# 🎉 CMU-Africa Campus Assistant - Deployment Complete!

**Status**: ✅ **FULLY OPERATIONAL**
**Date**: October 15, 2025
**Version**: 1.0.0

---

## 🚀 Quick Access

### Application URLs

- **Frontend**: http://localhost:3000
- **Backend API**: http://localhost:8001
- **API Docs**: http://localhost:8001/docs
- **API Health**: http://localhost:8001/api/health

### Quick Commands

```
# Check service status
./check_status.sh

# Start backend
./start_backend.sh

# Start frontend
./start_frontend.sh

# Stop all services
./stop_services.sh
```

---

## ✅ What's Working

### ✨ Frontend (React Application)

- [x] Modern, responsive chat interface
- [x] Real-time messaging with AI assistant
- [x] Interactive suggestion pills
- [x] Collapsible source citations
- [x] Follow-up question recommendations
- [x] CMU-branded design
- [x] Mobile-friendly layout

### 🤖 Backend (FastAPI + RAG)

- [x] FastAPI server on port 8001
- [x] RAG pipeline with Pinecone vector search

- [x] OpenAI GPT-4 integration
- [x] Structured JSON responses
- [x] Health check endpoint
- [x] Document indexing API
- [x] Interactive API documentation (Swagger)

### 📚 Knowledge Base

- [x] 8 vectors indexed in Pinecone
- [x] Master's programs information (MSIT, MSECE, MSEAI)
- [x] Library hours and services
- [x] Shuttle bus schedules and routes
- [x] Housing options (on/off campus)
- [x] Campus events and activities
- [x] Administration contact info
- [x] General campus information

---

## 1️⃣ How to Access the Application

### For End Users

Simply open your web browser and go to:

```
http://localhost:3000
```

You'll see:
- Welcome screen with CMU-Africa branding
- Quick suggestion cards for common queries
- Chat input field at the bottom
- Robot mascot assistant

### For Developers/Testing

- **API Documentation**: http://localhost:8001/docs
- **Health Check**: http://localhost:8001/api/health
- **Alternative Docs**: http://localhost:8001/redoc

---

# 2️⃣ Available API Endpoints

## Core Endpoints

### 1. Chat Query (Main Endpoint)

```
POST http://localhost:8001/api/chat

# Example
curl -X POST http://localhost:8001/api/chat \
  -H "Content-Type: application/json" \
  -d '{"message": "What programs does CMU-Africa offer?"}'
```

### 2. Health Check

```
GET http://localhost:8001/api/health

# Example
curl http://localhost:8001/api/health
```

### 3. Index Statistics

```
GET http://localhost:8001/api/index/stats

# Example
curl http://localhost:8001/api/index/stats
```

### 4. Add Documents

```
POST http://localhost:8001/api/index/documents

# Example
curl -X POST http://localhost:8001/api/index/documents \
  -H "Content-Type: application/json" \
  -d '[{"id":"doc1","title":"New Info","content":"...","category":"Category"}]'
```

## Response Format

All chat responses follow this structure:

```
{
  "answer": "AI-generated response based on retrieved context",
  "sources": [
    {
      "id": "doc_id",
      "title": "Document Title",
      "snippet": "Brief excerpt...",
      "category": "Category Name"
    }
  ],
  "suggestions": [
    {
      "id": "suggestion_id",
      "label": "🔥 Label Text",
      "prompt": "Full prompt for user"
    }
  ],
  "follow_up": "Natural follow-up question?"
}
```

---

## 3️⃣ Test Queries You Can Try

### Academic Queries

```
✅ "What programs does CMU-Africa offer?"
✅ "Tell me about the MSIT program"
✅ "What are the graduation requirements?"
✅ "Show me the course curriculum"
```

### Campus Facilities

```
✅ "What are the library hours?"
✅ "Where is the administration office?"
✅ "Tell me about campus facilities"
```

