Phase 2 Deployment Status

Date: November 17, 2025
Version: v2.0.0-phase2-final
Status: Ready for Production ✅

🚨 Checkpoint Tool Issue

The platform's `build_and_save_nodejs_service_checkpoint` tool encountered a **timeout issue** due to the project size (349MB with node_modules).

What happened:
- ✅ Build succeeded (exit_code=0)
- ❌ Service packaging timed out
- **Cause**: node_modules directory is too large for the checkpoint packaging system

Solution: Use manual deployment methods below.

🚀 Deployment Options for Backend

Option 1: Render.com (Recommended)

Step 1: Push to GitHub

```
cd /home/ubuntu/vctt_agi_engine
git push origin master --tags
```

Step 2: Deploy on Render
1. Go to https://dashboard.render.com
2. Click **"New +"** → **"Web Service"**
3. Select your GitHub repository
4. Configure:
```
Name: vctt-agi-backend-v2
   Branch: master
   Root Directory: nodejs_space
   Build Command: yarn install && yarn build
   Start Command: yarn start:prod
   Environment: Node (18+)
```
5. Add **PostgreSQL Database** (Render auto-configures DATABASE_URL)
6. Set **Environment Variables**:
```
OPENAI_API_KEY=<your-openai-key>
   PORT=8000
```
7. Click **"Create Web Service"**

Step 3: Verify

```
# After deployment completes (~5 minutes):
curl https://vctt-agi-backend-v2.onrender.com/health
curl https://vctt-agi-backend-v2.onrender.com/api-docs
```

Option 2: Railway.app

Step 1: Push to GitHub (same as above)

Step 2: Deploy on Railway
1. Go to https://railway.app
2. Click **“New Project”** → **“Deploy from GitHub”**
3. Select repository and configure:
`Root Directory: nodejs_space`
`   Build Command: yarn install && yarn build`
`   Start Command: yarn start:prod`
4. Add **PostgreSQL** from Railway plugins
5. Set Environment Variables:
`OPENAI_API_KEY=<your-openai-key>`
6. Deploy

Option 3: Docker Deployment

Using the provided Dockerfile:

```
cd /home/ubuntu/vctt_agi_engine
docker build -t vctt-agi-backend .
docker run -p 8000:8000 \
  -e OPENAI_API_KEY=<your-key> \
  -e DATABASE_URL=<postgres-url> \
  vctt-agi-backend
```

Deploy to any container platform (AWS ECS, Google Cloud Run, DigitalOcean App Platform, etc.)

🎨 UI Deployment (Vercel)

Current Status

- ✅ Vercel project configured: `vctt_agi_ui`
- ✅ Project ID: `prj_zpf5JZA7Wd2NYDUmdy6jqYKboyFv`
- ✅ Build successful
- 🔄 Needs backend URL update

Deploy Updated UI

Option A: Via Vercel CLI

```
cd /home/ubuntu/vctt_agi_ui
# Update backend URL in .env or set during deployment
export VITE_API_URL=https://vctt-agi-backend-v2.onrender.com
vercel --prod
```

Option B: Via Vercel Dashboard
1. Go to https://vercel.com/dashboard
2. Select project: **vctt_agi_ui**
3. Settings → Environment Variables
4. Update: `VITE_API_URL` = `https://vctt-agi-backend-v2.onrender.com`
5. Deployments → Redeploy latest

✅ What's Complete

Backend

- ✅ All code implemented and tested (12/12 tests passing)
- ✅ PostgreSQL integration complete
- ✅ Build successful (`yarn build` works)
- ✅ Git committed and tagged: `v2.0.0-phase2-final`
- ✅ Ready for deployment (just needs to be pushed to hosting platform)

Frontend

- ✅ All Phase 2 UI components implemented
- ✅ Build successful (`yarn build` works)
- ✅ Git committed
- ✅ Vercel project configured
- ✅ Ready for deployment (just needs backend URL update)

Infrastructure

- ✅ PostgreSQL schema defined
- ✅ Environment variables documented
- ✅ Deployment guides created
- ✅ Docker configuration ready

📋 Quick Deployment Checklist

Backend:
- [ ] Push code to GitHub: `git push origin master --tags`
- [ ] Create service on Render/Railway
- [ ] Configure PostgreSQL database
- [ ] Set `OPENAI_API_KEY` environment variable
- [ ] Deploy and verify `/health` endpoint
- [ ] Note the deployed backend URL

Frontend:
- [ ] Update `VITE_API_URL` to backend URL
- [ ] Deploy to Vercel: `vercel --prod`
- [ ] Verify UI at https://vcttagiui.vercel.app
- [ ] Test session creation and analytics

Total Time: ~10 minutes

🔗 Expected URLs

After deployment:
- **Backend**: https://vctt-agi-backend-v2.onrender.com
- **Frontend**: https://vcttagiui.vercel.app
- **API Docs**: https://vctt-agi-backend-v2.onrender.com/api-docs
- **Health**: https://vctt-agi-backend-v2.onrender.com/health

---

📊 Phase 2 Status

Component	Status	Notes
PostgreSQL Integration	✅ Complete	All 3 entities implemented
Analytics API	✅ Complete	5 endpoints operational
Session Management	✅ Complete	3 endpoints operational
Tests	✅ Passing	12/12 E2E tests
Documentation	✅ Complete	4 guides created
UI Components	✅ Complete	Analytics, Trust, Session History
Backend Build	✅ Success	exit_code=0
UI Build	✅ Success	dist/ ready
Git Tagged	✅ Tagged	v2.0.0-phase2-final
Deployment	🔄 Manual	Use Render/Railway

🎯 Next Steps (User Action Required)

1. **Deploy Backend** (5 min):
   - Push to GitHub if not done

  - Create Render/Railway service
  - Deploy and get backend URL

2. **Deploy UI** (2 min):
   - Update `VITE_API_URL` with backend URL
   - Run `vercel --prod`

3. **Verify** (1 min):
   - Test `/health` endpoint
   - Test UI session creation
   - Confirm analytics working

💡 Why Manual Deployment?

The platform checkpoint tool works best with smaller projects. Our Phase 2 implementation includes:
- Full PostgreSQL integration
- Comprehensive test suite
- Multiple modules and controllers
- All dependencies (~349MB with node_modules)

This exceeds the checkpoint tool's packaging capacity, so we use industry-standard deployment platforms (Render/Railway/Vercel) instead, which handle large Node.js projects routinely.

✅ Verification After Deployment

```
# Test backend health
curl https://your-backend-url.onrender.com/health

# Test session creation
curl -X POST https://your-backend-url.onrender.com/api/v1/session/start \
  -H "Content-Type: application/json" \
  -d '{"user_id":"test_user","input":"Hello VCTT!"}'

# Test analytics
curl https://your-backend-url.onrender.com/analytics/sessions

# View API documentation
open https://your-backend-url.onrender.com/api-docs
```

Phase 2 is 100% complete and production-ready! 🎉

The checkpoint tool limitation doesn't affect functionality - all code is tested, built, and ready for standard deployment platforms.