EchoDrop

A Prototype Multi-Platform Scheduled Messaging System

Presented By: **Parth Dilipbhai Patel**

Semester: **5th**

Guided By: **Mrs. Jyotsna Bhubanesh**

Organization: **R.K. DESAI ACHCHHARIWALA COLLEGE OF COMPUTER & APPLIED SCIENCES**

Table of contents

01 Acknowledgement

02 Introduction & Objective

03 Tech Stack

04 System Architecture

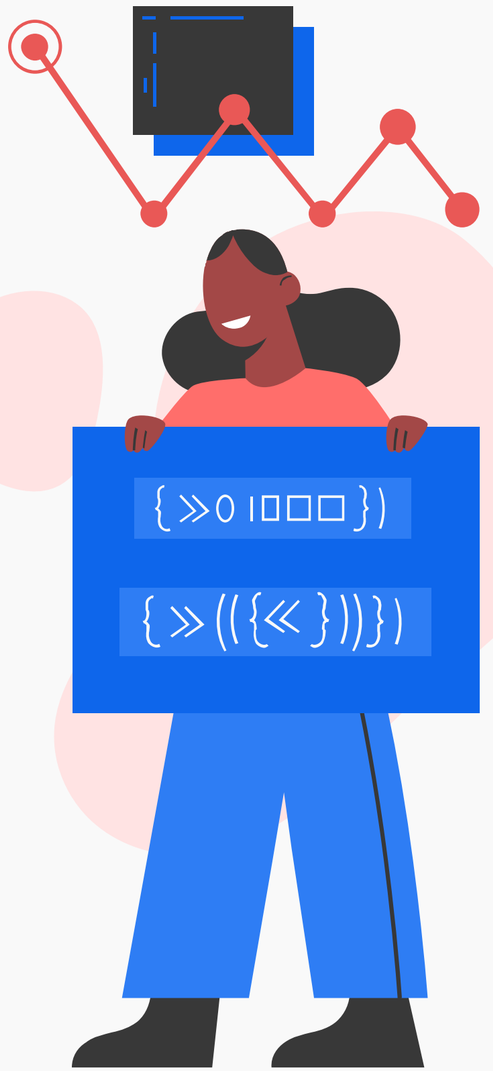


Table of contents

05

Workflow

(DFD Level 0 & 1)

