taken care of in integrity constraints

```
DELETE FROM NODE WHERE IP = IP_passed;
```

- Deleting an account:

```
DELETE FROM PERSON WHERE Id = Id;
```

- Providing an admin the rights to some node: Just adding an entry in admin_node table suffices

```
INSERT INTO ADMIN_NODE VALUES (A_Id, IP);
```

- Providing an user the rights to some node:

```
INSERT INTO USER_ADMIN_NODE VALUES (U_Id, Id, IP);
```

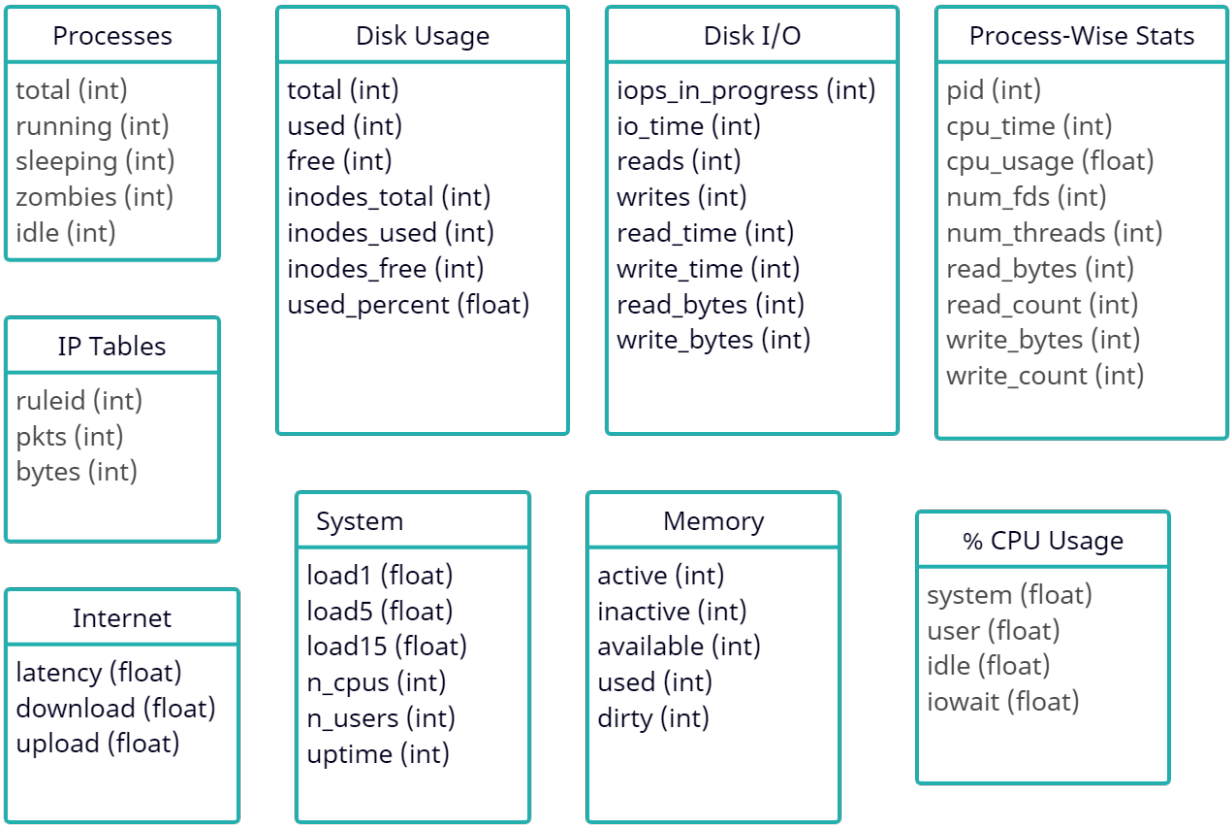
- Invoking the rights of an user for some node:

```
DELETE FROM USER_ADMIN_NODE VALUES WHERE U_Id = U_Id AND IP = IP;
```

- Invoking the rights of an admin for some node:

```
DELETE FROM ADMIN_NODE VALUES WHERE A_Id = A_ID AND IP = IP;
```

InfluxDB Bucket Setup & Schema



Schema Diagram InfluxDB Bucket

- Buckets can be thought of as a database.

- Most of the tables in a bucket have time stamp as a primary key.
- Some tables such as Process-Wise Stats can have variable number of entries per timestamp, so it's primary key is a combination of primary key if it were a postgresQL table and timestamp.

Materialized Views

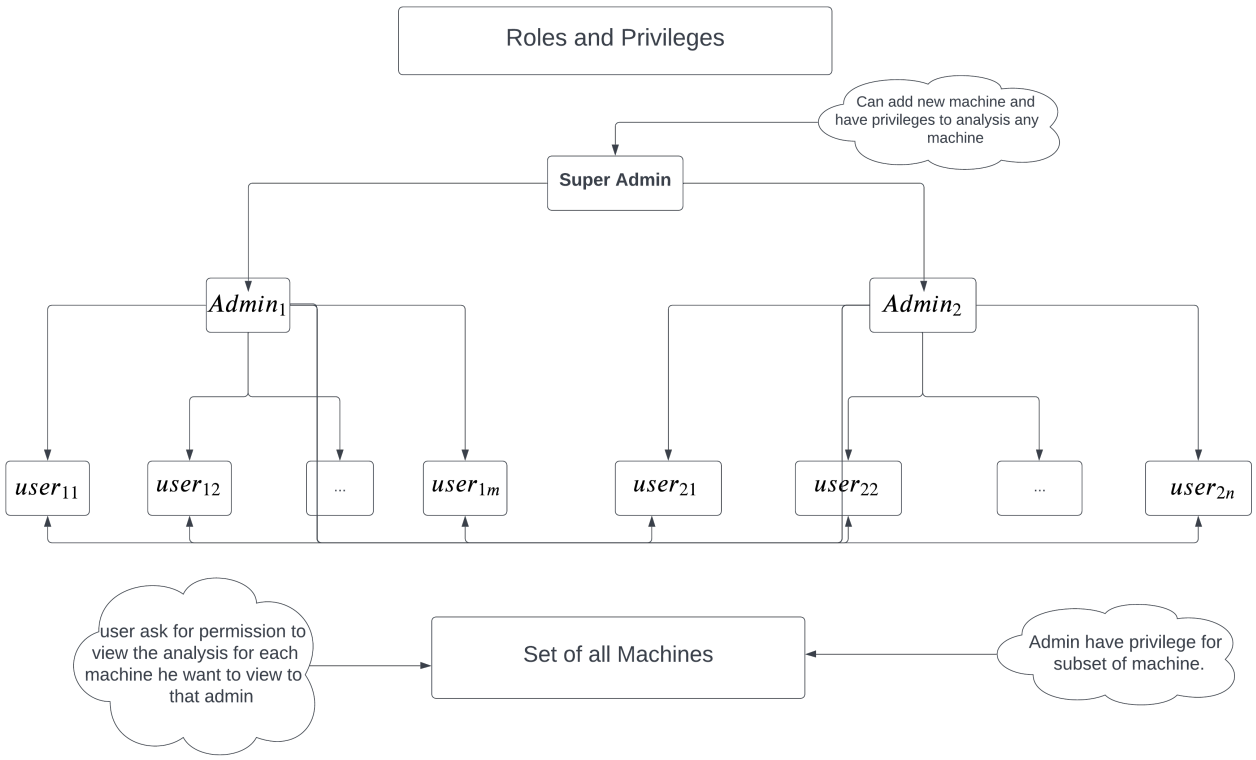


Figure 1: Roles & Privileges

- No user with any role (Super Admin, Admin, User) has the rights to alter the Influx database.
- Super Admin have the rights to grant privileges of nodes it has added and to grant the granting rights as well to admins.
- Admins can only grant privileges of nodes it has privilege on to Users and nothing else.
- Any user loses the privileges as soon as the corresponding entry is deleted in the postgresQL tables. (Views are updated)

Data Generation and Creation of tables

- Various Telegraf plugins are used for populating and creating tables in InfluxDB.
- A script is deployed on the host machines to run telegraf threads which feeds to the database hosted at the controller machine.
- The controller also maintains a link count for each host machine as multiple users may be monitoring the machine so we only have to run the script when link count goes from 0 to 1 and stop the threads when the link count goes back to 0.
- The controller uses ssh connection to start or terminate the script.

Model-View-Controller Design

For our application we follow an extended model-view-controller design. We have the traditional static view architecture coupled with a Socket server for real-time views. The socket server takes the triggers from the DRF and passes it on to Angular to update the charts and statistics in real-time.

We facilitate the **MVC** using **Django REST Framework**. Here we broadly explain what our Models, Views and Controllers are:

Models : We have two broad divisions of models. One is stored in PostgreSQL and one in Influx. PostgreSQL DB stores the models for the Users, SuperAdmin, Admin, Person, AdminNode, Node, UserAdminNode. InfluxDB has a bucket setup and stores all the data in a single table, in separate columns. It has fields for storing the hard and soft resource (listed in the **InfluxDB Bucket Setup & Schema** section) usage of the servers.

