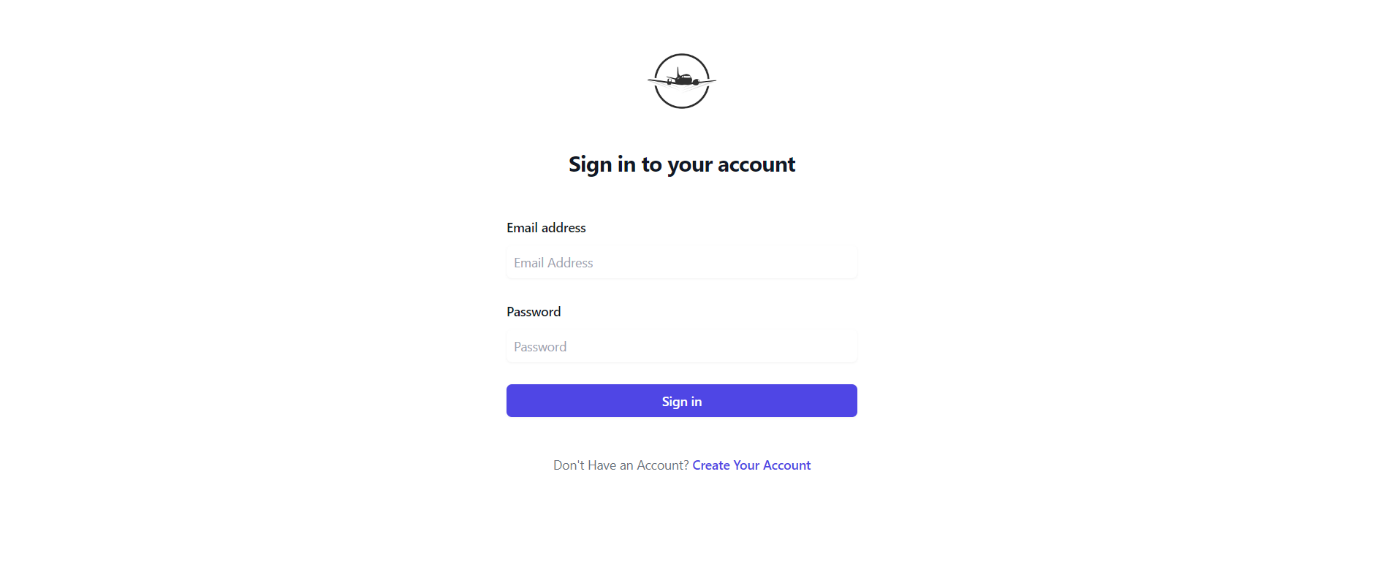
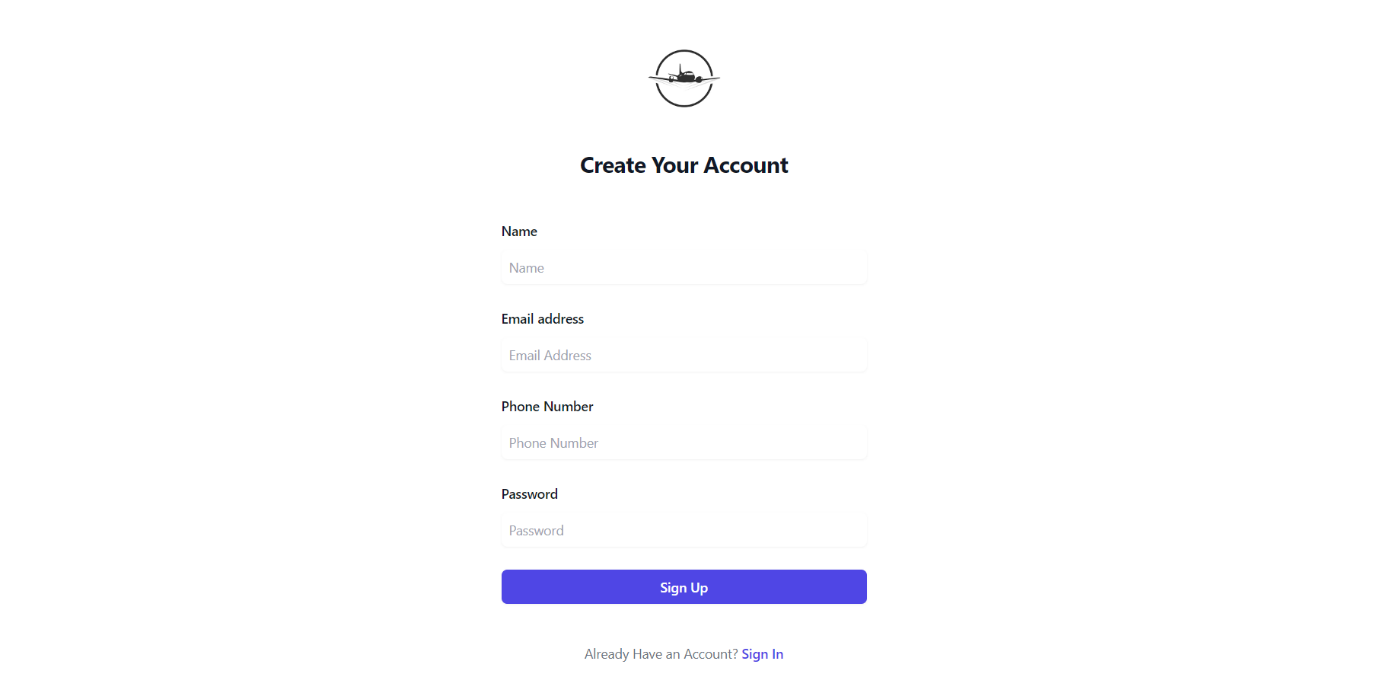