06

Key Features

07

Backend Deep Dive

08

Frontend Deep Dive

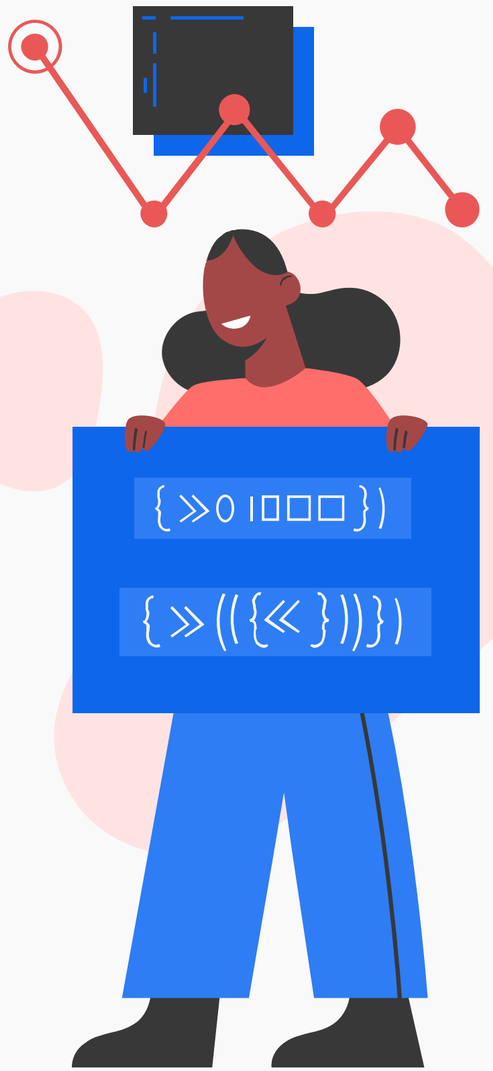


Table of contents

09 Database Schema

10 Screenshots - User Authentication

11 Screenshots - Scheduling & Management

12 Challenges & Solutions

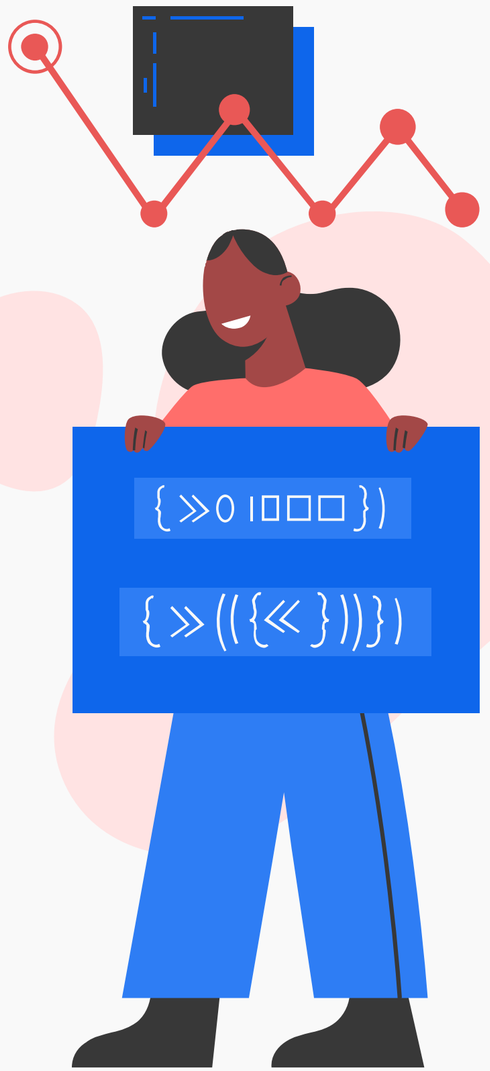


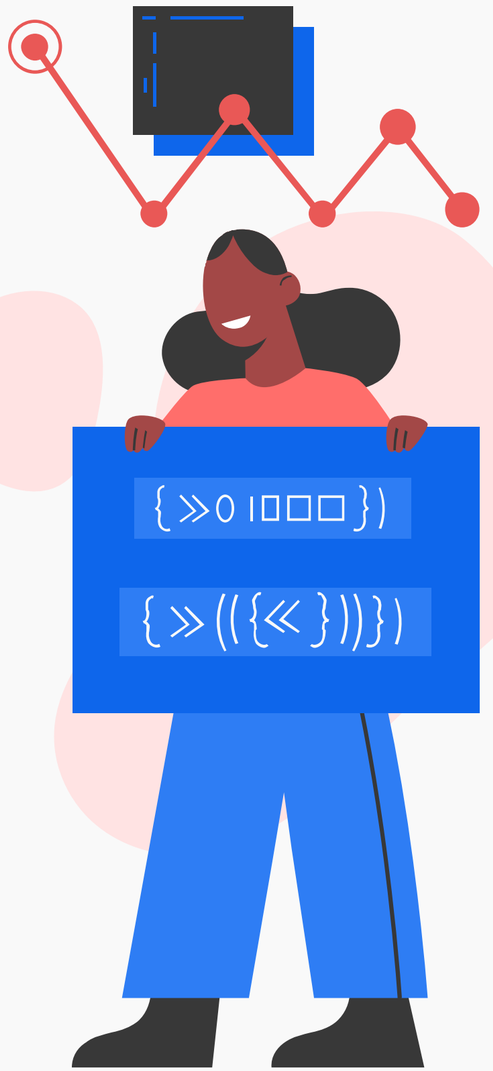
Table of contents

13

Conclusion & Future Scope

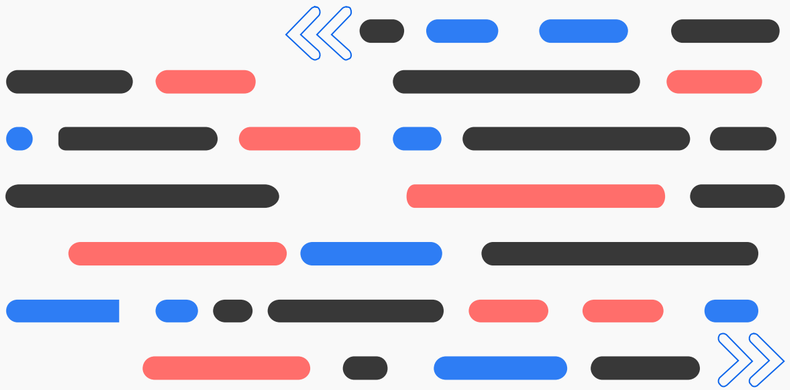
14

References



01

Acknowledgement

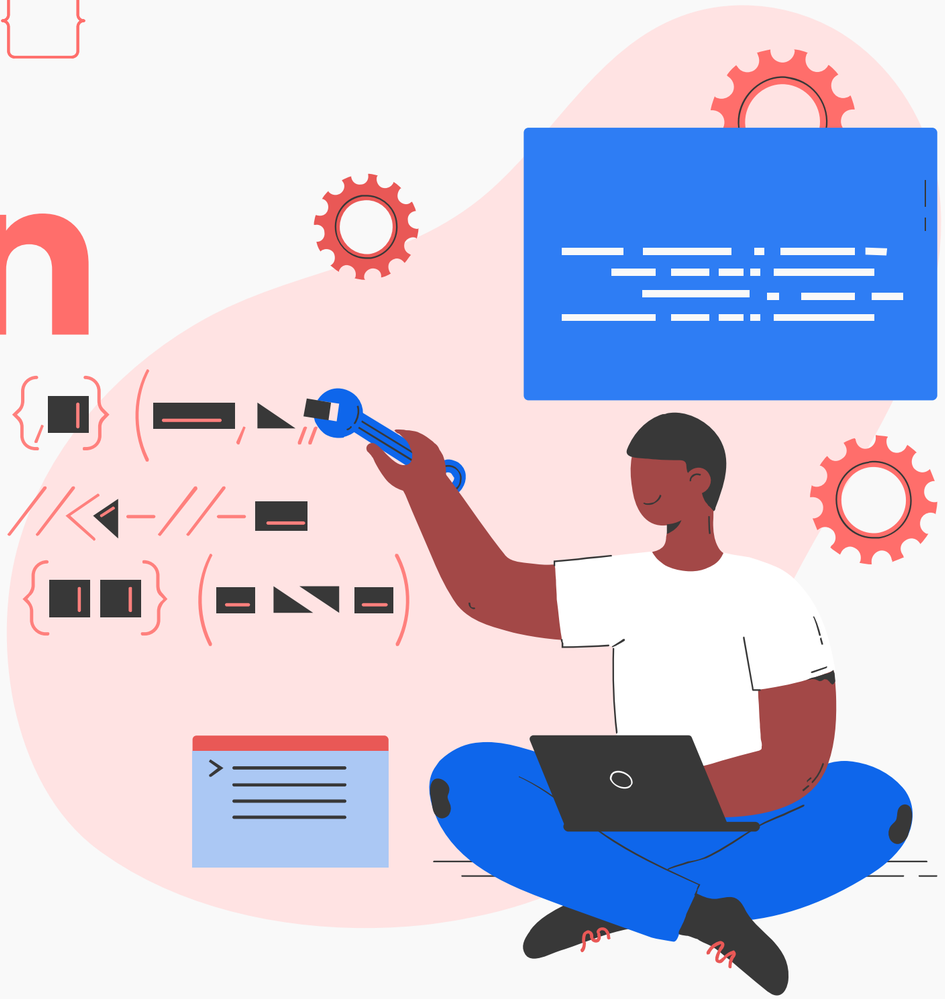


I express my sincere gratitude to

- My project guide, Mrs. Jyotsna Bhubanesh, for her invaluable guidance.
- The faculty for their unwavering support.
- My friends for their encouragement.
- My family for their constant motivation.

02

Introduction & Objective



- **What is EchoDrop?**
  A full-stack web application for scheduling messages across Email, SMS, and WhatsApp.

- **Core Objective:**
  To build a robust, secure, and user-friendly platform that allows users to plan their communication efficiently.

- **Problem Solved:**
  Eliminates the need to remember to send time-sensitive messages manually.

- **Current Status:**
  Functional prototype demonstrating core functionality in a controlled test environment.

- **Project Scope & Current Limitations**

1. **Authentication:** Google OAuth configured for test users only
2. **Messaging:** Twilio trial account limits messages to verified numbers only
3. **Deployment:** Local development environment (not deployed to production)
4. **Purpose:** Functional prototype demonstrating core architecture and capabilities

These limitations are expected for a academic prototype and can be addressed with production resources and budget.