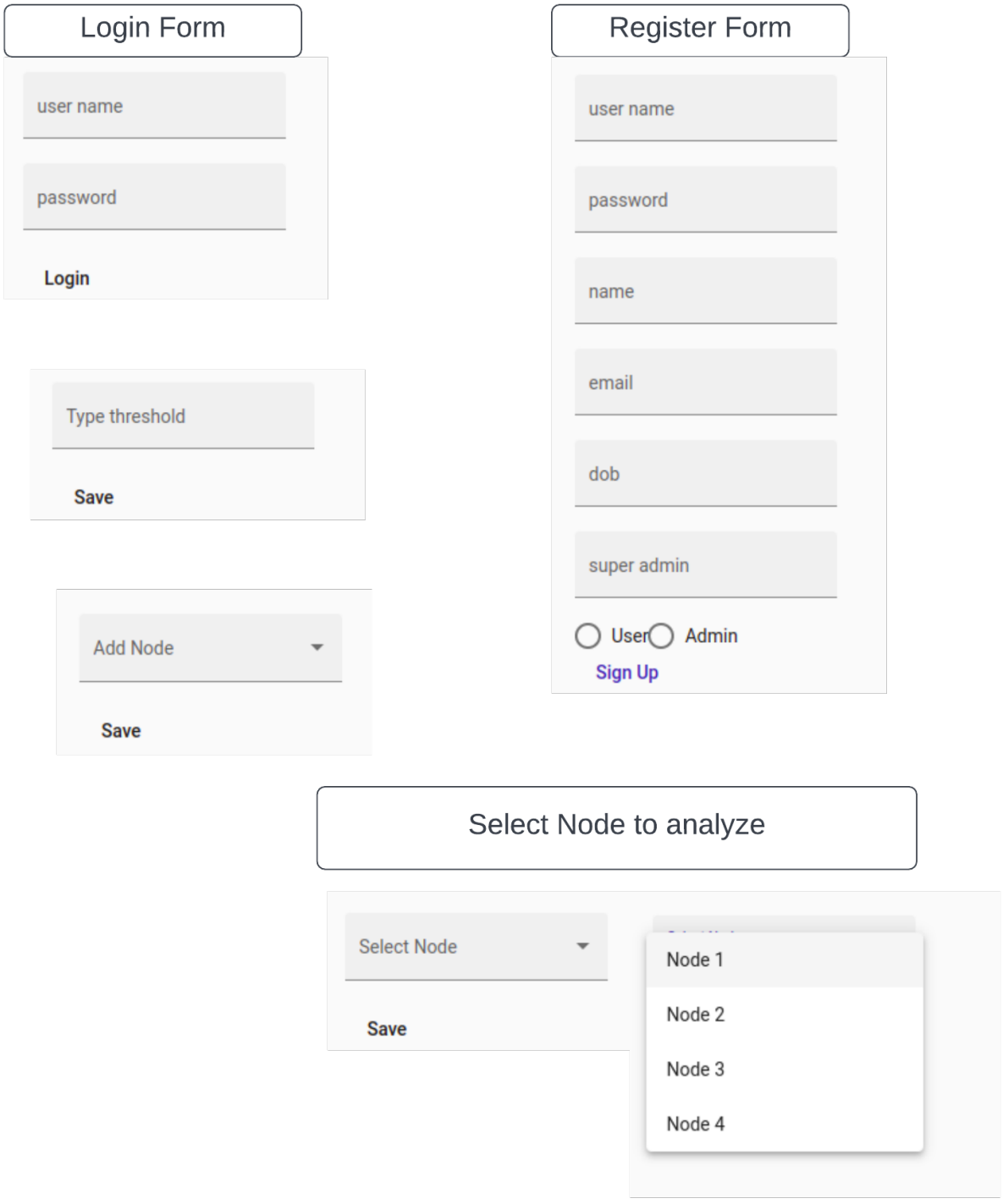
Views : We write our views in DRF. We will send the required data for rendering the views by exposing a REST API. We render the views using Angular. We will have multiple views for login, registration, admin page (for granting privileges and adding servers), main dashboard (for monitoring the statistics of the servers, and accessing the query panel).

Controller : Users can interact with our rendered views on the frontend (Angular). We have Login/Registration forms, query panel in the dashboard, forms to change the thresholds for different resources, adding new servers to monitor, etc. These serve as controllers for the users. DRF takes the response from these control parameters and sends required updates to PostgreSQL and InfluxDB.

User Interaction: Form Design and Technologies

We are using **Angular** for the frontend. All the required forms for the user interaction can be easily implemented using `angular forms` which will send data to backend for the appropriate action. Following are the rough list of the forms we will be using for the user interaction:

- Register User/ Sign Up
- Login
- Select Node(i.e Machine to be analyze) by the user
- New Node(i.e Machine to be analyze) added by the super admin
- Flux query written by the user in the query panel
- Threshold change (i.e user choice threshold for giving alert)


Login Form
user name
password
Login
Register Form
user name
password
name
email
dob
super admin
User
Admin
Sign Up
Type threshold
Save
Add Node
Save
Select Node to analyze
Select Node
Save
Node 1
Node 2
Node 3
Node 4

Form Sketch