Calendar Application for Communication Tracking

By
Likhitha Gudla
Likhithagudla7@gmail.com

Overview

The **Calendar Application for Communication Tracking** is a full-stack application that helps organizations track and manage their communications with other companies. It includes functionalities such as calendar integration, communication logging, reporting, notifications, and analytics to ensure timely and consistent follow-ups.

Table of Contents

Introduction

The **Calendar Application for Communication Tracking** helps users and administrators manage and track communication tasks with organizations. The app integrates a user-friendly interface with backend logic to ensure no communication task is missed.

Features

- **Communication Management**: Track the type, date, and details of each communication.
- **Calendar Integration**: Display past and future communications in an interactive calendar view.
- **Notifications**: Receive alerts for overdue and upcoming communications.
- **Analytics**: Generate reports and view communication trends and engagement.
- **Role-Based Access Control**: Admins can manage user roles and delegate tasks.
- **Customizable Communication Methods**: Define and manage communication methods such as emails, LinkedIn messages, etc.
- **Reporting**: Export communication logs in PDF and CSV formats.
- **Reminders**: Automated reminders for upcoming communications.

Technologies Used

- **Frontend**:
    - React.js
    - React Router
    - Axios for API calls
    - CSS/SCSS for styling
    - React Calendar (for calendar integration)
- **Backend**:
    - Node.js
    - Express.js
    - MongoDB (or MongoDB Atlas for cloud deployment)
- **Deployment**:
    - GitHub Pages (for frontend)
    - Heroku (for backend)

Setup Instructions

Prerequisites

Before setting up the project, ensure you have the following installed:

- **Node.js**: Download Node.js
- **MongoDB**: Download MongoDB or use MongoDB Atlas for cloud hosting.
- **Git**: Install Git

Backend Setup

1. Clone the repository to your local machine:

bash

Copy code

```
git clone https://github.com/your-repo-url/calendar-tracking-app.git
cd calendar-tracking-app
```

2. Navigate to the backend directory:

bash

Copy code

```
cd backend
```

3. Install the backend dependencies:

bash

Copy code

```
npm install
```

4. Set up your MongoDB connection in backend/config.js. If using MongoDB Atlas, replace the mongodb://localhost:27017/communication-tracking URL with your MongoDB Atlas connection string.

Example:

javascript

Copy code

```
mongoose. connect('mongodb://localhost:27017/communication-tracking', {
  useNewUrlParser: true,
  useUnifiedTopology: true
});
```

5. Start the backend server:

bash

Copy code

```
npm start
```

The backend will run on http://localhost:5000.

Frontend Setup

1. Navigate to the frontend directory:

bash
Copy code
cd frontend

2. Install the frontend dependencies:

bash
Copy code
npm install

3. Set up the API URL in the frontend to match the backend server. In frontend/src/api.js, update the API_URL variable:

javascript
Copy code
const API_URL = 'http://localhost:5000/api';

4. Start the frontend development server:

bash
Copy code
npm start
The frontend will run on http://localhost:3000.

Connecting Backend and Frontend

Ensure both the frontend and backend servers are running locally for proper communication. The frontend communicates with the backend using Axios for API requests.

Running the Application Locally

1. Start the backend server:

bash
Copy code
cd backend
npm start

2. Start the frontend server:

bash
Copy code
cd frontend
npm start

3. Visit http://localhost:3000 in your browser to access the application.

Deployment Instructions

Backend Deployment (Heroku)

1. Create a new Heroku application:

bash
Copy code
heroku create

2. Set up environment variables (e.g., MongoDB connection string, JWT secret) on Heroku.
3. Push the backend to Heroku:

bash
Copy code
git push heroku master

4. The backend will be accessible at a URL like https://your-backend-app.herokuapp.com.

Frontend Deployment (GitHub Pages)

1. Push the frontend to GitHub.
2. Link the GitHub repository to a deployment platform like **Netlify** or **Vercel** for automatic deployment.
3. Set the frontend to fetch data from the live backend API URL (e.g., https://your-backend-app.herokuapp.com).

After deployment, you can access the app at the live URL provided by your deployment platform.

Usage

1. **Admin Module**: The admin user can manage companies, communication methods, and schedule communications.
2. **User Module**: Users can view upcoming and overdue communications, log new communications, and access the calendar.
3. **Reporting and Analytics**: View and generate reports on communication trends and effectiveness.

Application Functionality

1. **Authentication**: Admin and User roles are supported with JWT-based authentication.
2. **Communication Tracking**: Users can log, edit, and delete communication entries.
3. **Calendar View**: Communications are displayed in a calendar interface.
4. **Notifications**: Alerts notify users of overdue and upcoming communications.
5. **Reports**: Communication logs can be exported in CSV or PDF formats.
6. **Analytics**: Basic analytics on communication frequency and effectiveness are available.

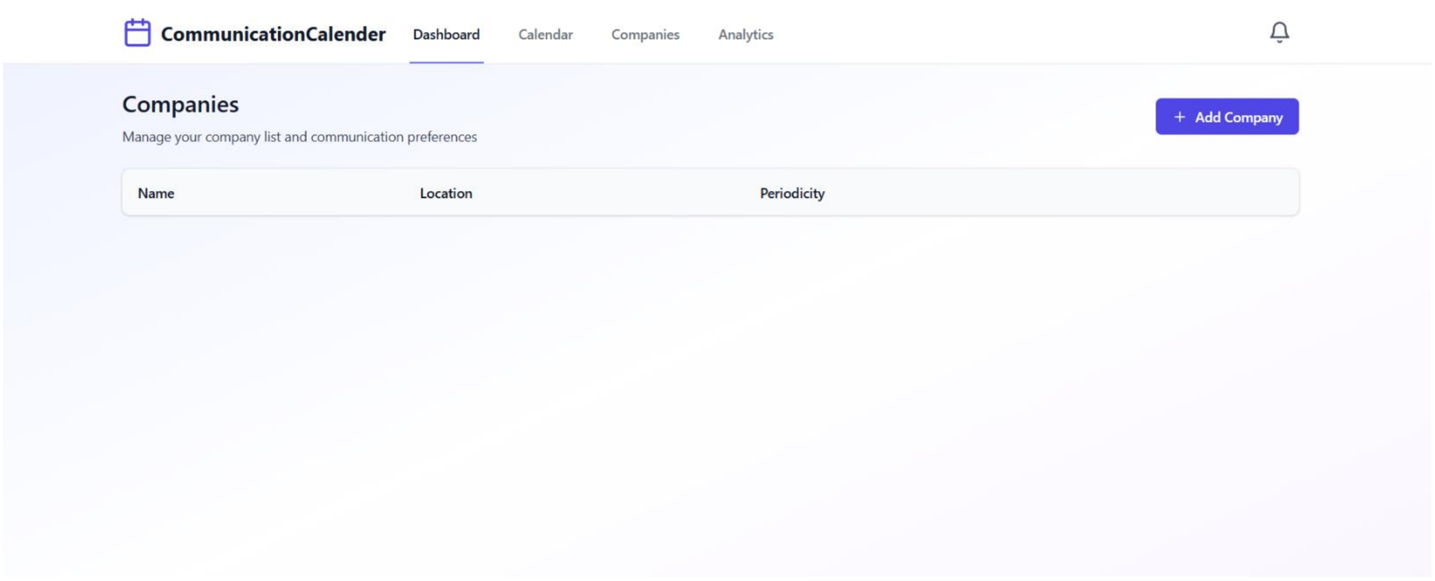
Known Limitations

1. **Limited Analytics**: The current analytics are basic and include only communication frequency. More advanced features will be added in future updates.
2. **Time zone Handling**: The calendar integration might not handle time zone differences perfectly.
3. **Scalability**: The application is designed for moderate use. Performance may degrade with very large datasets.
4. **User Roles**: Role-based access control is simple. Future versions will allow more complex user roles and permissions.
5. **Export Functionality**: Exporting data to CSV/PDF is functional, but future updates may allow more detailed export options (e.g., date ranges, communication types).

License

This project is licensed under the MIT License.

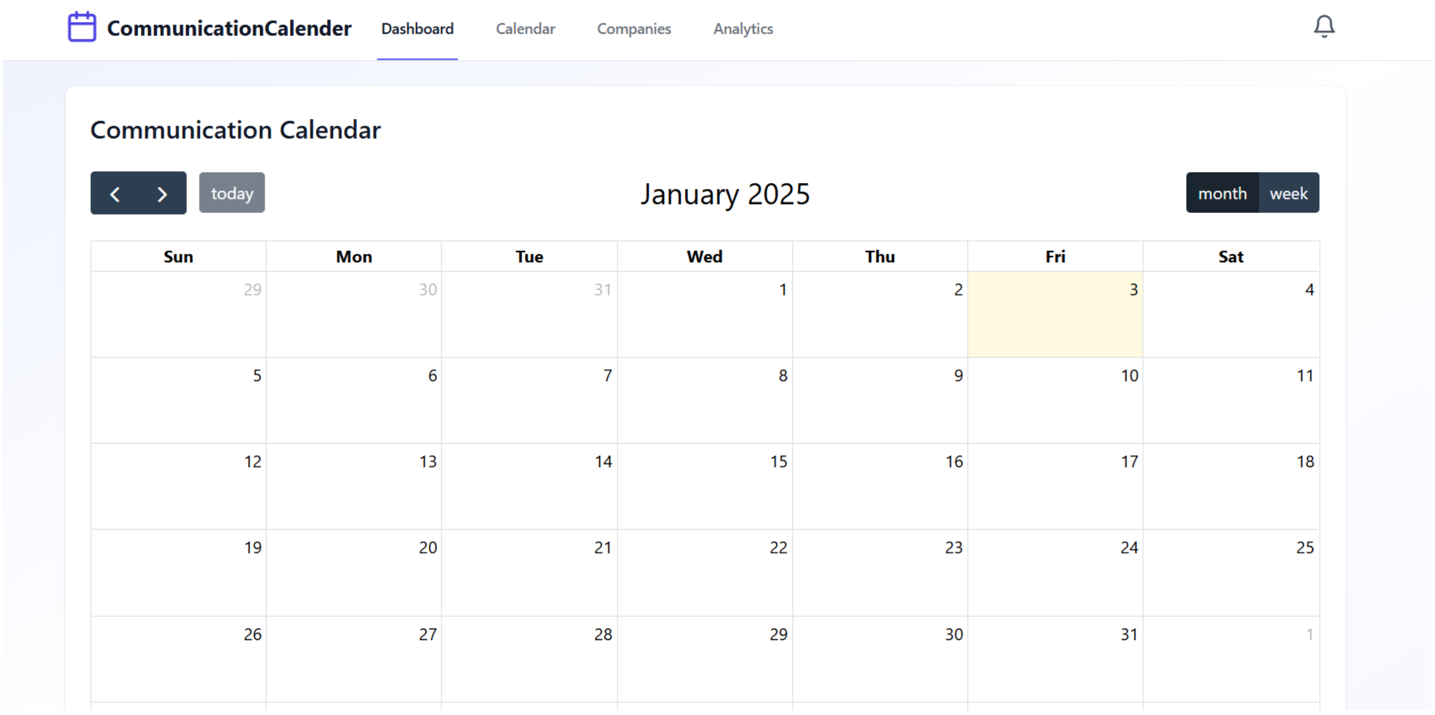
Output Screens



Companies

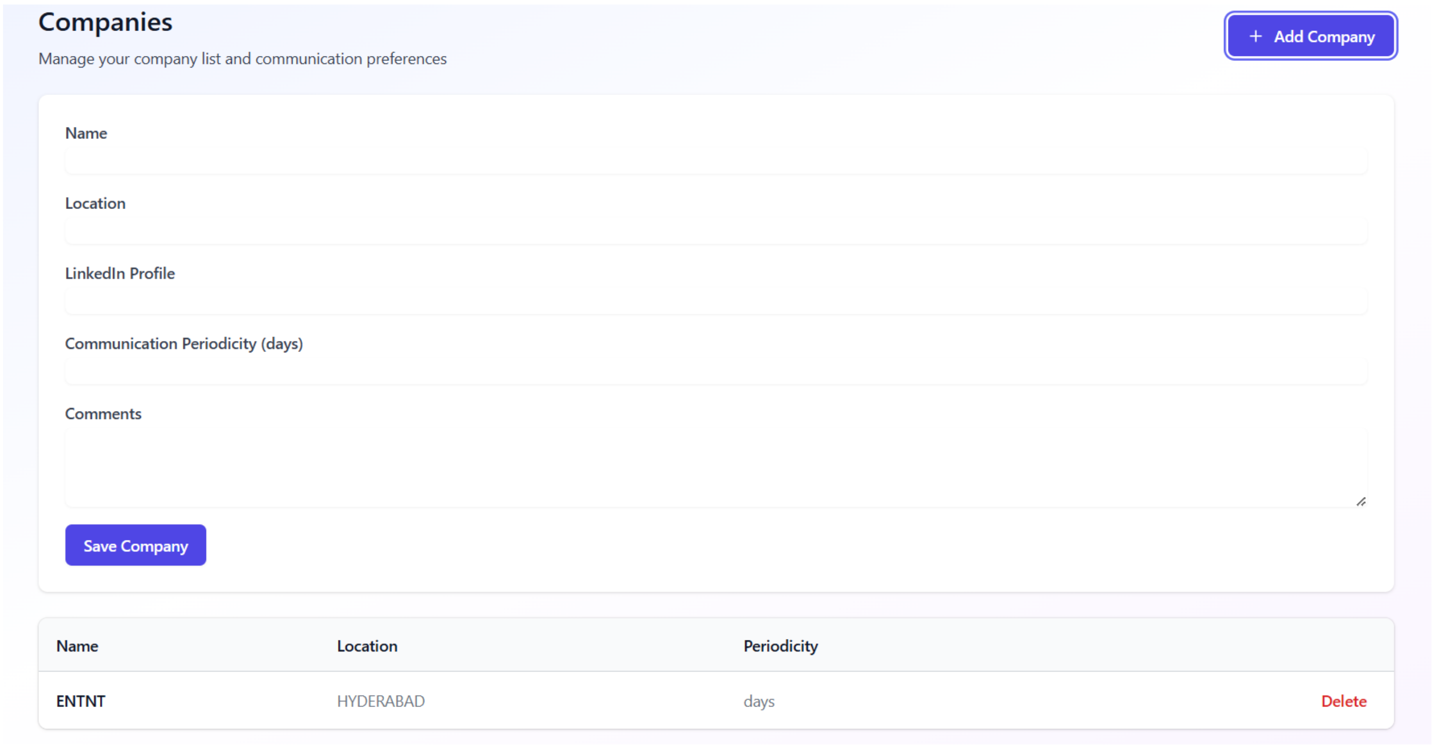
Manage your company list and communication preferences

Name	Location	Periodicity



Communication Calendar

Sun	Mon	Tue	Wed	Thu	Fri	Sat
29	30	31	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1



Companies

Manage your company list and communication preferences

Name

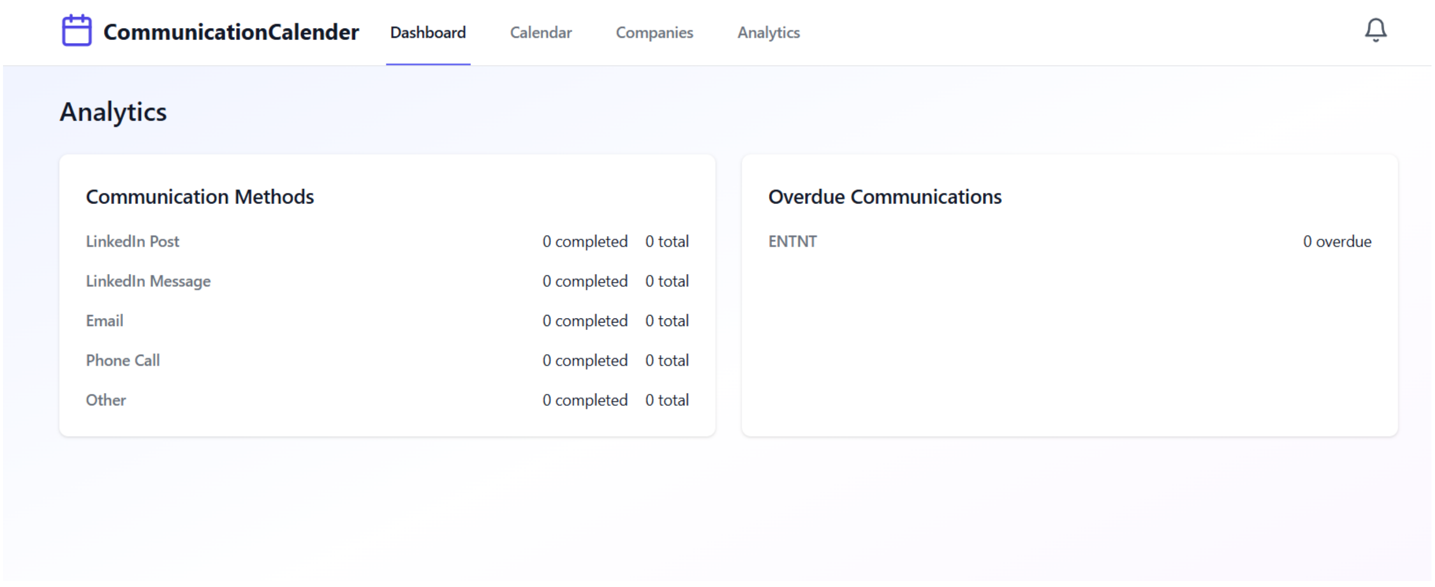
Location

LinkedIn Profile

Communication Periodicity (days)

Comments

Name	Location	Periodicity	
ENTNT	HYDERABAD	days	Delete



Analytics

Communication Methods

LinkedIn Post	0 completed	0 total
LinkedIn Message	0 completed	0 total
Email	0 completed	0 total
Phone Call	0 completed	0 total
Other	0 completed	0 total

Overdue Communications

ENTNT	0 overdue

Contact

For questions or issues, feel free to open an issue on the GitHub repository or reach out via the following:

- **GitHub Repository**: https://github.com/likhithagudla/Likhithagudla-Calendar-Application-for-Communication-Tracking
- **Email**: likhithagudla7@gmail.com