03

Tech Stack

Frontend

- Angular 17 (Standalone Components)
- TypeScript, HTML, SCSS
- RxJS, Bootstrap

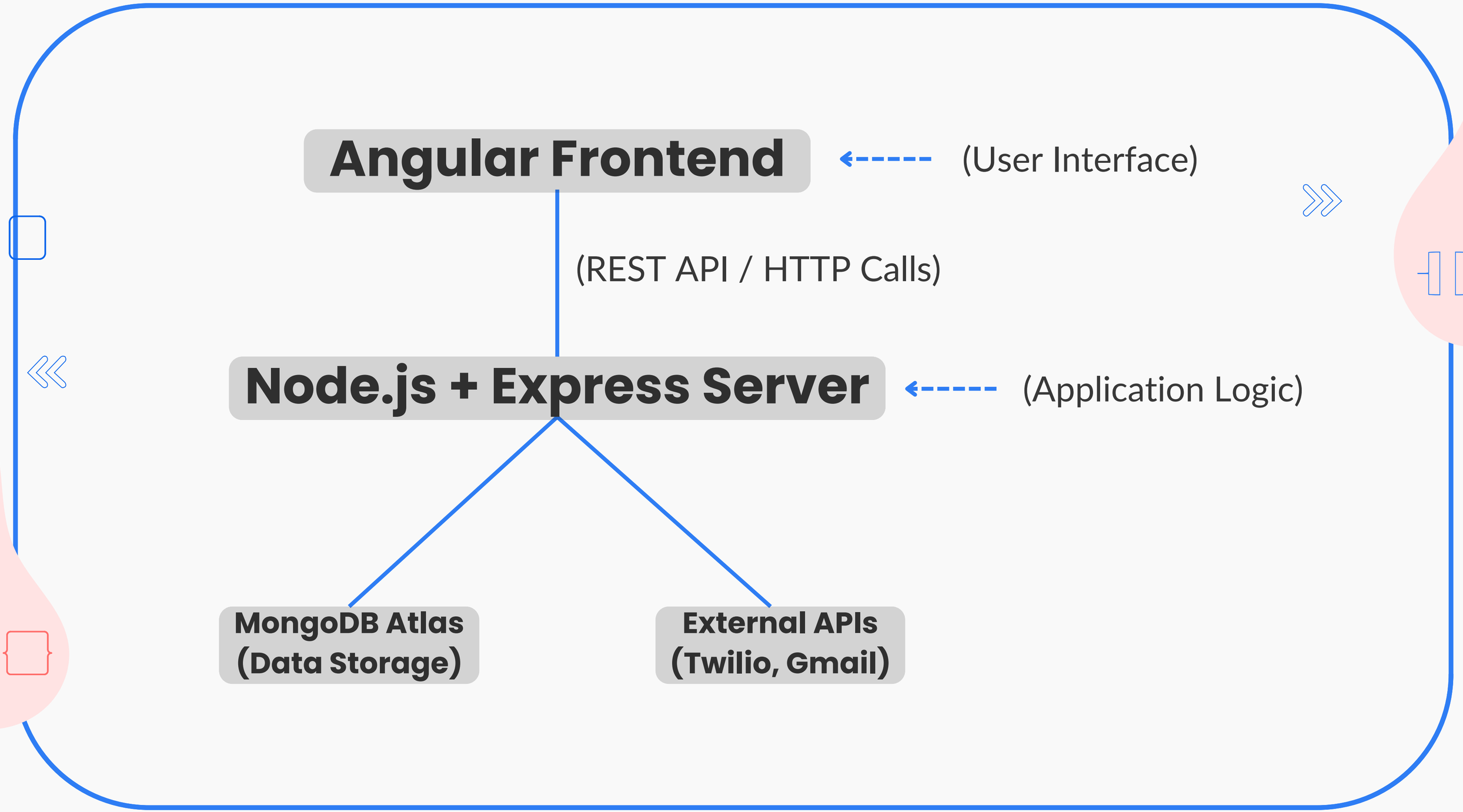
Backend

- Node.js, Express.js
- Mongo DB with Mongoose ODM

Services & APIs

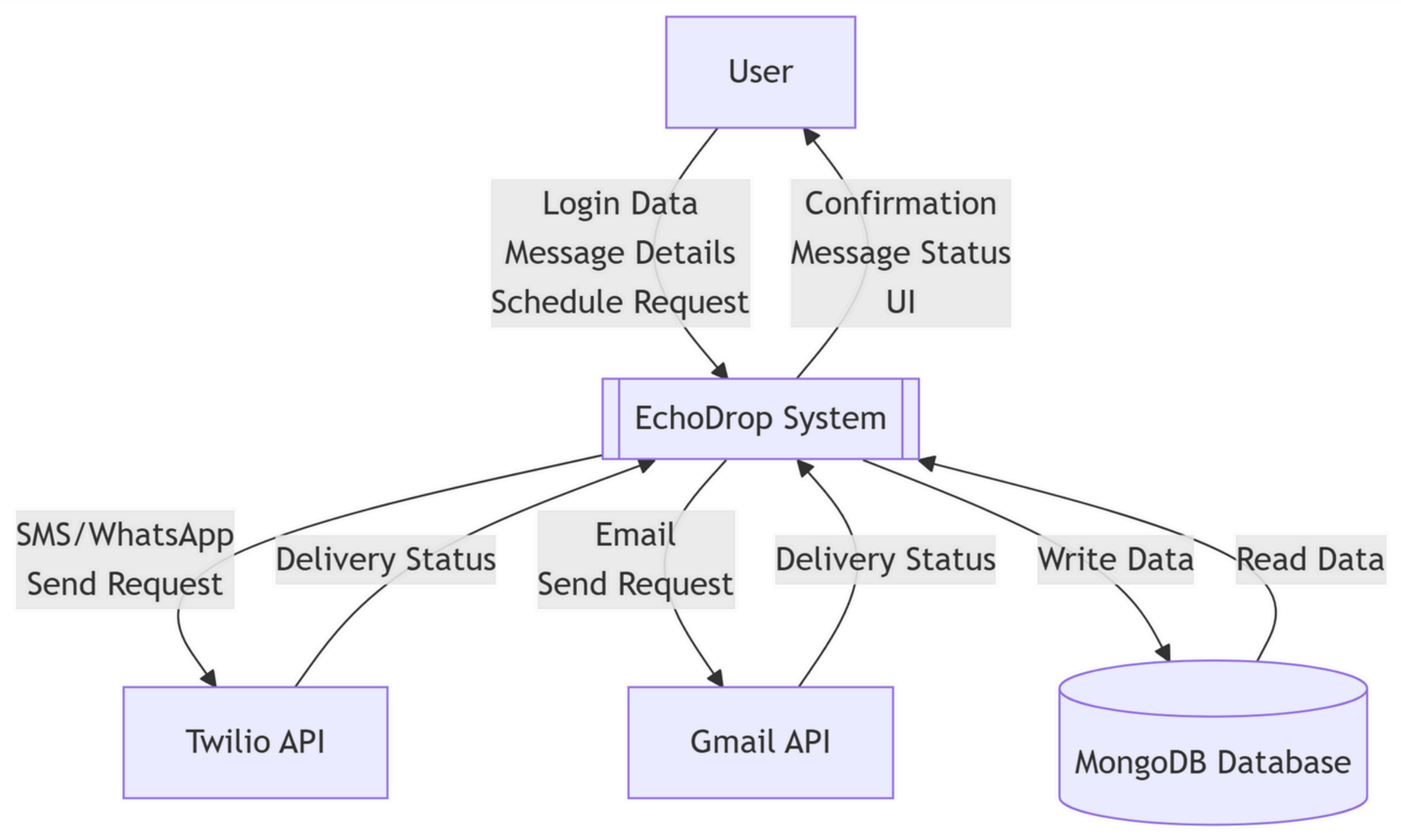
- Twilio API (SMS & WhatsApp)
- Gmail API (OAuth 2.0 for Emails)
- JWT for Authentication
- Passport.js for Google OAuth

04 System Architecture

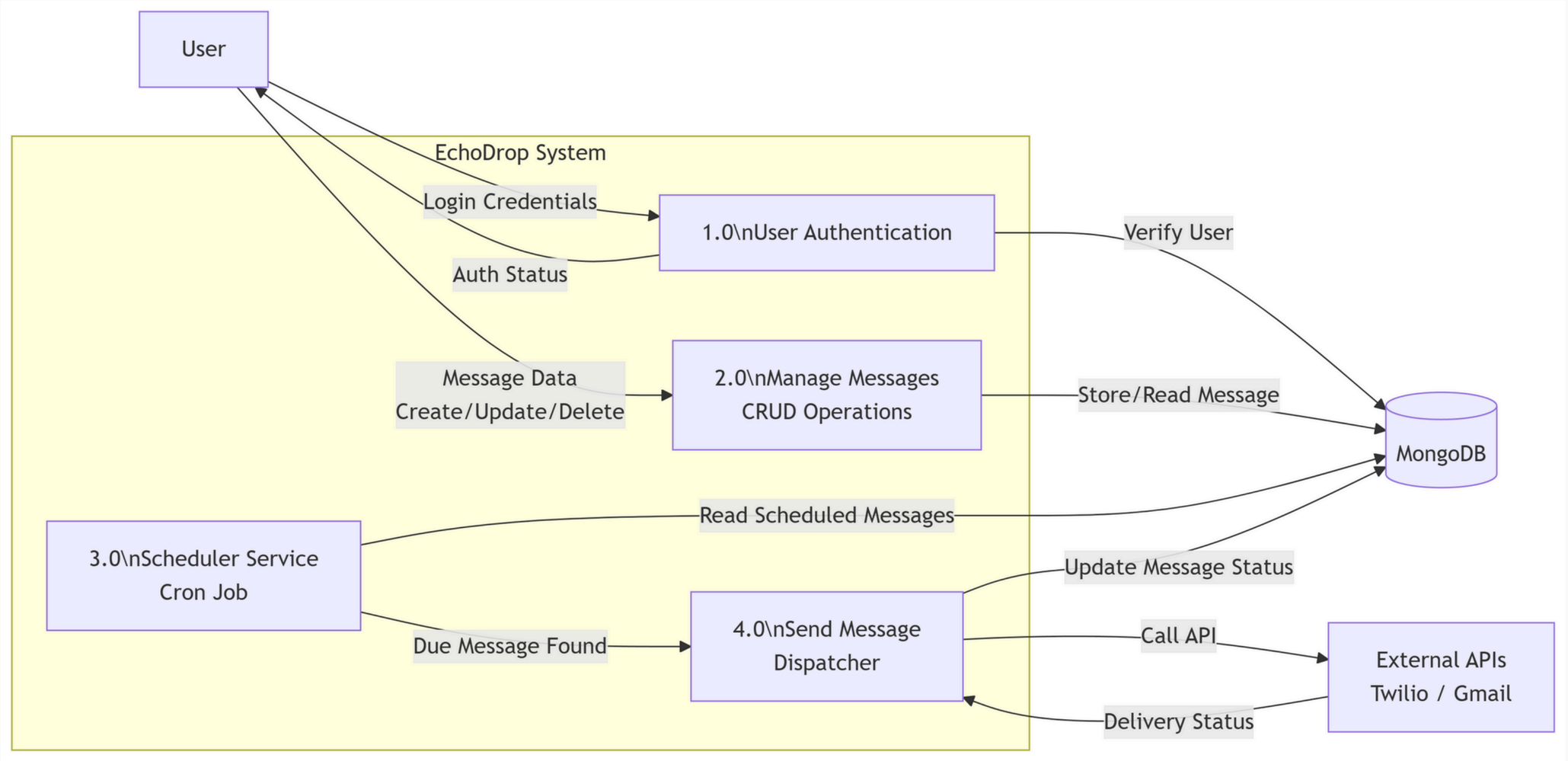

Angular Frontend
(User Interface)
(REST API / HTTP Calls)
Node.js + Express Server
(Application Logic)
MongoDB Atlas
(Data Storage)
External APIs
(Twilio, Gmail)

