**PART1**

**//install hadolint using curl**

**curl -Lo hadolint.exe https://github.com/hadolint/hadolint/releases/download/v2.12.0/hadolint-Windows-x86\_64.exe**

**//install trivy**

**docker pull aquasec/trivy:latest**

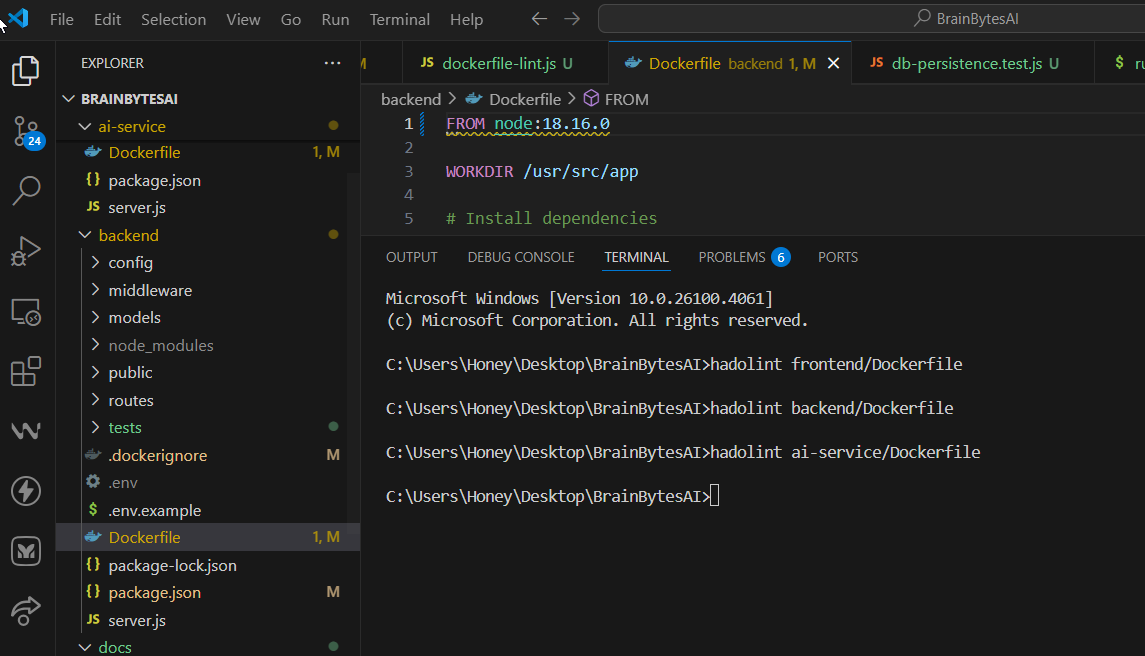
**//install jest**

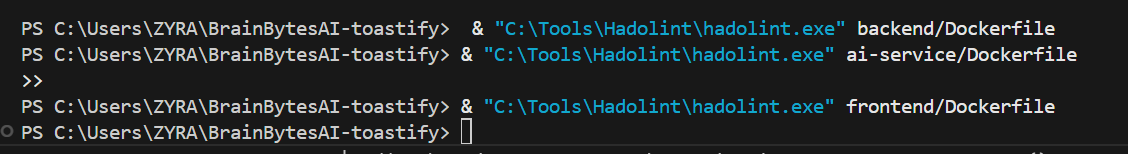
**cd backend && npm install --save-dev jest @types/jest supertest**

