**RESQNOW: A REAL-TIME DISASTER ALERT AND EMERGENCY RESPONSE SYSTEM.**

**1. Introduction**:

**ResQNow** is a fully serverless, web-based platform designed to help people in disaster-prone or conflict-affected areas send SOS alerts and get help faster. Whether someone is stuck during a flood, landslide, earthquake, or even in a war zone, ResQNow gives them a way to ask for help using their live GPS location. It connects **citizens, volunteers**, and **NGOs** in real-time to enable faster, smarter disaster response — all without relying on traditional infrastructure.

This project was built with compassion, keeping in mind people in vulnerable regions like **Palestine**, where government systems may fail or access is limited. ResQNow is my attempt to combine cloud technology and humanity to save lives when every second counts.

**2. Aim**:

* Allow victims to send **emergency SOS alerts** quickly using location and urgency level
* Notify **NGOs and volunteers** in real-time so help reaches faster
* Provide an **admin dashboard** with data insights to help make better decisions
* Show SOS locations on a **live map** so responders can locate victims instantly
* Build a secure, scalable, and cloud-based system that works even during extreme situations

**3. AWS Services which were used in ResQNow**:

These cloud services helped me build ResQNow without maintaining any physical servers. Each one plays a specific role in making the system **real-time**, **secure**, and **scalable**.

**1. Amazon Cognito – User Authentication**

* Used to handle **user signup and login** securely.
* It manages usernames, passwords, and tokens, and handles things like password resets and user sessions.
* This means I didn’t need to write my own login system or store sensitive credentials.

Example: When a new user signs up or logs in to ResQNow, Cognito verifies them before they can use the platform.

**2. AWS Lambda – Backend Without Servers**

* Lambda is used to run the backend code (like when a user sends an SOS or signs up).
* The best part? **No need to manage servers** — the code only runs when needed (event-driven), saving cost and time.
* I used Lambda for tasks like saving data to the database, generating presigned URLs, and handling form submissions.

Think of it like “functions in the cloud” that respond to API calls.

**3. API Gateway – The Entry Point for APIs**

* Acts as a **gatekeeper** between the frontend and the backend (Lambda).
* When the frontend wants to send or get data, it goes through API Gateway first.
* It ensures proper routing, handles errors, and also takes care of CORS (cross-origin permissions).

It helps securely expose your Lambda functions over the internet.

**4. DynamoDB – Fast NoSQL Database**

* DynamoDB stores all dynamic data like:
  + User profiles
  + SOS alerts
  + Volunteer responses
  + Alert statuses (e.g., Resolved, Active)
* It's fast, serverless, and scales automatically — perfect for real-time, high-volume applications.

Unlike SQL databases, DynamoDB stores data as flexible “items” in a table — great for speed and simplicity.

**5. Amazon S3 – Hosting the Website**

* I used S3 to **host my React + Vite website**. It acts like a web server for static files like index.html, CSS, JS, etc.
* Once I build the frontend using npm run build, the dist/ folder is uploaded to S3.
* S3 is fast, reliable, and great for hosting static websites.

Think of it as putting your website into a special online folder and making it public.

**6. CloudFront – Speed + HTTPS**

* CloudFront is a **Content Delivery Network (CDN)** that delivers the website fast to users globally.
* I also used it to connect a **custom domain** (resqnow.org) and enable **HTTPS** (SSL certificate from ACM).
* CloudFront caches content closer to users and adds security layers.

It’s like giving your S3-hosted website turbo speed + a secure green lock on the browser.

**7. Amazon AppSync – Real-time GraphQL**

* AppSync connects the **admin dashboard** to DynamoDB using **GraphQL**.
* It lets the admin fetch filtered, real-time data like:
  + Number of active SOS alerts
  + Alerts by location or urgency
  + Volunteer response times

It simplifies backend logic for dashboards and supports **real-time updates**.

**8. IAM (Identity & Access Management) – Roles & Permissions**

* IAM controls **who can access what** inside the AWS environment.
* I used IAM roles to securely allow:
  + Lambda functions to access DynamoDB
  + S3 buckets to connect with CloudFront
  + Cognito to read/write user data

It’s like giving each service its own ID card and limiting what it can do.

**3. Frontend & Other Technologies Used**

In addition to AWS, I also used modern frontend and third-party tools:

**a. React + Vite + Tailwind CSS:**

* **React**: JavaScript library used to build dynamic, component-based UI.
* **Vite**: Fast build tool that bundles the app for production.
* **Tailwind CSS**: Utility-first CSS framework for styling — helps build clean, responsive UIs quickly.

Together, they gave me speed, flexibility, and beautiful design with minimal effort.

**b. Mapbox (Third-party Service):**

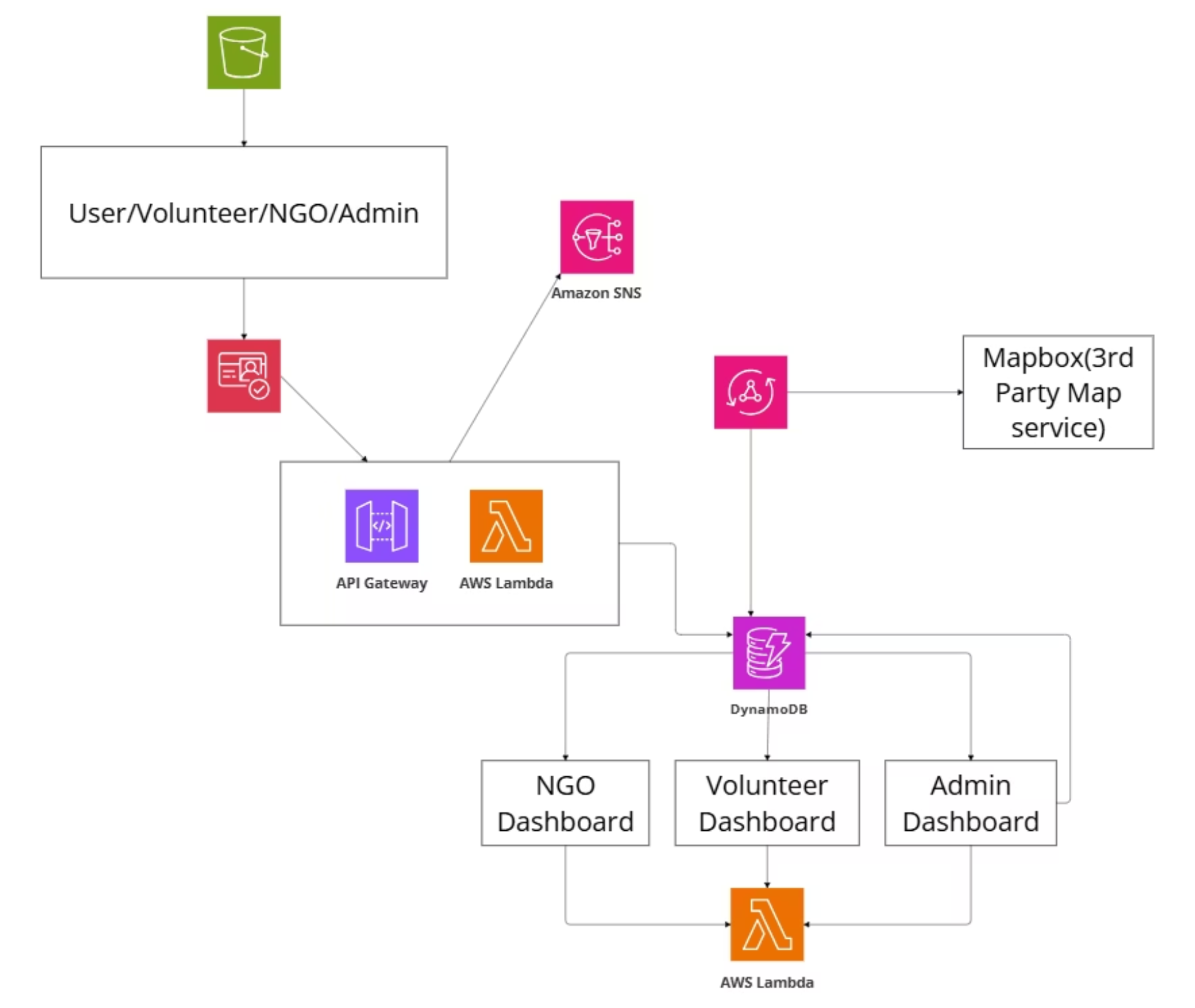
* I used **Mapbox** to display live **SOS alert locations on a map**.
* It’s highly customizable and mobile-friendly — perfect for visualizing GPS coordinates in real-time.

This helps volunteers and NGOs immediately see where help is needed most.

**4. Project Steps:**

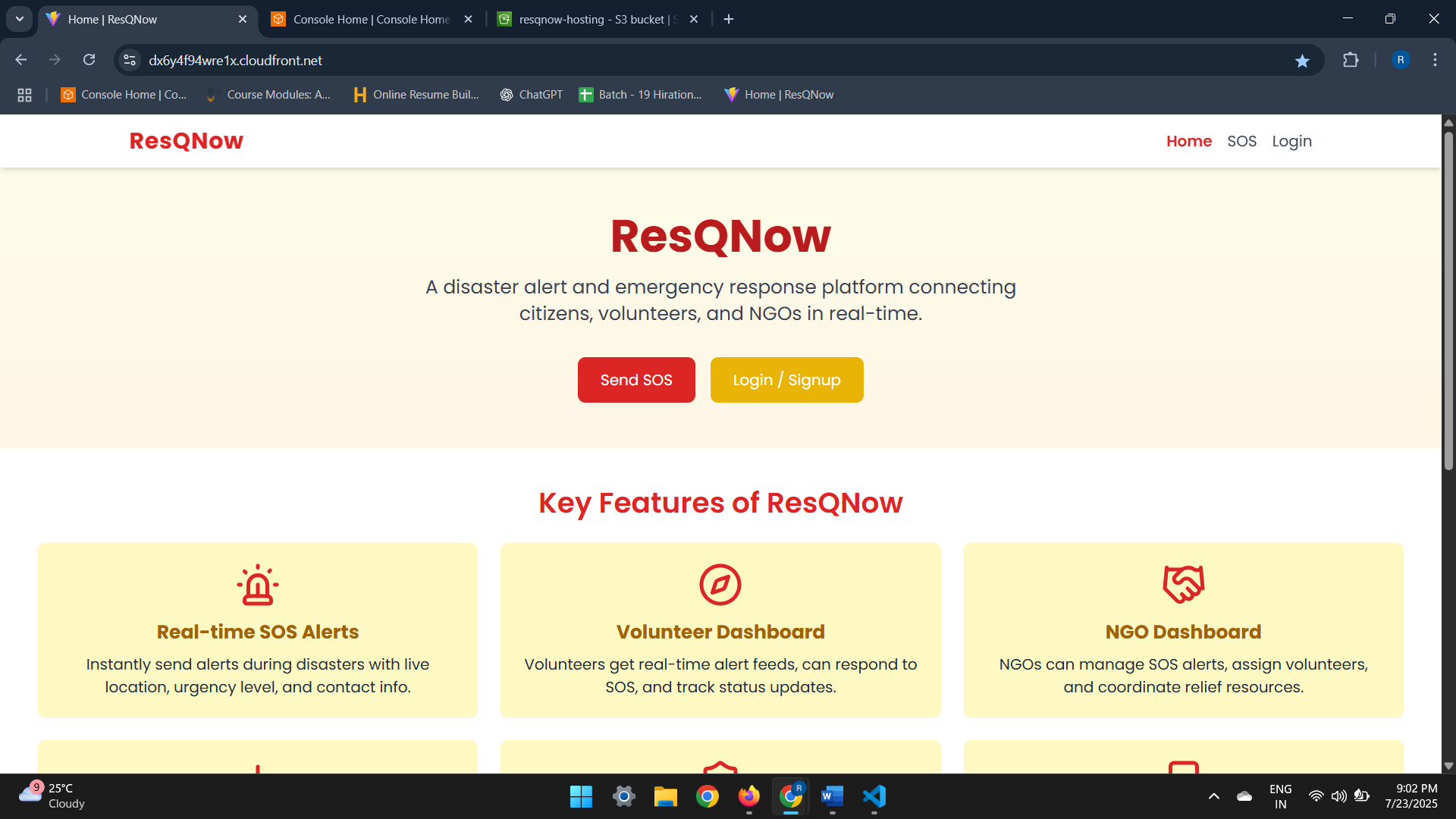
1. **User logs in / signs up**  
   → Authenticated securely using **AWS Cognito**
2. **User sends an SOS alert**  
   → Alert contains **live GPS location**, urgency, and contact info  
   → A confirmation **SMS is sent via AWS SNS** (e.g., “Help is on the way”)
3. **SOS alert is saved in DynamoDB**  
   → Automatically accessible to **NGO**, **Volunteer**, and **Admin dashboards**
4. **NGO Dashboard**  
   → NGO can:
   * View live alerts
   * **Assign volunteers**
   * **Change alert status** (Pending / Resolved)
5. **Volunteer Dashboard**  
   → Volunteers can:
   * View alerts
   * Join a mission
   * Report back to NGO on the **status** of the incident
   * View the **location on Mapbox map**
6. **Admin Dashboard**  
   → Admin can:
   * View real-time **graphs and analytics** (powered by **AWS AppSync + GraphQL**)
   * See all **users, NGOs, volunteers**
   * Monitor all **SOS alerts**
   * Visualize total alerts on a **live map (Mapbox)**

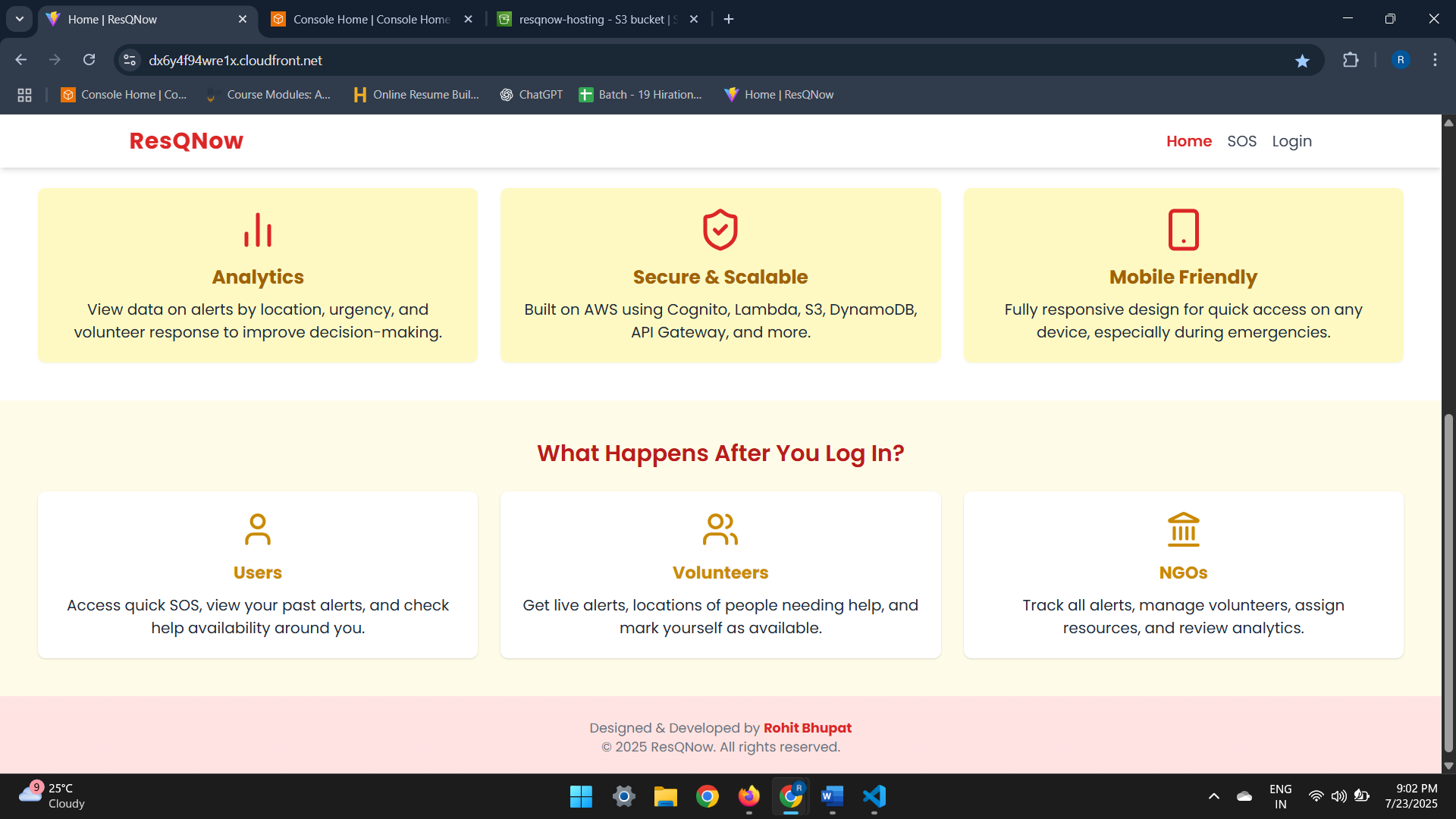
**5. Architecture**:



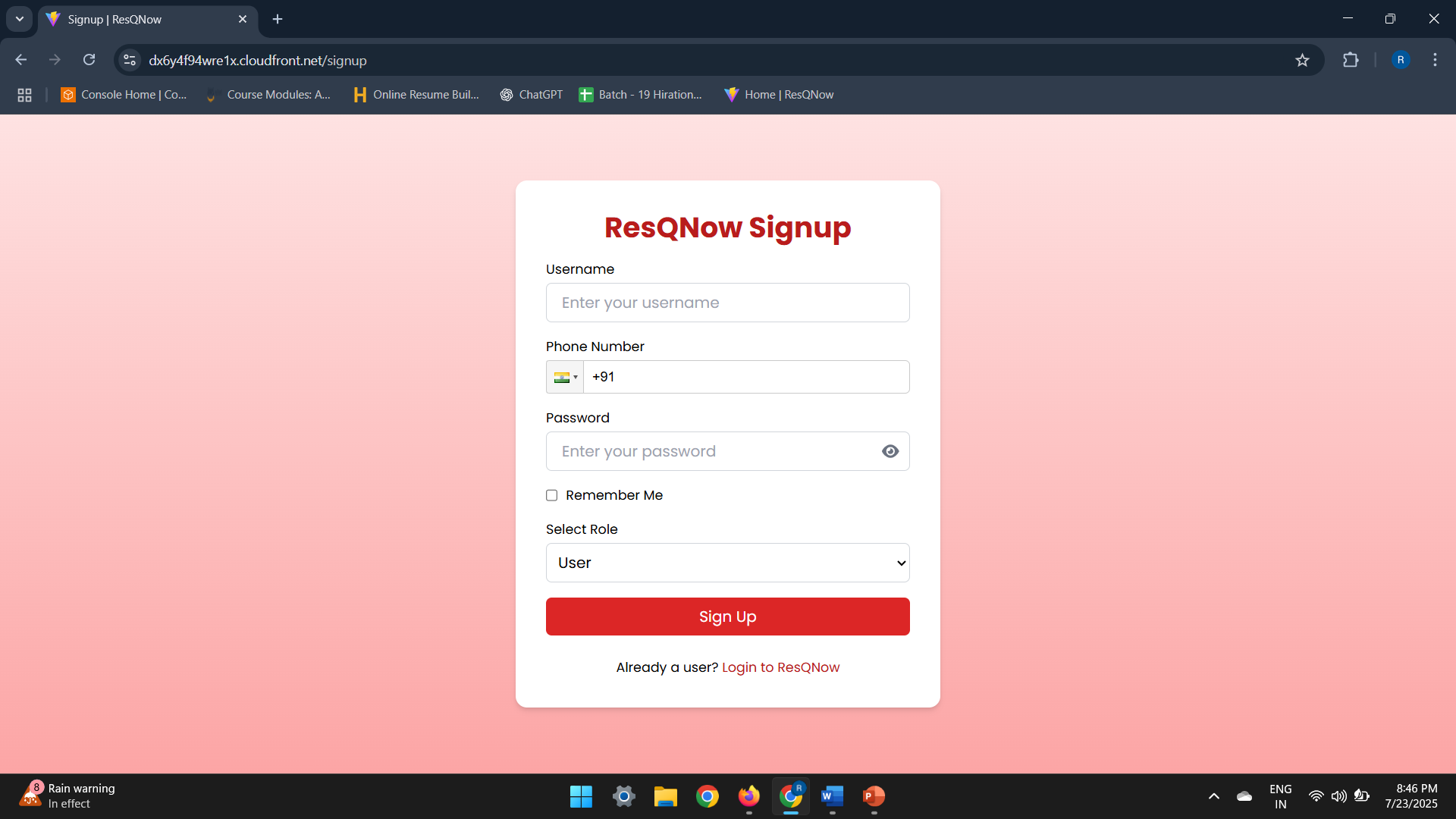
**6. Screenshots**:

**Home Page**:

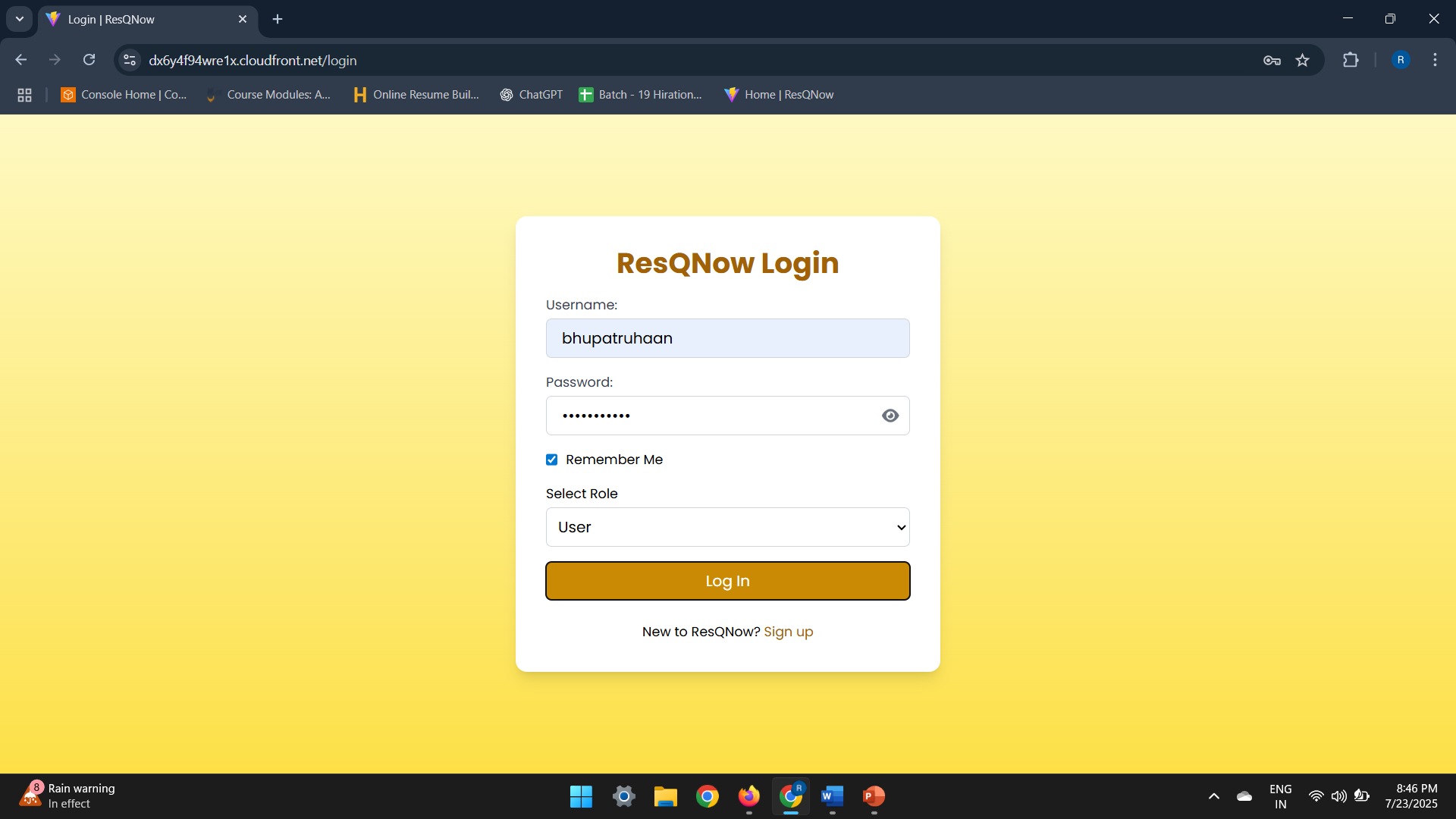




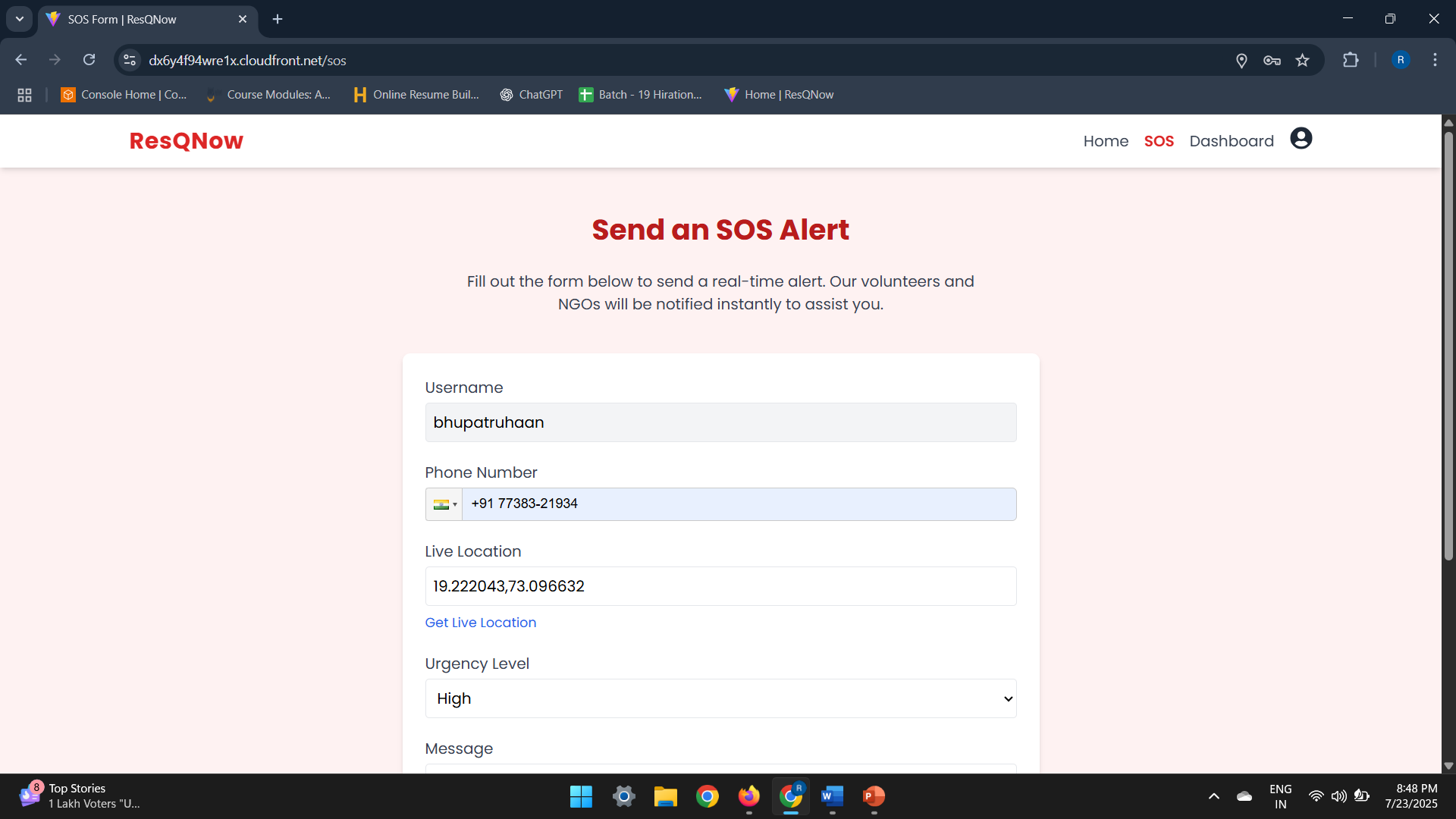
**Signup Page**:

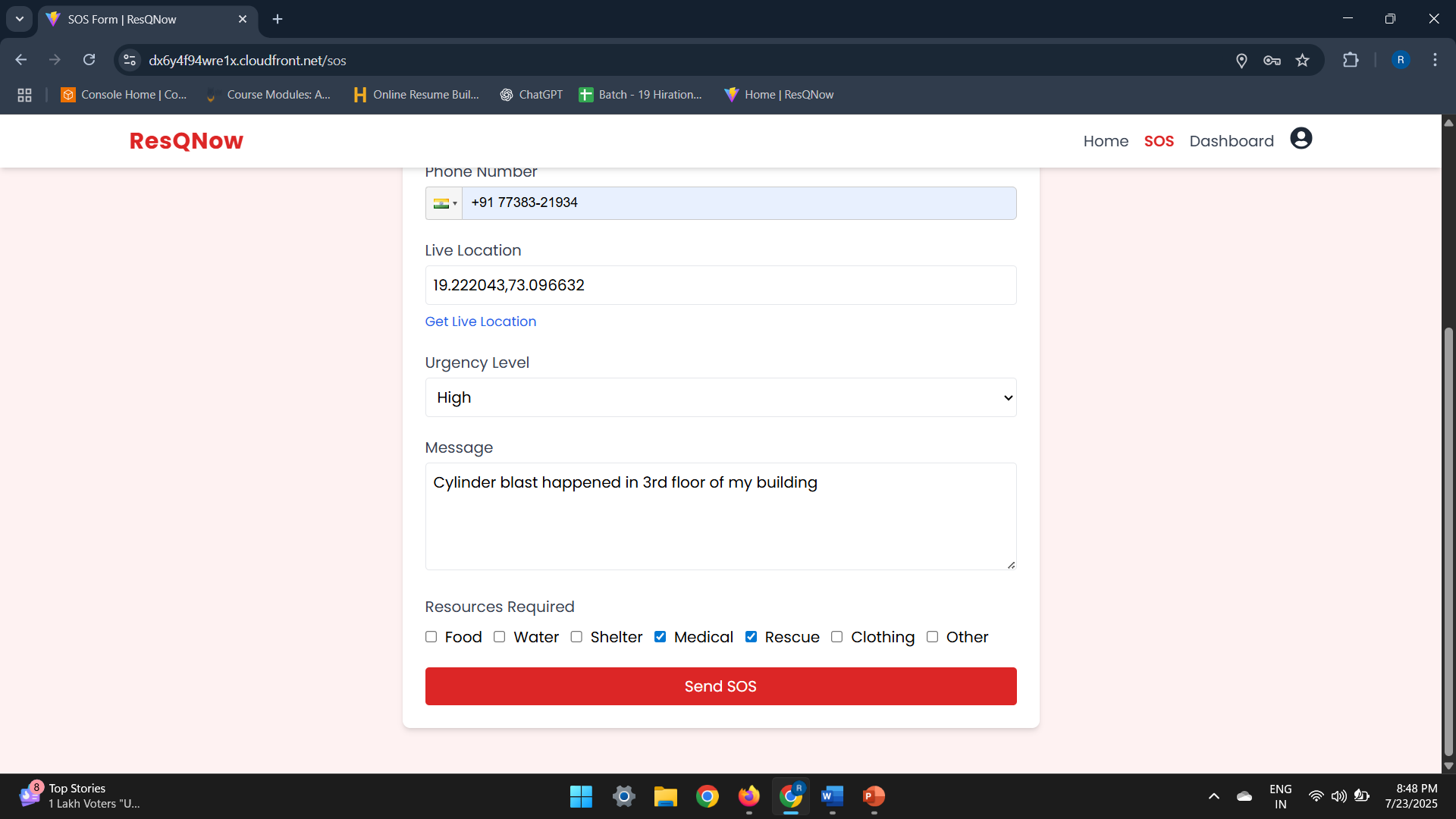


**Login Page**: User/Ngo/Volunteer/Admin can login from here

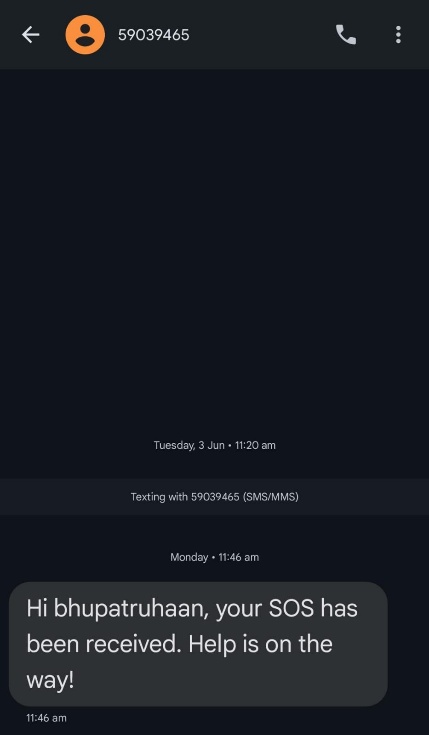


**SOS Alert form**:

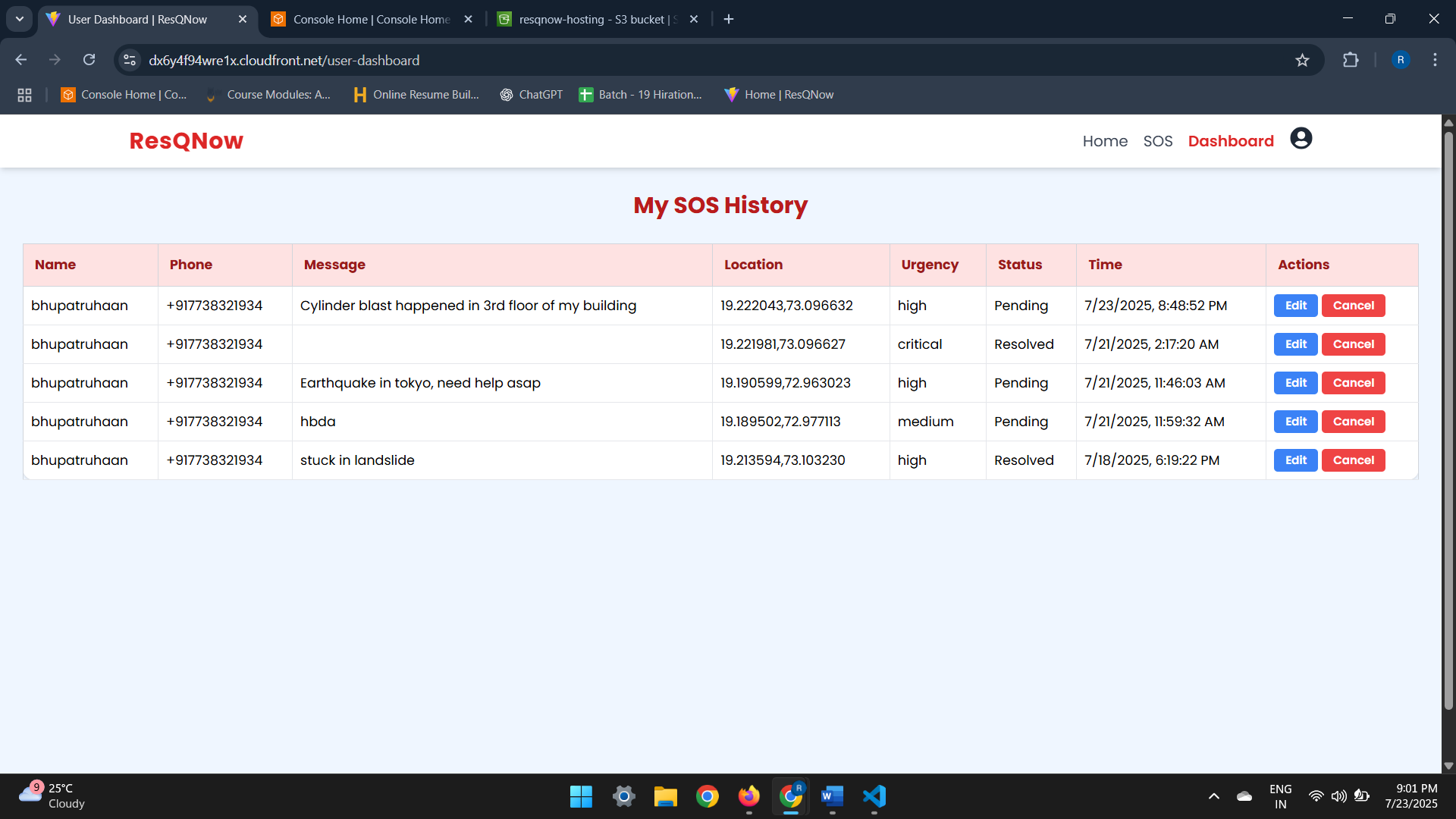




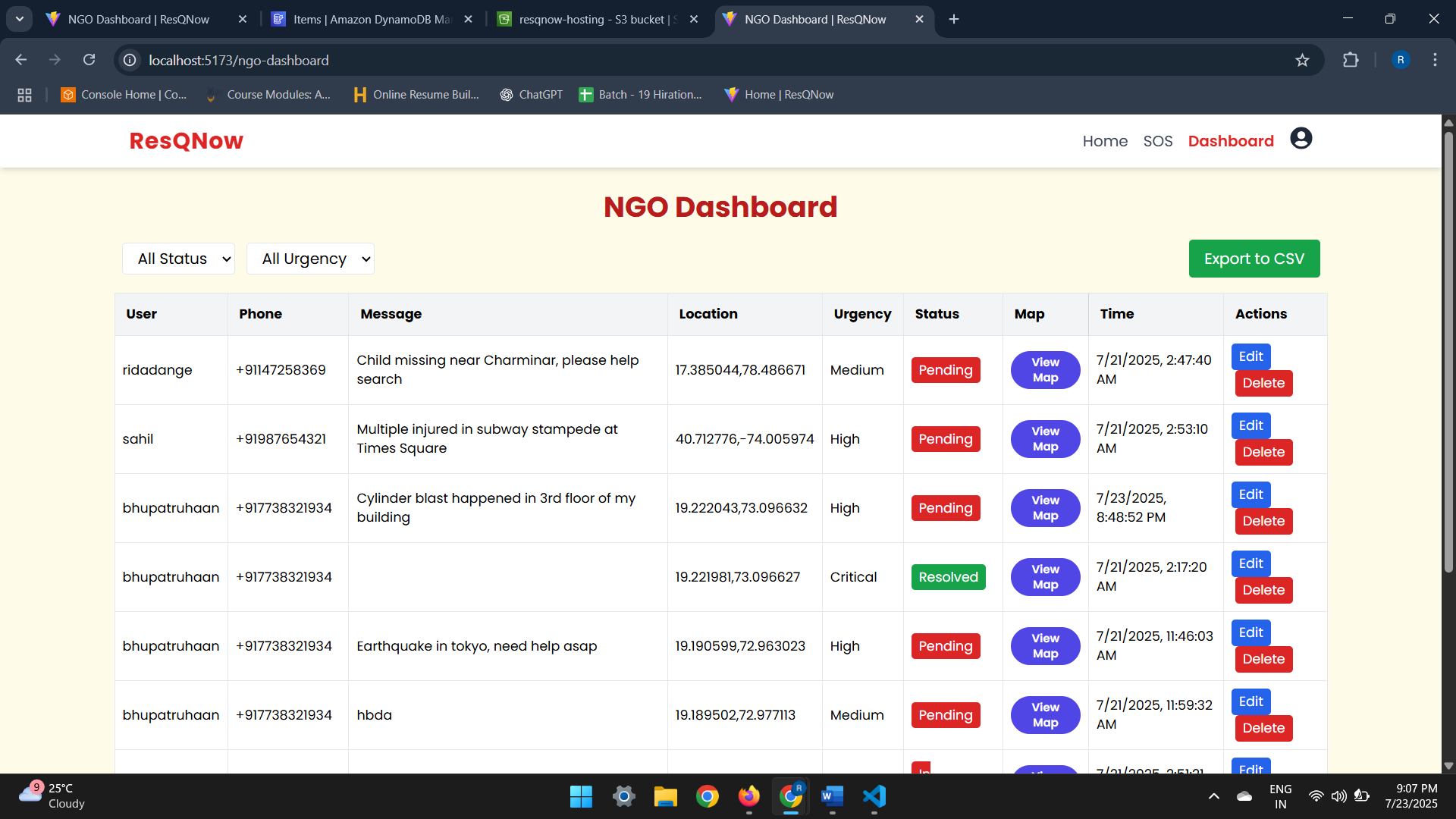
**SOS alert sent sms from Amazon SNS.**



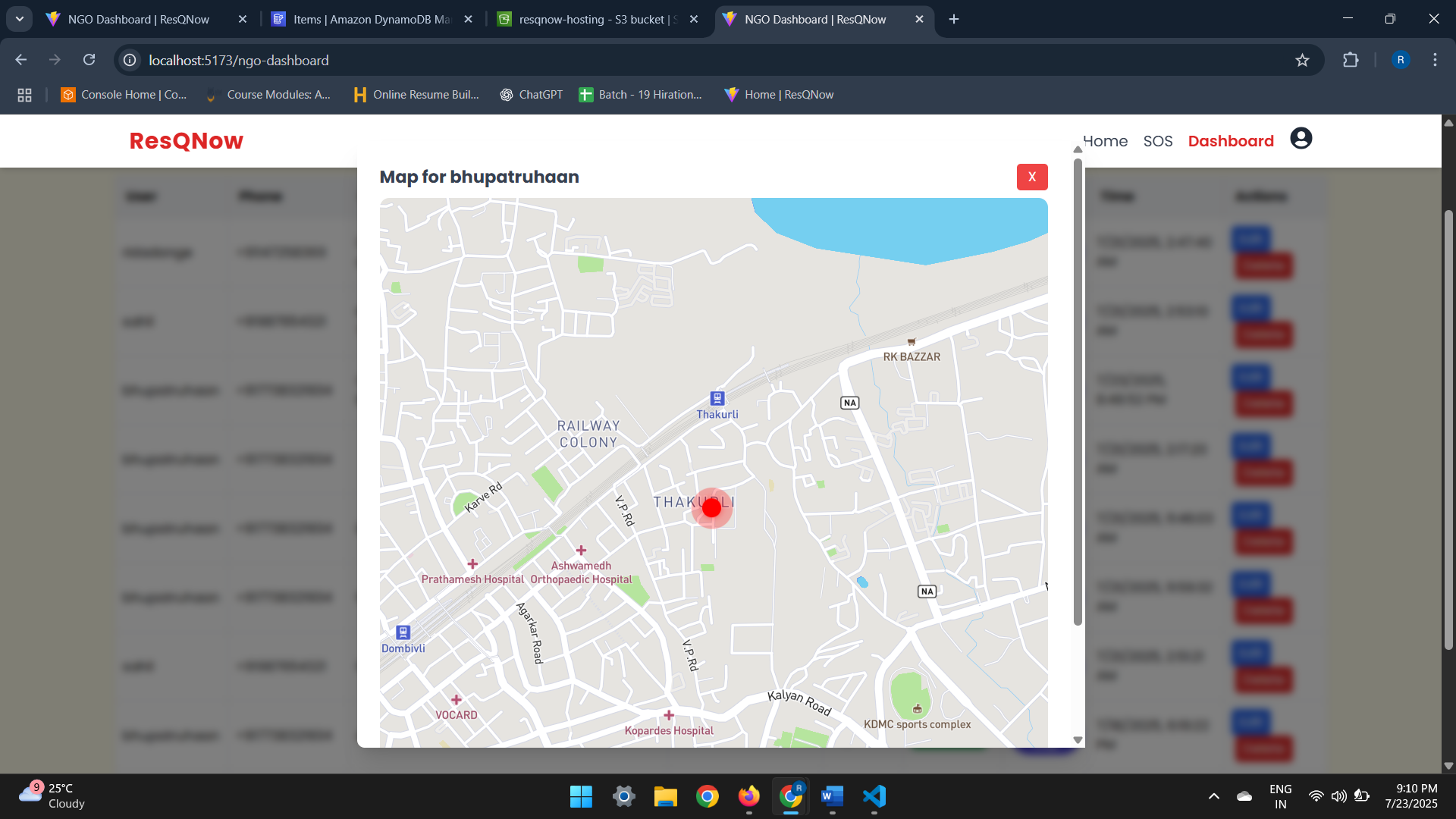
**SOS History of User**:



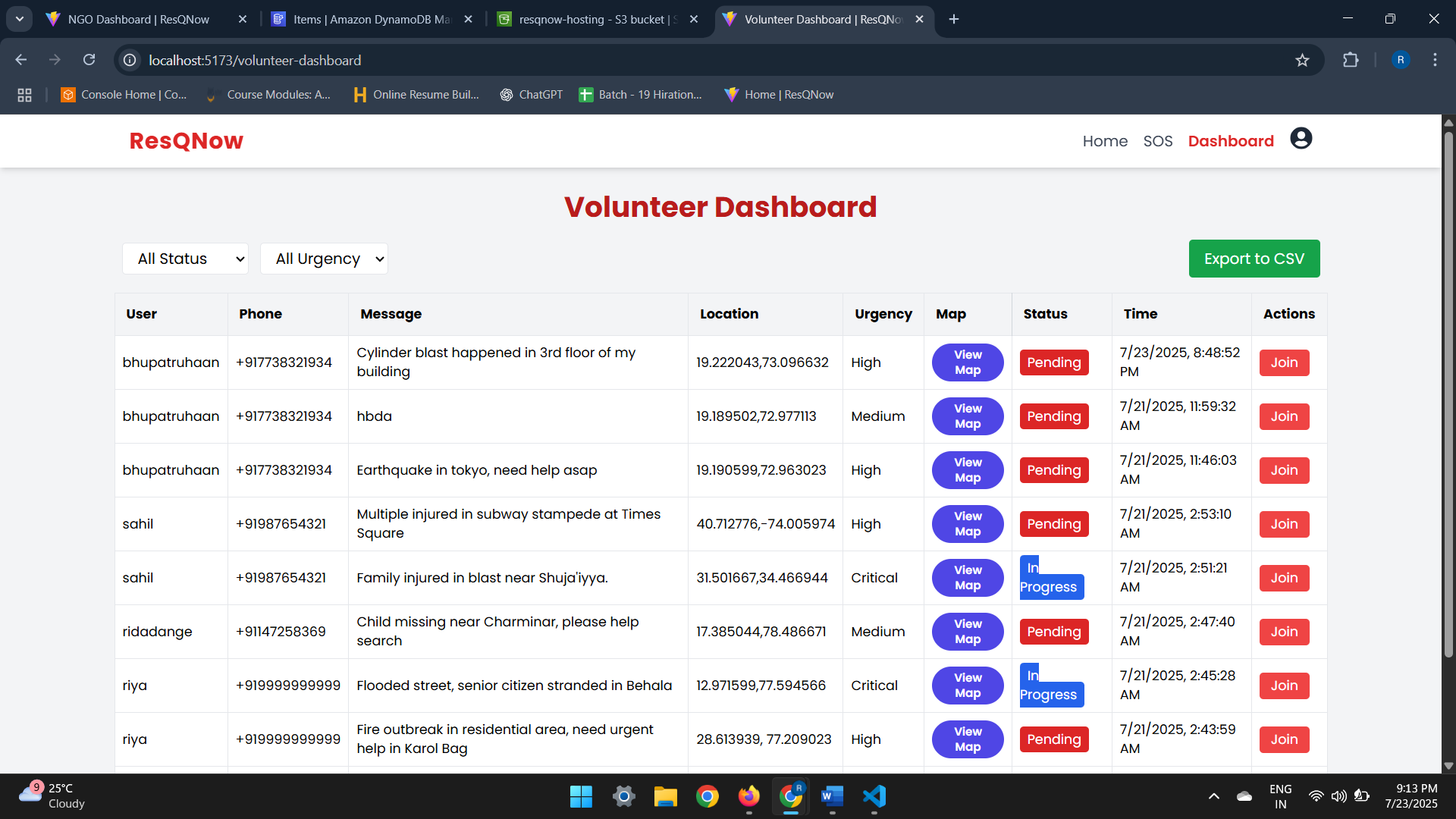
**NGO Dashboard**:



**View Map of sos from NGO Dashboard**:



**Volunteer Dashboard**:



**Admin Dashboard**:

