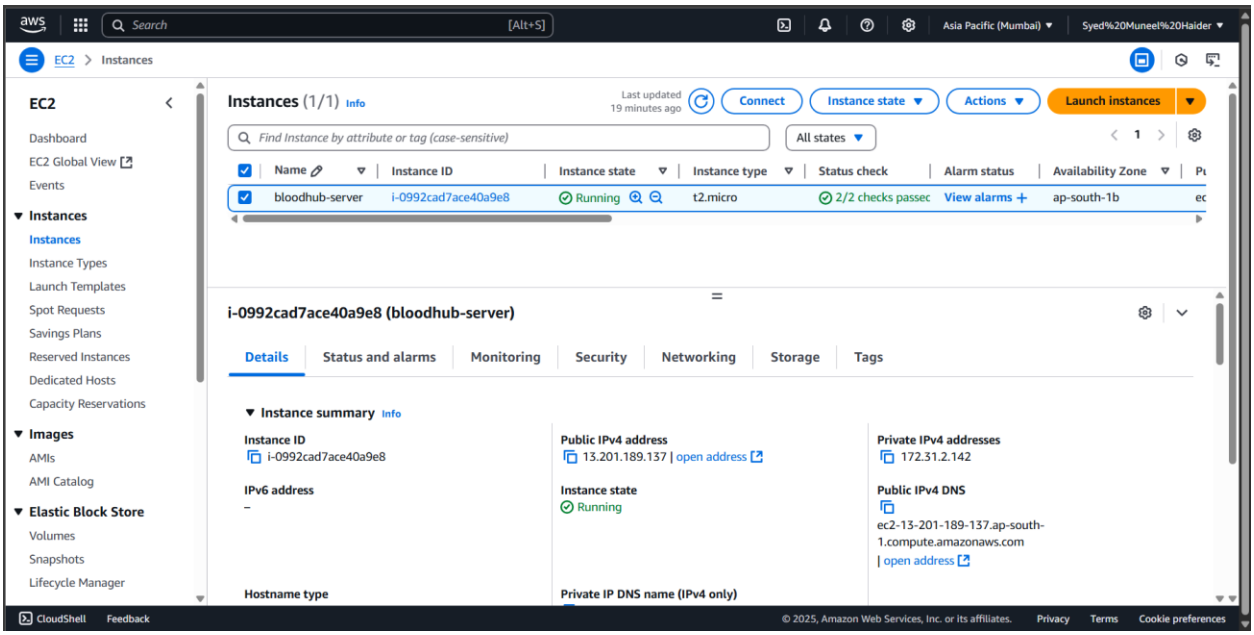
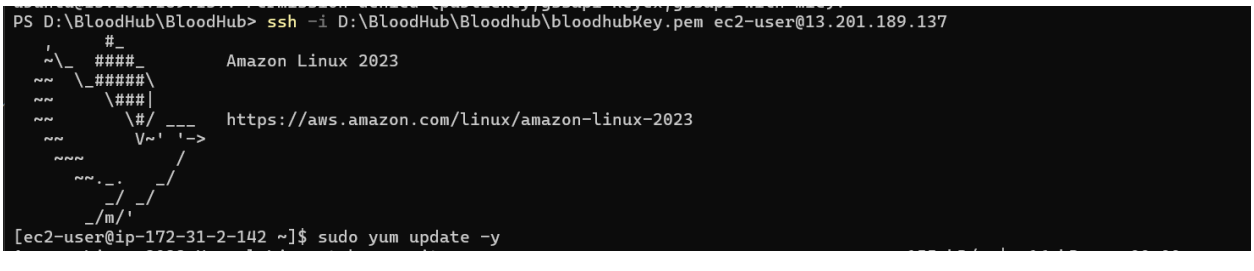
GitHub: https://github.com/MuneelHaider/BloodHub

BACKEND:

1. Launching EC2 Instance:



2. Connecting to EC2 Machine:



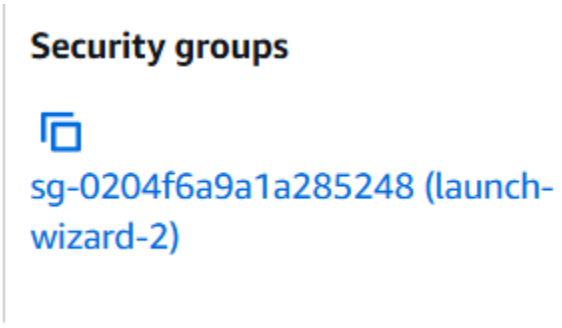
3. Using Docker to build the images and up the containers:

```
[ec2-user@ip-172-31-2-142 server]$ docker build -t bloodhub-backend .
[+] Building 33.8s (10/10) FINISHED                                                                                  docker:default
 => [internal] load build definition from Dockerfile                                                                          0.0s
 => => transferring dockerfile: 229B                                                                                          0.0s
 => [internal] load metadata for docker.io/library/node:18                                                                    2.3s
 => [internal] load .dockerignore                                                                                             0.0s
 => => transferring context: 2B                                                                                               0.0s
 => [1/5] FROM docker.io/library/node:18@sha256:867be01f97d45cb7d89a8ef0b328d23e8207412ebec4564441ed8cabc8cc4ecd             24.0s
 => => resolve docker.io/library/node:18@sha256:867be01f97d45cb7d89a8ef0b328d23e8207412ebec4564441ed8cabc8cc4ecd               0.0s
 => => sha256:867be01f97d45cb7d89a8ef0b328d23e8207412ebec4564441ed8cabc8cc4ecd 6.41kB / 6.41kB                                0.0s
```

4. Checking and confirmation if containers are up.

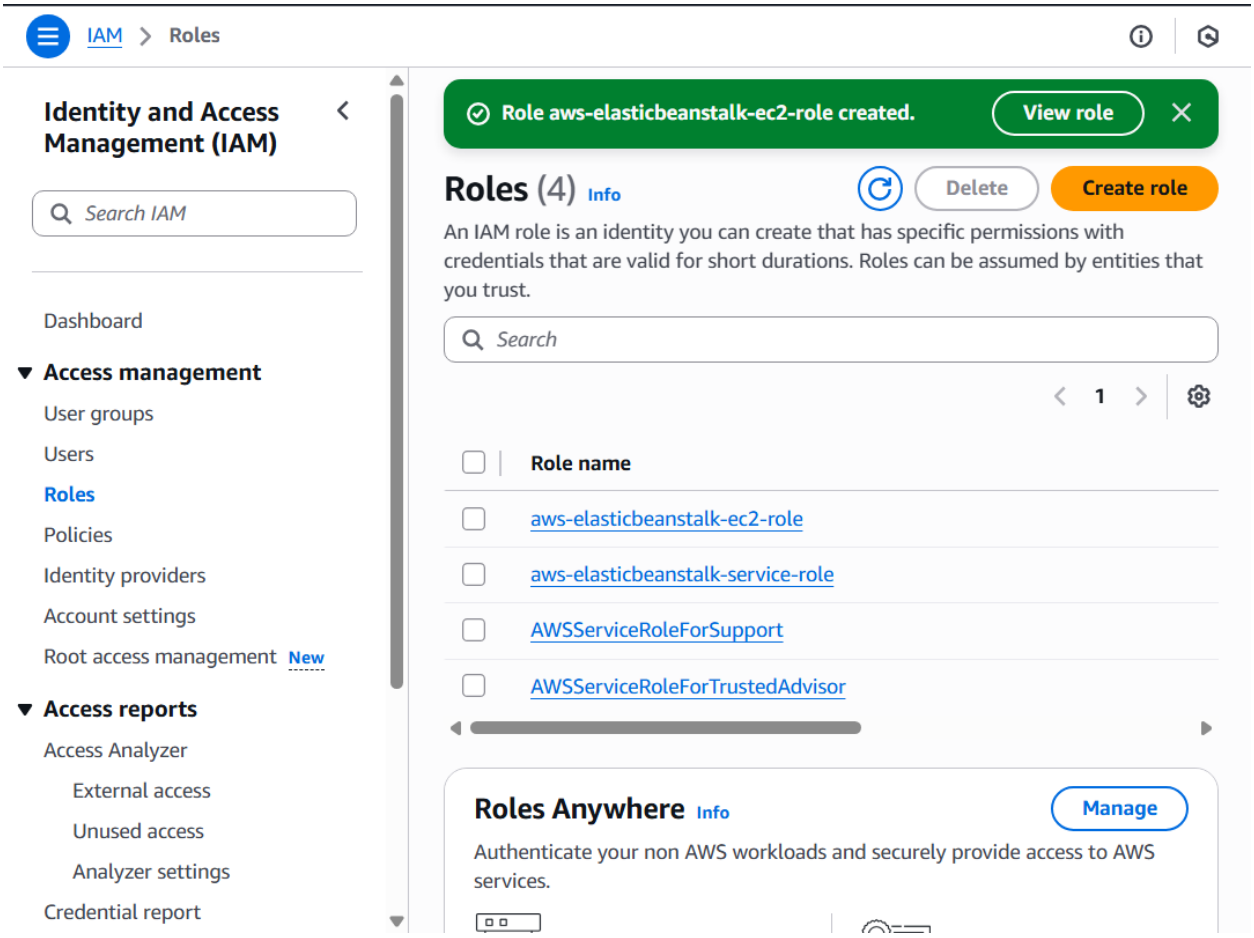
```
 => exporting to image                                                                                                        0.1s
 => => exporting layers                                                                                                       0.1s
 => => writing image sha256:5b8b37e626d3199a967df436d09eed3dab470f9f4fbd28ee9ad60d28412d219e                                  0.0s
 => => naming to docker.io/library/bloodhub-backend                                                                           0.0s
[ec2-user@ip-172-31-2-142 server]$ docker run -d -p 5000:5000 --env-file .env bloodhub-backend
d484e30b8c00b5ca137331bf82a9110f1bb89a2f898a1f08b08e7860a95bef0e
[ec2-user@ip-172-31-2-142 server]$ docker ps
CONTAINER ID   IMAGE              COMMAND                  CREATED          STATUS          PORTS
                NAMES
d484e30b8c00   bloodhub-backend   "docker-entrypoint.s…"   13 seconds ago   Up 12 seconds   0.0.0.0:5000->5000/tcp, :::5
000->5000/tcp   recursing_saha
[ec2-user@ip-172-31-2-142 server]$
```