**cd frontend && npm install --save-dev jest @types/jest @testing-library/react @testing-library/jest-dom**

**cd ai-service && npm install --save-dev jest @types/jest supertest**

**PART2**

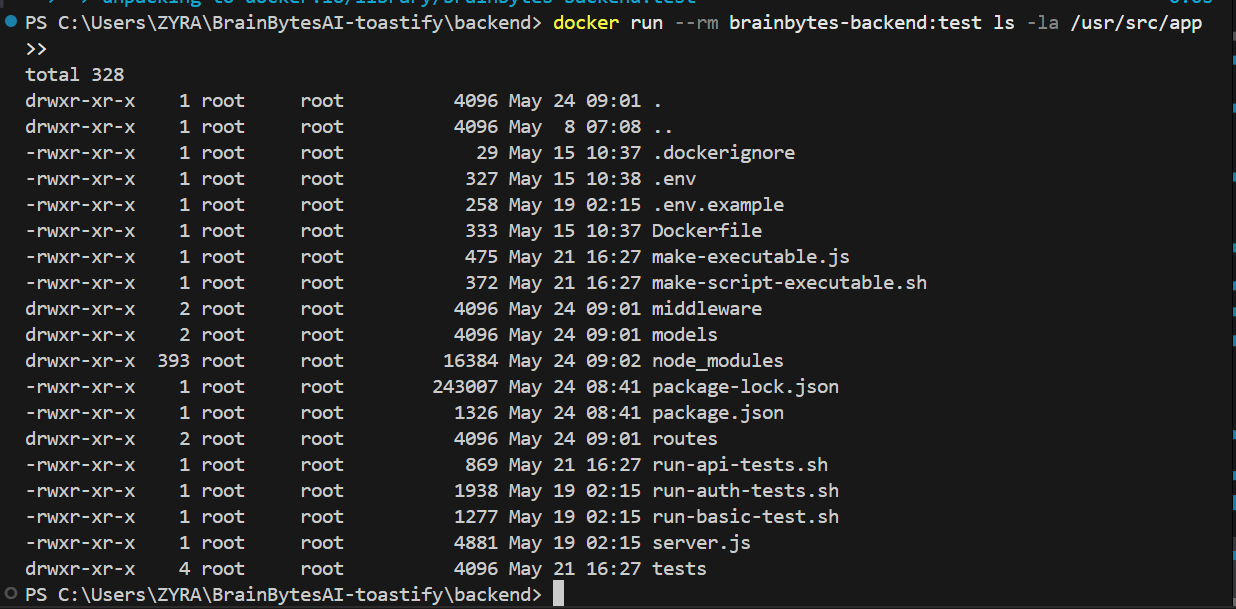
**Hadolint testing  
  
**

****

**cd backend**

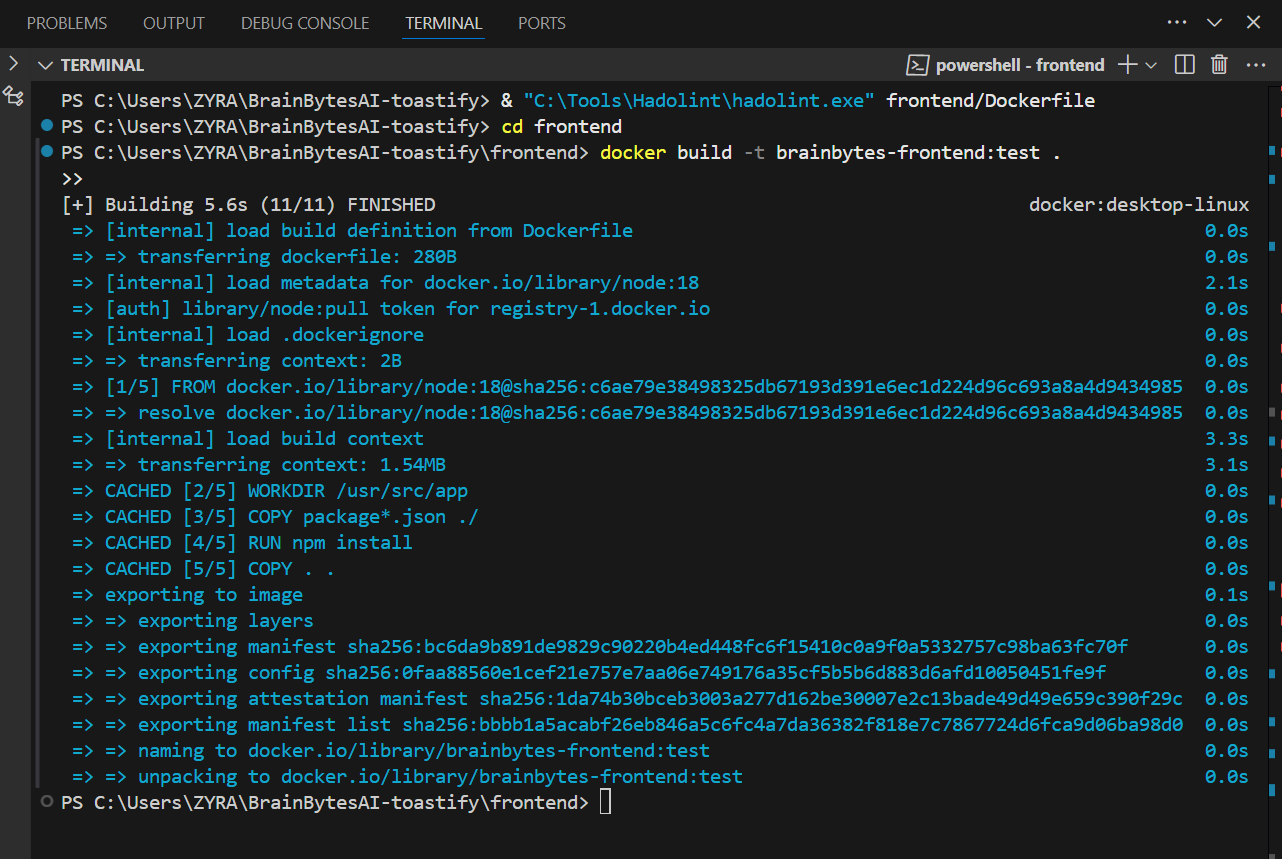
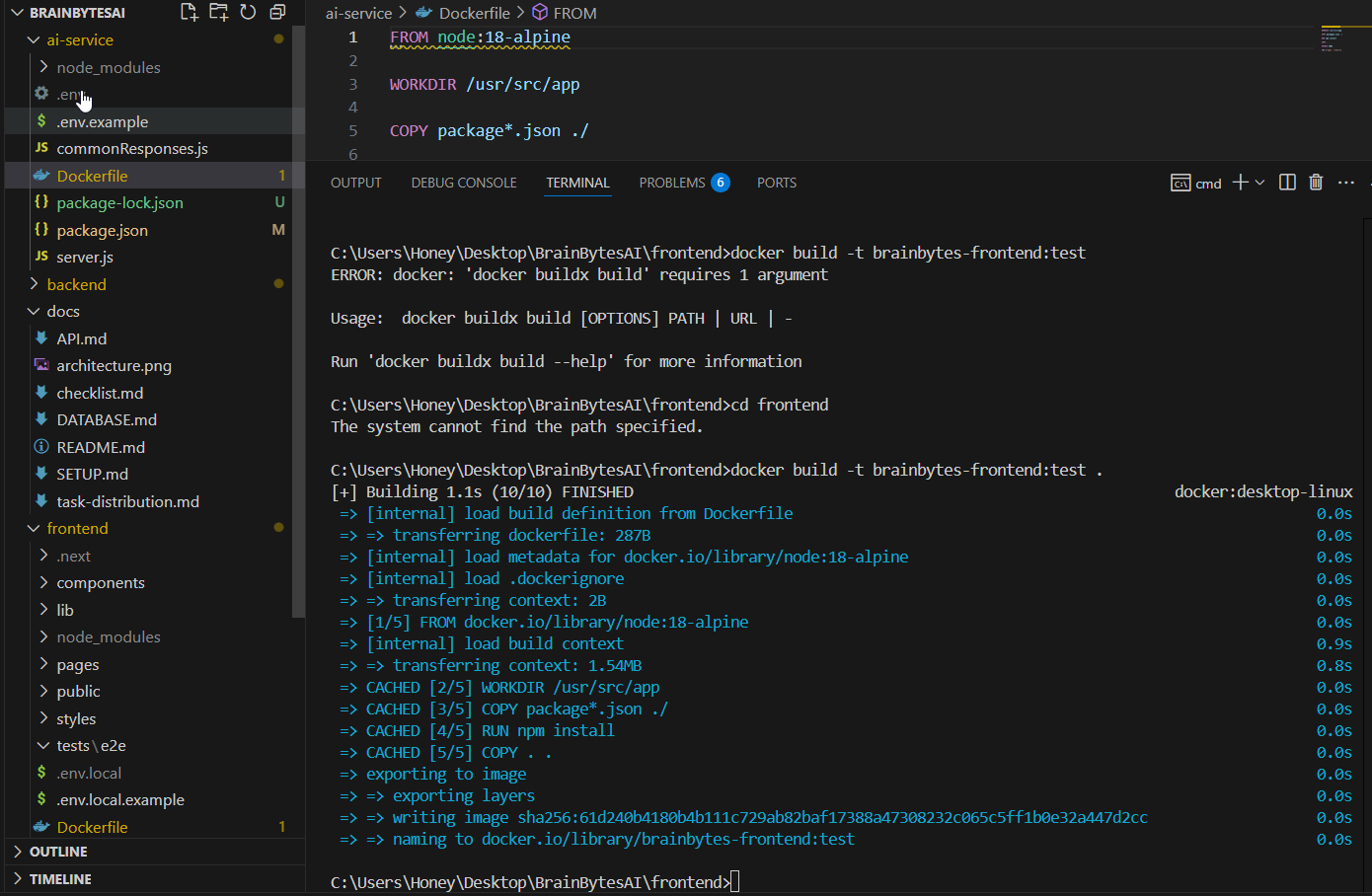
**docker build -t brainbytes-backend:test .  
  
