



FGF Brands

# HACKATHON

**Team 5:** Shaan, Navdeep,  
Jaimil, Nikhil, Simran

FGF

# PROBLEM 1:

## Meeting Schedule Automation



### Our Algorithm:

- **Saves time** by automating scheduling and room selection
- **Reduces Errors** in scheduling and room allocation
- **Optimizes** use of resources (time and space)



# INTRODUCING: ROOMMATE

## How does it work?

- RoomMate uses predefined rooms, with their capacities: ex. 1235 Ormont Blueberry (6 people)
- User inputs:
  - Attendee's name
  - Email
  - # of attendees
  - Date
  - Time
  - Duration
- RoomMate uses inputted data to check room availability and books an available room



```
products: storeProducts
)
{
  return (
    <React.Fragment>
      <div className="py-5">
        <div className="container">
          <Title name="ur" title="Roommate" />
          <div className="row">
            <ProductConsumer>
              {(value) =>
                <div>
                  <h1>Roommate</h1>
                  <p>Book a room</p>
                  <Form>
                    <FormRow>
                      <FormInput type="text" placeholder="Room Name" />
                      <FormInput type="text" placeholder="Attendees Name" />
                      <FormInput type="text" placeholder="Email" />
                      <FormInput type="text" placeholder="Date" />
                      <FormInput type="text" placeholder="Time" />
                      <FormInput type="text" placeholder="Duration" />
                    </FormRow>
                    <FormRow>
                      <FormText>Available</FormText>
                      <FormText>Booked</FormText>
                    </FormRow>
                    <FormRow>
                      <FormText>Available</FormText>
                      <FormText>Booked</FormText>
                    </FormRow>
                  </Form>
                </div>
              }
            </ProductConsumer>
          </div>
        </div>
      </div>
    </React.Fragment>
  )
}
```

# WHAT ELSE?

## Date and Time Conversion

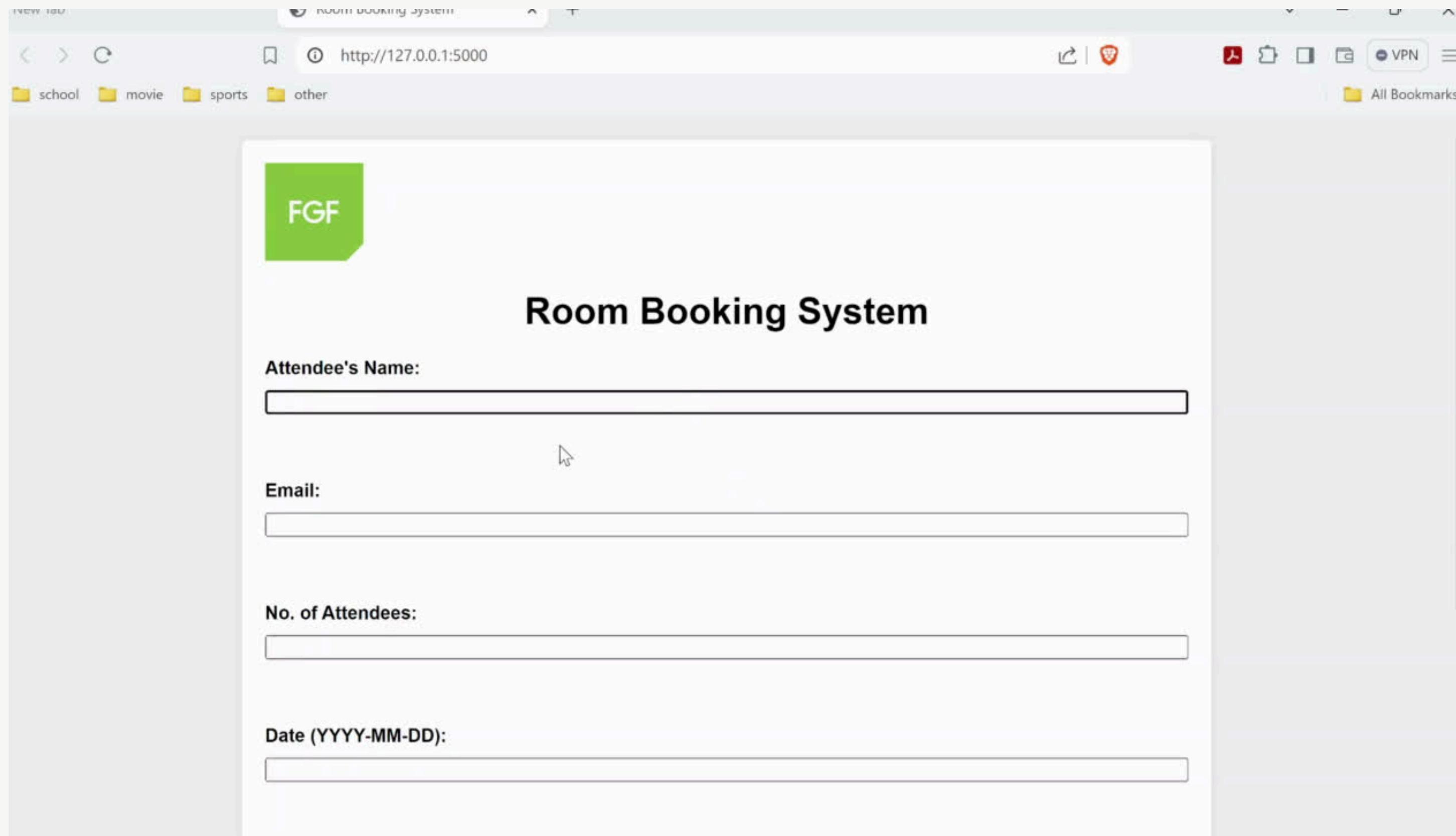
- Converts input date and time to timezone-aware datetime objects
- Book meetings from anywhere around the world!

## Checks Room Availability + Allocation

- Uses Outlook calendar to check room availability
- Sorts + filters appointments to find conflicts
- If a suitable room is found, RoomMate **creates a new appointment and sends a meeting invite** to attendees emails.



# USER INTERFACE



# PROBLEM 5:

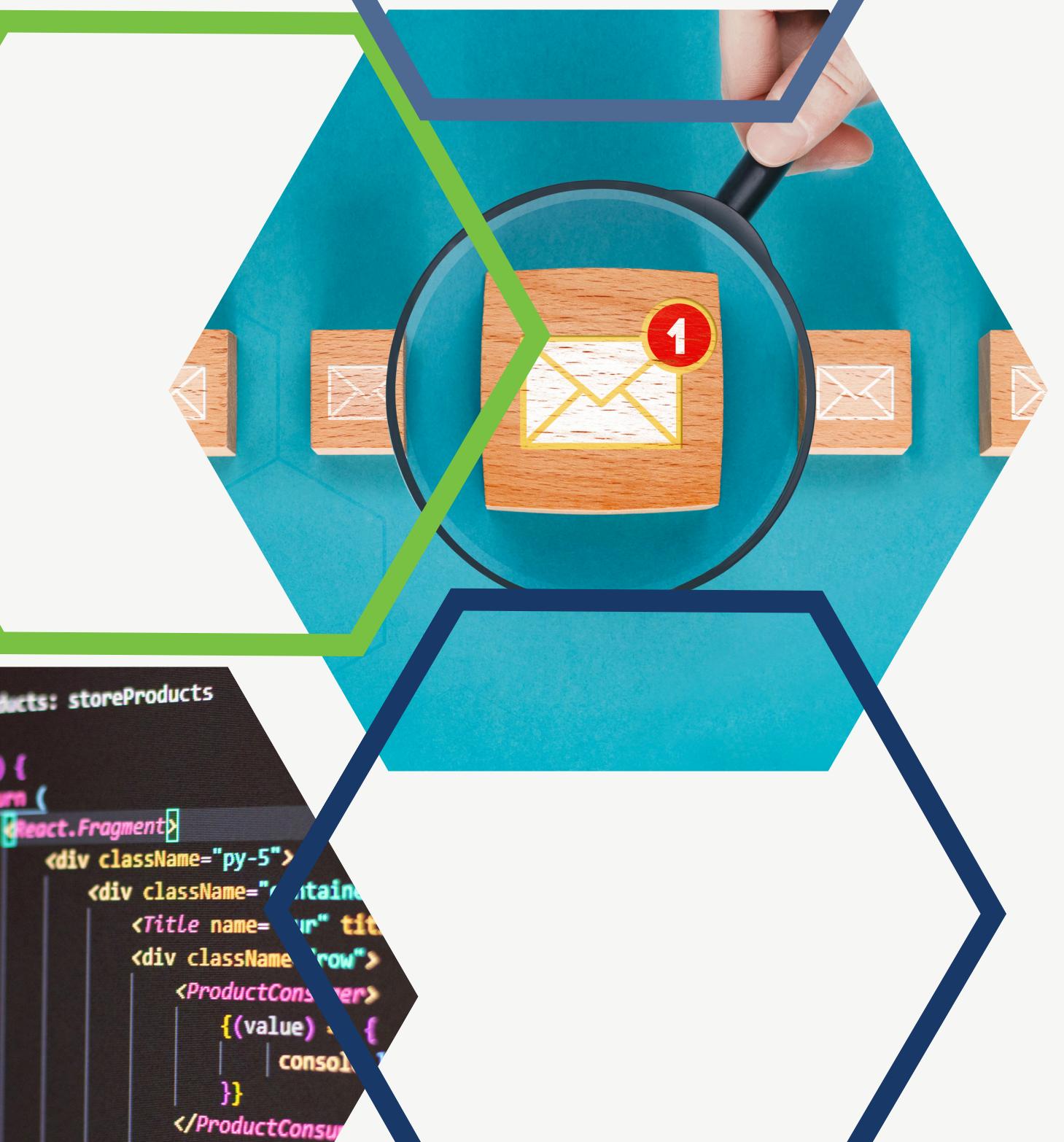
## Email Overload Management

### OUR ALGORITHM:

- **Automatically prioritizes** incoming emails based on urgency and relevance
- **Increases Productivity** for employees by displaying critical messages as a priority

# HOW DOES IT WORK?

- Using **NLP** (Natural Language Processor) the algorithm categorizes incoming emails into four categories:
  - **Urgent, Informational, Actional, Delegate**
- The model **predicts categories based on email content**
- The user interface provides a form for users to paste email content
  - Upon submission, the **algorithm categorizes the email and saves it** in the appropriate category



# CODE

The screenshot shows a code editor interface with a dark theme. The top bar includes icons for file operations, a project dropdown labeled "pythonProject", a "Version control" dropdown, and various toolbars. The main area displays a Python file named "APP.PY". The code imports several libraries including os, nltk, datetime, and various sklearn modules for text processing and machine learning. It then initializes a Flask application and uses NLTK to download stopwords, punkt, and wordnet datasets. The bottom panel shows a terminal window with the output of the application's execution, indicating it is active and listing several log entries from a local host IP address.

```
import os
import nltk
import datetime
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import SVC
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from nltk.stem import WordNetLemmatizer
import string
from flask import Flask, request, render_template

app = Flask(__name__)

# Initialize NLTK stopwords and tokenizer
nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')
```

```
* Debugger is active!
* Debugger PIN: 258-811-220
127.0.0.1 - - [05/Jul/2024 16:08:54] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [05/Jul/2024 16:08:54] "GET /static/logo.png HTTP/1.1" 304 -
127.0.0.1 - - [05/Jul/2024 16:10:34] "POST /submit HTTP/1.1" 200 -
127.0.0.1 - - [05/Jul/2024 16:10:34] "GET /static/logo.png HTTP/1.1" 304 -
127.0.0.1 - - [05/Jul/2024 16:10:34] "GET /static/informational.png HTTP/1.1" 304 -
```

# USER INTERFACE

The screenshot shows a web page titled "FGF Internal Website". At the top left is a green square icon containing the letters "FGF". Below the title, there is a legend with four categories: "Urgent: Emails needing immediate attention.", "Informational: Emails providing updates or reports.", "Actionable: Emails requiring specific actions or responses.", and "Delegate: Emails to be forwarded or delegated to others.". A large text input field is present with the placeholder text "Paste your email content here...". A blue "Submit" button is located at the bottom right of the input field.

The screenshot shows a web page with a green "FGF" icon at the top left. Below it, a message states "The email has been categorized as: Informational". Underneath this message is a circular icon with a white "i" inside, representing information. Below the icon is a blue "Check Again" button.

# PROBLEM 3:

## Email Reply Suggestions

### OUR ALGORITHM:

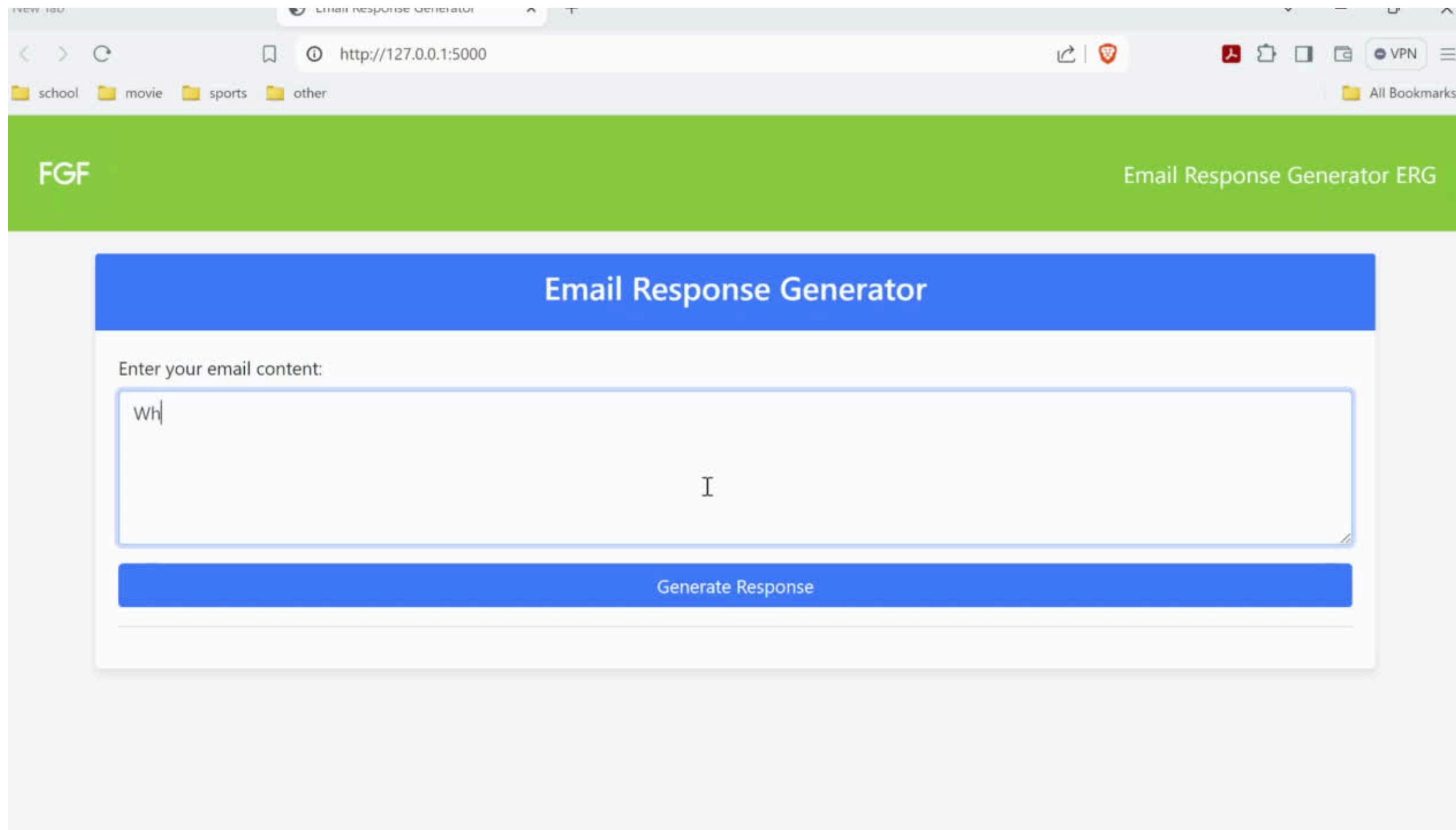
- **Saves Time** by suggesting relevant responses quickly
- **Ensures Consistency** through responses that align with the tone and intent of the email
- **Can Adapt** suggestions based on context of the sender

# HOW DOES IT WORK?

- Using predefined patterns, the algorithm automates email responses based on keywords
- The user inputs the email on the interface
- Utilizes NLP using spacy API



# USER INTERFACE:



# THANK YOU!

-TEAM 5!

FGF

