網程期末專案

RESTAURANT WEB ORDERING

第19組
01157161 陸楷軒
01157155鍾沐良

動機

我們知道到線上餐點訂購系統在現代餐飲業的重要性。

所以我們設計了一個餐廳線上外送訂購系統

BIG-BITE

技術

FRONTEND

React

REACT ROUTER

AXIOS

BACKEND

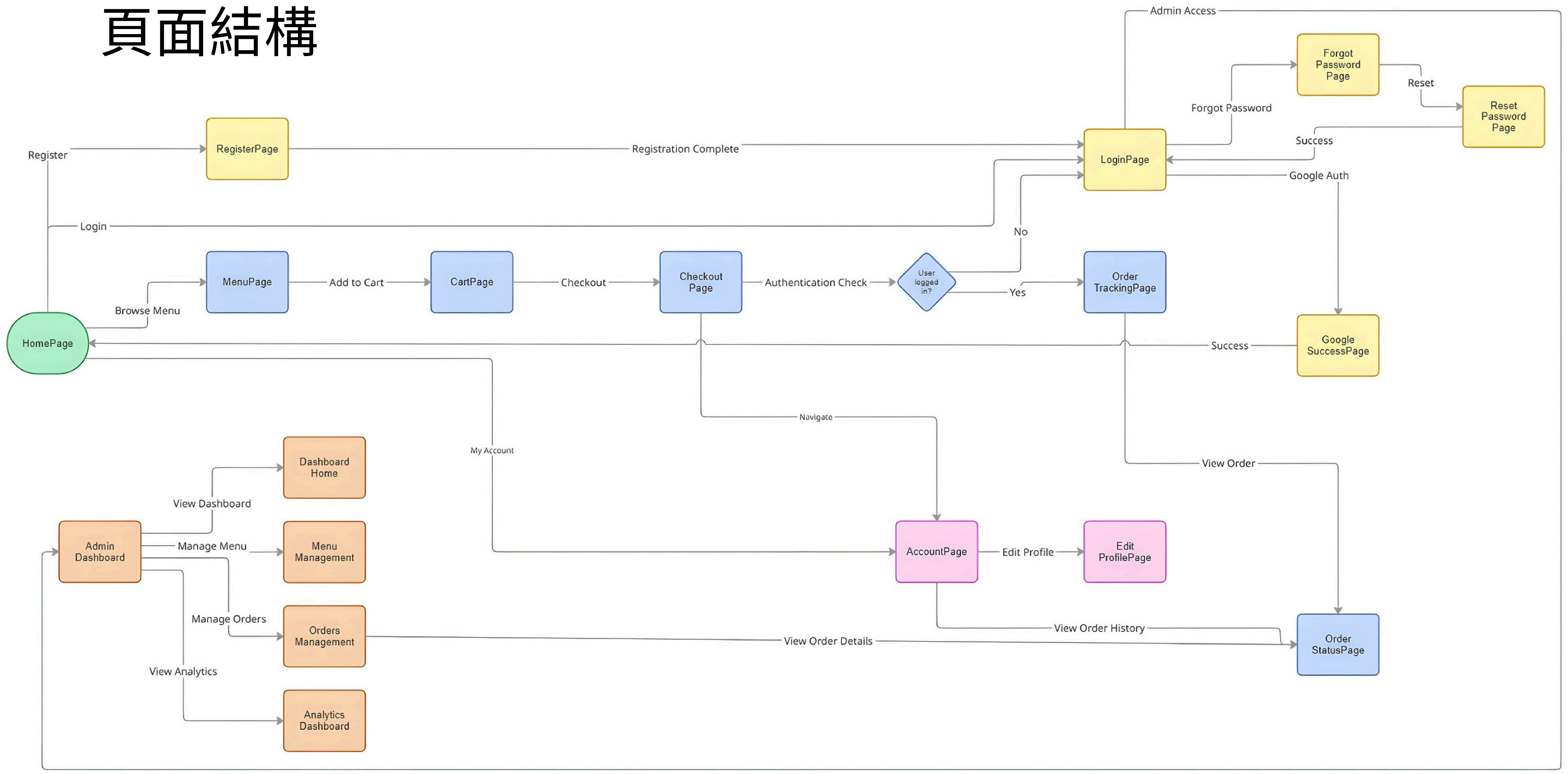
node

express

mongoose

DATABASE mongoDB.