5. Security group created for EC2 Instance.

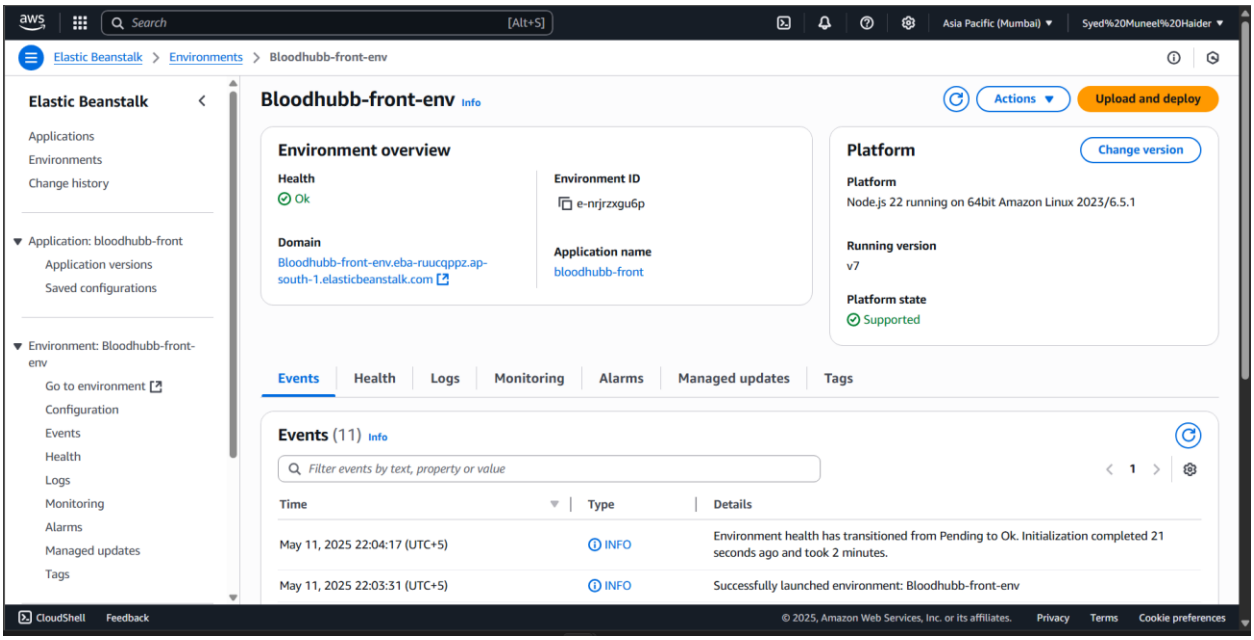


FRONTEND:

1. Created new IAM role for Elastic Beanstalk Instance:



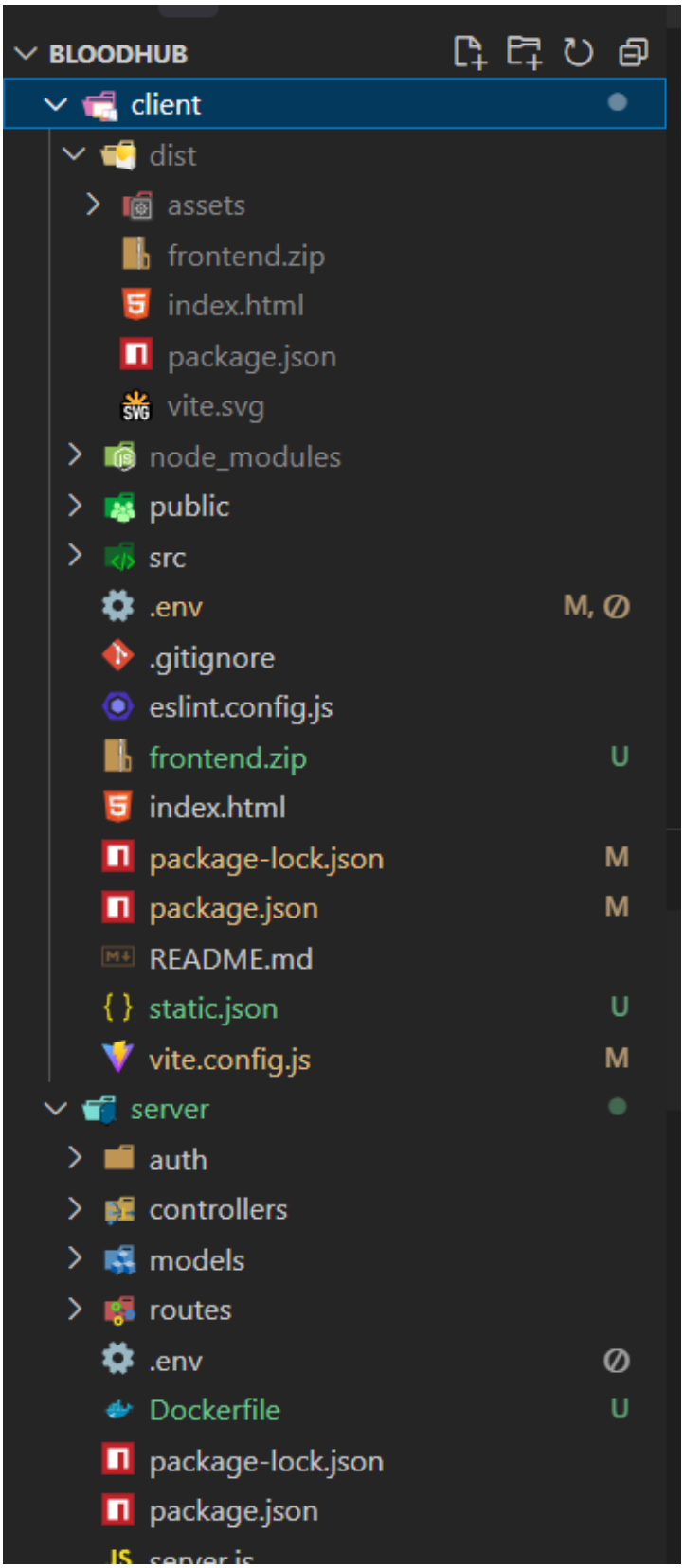
2. New environment and application created:



3. Proof that everything is working:

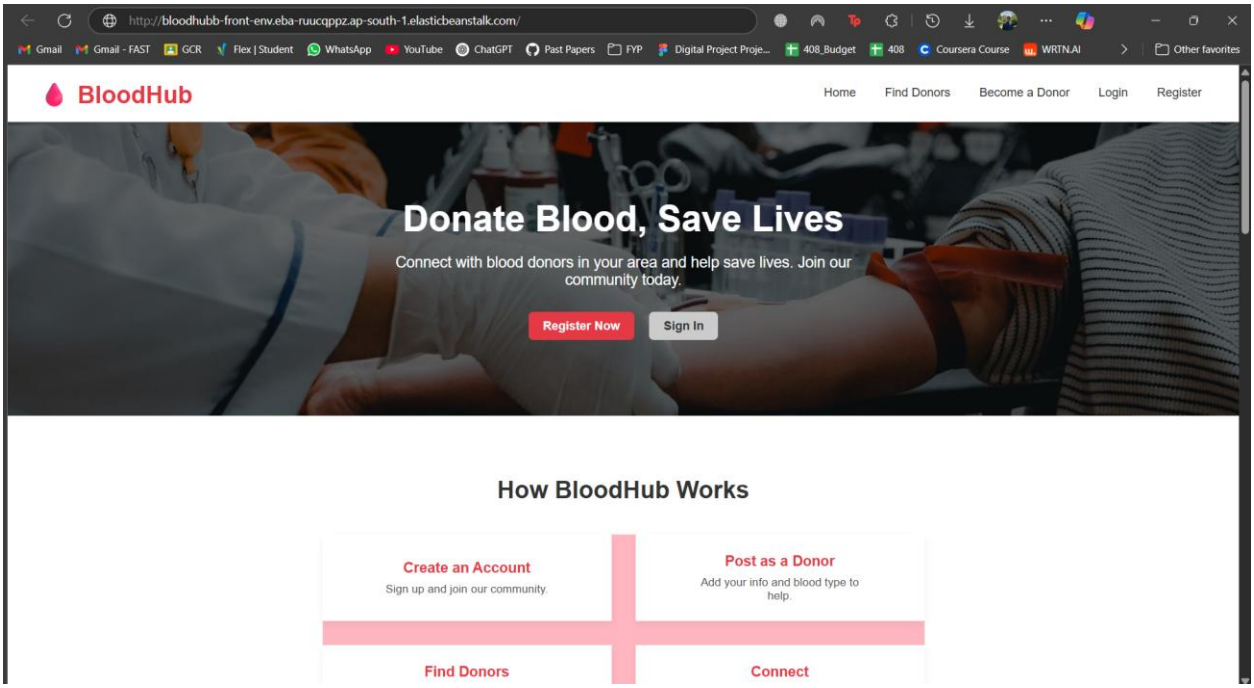
Time	Type	Details
May 11, 2025 22:04:17 (UTC+5)	INFO	Environment health has transitioned from Pending to Ok. Initialization completed 21 seconds ago and took 2 minutes.
May 11, 2025 22:03:31 (UTC+5)	INFO	Successfully launched environment: Bloodhubb-front-env
May 11, 2025 22:03:30 (UTC+5)	INFO	Application available at Bloodhubb-front-env.eba-ruucqppz.ap-south-1.elasticbeanstalk.com.
May 11, 2025 22:03:17 (UTC+5)	INFO	Added instance [i-011e48483558191d4] to your environment.
May 11, 2025 22:03:13 (UTC+5)	INFO	Instance deployment completed successfully.
May 11, 2025 22:02:17 (UTC+5)	INFO	Environment health has transitioned to Pending. Initialization in progress (running for 47 seconds). There are no instances.
May 11, 2025 22:02:03 (UTC+5)	INFO	Waiting for EC2 instances to launch. This may take a few minutes.
May 11, 2025 22:01:47 (UTC+5)	INFO	Created EIP: 52.66.107.94
May 11, 2025 22:01:32 (UTC+5)	INFO	Created security group named: awseb-e-nrjrzxgu6p-stack-AWSEBSecurityGroup-Z47p6mrqLFbp
May 11, 2025 22:01:12 (UTC+5)	INFO	Using elasticbeanstalk-ap-south-1-058264539896 as Amazon S3 storage bucket for environment data.
May 11, 2025 22:01:11 (UTC+5)	INFO	createEnvironment is starting.