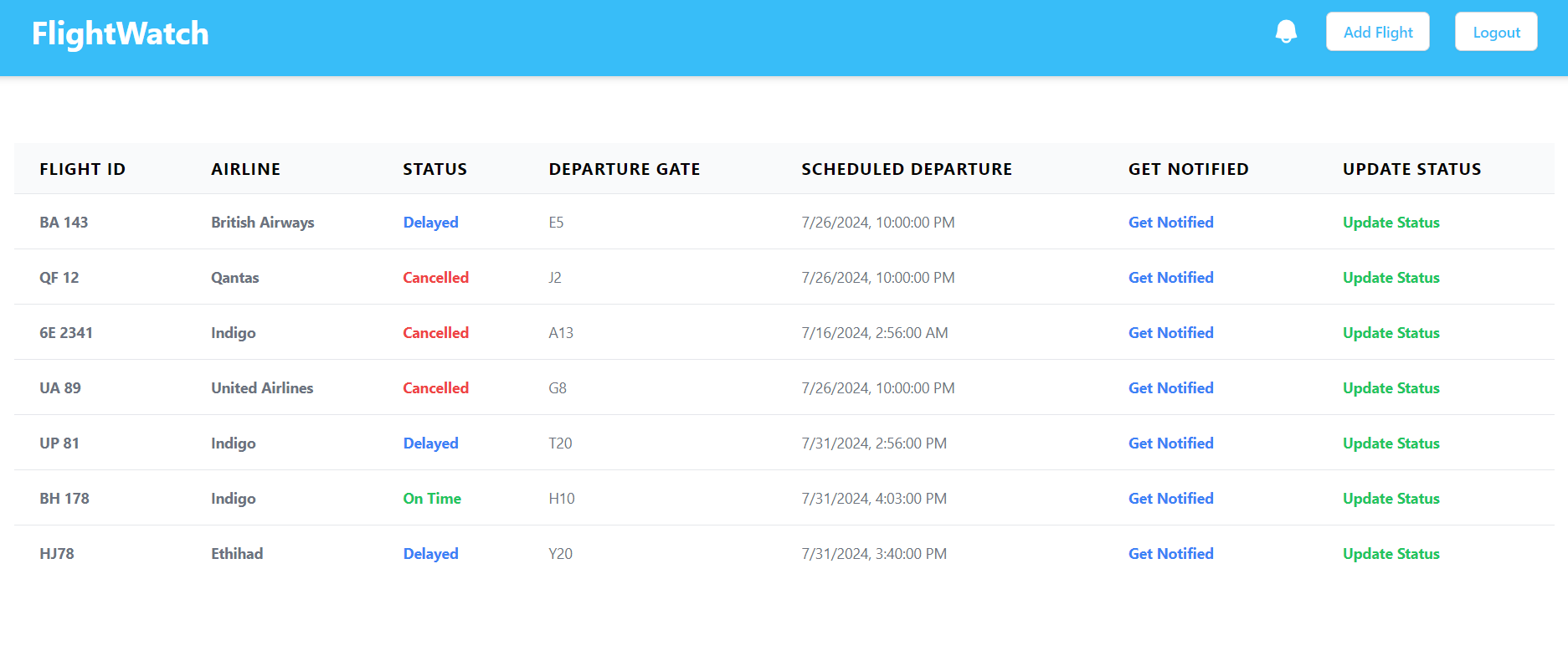
**Hack To Hire**

1. I have completed all the requirements as outlined in the case study.
2. I have implemented role-based access control, allowing administrators to create flights and update flight statuses.
3. Users can register to receive notifications via email, SMS, or in-app notifications.
4. When a flight is updated, all users will receive notifications through their selected channels.
5. Real-time data updates are implemented so that any changes to flight information are immediately reflected on the users' screens.
6. Authentication has been implemented using JWT (JSON Web Tokens).

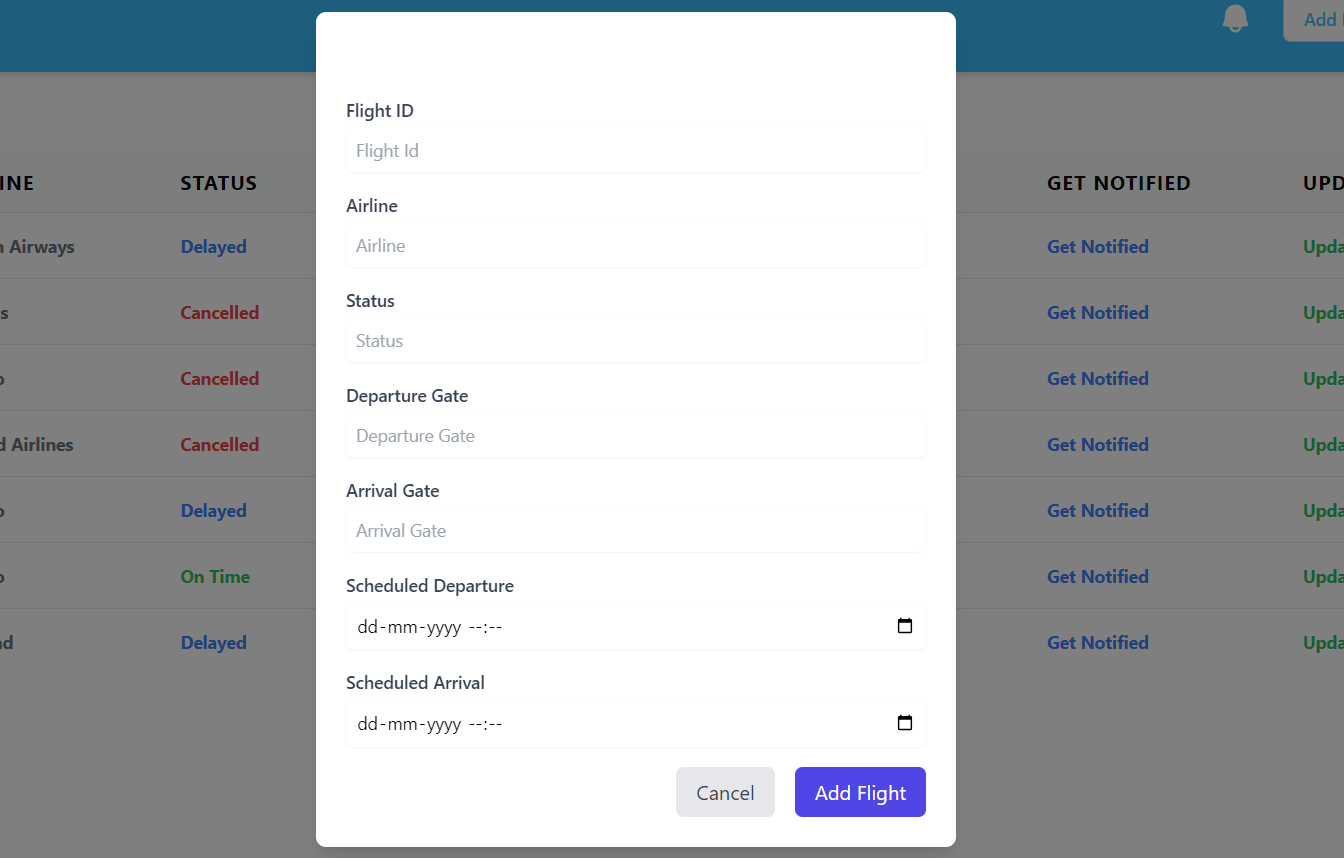




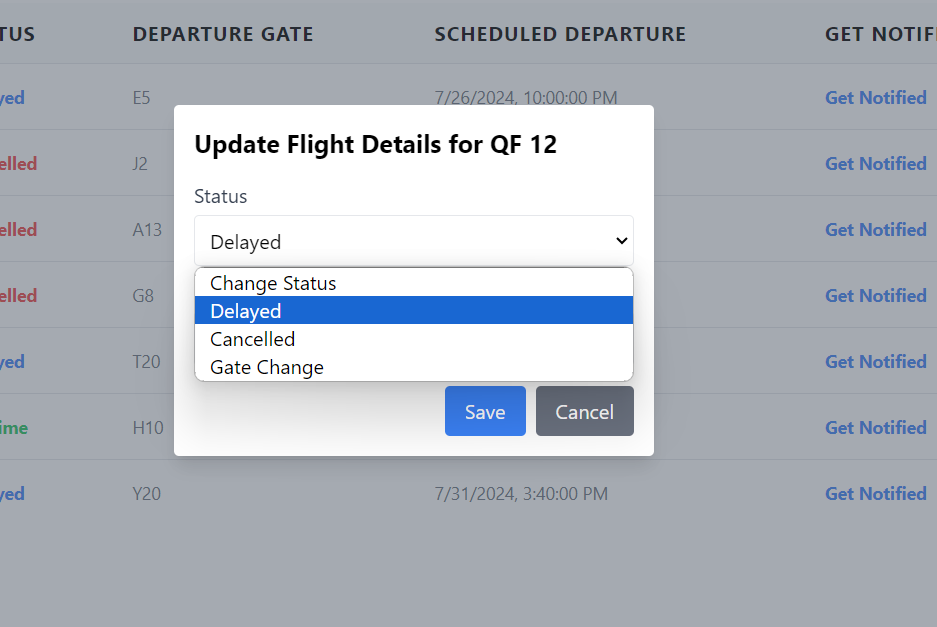
Once the user logs in, they will be presented with a list of available flights.



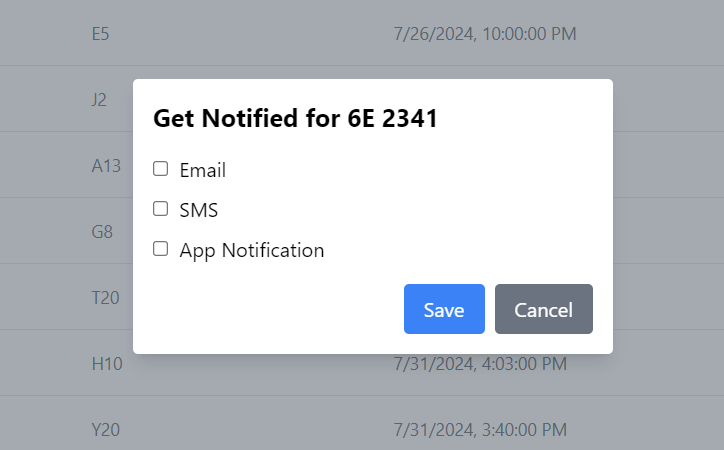
Only administrators have the permission to create flights.



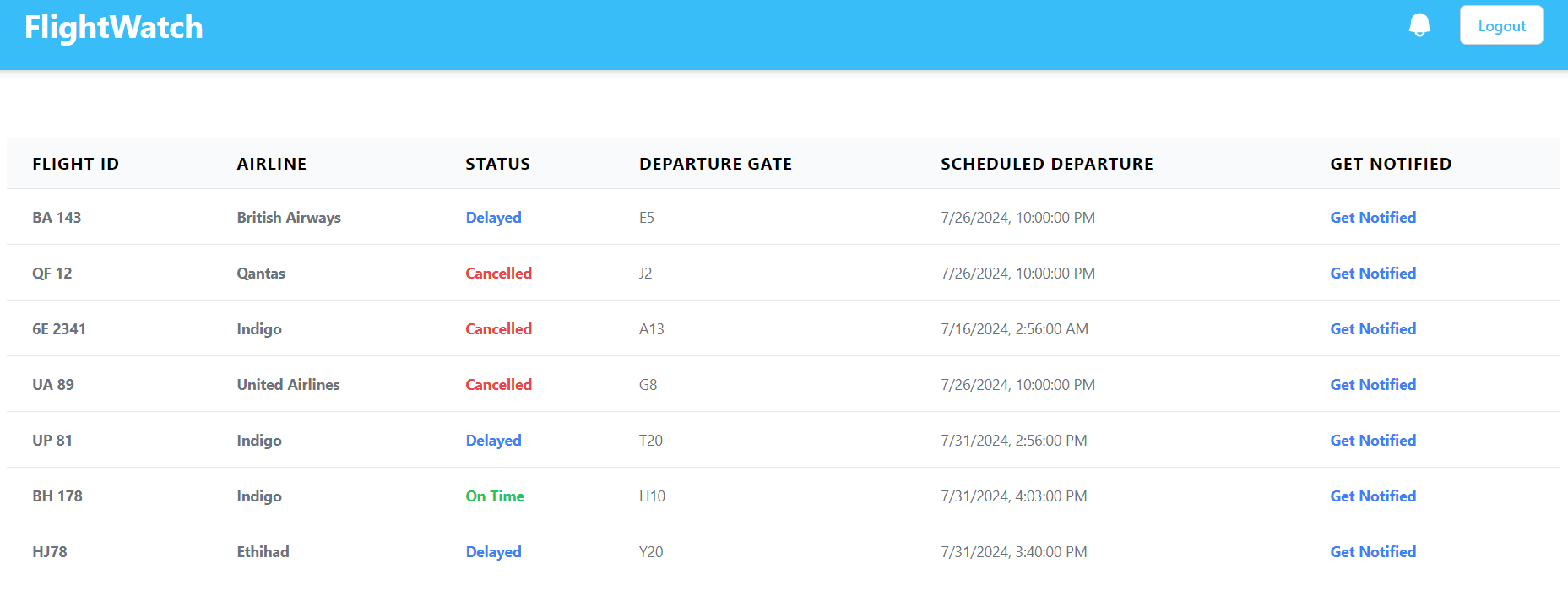
Administrators have the ability to update the status of a flight, including changes such as "Delayed," "Cancelled," or "Gate Change," among others.



Users can select multiple modes to receive notifications, allowing them to choose from options such as email, SMS, and in-app notifications.



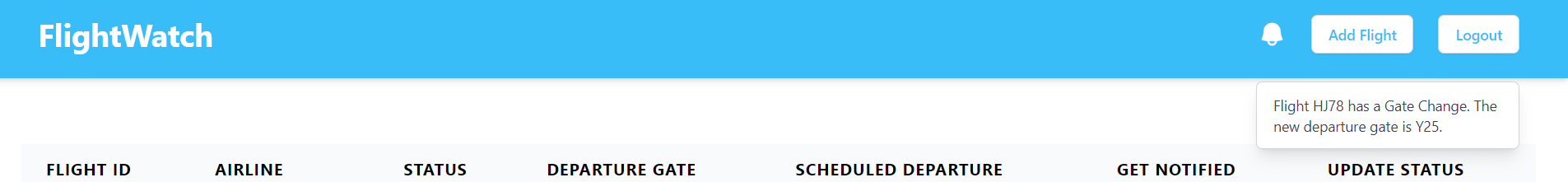
This page is what a regular user sees, who does not have permissions to add or update flights. They can only subscribe to flights and receive notifications based on their selected preferences.



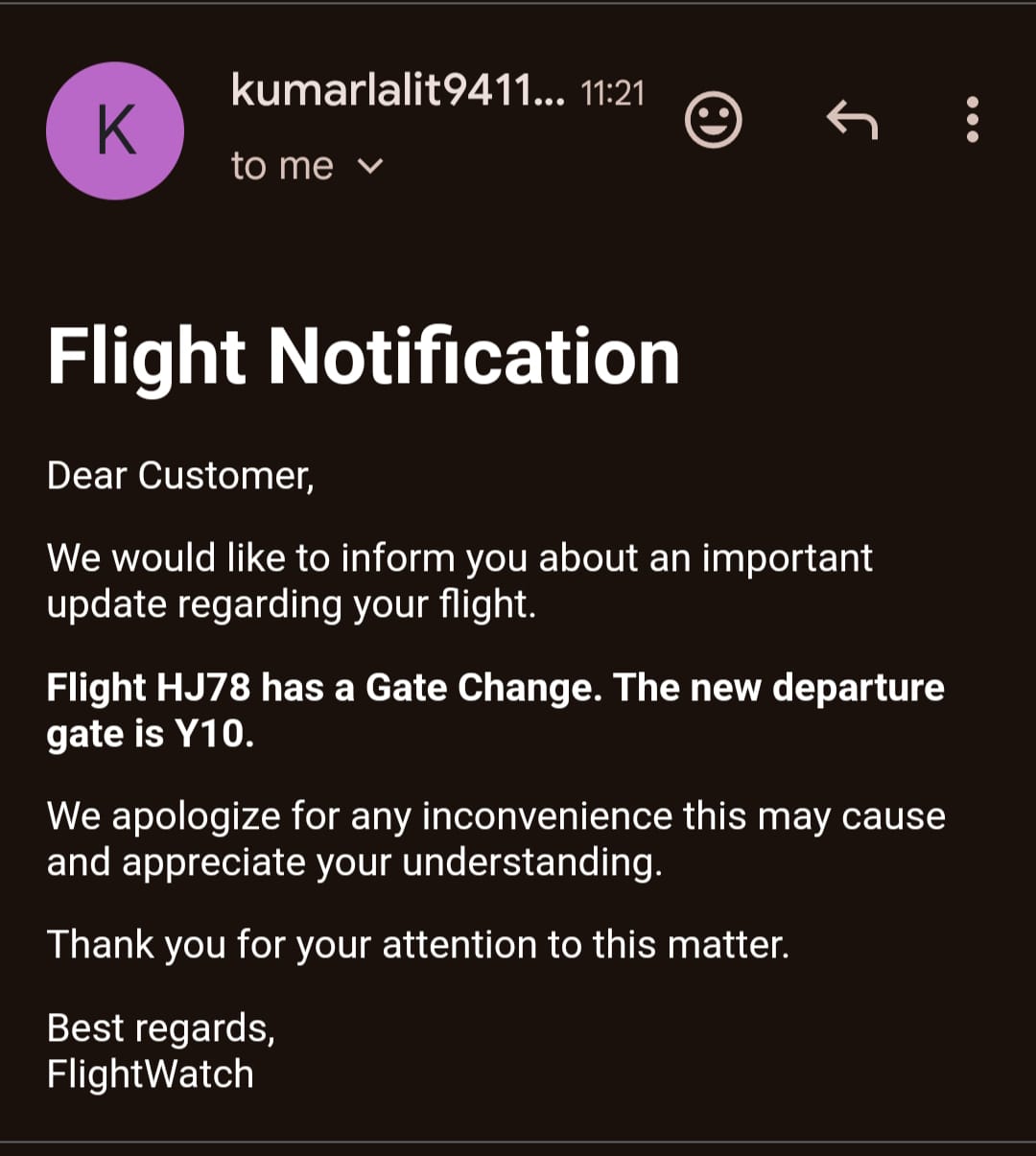
Users can view notifications of recent updates by clicking on the bell icon.