頁面結構



Admin Access
Forgot Password Page
Reset
Reset Password Page
Forgot Password
Register
RegisterPage
Registration Complete
LoginPage
Success
Google Auth
Login
No
MenuPage
Add to Cart
CartPage
Checkout
Checkout Page
Authentication Check
User logged in?
Yes
Order TrackingPage
Browse Menu
HomePage
Success
Google SuccessPage
Navigate
My Account
Dashboard Home
View Dashboard
View Order
Admin Dashboard
Manage Menu
Menu Management
AccountPage
Edit Profile
Edit ProfilePage
Manage Orders
Orders Management
View Order History
View Order Details
Order StatusPage
View Analytics
Analytics Dashboard

FEATURES

用戶功能

驗證：郵件信箱/密碼登入 + Google OAuth、密碼重置

菜單和購物車：瀏覽菜單、按類別篩選、添加/移除商品、查看即時總價

結帳：僅限已認證用戶結帳、姓名、電話號碼、地址、訂單總覽確認

訂單追蹤：即時狀態更新（待處理 → 準備中 → 配送中 → 已完成）

使用者個人資料：查看訂單歷史記錄、編輯收貨地址

Admin功能

Dashboard: KPI cards (Total Revenue, Total Orders, Pending Orders, Cancelled Orders), 訂單趨勢圖（過去 7 天）