4. Project Folder Structure:


BLOODHUB
client
dist
assets
frontend.zip
index.html
package.json
vite.svg
node_modules
public
src
.env M, Ø
.gitignore
eslint.config.js
frontend.zip U
index.html
package-lock.json M
package.json M
README.md
static.json U
vite.config.js M
server
auth
controllers
models
routes
.env Ø
Dockerfile U
package-lock.json
package.json

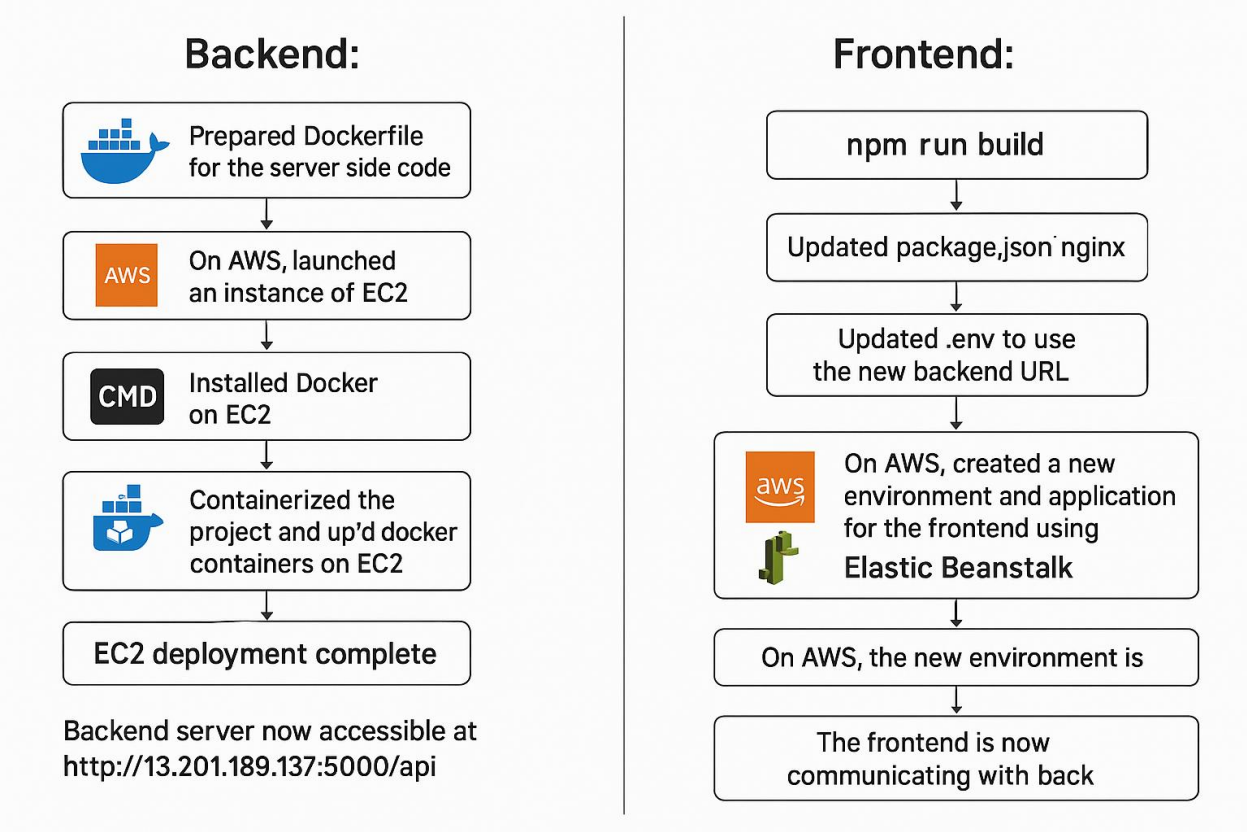
WEBSITE:

1. THE WEBSITE IS FULLY FUNCTIONAL:



2. ALL CRUD OPERATION ARE FUNCTIONAL.

3. DIAGRAM:



4. DIAGRAM: