# StudySpot

System Design
Ammar Mohamed, Andrew Maged, Faris Maali
EECS 3311
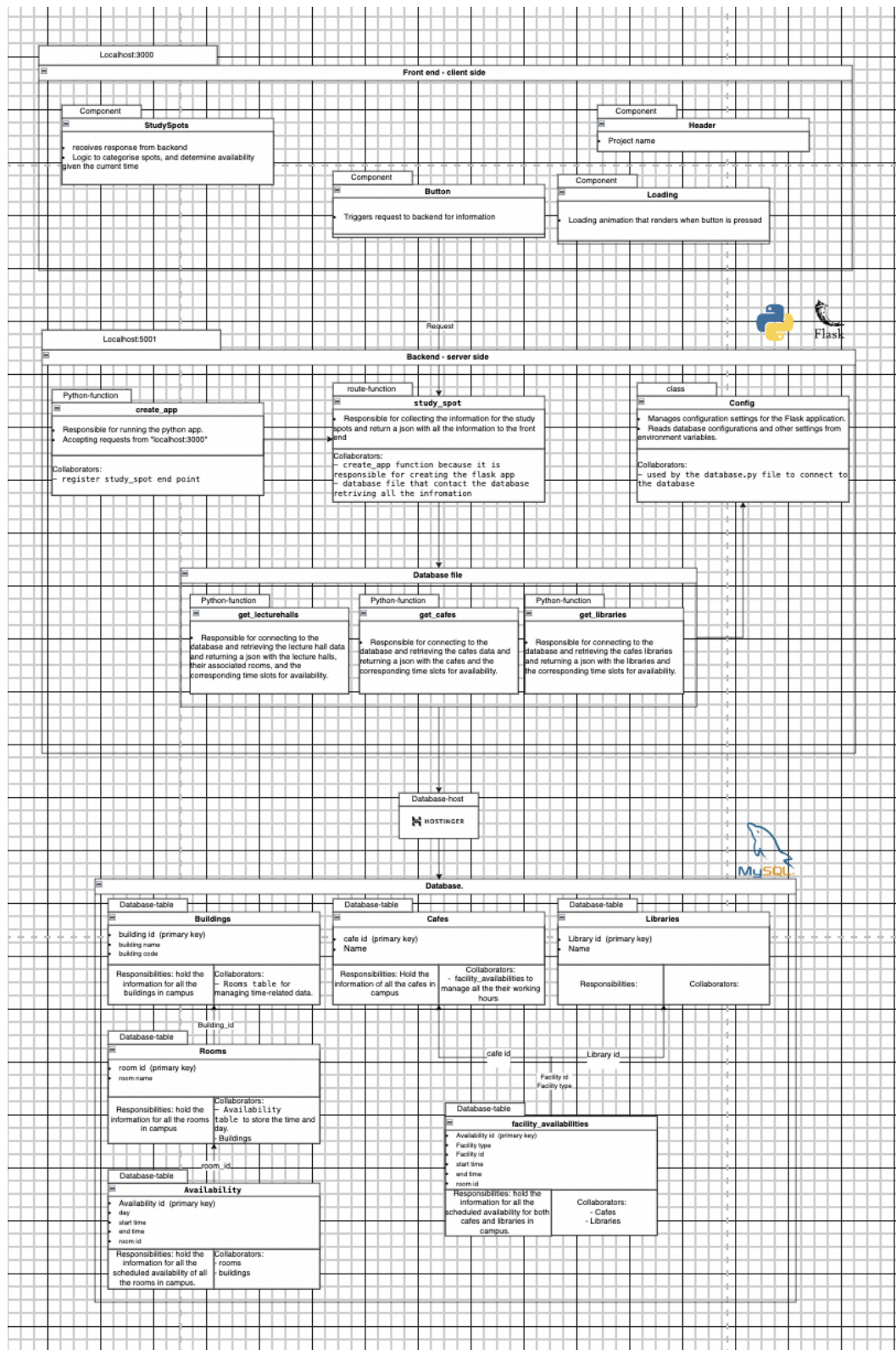21 November 2024
Sprint 1

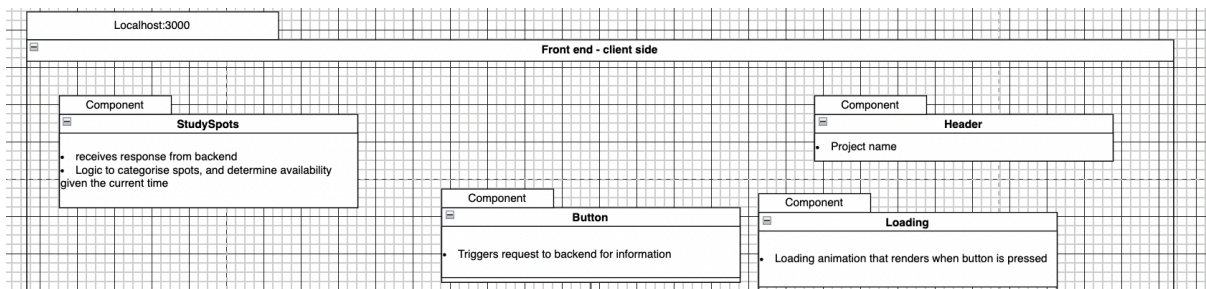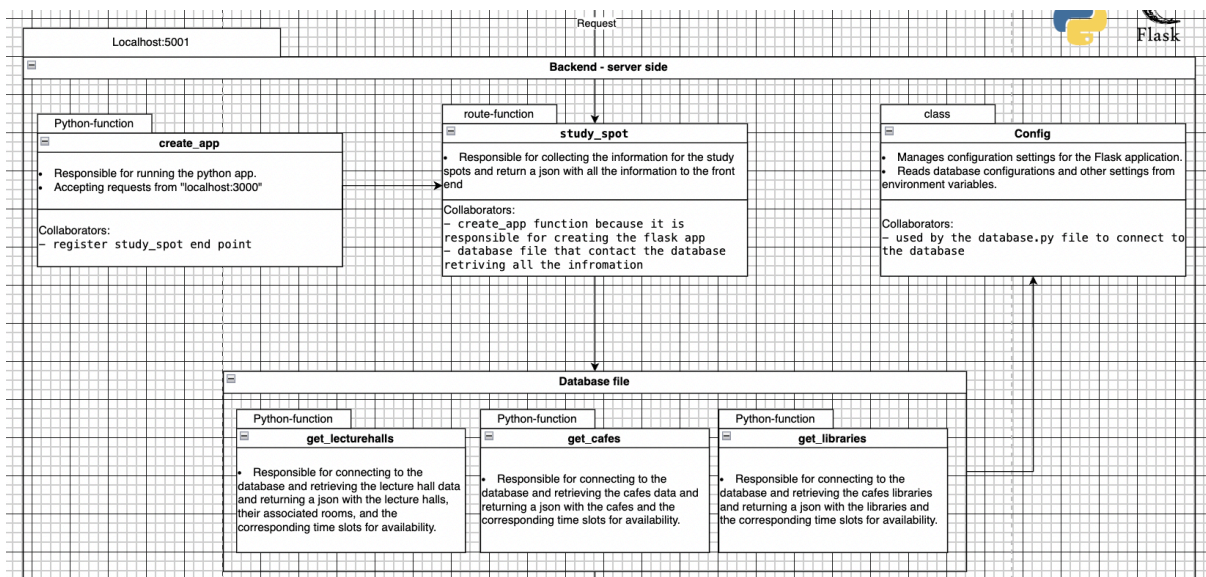Table of contents

# CRC cards:

We show detailed CRC cards for our backend and database, and we show the various components of the frontend. Here is a detailed diagram:
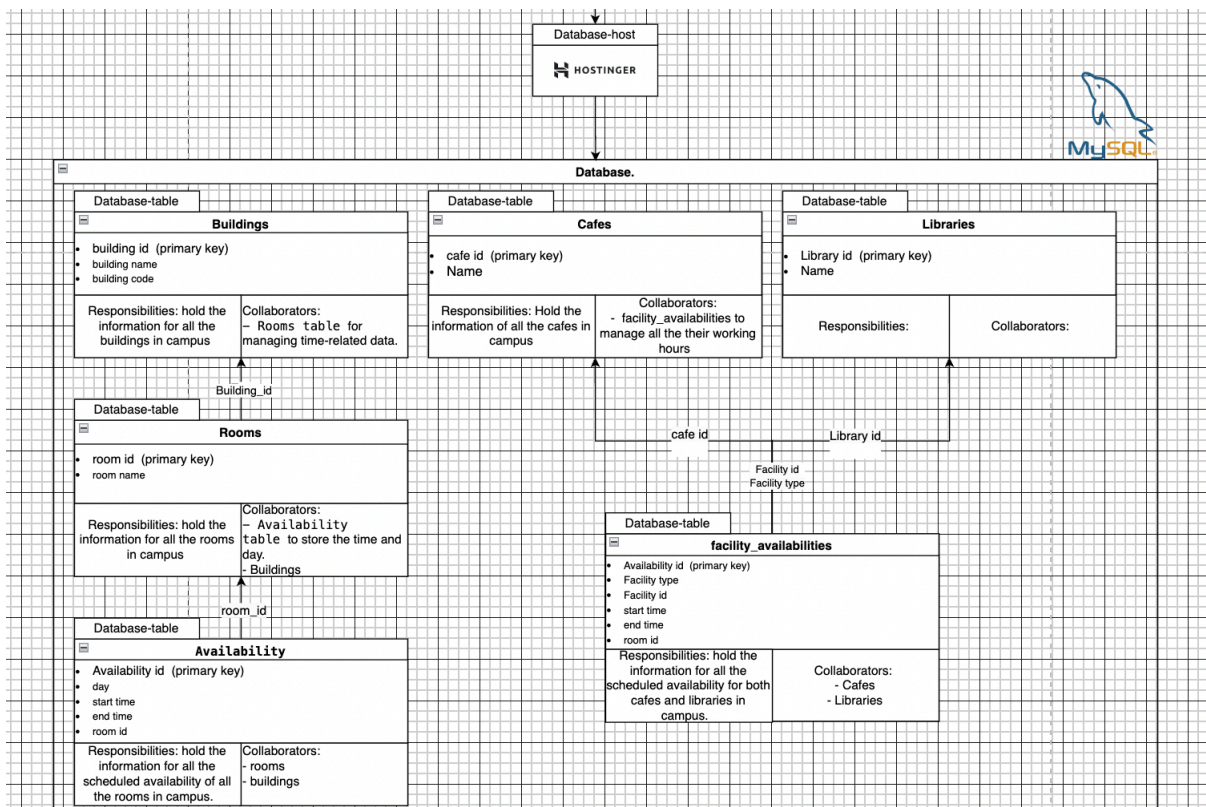
# Frontend

**Localhost:3000**

**Front end - client side**

**Component**

**StudySpots**

- receives response from backend
- Logic to categorise spots, and determine availability given the current time

**Component**

**Header**

- Project name

**Component**

**Button**

- Triggers request to backend for information

**Component**

**Loading**

- Loading animation that renders when button is pressed

# Backend:

Request

Flask

**Localhost:5001**

**Backend - server side**

**Python-function**

**create_app**

- Responsible for running the python app.
- Accepting requests from "localhost:3000"

Collaborators:
- register study_spot end point

**route-function**

**study_spot**

- Responsible for collecting the information for the study spots and return a json with all the information to the front end

Collaborators:
- create_app function because it is responsible for creating the flask app
- database file that contact the database retriving all the infromation

**class**

**Config**

- Manages configuration settings for the Flask application.
- Reads database configurations and other settings from environment variables.

Collaborators:
- used by the database.py file to connect to the database

**Database file**

**Python-function**

**get_lecturehalls**

- Responsible for connecting to the database and retrieving the lecture hall data and returning a json with the lecture halls, their associated rooms, and the corresponding time slots for availability.

**Python-function**

**get_cafes**

- Responsible for connecting to the database and retrieving the cafes data and returning a json with the cafes and the corresponding time slots for availability.

**Python-function**

**get_libraries**

- Responsible for connecting to the database and retrieving the cafes libraries and returning a json with the libraries and the corresponding time slots for availability.

# Database:

**Database-host**

HOSTINGER

MySQL

**Database.**

**Database-table**

**Buildings**

- building id  (primary key)
- building name
- building code

| Responsibilities: hold the information for all the buildings in campus | Collaborators:<br>- Rooms table for managing time-related data. |

**Database-table**

**Cafes**

- cafe id (primary key)
- Name

| Responsibilities: Hold the information of all the cafes in campus | Collaborators:<br>- facility_availabilities to manage all the their working hours |

**Database-table**

**Libraries**

- Library id (primary key)
- Name

| Responsibilities: | Collaborators: |

Building_id

**Database-table**

**Rooms**

- room id  (primary key)
- room name

| Responsibilities: hold the information for all the rooms in campus | Collaborators:<br>- Availability table to store the time and day.<br>- Buildings |

room_id

**Database-table**

**Availability**

- Availability id  (primary key)
- day
- start time
- end time
- room id

| Responsibilities: hold the information for all the scheduled availability of all the rooms in campus. | Collaborators:<br>- rooms<br>- buildings |

cafe id

Library id

Facility id
Facility type

**Database-table**

**facility_availabilities**

- Availability id  (primary key)
- Facility type
- Facility id
- start time
- end time
- room id

| Responsibilities: hold the information for all the scheduled availability for both cafes and libraries in campus. | Collaborators:<br>- Cafes<br>- Libraries |

# Software Architecture Diagram:

Our project follows the MVC architecture closely, with the view being our front end, implemented with Next.js, React.js, TailwindCSS and Typescript. Our controller is the python-flask api, and we have our model that communicates with our database to fetch the desired results. The diagram is given below.