05 Workflow

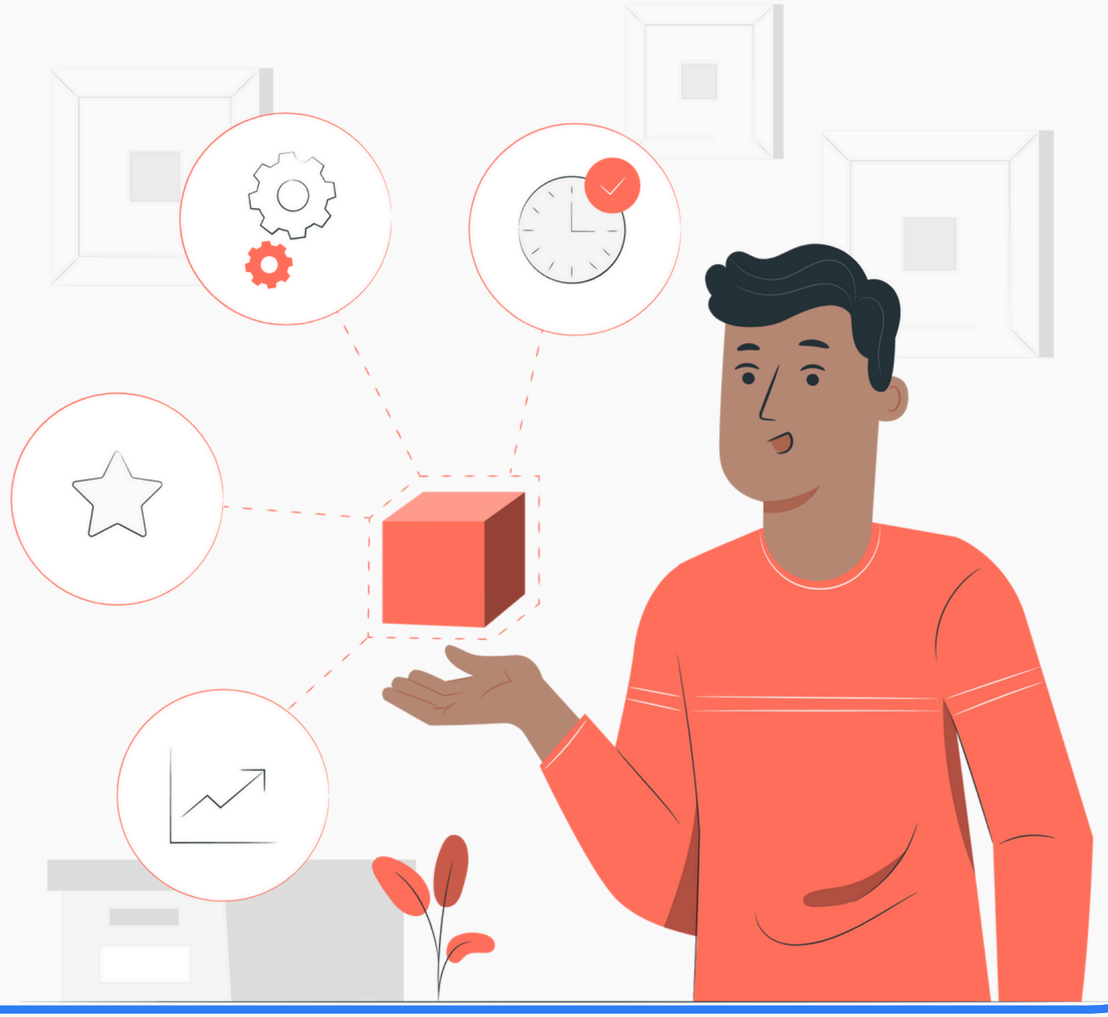
Data Flow Diagram Level 0 (Context Diagram)



Data Flow Diagram Level 1 (Main Processes)



06 Key Features



- 🔐 **Secure JWT Authentication & Google OAuth 2.0**
- 📅 **Intuitive DateTime Scheduling Interface**
- 📊 **Real-Time Message Status Tracking** (Pending, Processing, Sent, Failed)
- ✏️ **CRUD Operations on Pending Messages**
- 📋 **Detailed Delivery Logs & History**
- ⚡ **Responsive Angular UI with Modern Design**

07 Backend Deep Dive

Scheduler Service – The Brain (scheduler.js)

```
// Runs every minute
cron.schedule("* * * * *", async () => {
  const dueMessages = await ScheduledMessage.find({
    status: "pending",
    scheduledTime: { $lte: new Date() },
  });
  dueMessages.forEach(msg => sendMessage(msg));
});
```

- The cron job that fatches and processes all due messages automatically.

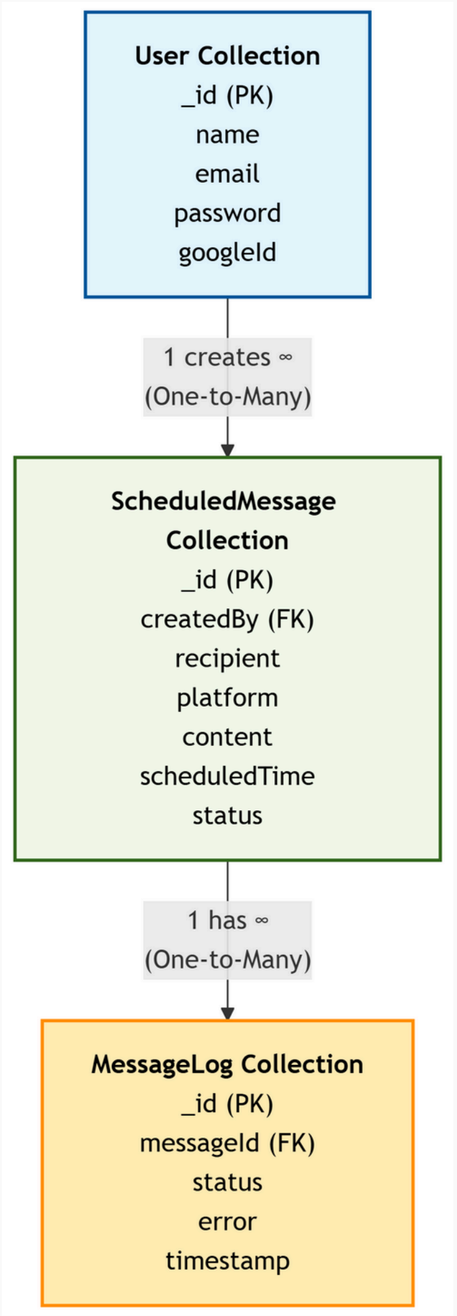
08 Frontend Deep Dive

AuthGuard – Route Protection (auth.guard.ts)

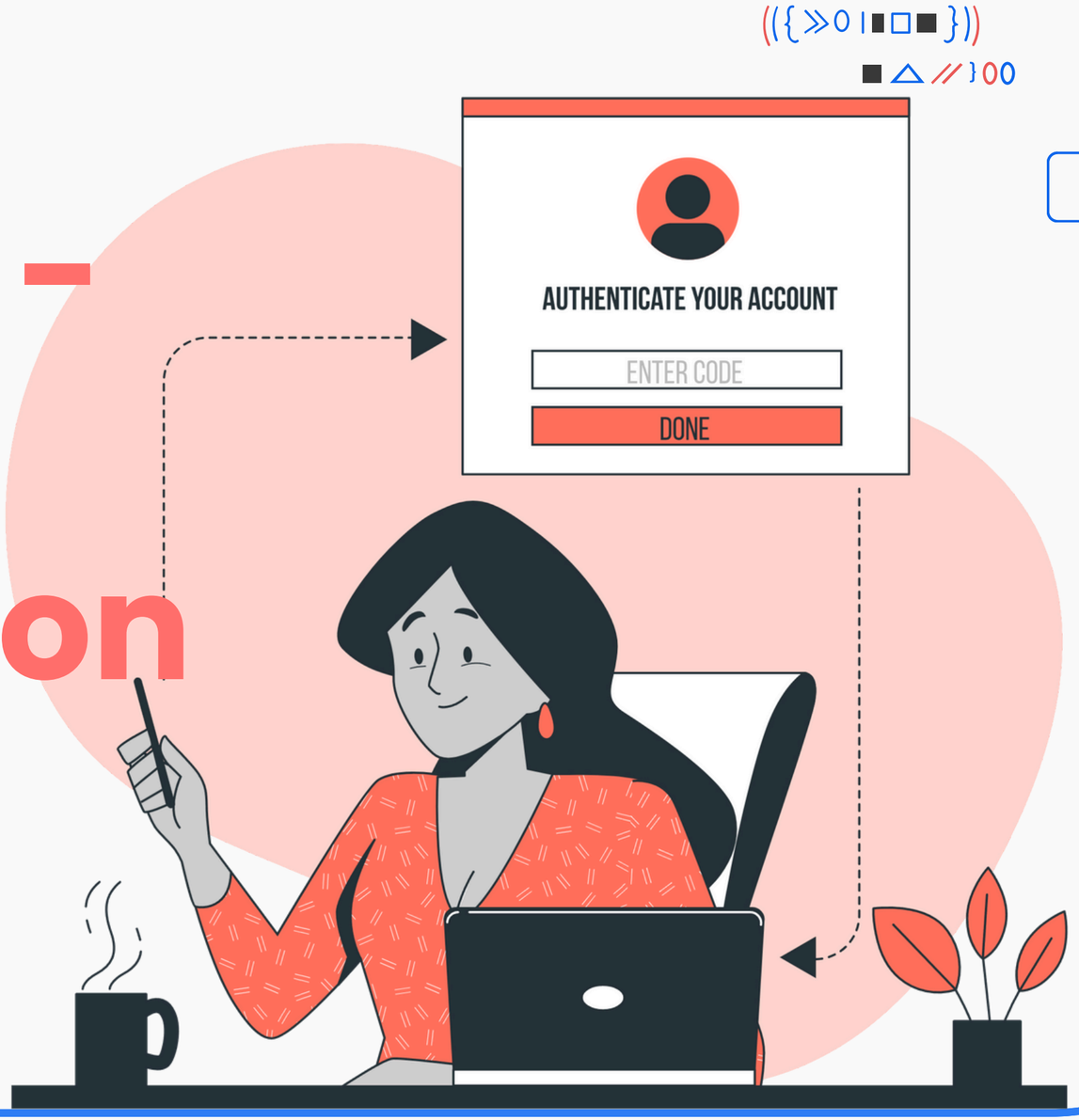
```
export const AuthGuard: CanActivateFn = (route, state) => {
  const token = route.queryParamMap.get('token'); // Handle Google OAuth redirect
  if (token) auth.setToken(token);
  return auth.isLoggedIn() ? true : router.navigate(['/login']);
};
```

- Seamlessly handles both manual login and OAuth redirects while protecting routes.

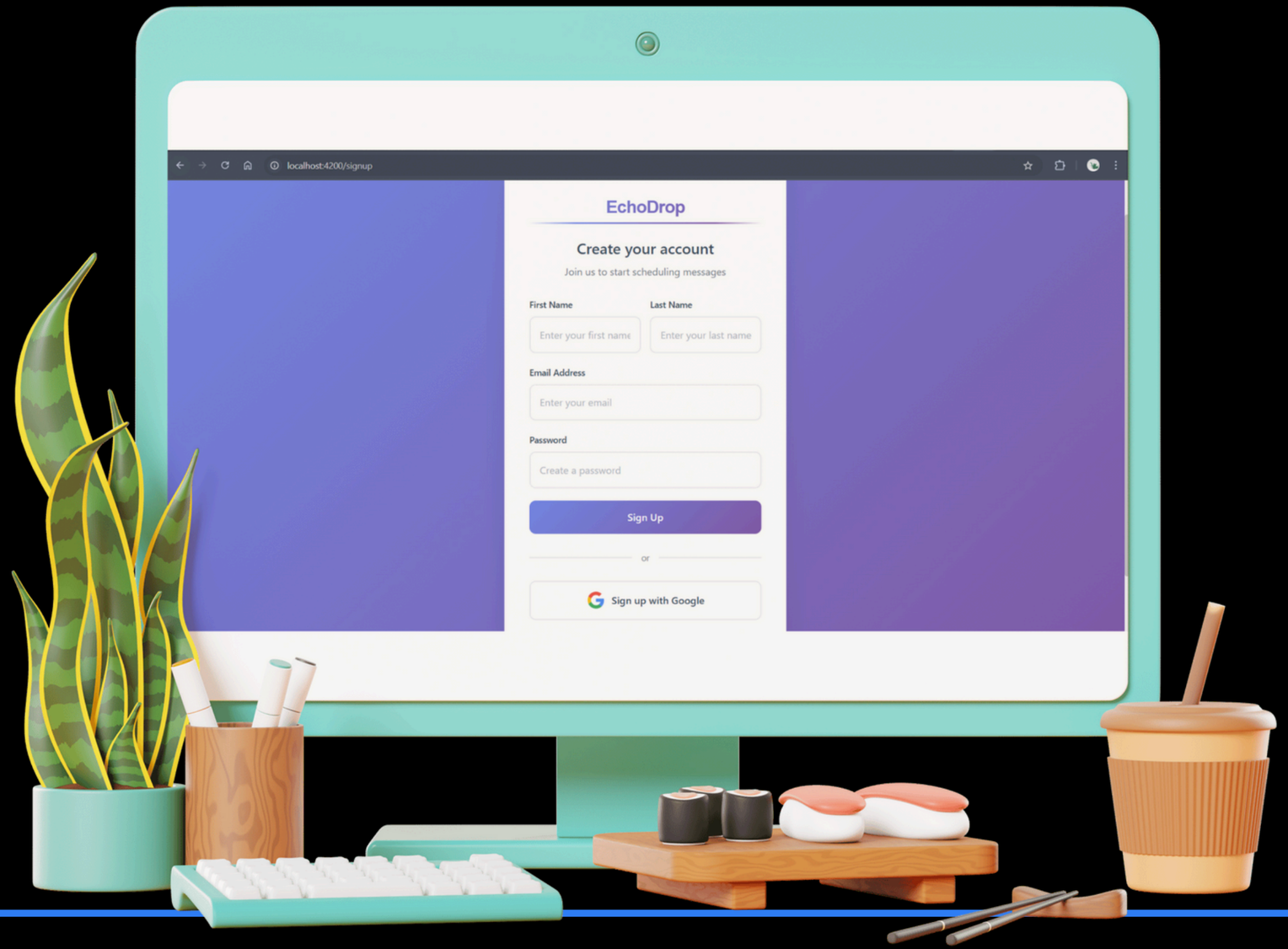
09 Database Schema


User Collection
_id (PK)
name
email
password
googleId
1 creates ∞
(One-to-Many)
ScheduledMessage
Collection
_id (PK)
createdBy (FK)
recipient
platform
content
scheduledTime
status
1 has ∞
(One-to-Many)
MessageLog Collection
_id (PK)
messageId (FK)
status
error
timestamp

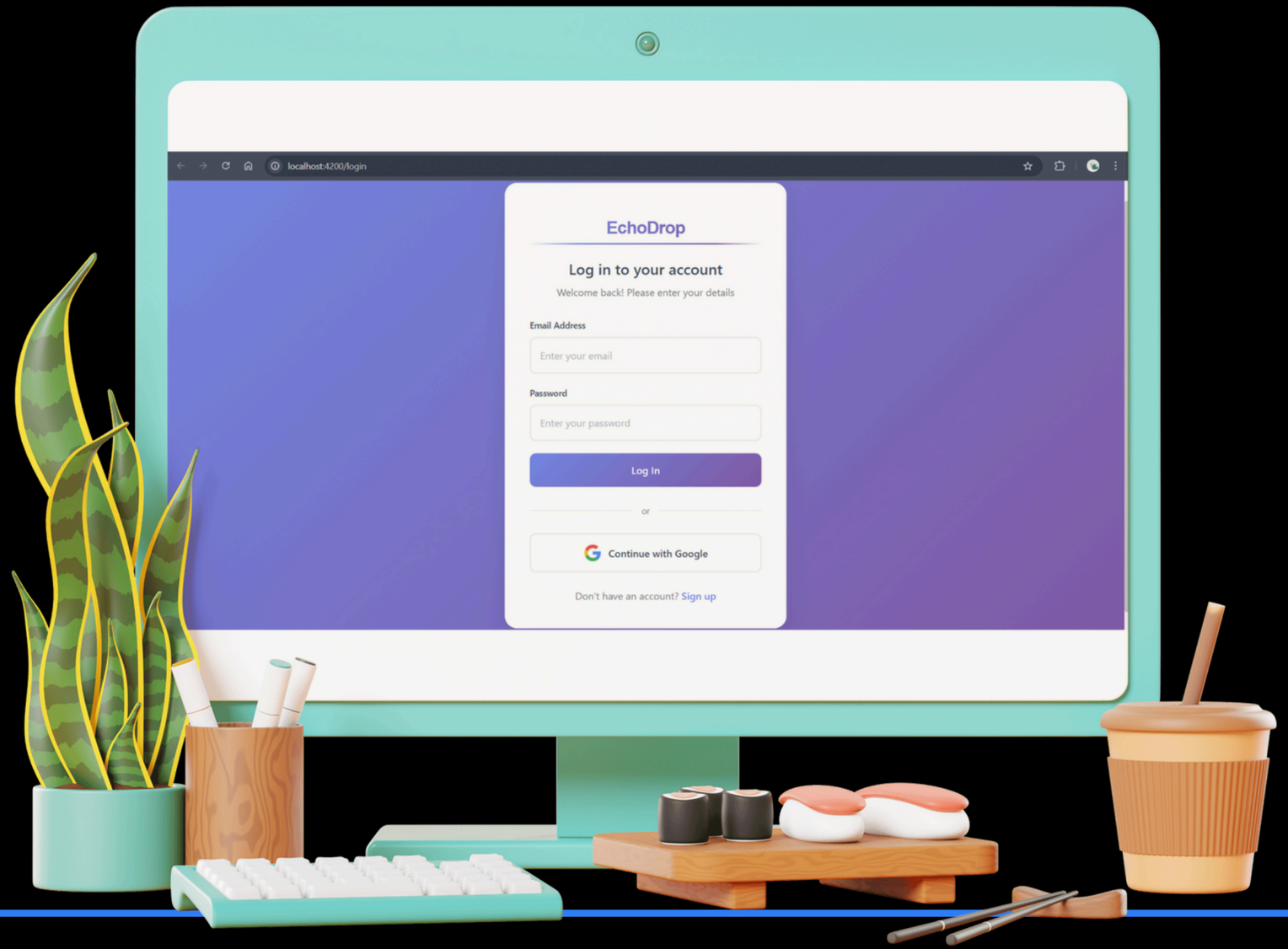
10 Screenshots – User Authentication



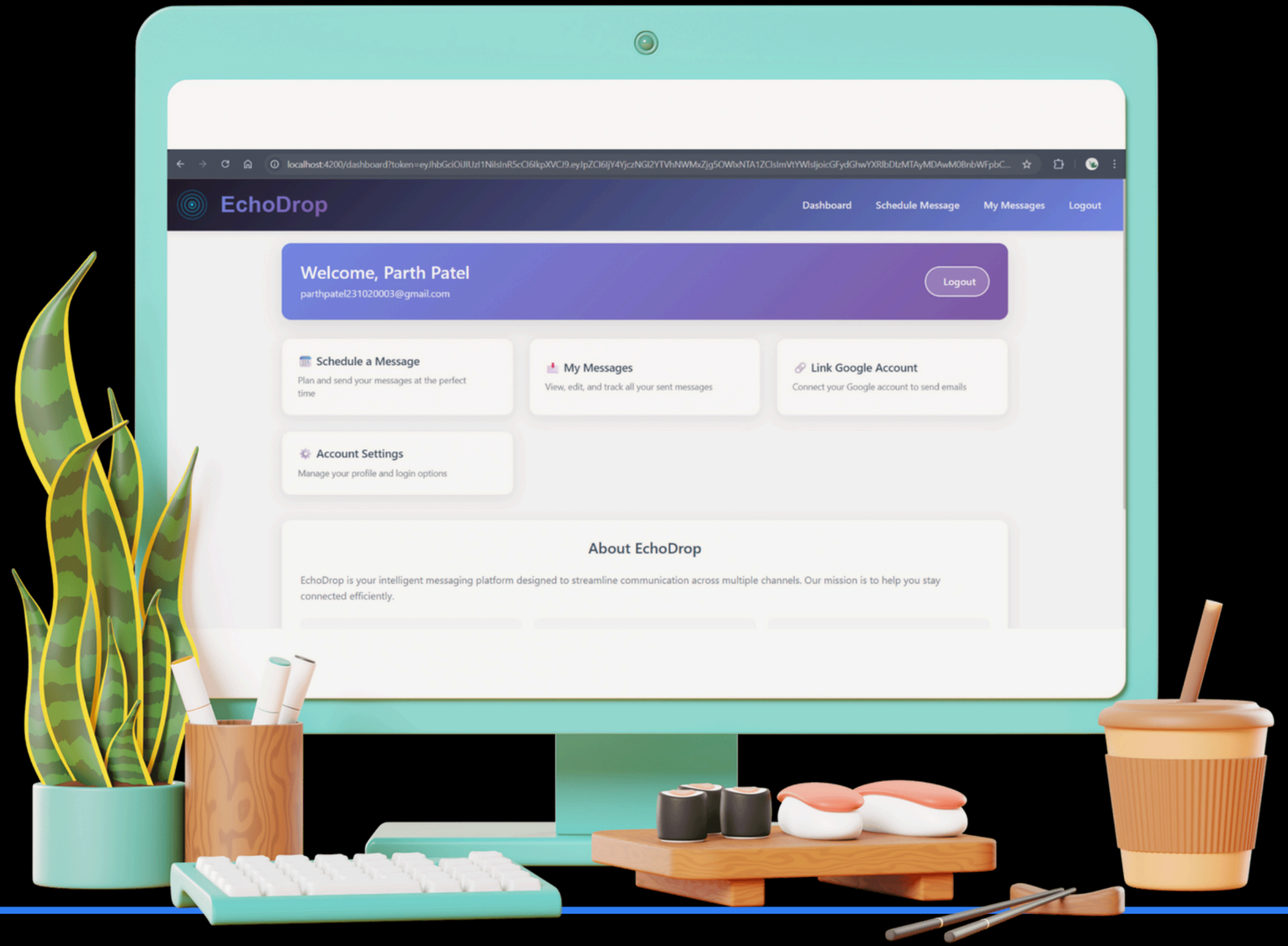
Clean Signup Page with Google OAuth option



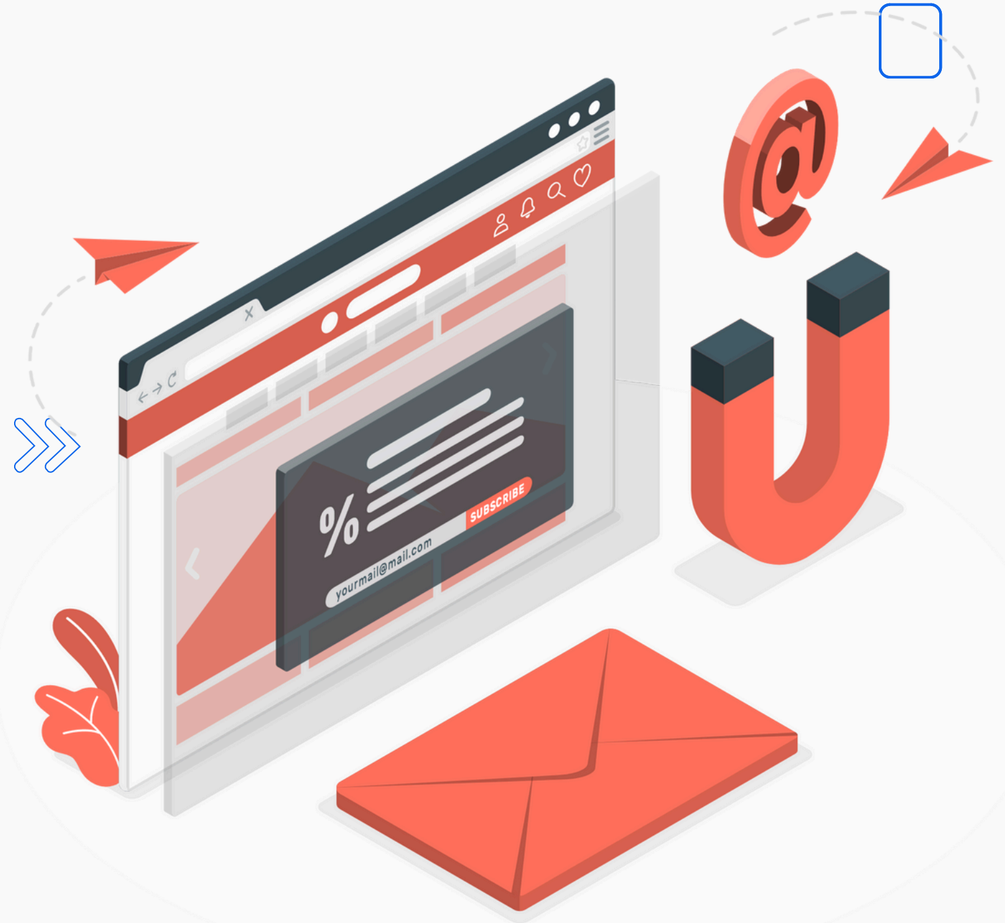
Clean Login Page with Google OAuth option



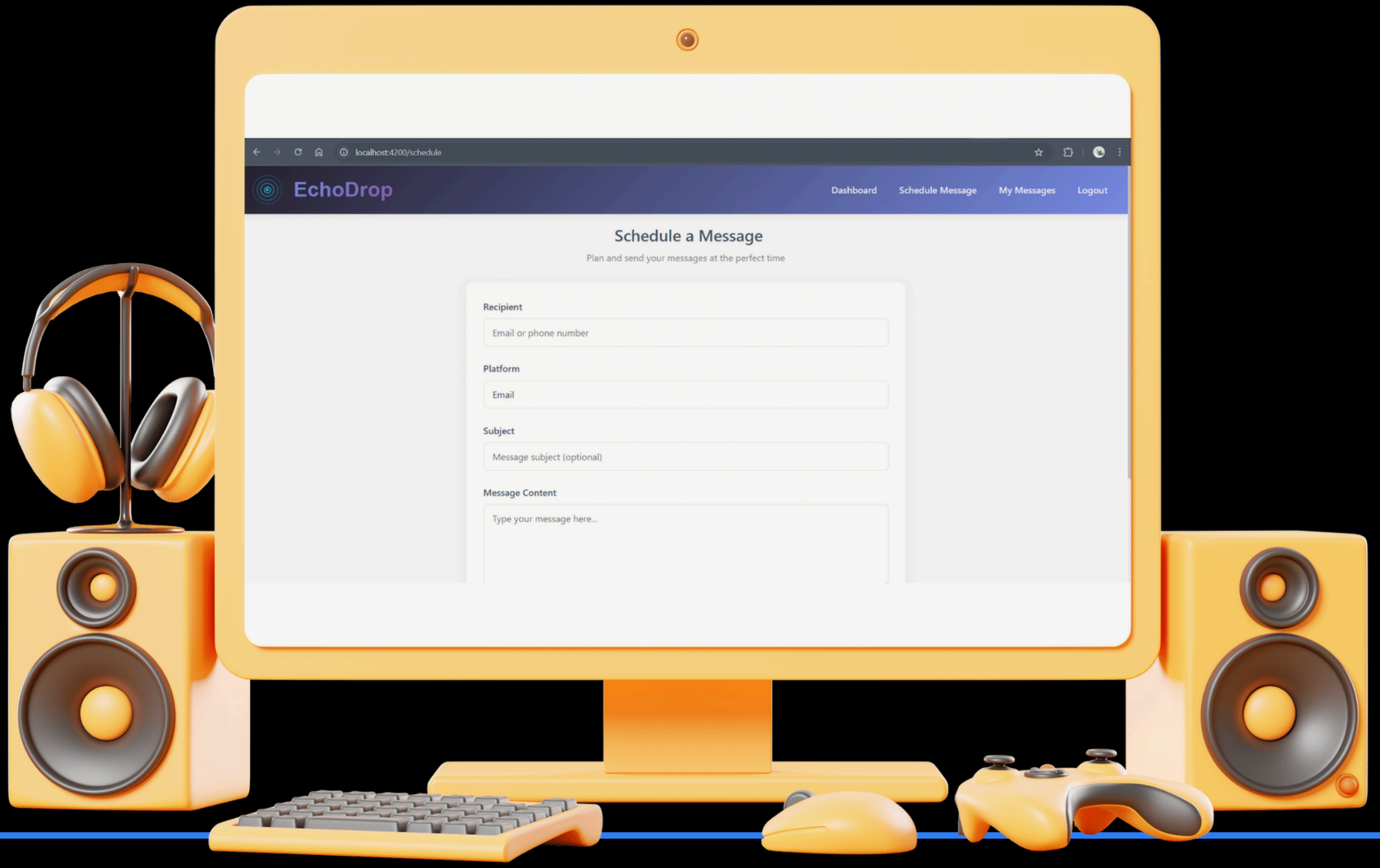
Dashboard Post-Login showing user greeting and access cards



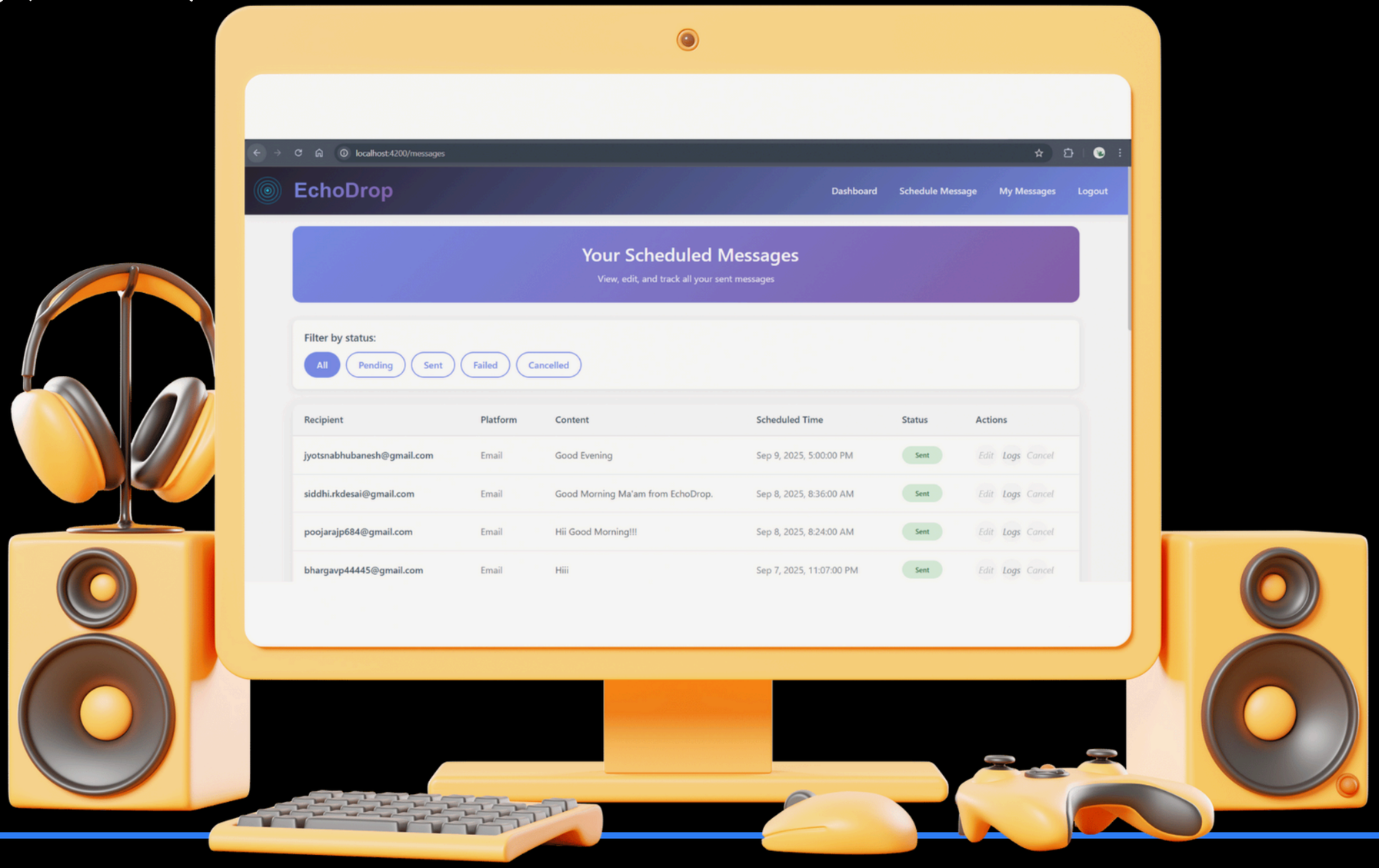
11 Screenshots – Scheduling & Management



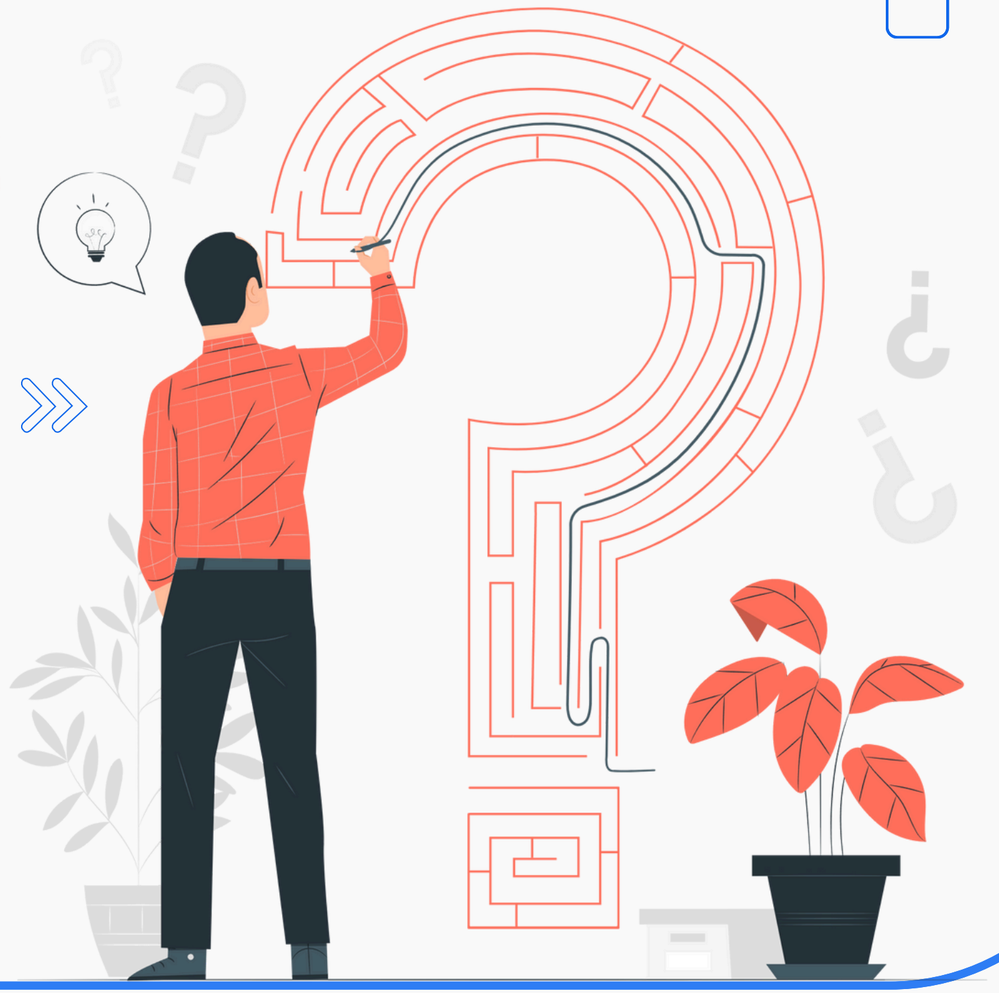
Schedule Form: Platform selector, recipient input, datetime picker



Messages List View: Table with status badges, filter buttons, and action buttons (Edit, Logs, Cancel)

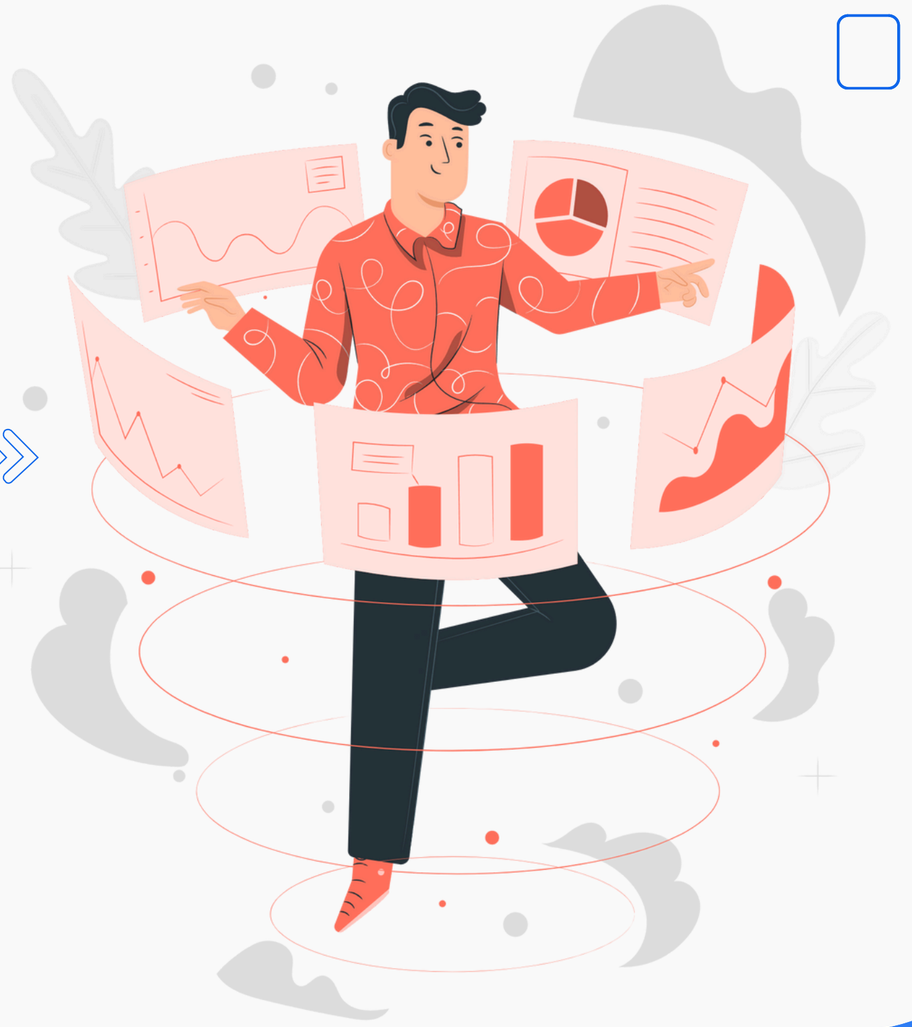


12 Challenges & Solutions



Challenge	Solution
Gmail API Integration	Used OAuth 2.0 with refresh tokens instead of insecure app passwords.
Real-Time Status Updates	Implemented automatic frontend polling every 5 seconds to refresh the message list.
Angular SSR Compatibility	Used isPlatformBrowser() to guard against server-side excution of localStorage code.
Token Handling	AuthGuard captures token from URL after OAuth redirect before navigation.

13 Conclusion & Future Scope



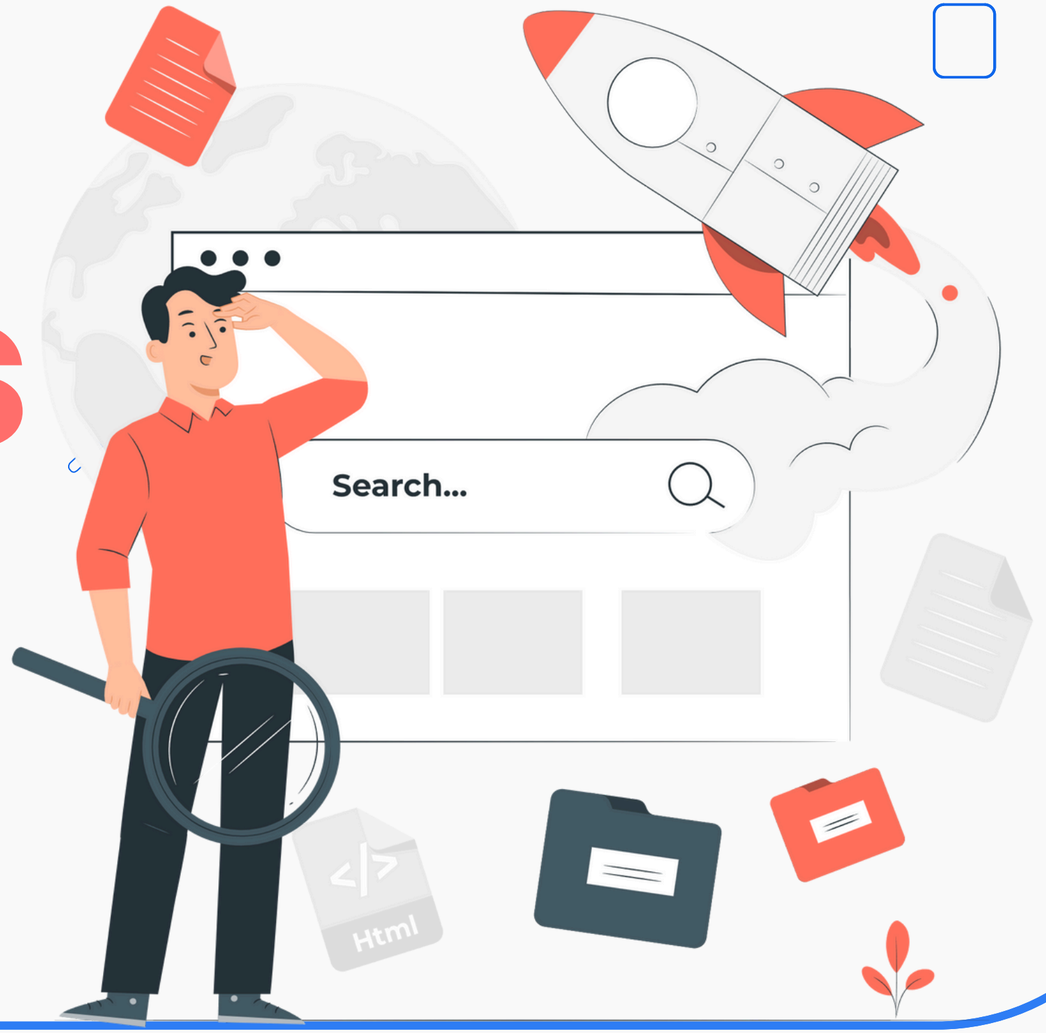
- **Conclusion:**

1. **Successfully developed a fully functional, secure, and scalable scheduled messaging platform.**
2. **Mastered full-stack integration with complex third-party APIs.**

- **Future Scope:**

1. **Group Messaging: Send to multiple recipients.**
2. **Message Templates: Save and reuse common messages.**
3. **Advanced Analytics: Charts and graphs for message performance.**
4. **Mobile App: React Native or Ionic version.**
5. **Current Limitations: Production deployment with full authentication and messaging capabilities.**

14 References



- References:

1. Angular Documentation
2. Node.js & Express.js Guides
3. Twilio API Docs
4. Google Gmail API Docs
5. MongoDB University

Thank You!