docker run --rm brainbytes-backend:test ls -la /usr/src/app**

**docker run --rm brainbytes-backend:test node -v**

****

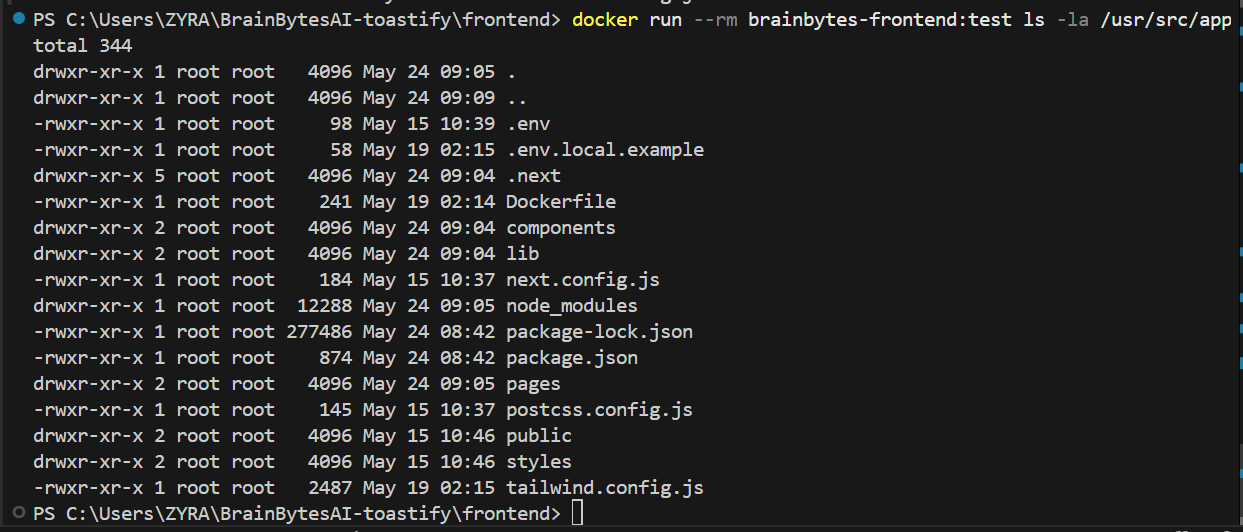
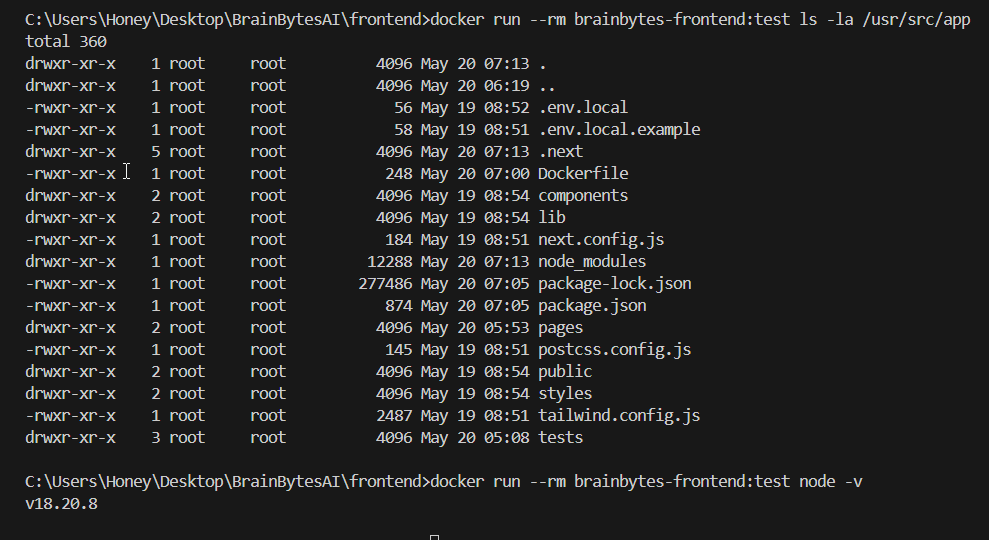
**cd frontend**

**docker build -t brainbytes-frontend:test .**

****

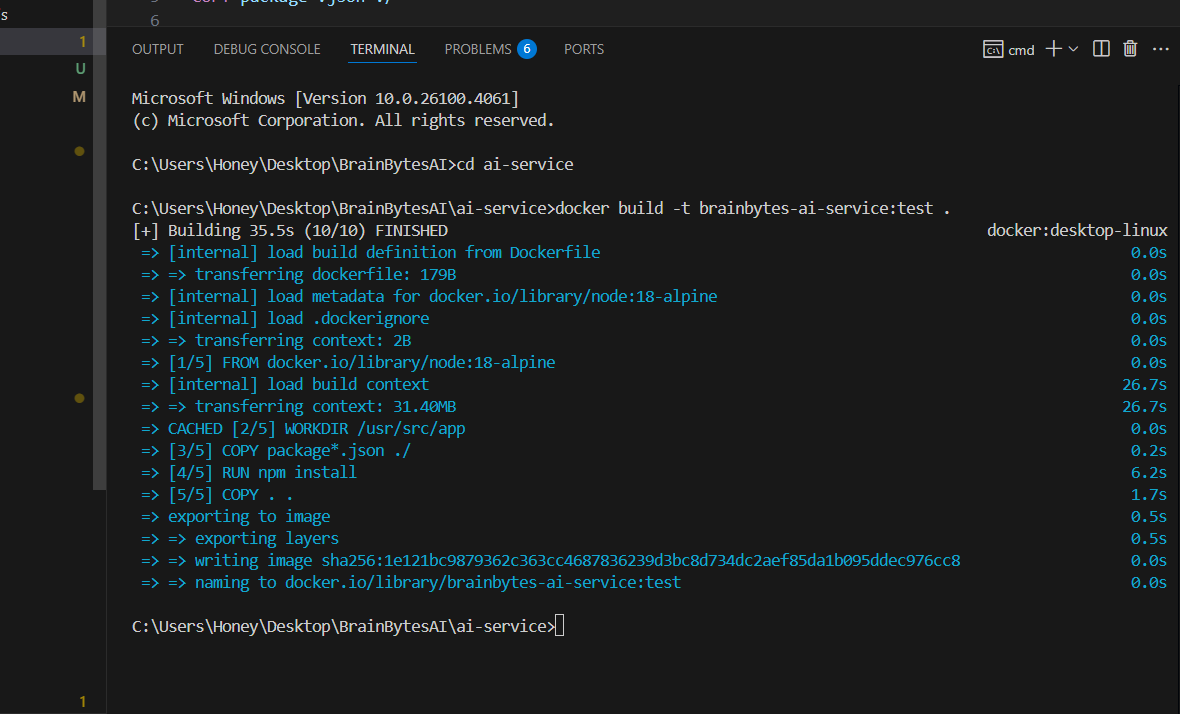
**docker run --rm brainbytes-frontend:test ls -la /usr/src/app**

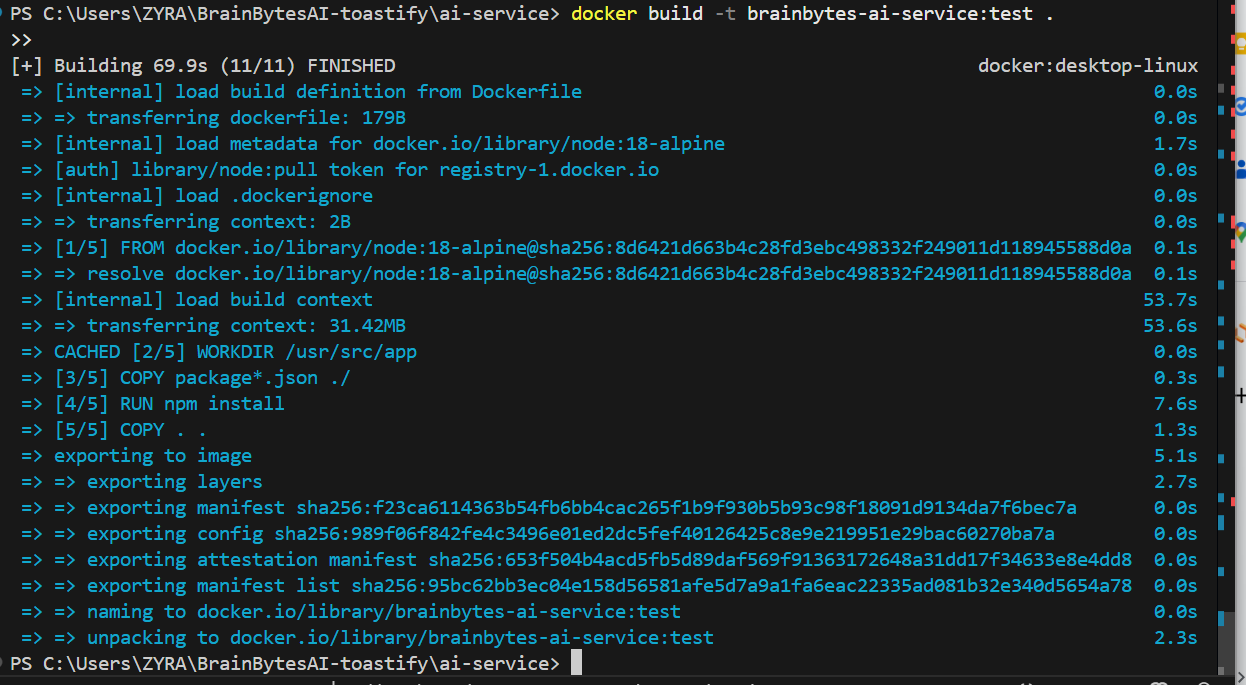
**docker run --rm brainbytes-frontend:test node -v**

****

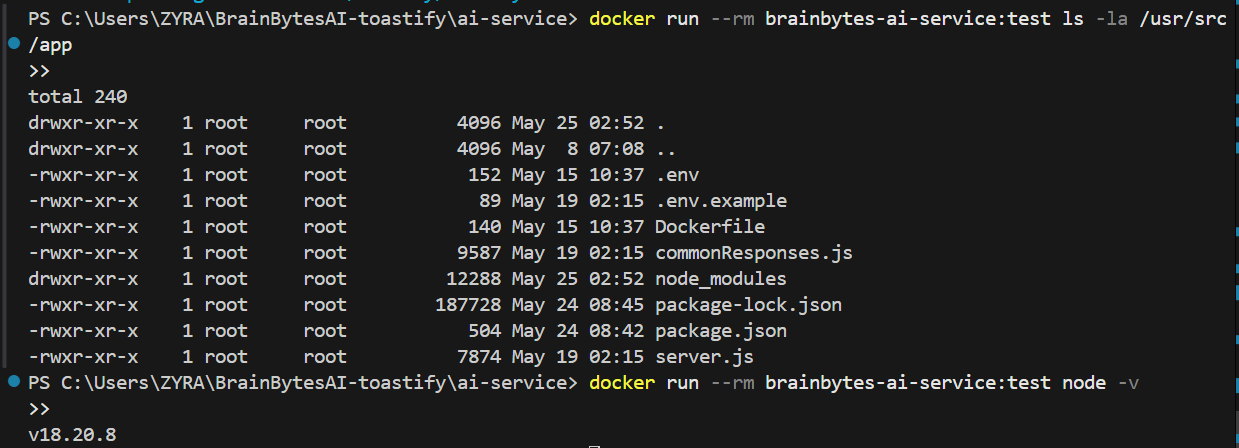
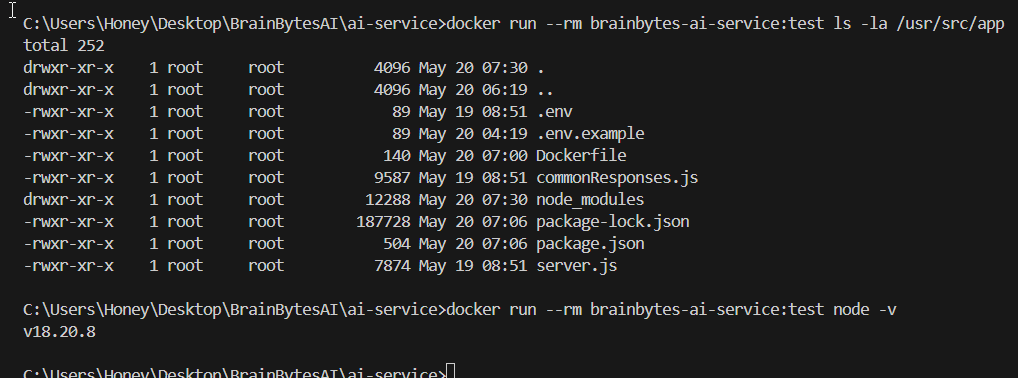
**cd ai-service**