菜單管理：新增、編輯、刪除選單項目（含圖片和分類）

訂單管理：檢視和更新訂單狀態，管理客戶訂單

分析：收入圖表（過去 7 天）、暢銷商品、每日趨勢

Technical Stack

JWT後端：JWT 身份驗證

安全性：Password hashing (bcrypt)、protected routes、token-based身份驗證

UX：載入狀態、流暢的animation

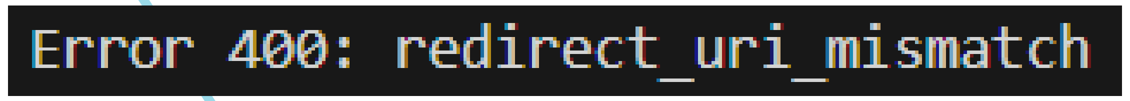
PROBLEMS

- Google OAuth redirect_uri_mismatch Error
- API REQUEST AUTHENTICATION

PROBLEM 1: GOOGLE OAUTH REDIRECT_URI_MISMATCH ERROR

Problems

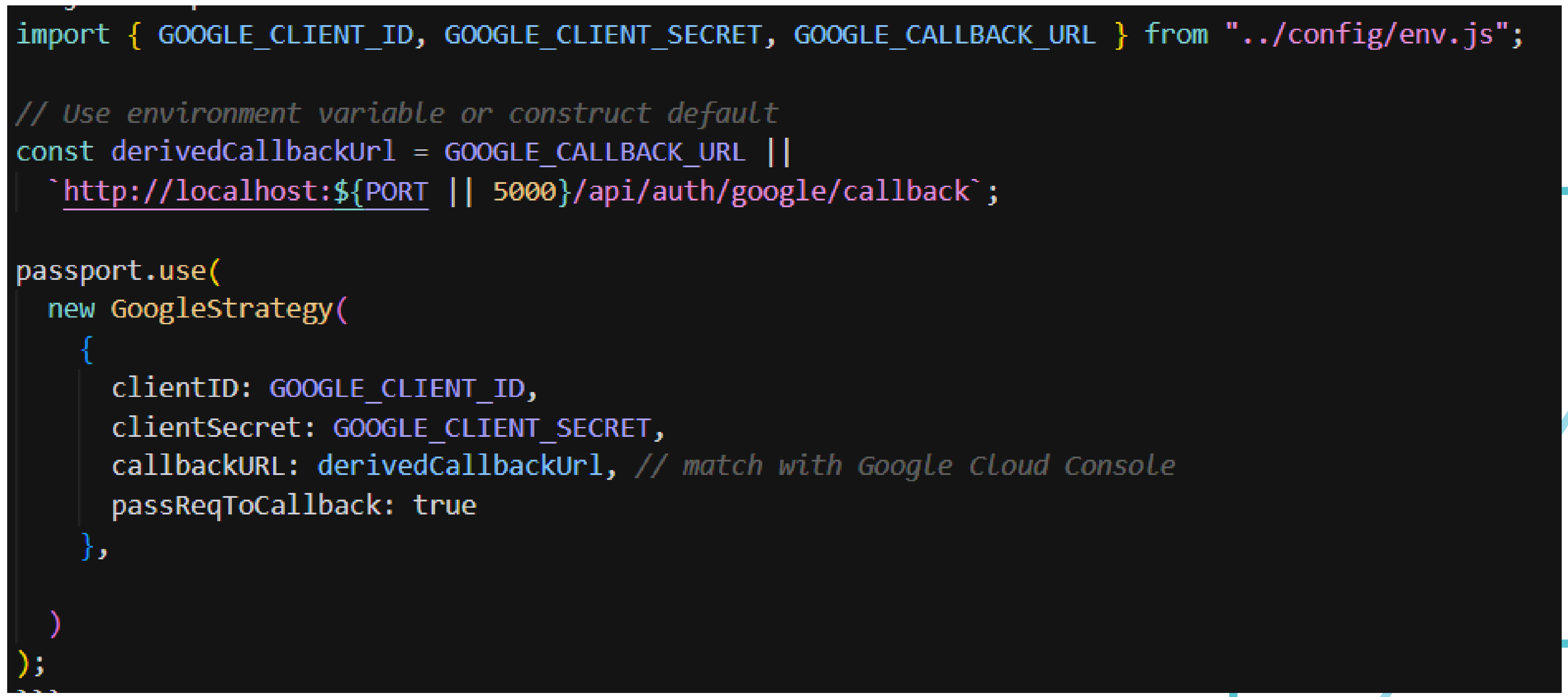
When implementing Google OAuth authentication, users encountered a `400: redirect_uri_mismatch` error. This prevented users from signing in with Google.



```
Error 400: redirect_uri_mismatch
```

solution

- step 1: Verify Backend Callback URL Configuration in passport.js:



```
import { GOOGLE_CLIENT_ID, GOOGLE_CLIENT_SECRET, GOOGLE_CALLBACK_URL } from "../config/env.js";

// Use environment variable or construct default
const derivedCallbackUrl = GOOGLE_CALLBACK_URL ||
  `http://localhost:${PORT || 5000}/api/auth/google/callback`;

passport.use(
  new GoogleStrategy(
    {
      clientID: GOOGLE_CLIENT_ID,
      clientSecret: GOOGLE_CLIENT_SECRET,
      callbackURL: derivedCallbackUrl, // match with Google Cloud Console
      passReqToCallback: true
    },

  )
);
```

PROBLEM 1: GOOGLE OAUTH REDIRECT_URI_MISMATCH ERROR

- step 2: Configure Google Cloud Console

go to Google Cloud Console

Under **Authorized redirect URIs**

http://localhost:5000/api/auth/google/callback (for development)

https://big-bite.onrender.com/api/auth/google/callback (for production)