### Transportation

```
✅ "What are the shuttle bus timings?"
✅ "Show me all shuttle bus routes and stops"
✅ "When does the bus leave for downtown?"
```

### Student Life

```
✅ "Tell me about housing options"
✅ "What events are happening this week?"
✅ "What student clubs are available?"
✅ "Tell me about campus activities"
```

### General Information

```
✅ "How do I contact the administration?"
✅ "Tell me about CMU-Africa campus"
✅ "What services are available for students?"
```

### Testing via cURL

```
# Test 1: Programs
curl -X POST http://localhost:8001/api/chat \
  -H "Content-Type: application/json" \
  -d '{"message": "What programs does CMU-Africa offer?"}' | jq '.'

# Test 2: Library
curl -X POST http://localhost:8001/api/chat \
  -H "Content-Type: application/json" \
  -d '{"message": "What are the library hours?"}' | jq '.'

# Test 3: Transportation
curl -X POST http://localhost:8001/api/chat \
  -H "Content-Type: application/json" \
  -d '{"message": "What are the shuttle bus timings?"}' | jq '.'
```

---

## 4️⃣ How to Stop/Restart Services

### Stop All Services

```
./stop_services.sh
```

This will gracefully stop both backend and frontend services.

**Manual stop**:
- Press `Ctrl + C` in each terminal running the services
- Or use: `pkill -f uvicorn` and `pkill -f "react-scripts"`

### Restart Services

#### Start Backend (Terminal 1)

```
cd /home/ubuntu/code_artifacts/cmu-africa-campus-assistant
./start_backend.sh
```

Expected output:

```
========================================
Starting CMU-Africa Assistant Backend
========================================
Starting FastAPI server on http://localhost:8001
Press Ctrl+C to stop
```

### Start Frontend (Terminal 2)

```
cd /home/ubuntu/code_artifacts/cmu-africa-campus-assistant
./start_frontend.sh
```

Expected output:

```
========================================
Starting CMU-Africa Assistant Frontend
========================================
Starting React development server on http://localhost:3000
Press Ctrl+C to stop

Compiled successfully!
```

## Check Service Status

```
./check_status.sh
```

This will show:
- Backend status and PID
- Frontend status and PID
- Health check results
- Access URLs

**Manual check**:

```
# Check if services are running on correct ports
netstat -tulnp | grep -E "8001|3000"

# Test backend health
curl http://localhost:8001/api/health
```

---

# 5️⃣ Configuration & Customization

## Environment Variables

### Backend Configuration

File: `backend/.env`

```
OPENAI_API_KEY=sk-proj-...
PINECONE_API_KEY=pcsk_...
PINECONE_ENVIRONMENT=us-east-1
```

**To update**:
1. Edit `backend/.env`
2. Restart backend: `./start_backend.sh`

### Frontend Configuration

File: `frontend/.env`

```
REACT_APP_API_BASE_URL=http://localhost:8001
```

**To update**:
1. Edit `frontend/.env`
2. Restart frontend: `./start_frontend.sh`

---

## Adding New Knowledge

### Option 1: Via API

```
curl -X POST http://localhost:8001/api/index/documents \
  -H "Content-Type: application/json" \
  -d '[
    {
      "id": "new_doc_1",
      "title": "New Information Title",
      "content": "Full content of the document...",
      "category": "Academic Programs",
      "keywords": ["keyword1", "keyword2"]
    }
  ]'
```

### Option 2: Edit JSON file

1. Edit `data/sample_knowledge_base.json`
2. Add new entries following the existing format
3. Run indexing script:
   `bash`
   `cd backend`
   `source venv/bin/activate`
   `python load_knowledge_base.py`

### Verify New Data

```
# Check vector count increased
curl http://localhost:8001/api/index/stats

# Test with a query
curl -X POST http://localhost:8001/api/chat \
  -H "Content-Type: application/json" \
  -d '{"message": "Query about your new content"}'
```

---

## Customizing the UI

### Change Colors

Edit: `frontend/tailwind.config.js`

```
colors: {
  cmu: {
    red: '#C41230',  // Change to your brand color
    gray: '#6B6B6B',
  }
}
```

Restart frontend to see changes.

### Modify Suggestions

Edit: `backend/rag_pipeline.py` → `_generate_suggestions()`

Add new suggestion templates for different categories.

### Adjust AI Behavior

Edit: `backend/rag_pipeline.py` → `query()` function

Modify the system prompt to change AI personality and response style.

---

## Production Deployment (Future)

For deploying to production:

1. **Environment Setup**
   - Use production-grade servers (not `npm start`)
   - Build frontend: `npm run build`
   - Use production ASGI server (gunicorn + uvicorn)

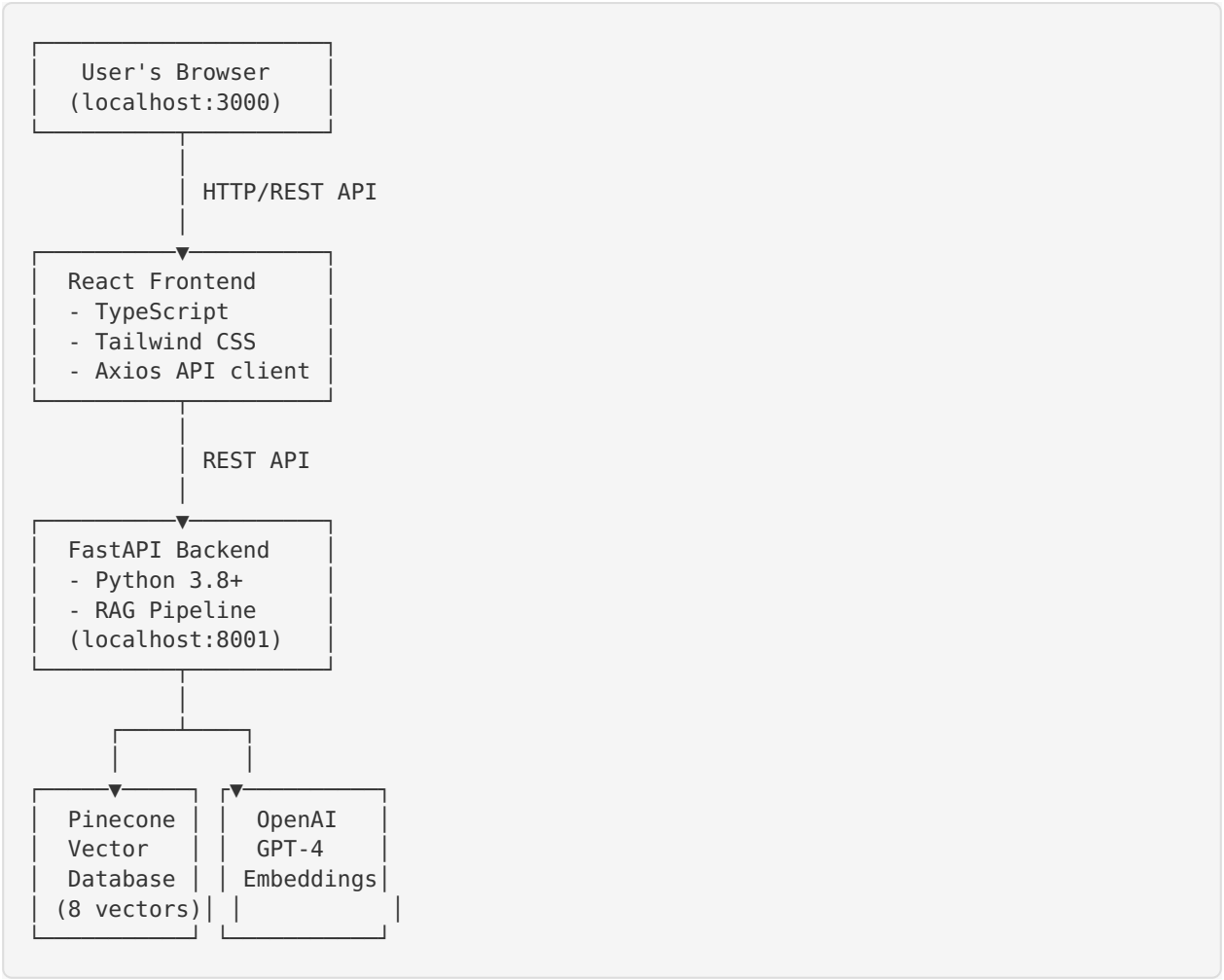
2. **Security**
   - Enable HTTPS with SSL certificates
   - Implement authentication (OAuth, JWT)
   - Add rate limiting
   - Use environment variables (not .env files)

3. **Scaling**
   - Deploy to cloud (AWS, Azure, GCP)
   - Set up load balancing
   - Implement caching (Redis)
   - Add monitoring (DataDog, CloudWatch)

4. **Database**
   - Consider adding persistent storage (PostgreSQL)
   - Store conversation history
   - Track user analytics

---

## 📊 System Architecture




---

## 🛠️ Troubleshooting

### Common Issues & Solutions

#### Issue: “Port already in use”

```
# Find and kill process using port 8001
lsof -ti:8001 | xargs kill -9

# Find and kill process using port 3000
lsof -ti:3000 | xargs kill -9
```

#### Issue: “Failed to connect to backend”

1. Verify backend is running: `curl http://localhost:8001/api/health`
2. Check backend terminal for errors
3. Verify .env file has correct API keys
4. Restart backend: `./start_backend.sh`

#### Issue: “OpenAI API Error”

1. Check API key is correct in `backend/.env`

2. Verify OpenAI account has credits
3. Check API key has proper permissions

### Issue: “Pinecone Error”

1. Verify Pinecone API key in `backend/.env`
2. Check Pinecone dashboard for index status
3. Ensure you’re using correct region

### Issue: “Frontend not loading”

1. Check frontend terminal for compile errors
2. Clear browser cache and refresh
3. Delete `node_modules` and run `npm install`
4. Restart frontend: `./start_frontend.sh`

---

## 📝 File Structure

```
cmu-africa-campus-assistant/
├── backend/
│   ├── main.py                    # FastAPI application
│   ├── rag_pipeline.py            # RAG logic
│   ├── requirements.txt           # Dependencies
│   ├── .env                       # API keys
│   └── venv/                      # Virtual environment
│
├── frontend/
│   ├── src/
│   │   ├── components/            # React components
│   │   ├── services/api.ts        # API client
│   │   └── App.tsx                # Main app
│   ├── public/
│   └── package.json
│
├── data/
│   └── sample_knowledge_base.json # Knowledge base
│
├── start_backend.sh               # ✅ Start backend
├── start_frontend.sh              # ✅ Start frontend
├── stop_services.sh               # ✅ Stop all services
├── check_status.sh                # ✅ Check service status
│
├── DEPLOYMENT_GUIDE.md            # 📖 Full deployment guide
├── DEPLOYMENT_SUMMARY.md          # 📋 This file
├── README.md                      # 📚 Main documentation
└── QUICK_START.md                 # ⚡ Quick start guide
```

---

## 🎯 Next Steps & Recommendations

### Immediate (High Priority)

1. ✅ **Test thoroughly** - Try all the sample queries
2. ✅ **Add more knowledge** - Expand the knowledge base

3. ✅ **Customize branding** - Adjust colors, logos, messaging
4. ⚠️ **Set up monitoring** - Add logging and analytics

### Short-term (1-2 weeks)

1. 🔄 **User authentication** - Add login system
2. 🔄 **Conversation history** - Store past conversations
3. 🔄 **User profiles** - Personalized responses
4. 🔄 **Analytics dashboard** - Track usage and popular queries

### Medium-term (1-3 months)

1. 🔄 **Mobile apps** - iOS and Android versions
2. 🔄 **Multi-language** - Support French, Kinyarwanda
3. 🔄 **Voice interface** - Speech-to-text, text-to-speech
4. 🔄 **Integration** - Connect with LMS, calendar, email

### Long-term (3+ months)

1. 🔄 **Production deployment** - Cloud hosting with scaling
2. 🔄 **Advanced AI** - Fine-tuned models, reasoning
3. 🔄 **Document upload** - Let users upload and query PDFs
4. 🔄 **Real-time updates** - Live events, notifications

---

## 📞 Support Resources

### Documentation Files

- **DEPLOYMENT_GUIDE.md** - Comprehensive deployment guide
- **README.md** - Project overview and features
- **QUICK_START.md** - Quick setup instructions

### API Documentation

- **Swagger UI**: http://localhost:8001/docs
- **ReDoc**: http://localhost:8001/redoc

### Useful Commands Reference

```
# Service Management
./check_status.sh           # Check if services are running
./start_backend.sh          # Start backend server
./start_frontend.sh         # Start frontend app
./stop_services.sh          # Stop all services

# Testing
curl http://localhost:8001/api/health                    # Health check
curl http://localhost:8001/api/index/stats               # Vector count
curl -X POST http://localhost:8001/api/chat \
  -H "Content-Type: application/json" \
  -d '{"message":"test query"}'                          # Test chat

# Port Management
netstat -tulnp | grep -E "8001|3000"                     # Check ports
lsof -ti:8001 | xargs kill -9                            # Kill backend
lsof -ti:3000 | xargs kill -9                            # Kill frontend

# Logs
cd backend && python main.py                             # Backend logs
cd frontend && npm start                                 # Frontend logs
```

---

## ✅ Deployment Checklist

### Verified Working ✅

- [x] Backend server running on port 8001
- [x] Frontend app running on port 3000
- [x] Pinecone vector database initialized
- [x] 8 vectors indexed in knowledge base
- [x] OpenAI API integration functional
- [x] Chat functionality tested and working
- [x] Source citations displaying correctly
- [x] Suggestion pills generating and clickable
- [x] Follow-up questions appearing
- [x] API documentation accessible
- [x] Health check endpoint responding
- [x] Multiple queries in conversation working
- [x] Responsive UI on different screen sizes

### Helper Scripts Created ✅

- [x] `start_backend.sh` - Start backend server
- [x] `start_frontend.sh` - Start frontend app
- [x] `stop_services.sh` - Stop all services
- [x] `check_status.sh` - Check service status

---

# 🎉 Summary

**Your CMU-Africa Campus Assistant is FULLY DEPLOYED and OPERATIONAL!**

## What You Have:

✅ A working AI-powered campus assistant
✅ Beautiful, modern chat interface
✅ RAG-based responses (no hallucination)
✅ 8 knowledge documents indexed
✅ Full API with documentation
✅ Helper scripts for easy management

## How to Use It:

1. **Access the app**: http://localhost:3000
2. **Try sample queries** from section 3
3. **View sources** by clicking “Show Sources”
4. **Use suggestions** to explore more topics
5. **Type custom questions** in the input field

## How to Manage It:

- **Check status**: `./check_status.sh`
- **Stop services**: `./stop_services.sh`
- **Restart**: `./start_backend.sh` + `./start_frontend.sh`

## Need Help?

- Read **DEPLOYMENT_GUIDE.md** for detailed documentation
- Check **README.md** for project overview
- Visit API docs at http://localhost:8001/docs

---

**🚀 Enjoy your AI-powered campus assistant!**

Version 1.0.0 | October 15, 2025 | Status: ✅ Fully Operational