Backend Server (Node.js + Express)

🎯 Purpose

This Express server acts as a proxy between your frontend and the model server, providing:

- API rate limiting
- CORS handling
- Conversation history management
- Security layer

📋 Setup Instructions

1. Install Dependencies

bash

```bash
cd backend
npm install
```

2. Configure Environment Variables

bash

```bash
# Copy example env file
cp .env.example .env

# Edit .env with your values
nano .env
```

Update these values:

env

```env
MODEL_SERVER_URL=https://xxxx.ngrok.io  # Your model server URL
MODEL_API_KEY=your-super-secret-key-change-this  # Same key as model server
FRONTEND_URL=http://localhost:5173  # Your frontend URL
```

3. Run Development Server

bash

```bash
npm run dev
```

4. Test the API

bash

```bash
# Health check
curl http://localhost:3000/api/health

# Test chat
curl -X POST http://localhost:3000/api/chat \
  -H "Content-Type: application/json" \
  -d '{
    "message": "Hello!",
    "sessionId": "test-session"
  }'
```

📡 API Endpoints

Health Check

```
GET /api/health
```

Returns backend and model server status.

Chat (with history)

```
POST /api/chat
```

Request body:

json

```json
{
  "message": "Your message here",
  "sessionId": "optional-session-id",
  "options": {
    "maxLength": 512,
    "temperature": 0.7,
    "topP": 0.9
  }
}
```

Response:

json

```
{
  "response": "AI response here",
  "sessionId": "session-id",
  "modelName": "model-name",
  "timestamp": "2024-01-01T00:00:00.000Z"
}
```

Generate (without history)

```
POST /api/generate
```

Request body:

```json
{
  "prompt": "Your prompt here",
  "options": {
    "maxLength": 512,
    "temperature": 0.7,
    "topP": 0.9
  }
}
```

Get Conversation History

```
GET /api/chat/:sessionId
```

Clear Conversation

```
DELETE /api/chat/:sessionId
```

🚀 Deployment

Deploy to Railway

1. Install Railway CLI:

```bash
npm install -g @railway/cli
```

2. Login and initialize:

```bash
railway login
railway init
```

3. Add environment variables in Railway dashboard
4. Deploy:

```bash
railway up
```

Deploy to Render

1. Push code to GitHub
2. Create new Web Service on Render
3. Connect your repo
4. Add environment variables
5. Deploy!

Deploy to Heroku

1. Install Heroku CLI
2. Login:

```bash
heroku login
```

3. Create app:

```bash
heroku create your-app-name
```

4. Set environment variables:

```bash
heroku config:set MODEL_SERVER_URL=https://xxxx.ngrok.io
heroku config:set MODEL_API_KEY=your-key
```

5. Deploy:

```bash
git push heroku main
```

🔒 Security Checklist

- [ ] Changed MODEL_API_KEY from default
- [ ] Set FRONTEND_URL to your actual domain (not *)
- [ ] Enabled rate limiting (already configured)
- [ ] Using HTTPS in production
- [ ] Environment variables not committed to git
- [ ] Added helmet for security headers
- [ ] Monitoring enabled

📊 Monitoring

View Logs

```bash
# Local
npm run dev

# Railway
railway logs

# Render
# Check dashboard

# Heroku
heroku logs --tail
```

🔧 Troubleshooting

"Failed to connect to model server"

- Check MODEL_SERVER_URL is correct
- Verify model server is running
- Check API key matches
- Test model server directly with curl

CORS errors

- Update FRONTEND_URL in .env

- Check corsOptions in server.js
- Verify frontend is using correct backend URL

Rate limit hit

- Adjust rate limits in server.js
- For development, comment out rate limiters

🎨 Customization

Change rate limits

Edit in `server.js`:

```javascript
const chatLimiter = rateLimit({
  windowMs: 1 * 60 * 1000, // 1 minute
  max: 10, // 10 requests per minute
});
```

Add authentication

```javascript
// Add JWT or session-based auth
import jwt from 'jsonwebtoken';

const authMiddleware = (req, res, next) => {
  // Your auth logic
};

app.use('/api/chat', authMiddleware);
```

Use database for history

Replace `conversationStore` with MongoDB/PostgreSQL:

```javascript
import { MongoClient } from 'mongodb';
// Store conversations in DB instead of memory
```

📝 Next Steps

After backend is deployed:

1. Note down your backend URL
2. Move to frontend setup
3. Configure frontend to use your backend URL