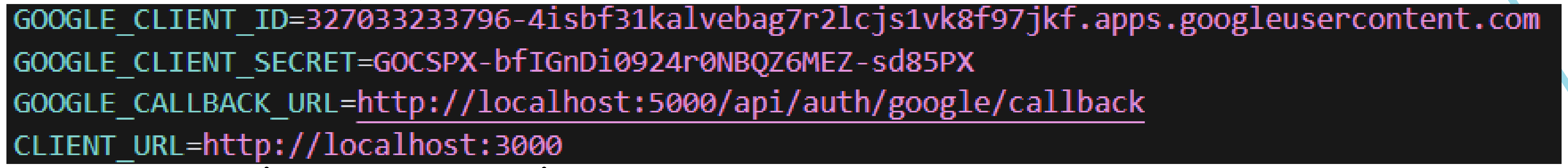
Under **Authorized JavaScript origins**

http://localhost:5000 (for development)

https://big-bite.onrender.com (for production)

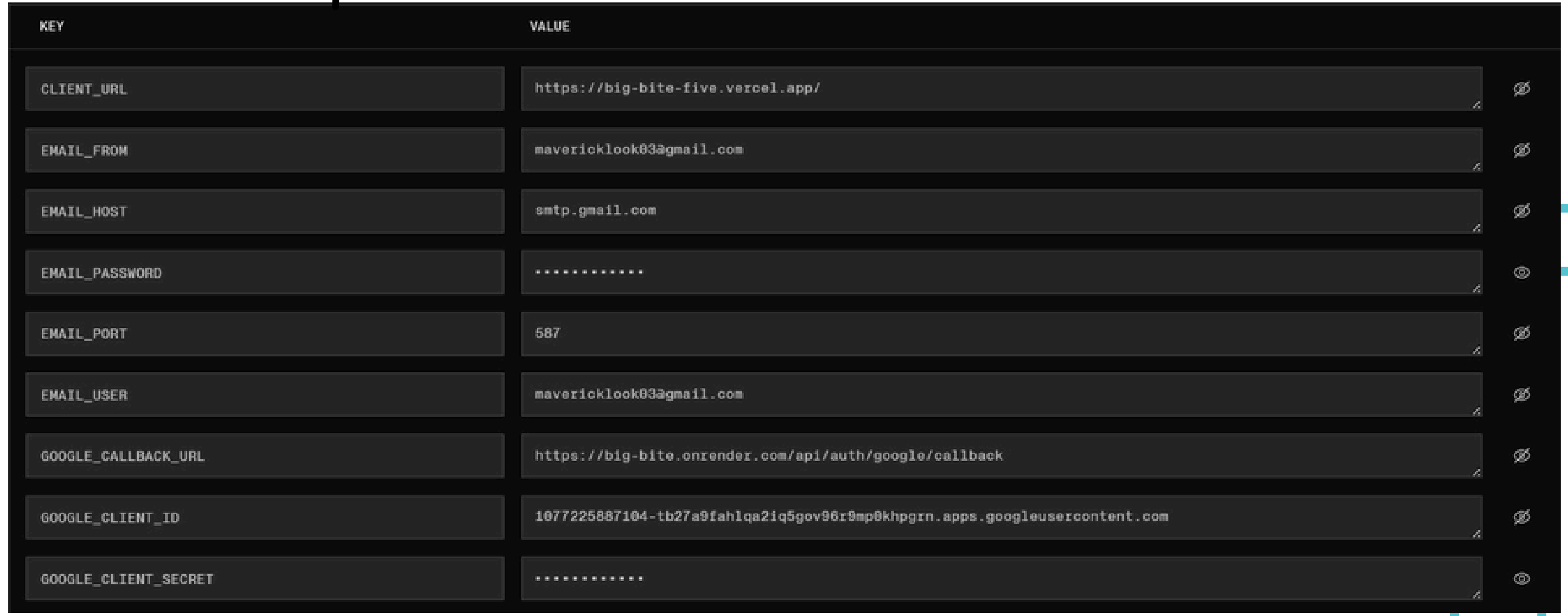
step 3: Environment Variables

in .env (development)



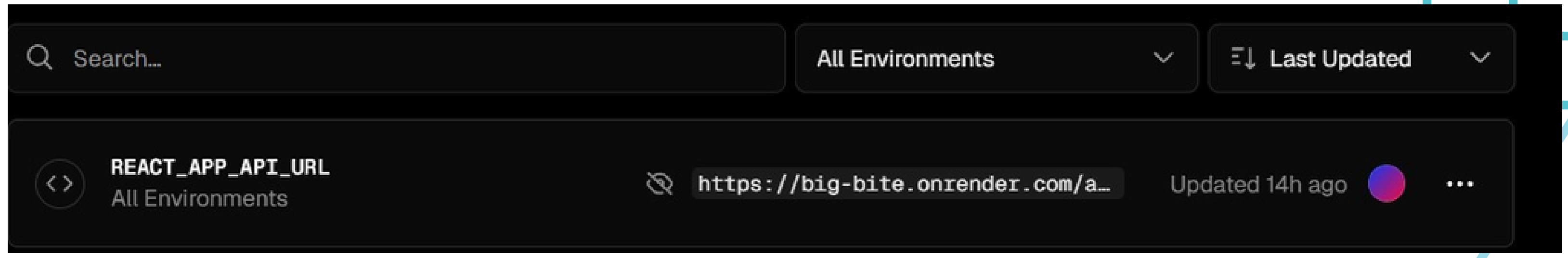
```
GOOGLE_CLIENT_ID=327033233796-4isbf31kalvebag7r2lcjs1vk8f97jkf.apps.googleusercontent.com
GOOGLE_CLIENT_SECRET=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
GOOGLE_CALLBACK_URL=http://localhost:5000/api/auth/google/callback
CLIENT_URL=http://localhost:3000
```

in Render (production)



KEY	VALUE
CLIENT_URL	https://big-bite-five.vercel.app/
EMAIL_FROM	mavericklook03@gmail.com
EMAIL_HOST	smtp.gmail.com
EMAIL_PASSWORD	••••••••••••
EMAIL_PORT	587
EMAIL_USER	mavericklook03@gmail.com
GOOGLE_CALLBACK_URL	https://big-bite.onrender.com/api/auth/google/callback
GOOGLE_CLIENT_ID	1077225887104-tb27a9fahlqa2iq5gov96r9mp0khpgrn.apps.googleusercontent.com
GOOGLE_CLIENT_SECRET	••••••••••••

in Vercel(production)



REACT_APP_API_URL
All Environments
https://big-bite.onrender.com/a...
Updated 14h ago

PROBLEM 2: API REQUEST AUTHENTICATION

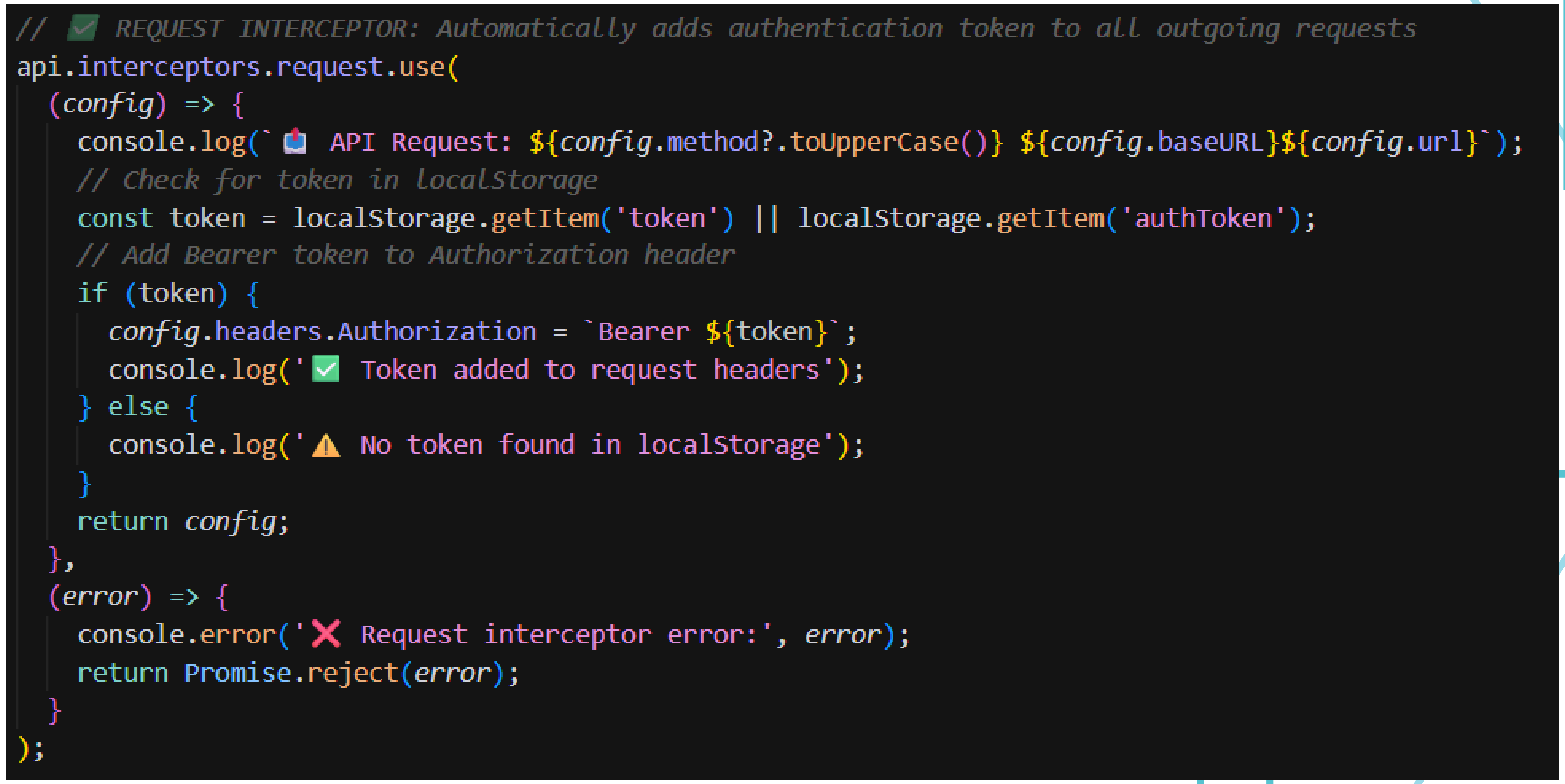
Problems

each API call required the same 3 repetitive steps: get token from storage, add to headers, and handle expired token errors - repeated in every single function.

solution

Axios Interceptors - middleware that run automatically on every request and every response.