**docker build -t brainbytes-ai-service:test .**

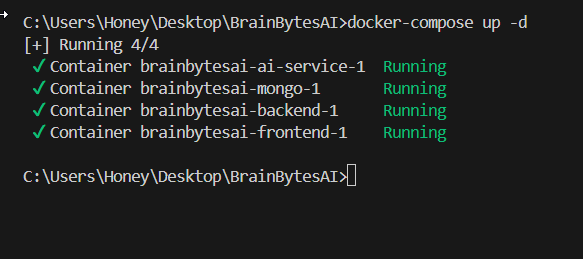
****

****

**docker run --rm brainbytes-ai-service:test ls -la /usr/src/app**

**docker run --rm brainbytes-ai-service:test node -v  
  
Part 3: Container Composition Tests**

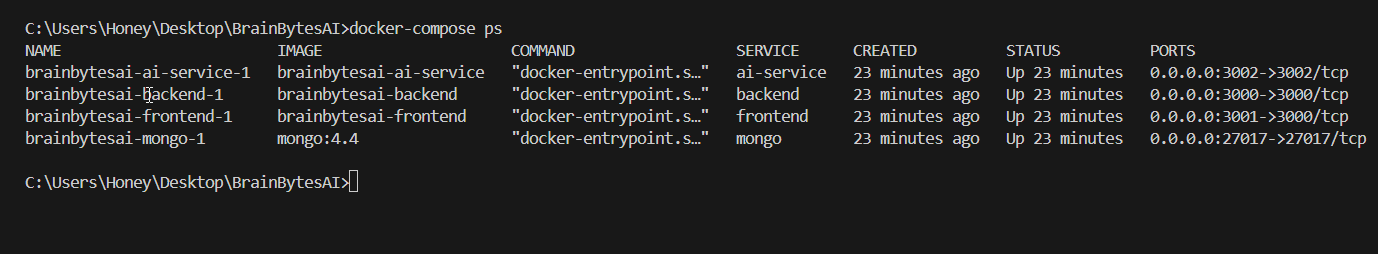
**Spin Up Your Containers -**

* **Run docker-compose up -d to launch all services.**
* ****

**Check Service Health**

* **Confirm that each container is up and not in an unhealthy state:**

**docker-compose ps**

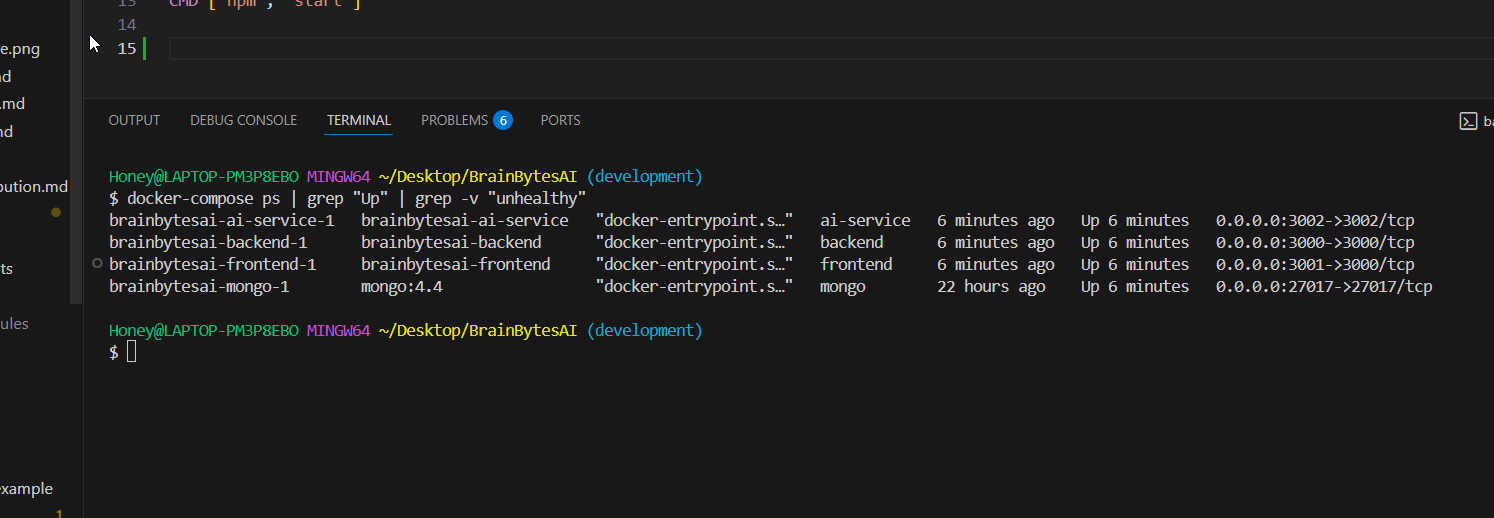
****

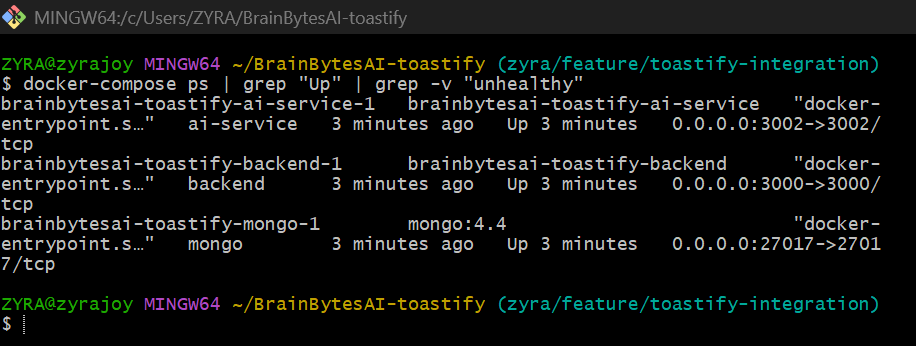
**docker-compose ps | grep "Up" | grep -v "unhealthy"**

Install [Git for Windows](https://git-scm.com/download/win) if you haven't already.

1. After installation, right-click inside your project folder and choose “Git Bash Here.”
2. Then run:

docker-compose ps | grep "Up" | grep -v "unhealthy"



****

* **Automate Composition Testing**
* **Create or adapt a test-composition.sh script (as shown in the example) that:**
* **Waits for containers to initialize**
* **Checks that frontend, backend, and database services are responding correctly**
* **Exits with a non-zero status if any service is offline or unhealthy**

**frontend/pages/api//**[**health.js**](http://health.js)

**export default function handler(req, res) {**

**if (req.method !== 'GET') {**

**return res.status(405).json({ error: 'Method not allowed' });**

**}**

**res.status(200).json({**

**status: 'healthy',**

**timestamp: new Date().toISOString(),**

**service: 'frontend'**

**});**

**}**

**backend/routes/**[**health.js**](http://health.js) **const express = require('express');**

**const router = express.Router();**

**router.get('/', (req, res) => {**

**res.status(200).json({**

**status: 'healthy',**

**timestamp: new Date().toISOString(),**

**service: 'backend'**

**});**

**});**

**module.exports = router;**

**ai-services/routes/**[**health.js**](http://health.js) **const express = require('express');**

**const router = express.Router();**

**router.get('/', (req, res) => {**

**res.status(200).json({**

**status: 'healthy',**

**timestamp: new Date().toISOString(),**

**service: 'backend'**

**});**

**});**

**module.exports = router;**

**backend/**[**server.js**](http://server.js) **//add**

**// Health check endpoint**

**app.get('/health', (req, res) => {**

**res.status(200).send('OK');**

**});**

**Create new folder   
  
tests/**[**test-composition.sh**](http://test-composition.sh)

**#!/bin/bash**

**# test-composition.sh**

**# Start all containers**

**docker-compose up -d**

**# Wait for services to be ready**

**echo "Waiting for containers to initialize..."**

**sleep 15**

**# Check Frontend**

**frontend\_status=$(curl -s -o /dev/null -w "%{http\_code}" http://localhost:3001)**

**# Check Backend**

**backend\_status=$(curl -s -o /dev/null -w "%{http\_code}" http://localhost:3000/health)**

**# Check MongoDB by evaluating a basic command**

**db\_status=$(docker-compose exec -T mongo mongo --quiet --eval "db.stats().ok")**

**# Verify if services are healthy**

**if [[ "$frontend\_status" == "200" && "$backend\_status" == "200" && "$db\_status" == "1" ]]; then**

**echo "✅ All services are running properly."**

**exit 0**

**else**

**echo "❌ Service check failed!"**

**echo "Frontend status: $frontend\_status"**

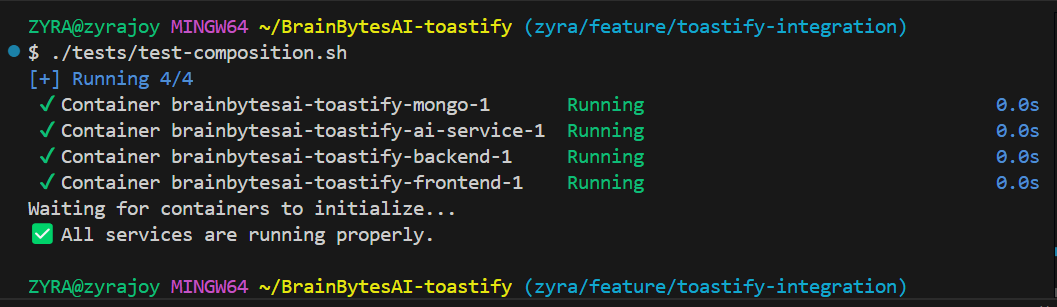
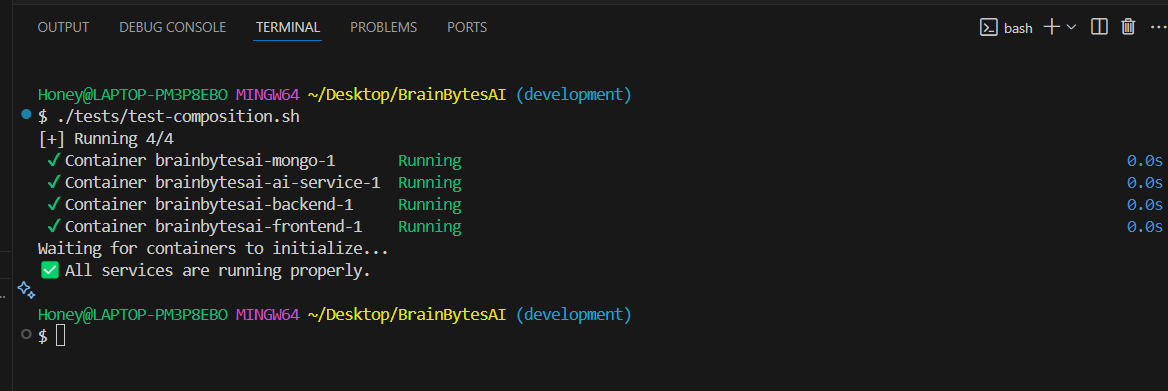
**echo "Backend status: $backend\_status"**

**echo "Database status: $db\_status"**

**exit 1**

**fi**

**Run use bash:  
./tests/**[**test-composition.sh**](http://test-composition.sh)

**Result:  
**

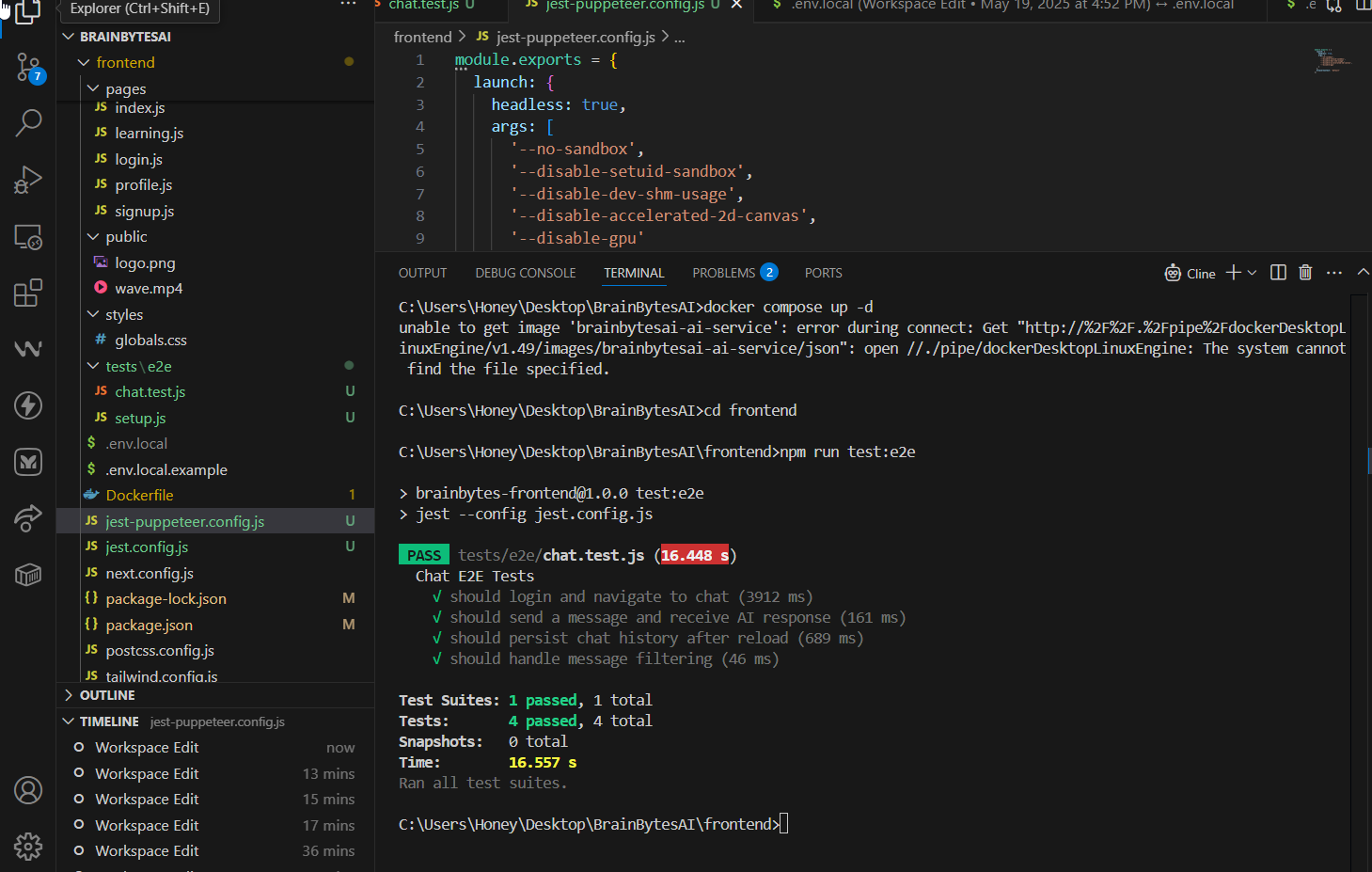
**Part4: API and Backend Test**

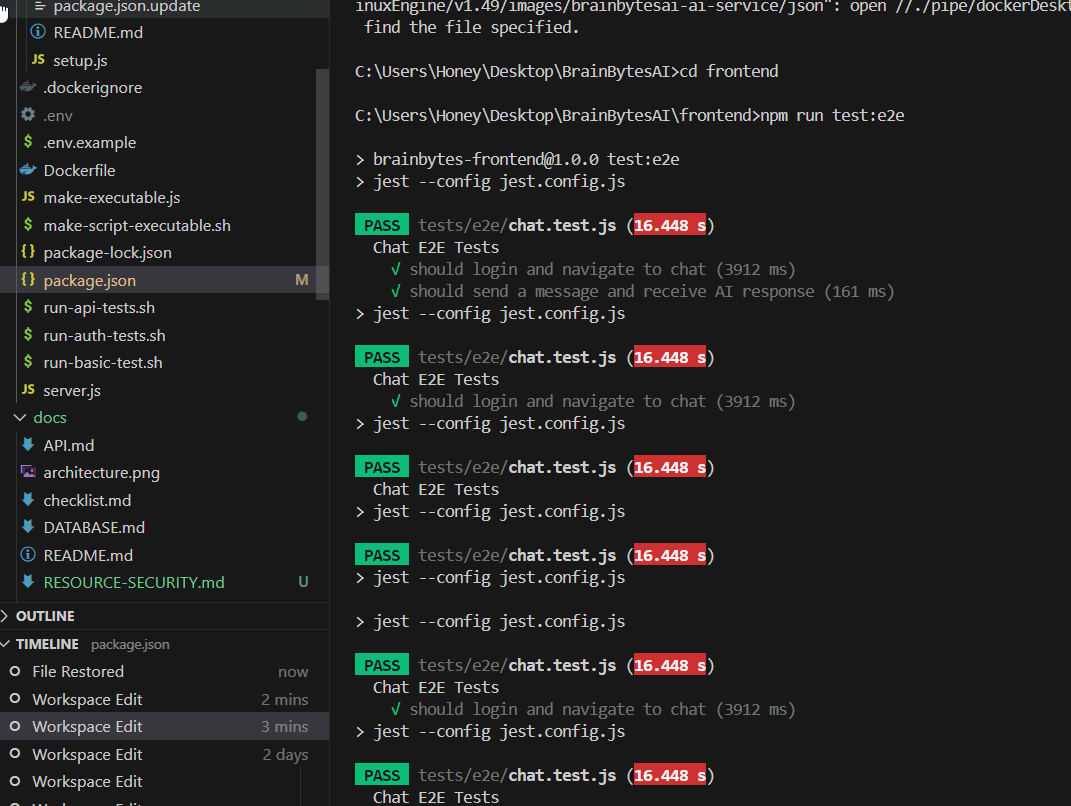
**Auth Unit Test:**

**cd backend**

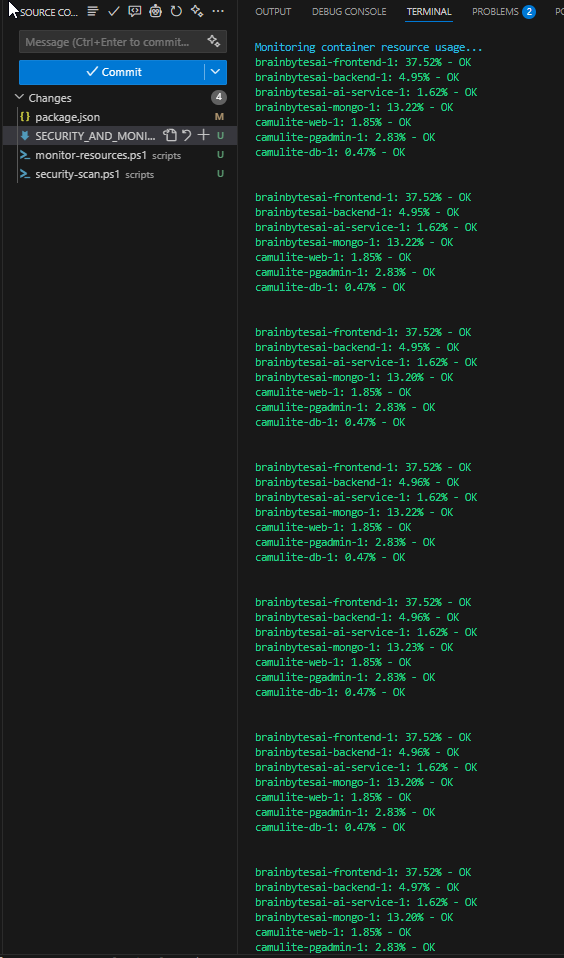
**./**[**run-auth-tests.sh**](http://run-auth-tests.sh)

**cd backend && npm test**

**Part5: EtoE Testing:**



**Part 6:  
Monitoring**

****