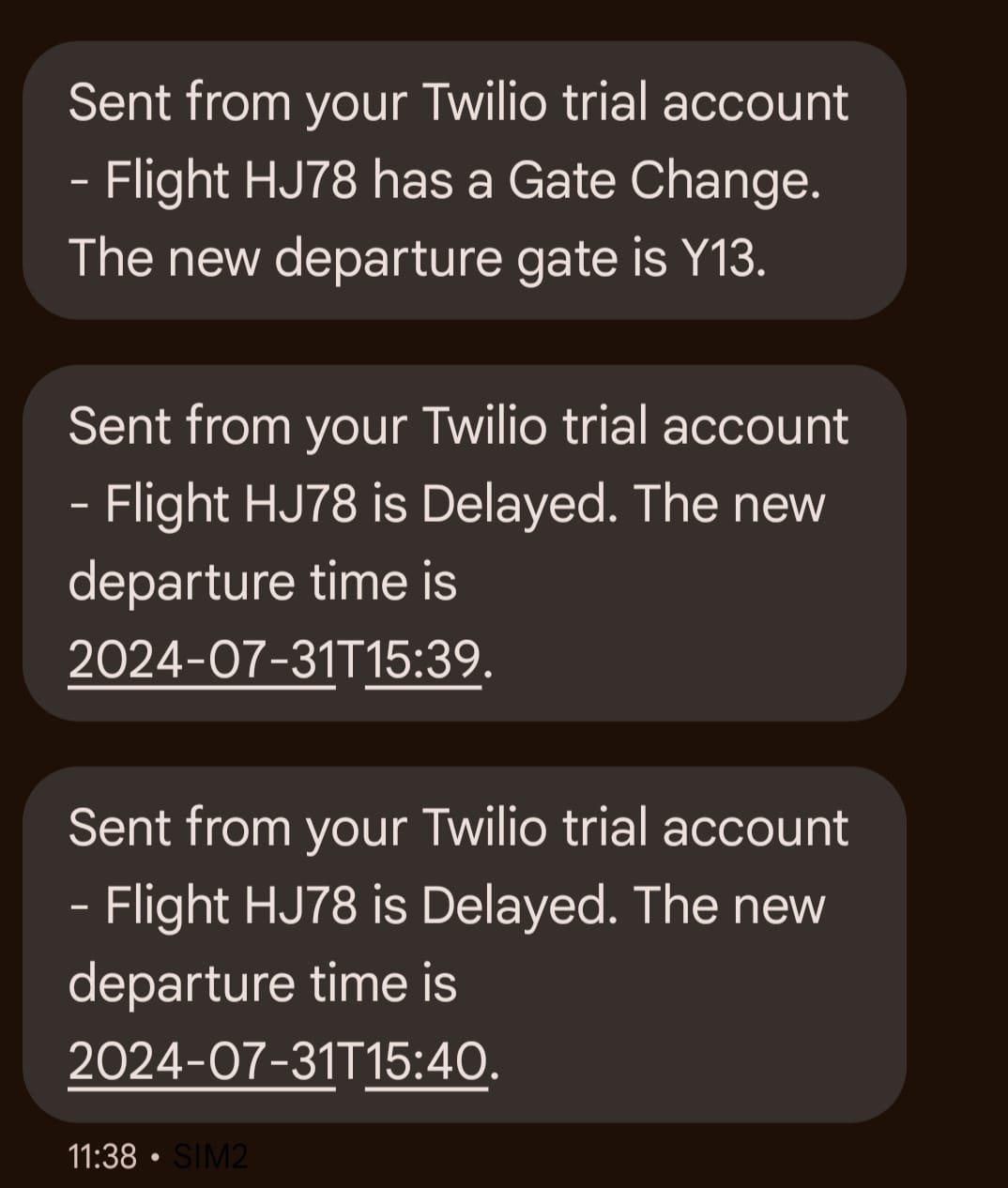


When a user clicks on the bell icon, the notifications are displayed, and the notification count is reset.



Here are sample structures for Email and SMS notifications:





Here’s a formal overview of your tech stack and its components:

* **Frontend**: React.js
* **Backend**: Node.js
* **Message Queue**: RabbitMQ (used to send SMS and email notifications asynchronously)
* **Real-time Communication**: Socket.IO (used for real-time updates on flight status and notifications)
* **Database**: PostgreSQL
* **SMS Service**: Twilio

Here is a formal description of your database tables:

**Flight Table**

* **flight\_id** (Primary Key): Unique identifier for each flight.
* **airline**: The airline operating the flight.
* **status**: Current status of the flight (e.g., OnTime, Delayed, Cancelled).
* **departure\_gate**: Gate from which the flight departs.
* **arrival\_gate**: Gate at which the flight arrives.
* **scheduled\_departure**: Scheduled departure time.
* **scheduled\_arrival**: Scheduled arrival time.
* **actual\_departure**: Actual departure time.
* **actual\_arrival**: Actual arrival time.

**Notification Table**

* **notification\_id** (Primary Key): Unique identifier for each notification.
* **flight\_id** (Foreign Key): Reference to the related flight.
* **message**: Content of the notification.
* **timestamp**: Time when the notification was sent.
* **method**: Method of notification (Enum: Email, SMS, App).
* **recipient**: Recipient of the notification (Email address, Phone number, or User ID).

**Passenger Table**

* **user\_id** (Primary Key): Unique identifier for each passenger.
* **name**: Name of the passenger.
* **role**: Role of the passenger (Admin, User).
* **email**: Email address of the passenger.
* **password\_hash**: Hashed password for authentication.
* **phone\_number**: Contact phone number of the passenger.

**Notification Preference Table**

* **id** (Primary Key): Unique identifier for each preference entry.
* **user\_id** (Foreign Key): Reference to the related user.
* **flight\_id** (Foreign Key): Reference to the related flight.
* **notificationMode**: Bitmask indicating user preferences for notification methods.
* **isDeleted**

First Bit: Represents Email notifications.

Second Bit: Represents SMS notifications.

Third Bit: Represents In-app notifications.

For example:

7 (Binary 111): Indicates that the user wants notifications via all mediums (Email, SMS, and App).

5 (Binary 101): Indicates that the user wants notifications via Email and App only.