On-Demand Event Scheduler

About the project

· The On-demand Event Scheduling platform seeks to develop a meeting management tool that makes it simple for users to schedule, monitor, plan, and reschedule meetings from anytime, anywhere.

· It eliminates the need for back-and-forth emails and phone calls, guaranteeing a smooth scheduling experience for all participants.

· This web application offers a simple and user-friendly interface, which increases reliability and usage.

· The application simplifies the process by providing a visual representation of the optimal meeting times.

· It includes support for different time zones, facilitating coordination with individuals in various locations of the world.

Features of the project

· The application allows users to register and log in to access its features.

· Users have the option to create a poll to find common availability, whether they have an account or not.

· A user's schedule is synced with their personal calendar after they register an account and confirm their availability via the poll.

· The application takes the user's location and calendar information into account when handling time zone restrictions. This guarantees proper scheduling and cooperation across several time zones.

· Users are given access to a booking page, allowing them to set up meetings with or without recurrence. Meetings between one person and numerous people are supported by this feature.

· End users can make use of tools like personalized task, birthday, and to-do reminders based on their individual needs.

Motivation/Problem approached

The motivation behind the development of this product is to address the challenges and inefficiencies associated with scheduling and managing events, particularly meetings, in a streamlined and effective manner.

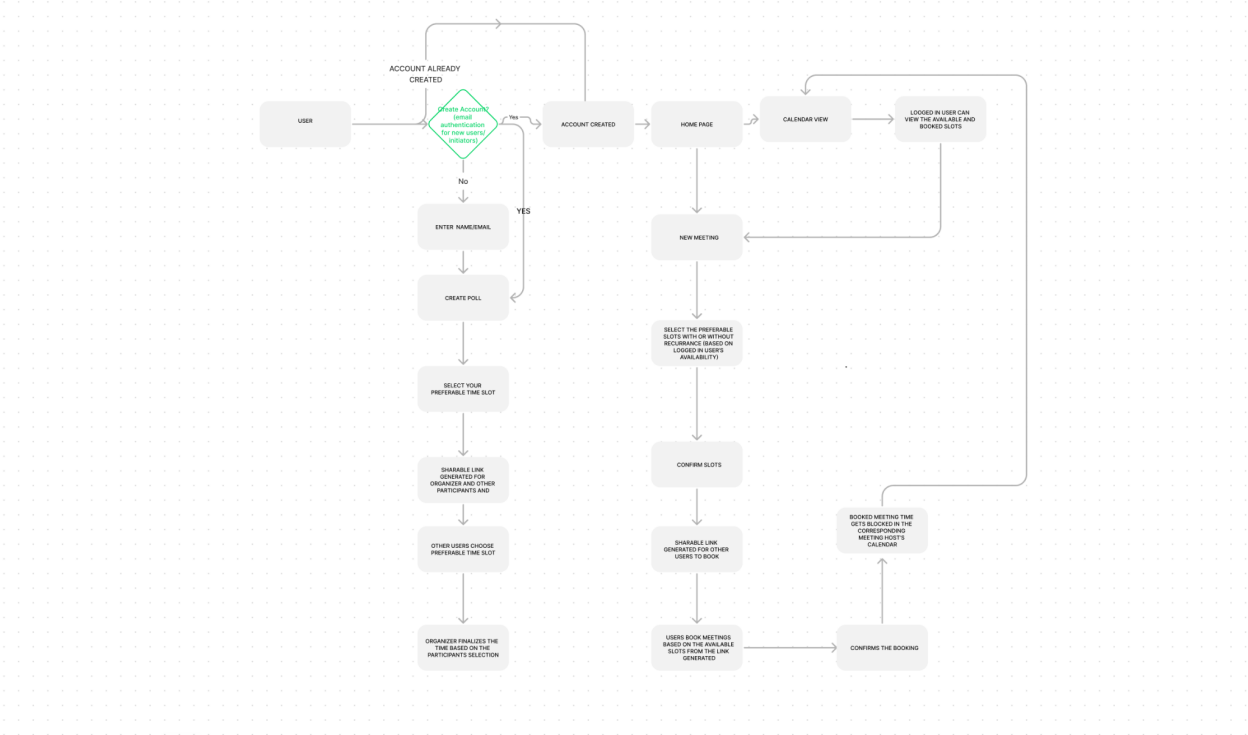
· The solution offers a centralized platform where users can effectively organize and schedule different event kinds, such as one-on-one meetings, group meetings, and create polls to fix on a particular meeting time. It saves time and effort that would otherwise be required to coordinate and organize these events through several channels by providing a one-stop solution.

· Users can easily check the availability of individuals or their own schedule within the system. This eliminates the need for contacting each person individually to inquire about their availability, saving time and avoiding unnecessary communication.

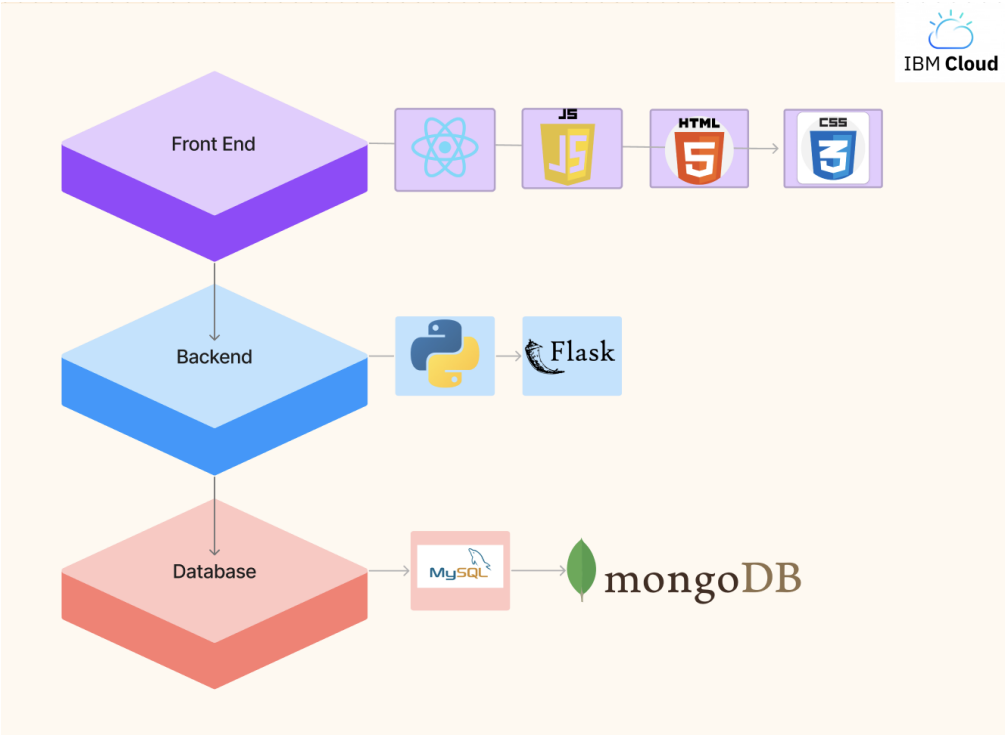
· By showing a visual depiction of the ideal meeting times, the platform aids in the settlement of disagreement during meetings. By identifying timeslots during which everyone is available, this tool lessens scheduling disputes and promotes better team member collaboration.

. One significant improvement is the elimination of the need for users to create accounts or log in to access and book appointments with professors or corporates. This decentralization approach allows users to simply poll available times and schedule appointments without the burden of signing up or remembering passwords.