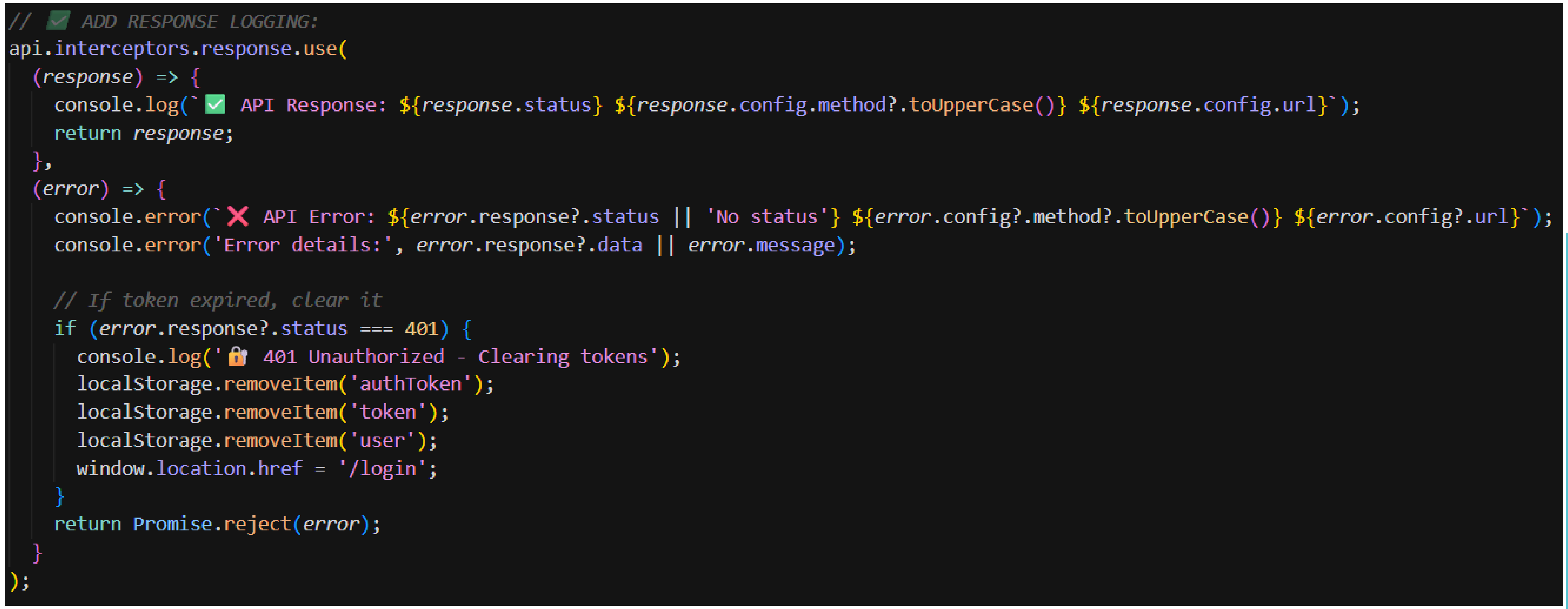
in API.js



```
// ✅ REQUEST INTERCEPTOR: Automatically adds authentication token to all outgoing requests
api.interceptors.request.use(
  (config) => {
    console.log(`📤 API Request: ${config.method?.toUpperCase()} ${config.baseURL}${config.url}`);
    // Check for token in localStorage
    const token = localStorage.getItem('token') || localStorage.getItem('authToken');
    // Add Bearer token to Authorization header
    if (token) {
      config.headers.Authorization = `Bearer ${token}`;
      console.log('✅ Token added to request headers');
    } else {
      console.log('⚠️ No token found in localStorage');
    }
    return config;
  },
  (error) => {
    console.error('❌ Request interceptor error:', error);
    return Promise.reject(error);
  }
);
```

PROBLEM 2: API REQUEST AUTHENTICATION

in API.js



```
// ✅ ADD RESPONSE LOGGING:
api.interceptors.response.use(
  (response) => {
    console.log(`✅ API Response: ${response.status} ${response.config.method?.toUpperCase()} ${response.config.url}`);
    return response;
  },
  (error) => {
    console.error(`❌ API Error: ${error.response?.status || 'No status'} ${error.config?.method?.toUpperCase()} ${error.config?.url}`);
    console.error('Error details:', error.response?.data || error.message);

    // If token expired, clear it
    if (error.response?.status === 401) {
      console.log('🔐 401 Unauthorized - Clearing tokens');
      localStorage.removeItem('authToken');
      localStorage.removeItem('token');
      localStorage.removeItem('user');
      window.location.href = '/login';
    }
    return Promise.reject(error);
  }
);
```

THANK YOU