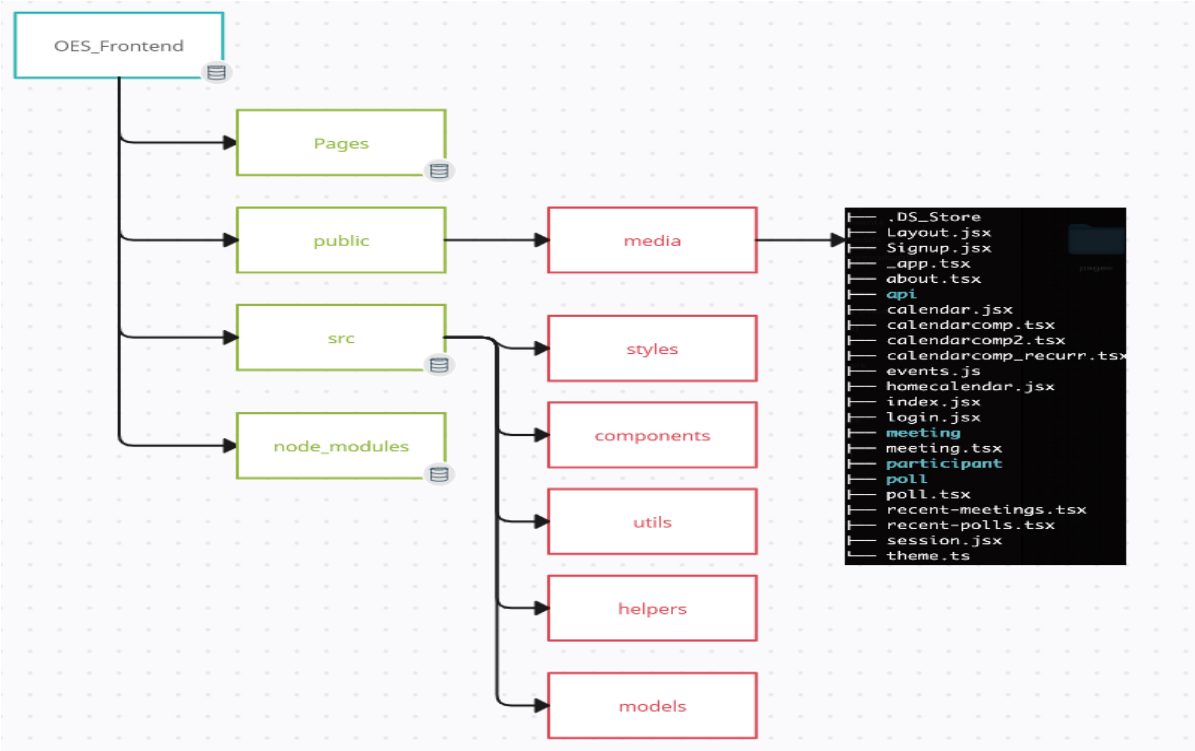
Architecture



Technical Design:



Directory Organization:



Deployment Steps

Frontend

Steps to be done before deployment:

1.Update the config in the .env file as given below:

.env

```
NEXT_MONGODB_URI=NEXT_MONGODB_URI is the connection string

ROOT_URL=http://localhost:3001 #ROOT_URL is the base URL of your deployment; like https://www.oes.com


# Key and IV for data encryption and decryption
# Generate NEXT_PUBLIC_ENCRYPTION_KEY using $openssl rand -hex 16
# Generate NEXT_PUBLIC_ENCRYPTION_IV using $openssl rand -hex 8

NEXT_PUBLIC_ENCRYPTION_KEY=
NEXT_PUBLIC_ENCRYPTION_IV=

```

- Build the Docker Image:
    - Navigate to the directory cse611-spring2023-OnlineEventScheduler/Frontend/OES_Fronten and run the command: docker build --platform linux/amd64 -t oes:frontend
- Push the Docker Image to Docker Hub:
    - Login to the Docker Hub account using docker login command
    - Tag the image appropriately with Docker Hub username with command: docker tag oes:frontend <dockerhub-username>/oes:frontend<version>
    - Push the image with command: docker push <dockerhub-username>/oes:frontend<version>
- Login to IBM Cloud, navigate to Code Engine:
    - Select the Project
    - Navigate to Applications > Create
    - Choose Container image and fill in the Docker Hub image registry details (will be in the format docker.io/<dockerhub-username>/oes:frontend<version>)
    - Once ready, the endpoint can be found under Domain Mappings > Public
    - The endpoint generated must be added to the env file.

If the application needs to be tested locally:
Run the below command in terminal after going to the root directory

-npm install --legacy-peer-deps
-npm run dev

Backend

- Build the Docker Image:
    - Navigate to the directory cse611-spring2023-OnlineEventScheduler/Backend and run the command: docker build --platform linux/amd64 -t oes:backend .
- Push the Docker Image to Docker Hub:
    - Login to the Docker Hub account using docker login command
    - Tag the image appropriately with Docker Hub username with command: docker tag oes:backend <dockerhub-username>/oes:backend<version>
    - Push the image with command: docker push <dockerhub-username>/oes:backend<version>
- Login to IBM Cloud, navigate to Code Engine:

    - Select the Project
    - Navigate to Applications > Create
    - Choose Container image and fill in the Docker Hub image registry details (will be in the format docker.io/<dockerhub-username>/oes:backend<version>)
    - Once ready, the endpoint can be found under Domain Mappings > Public

Database

- Create a Google Cloud SQL instance:
    - Go to the Google Cloud Console (console.cloud.google.com) and create a new project.
    - Navigate to the "SQL" section in the left-hand menu.
    - Click on "Create Instance" to start setting up your MySQL database.
    - Configure the instance settings such as region, database version, and instance type. Provide a root password for the MySQL instance.
- Export your local database:
    - On your local machine, use the mysqldump command to export your MySQL database. This will create a SQL dump file containing your database schema and data.
    - Run the following command:
        - mysqldump -u [username] -p [database_name] > [dump_file_name].sql
- Upload the SQL dump file to Google Cloud Storage:
    - Go to the Google Cloud Console and navigate to the "Storage" section in the left-hand menu.
    - Create a new bucket to store your SQL dump file.
    - Upload the SQL dump file to the selected bucket.
- Import the SQL dump file to the Cloud SQL instance:
    - Go back to the Google Cloud Console and navigate to the "SQL" section.
    - Click on your created Cloud SQL instance.
    - In the instance details page, select the "Import" tab.
    - Specify the location of the SQL dump file you uploaded to Google Cloud Storage.
    - Choose the database version, database, and other import settings as necessary.
    - Start the import process.

- Connect to the Cloud SQL instance:
    - After the import process completes, you can connect to your Cloud SQL instance to verify the database.
    - From the instance details page, click on "Connect using Cloud Shell" or use any other MySQL client of your choice to connect to the database.
    - Provide the necessary connection details (host, port, username, and password) to establish the connection.

Testing

- Import the Selenium Project:
    - Launch Eclipse IDE.
    - Select "File" -> "Import" from the menu.
    - In the import wizard, expand the "General" folder and select "Existing Projects into Workspace". Click "Next".
    - Choose the root directory where your Selenium project is located or select the archive file (.zip) containing the project.
    - Click "Finish" to import the project into Eclipse.
- Set Up Selenium WebDriver:
    - Download the appropriate WebDriver executable for the web browser you intend to automate (e.g., ChromeDriver for Google Chrome).
    - Place the WebDriver executable in a specific location within your project directory.
    - Update the path to the WebDriver executable in your code to match the location on your system.
- Resolve Dependencies:
    - Right-click on the project in Eclipse and select "Properties".
    - In the properties dialog, navigate to "Java Build Path" -> "Libraries" tab.
    - Click "Add External JARs" and browse to the location where you have saved the Selenium JAR files.
    - Select all the JAR files and click "Open" to add them to the project's build path.
    - Click "Apply" and then "OK" to save the changes.
- Run the Code:
    - Open the Java file containing the main method (e.g., TestPM.java) in Eclipse.
    - Right-click on the file and select Run As -> “Java Application”.

- Eclipse will compile the code and execute it using the specified WebDriver and browser configuration.
- You will see the execution progress and any output or errors in the Console view.

Work Breakdown:

Team Members	Modules Worked on	Work percentage split up across various aspects
Lakshaye Vaikunth	1.Poll Flow: ● Link Generation module with encryption ● Participant polling page ● Poll finalized page ● REST API Handler for GET, PUT and POST methods. 2.Meeting Flow: ● Account sign up page ● Meeting form ● Appointment slot selection component. ● REST API handler for fetch and update slots. ● Meeting slots generated page. ● Recurrence flow-Monthly 3.Standalone Pages: Homescreen 4. Deployment: ● Created docker file based on the project requirement. ● Built docker container for deployment. ● Deployed the front-end	● Frontend Development-70% ● Standalone UI components development-5% ● Backend endpoints Design Discussions-10% ● Deployment-10% ● Testing - 5%

	docker container on the IBM cloud. Documentation: Worked on functional flow design of the application. Worked on data models and schemas.	
Arjun	1.Poll Flow: ● Poll form ● Timezone component ● Survey end date validation 30 days validation and expiry. ● Calendar component 2.Meeting Flow: ● Login page and validations ● REST API handler for create meeting and retrive to calendar component. ● Recurrence flow-Daily, Weekly ● My calendar functionality 3.Navigation bar and routing modules 4.Standalone Pages: About 5. Deployment: ● Created docker file based on the project requirement. ● Built docker container for deployment. ● Deployed the front-end docker container on the IBM cloud.	● Frontend Development-60% ● Standalone UI components development-25% ● Deployment-10% ● Testing - 5%

	6. Documentation: Worked on functional flow design of the application. Worked on data models and schemas.	
Harshitha	Final Meeting Flow: ● Create Meeting API ● Get Meeting API ● Meeting Registration API Old Poll Flow: ● Get Participant Details API ● Updates to Get Poll API ● Updates to Initiate Poll API ● Participant Availability API ● Get Participant Poll API ● Finalize Poll API SignUp/Login: ● SignUp API ● Login API GCP: ● OES Instance creation ● Backend Deployment ● MySQL buckets creation ● MySQL Database Deployment ● Configuration of GCP Database to Backend Swagger: ● Create Meeting Availabilities ● Retrieve Meeting Details ● Register for Meetings Documentation: ● Backend complete documentation: Intro, Poll and Meeting descriptions, Tech Stack, All APIs -	Backend - 50% Database - 50%

	Endpoints, RequestBody, Responses Miscellaneous: ● Initial Backend SetUp ● Schema Changes for Poll ● Schema Changes for Meetings ● Updates on Poll APIs ● Updates to Meeting APIs ● Backend unit testing through Postman ● Poster Design for Demo Day ● Queries for all backend APIs - Poll and Meeting ● Connection and storing all data from Backend to Database	
Divya	Final Poll Flow: ● Initiating poll POST API ● Get poll details Initiator View GET API ● Marking poll times PUT API ● Finalizing poll PUT API ● Get poll details Participant View GET API ● Removing poll DELETE API ● SignUp POST API ● Login POST API ● Implemented Initiator and participant view for APIs by validating secret while returning information SignUp/Login: ● Implemented Login backend module ● Implemented Signup backend module Deployment: ● Created backend	Backend - 50% Deployment - 60%

	Dockerfile with requirements and built docker container for deployment ● Deployed backend on IBM Swagger: ● Initiating poll ● Get poll details Initiator View ● Marking poll times ● Finalizing poll ● Get poll details Participant View ● Deleting poll ● SignUp for application ● Login for application Old Poll & Meeting Flows: ● Created APIs as per older design for ○ Initiate poll ○ Get poll ○ Participant info ○ Get participant details ○ Mark availability for participants ○ Finalizing poll ● Created APIs as per initial design for ○ Meeting registration ○ Get meeting details ● Created initial schemas for poll and meeting API requests Documentation: ● Poll Functionality ● Backend Deployment Miscellaneous: ● Handled Time Zones at backend ● Handled expiry date	

	validation at backend ● Created schemas for signup/login, poll and meeting API requests ● Tested deployment of docker containers on GCP for frontend and backend ● Backend unit testing through Postman ● Updated SQL Schema as per new poll design for Poll tables	
Venkatesh	Database: ● Polling Flow ○ Schema for Polling. ○ Schema for Poll participants. ○ SQL queries for the backend. ○ Dummy data for backend testing. ● Meeting Flow ○ Schema for Meeting. ○ Schema for recurring Meetings ○ SQL queries for the backend. ○ Dummy data for backend testing. ● Schema Changes for the complete flow. ● Database connection to backend. Deployment: ● Created instance for Database and deployed in GCP. ● Created buckets for MySQL instances. ● Configuration for OES database.	Database: 50% Frontend Testing: 100%

	Testing: ● Selenium Automation for Frontend Testing. ● Frontend test script for Meeting and Polling flow. ● Edge case testing with boundary conditions. ● Test script for all use cases.	

Course/Meeting Discussions:

- Throughout the course, we had the privilege of engaging in regular meetings with you, during which we received valuable feedback and guidance.
- The meetings with the professor and TA served as invaluable opportunities for us to retrospect on our progress and identify areas for improvement.
- We actively listened to professor's and TA's comments, carefully documented them, and embraced them as stepping stones towards achieving our project goals.
- One of the key takeaways from our meetings was the importance of creating a user-friendly interface that promotes efficiency and ease of use.
- Based on Professor's suggestions, we dedicated significant effort to revamp the user interface, streamlining the navigation and improving the overall aesthetics.
- By implementing a visually appealing design, we aimed to provide a seamless scheduling experience for our users.
- Incorporating the professor's guidance, we seamlessly integrated the recurrence functionality into our meeting system, resulting in a harmonious convergence of user-friendly interface and impactful functional flow.