Google Gemini AI Integration Guide for TimeTravelers Backend

Overview

This document provides a comprehensive guide for integrating Google Gemini AI into the TimeTravelers backend application. The integration replaces the previously planned Manus AI API (which doesn't exist) with Google's powerful Gemini AI service, providing real AI-powered mentoring, career advice, and learning plan generation capabilities.

Table of Contents

Prerequisites

Before integrating Google Gemini AI, ensure you have:

- Node.js 16+ installed
- MongoDB database running
- Google Cloud Platform account
- Google AI Studio access (for API key generation)
- Basic understanding of Express.js and MongoDB

Installation and Setup

1. Install Dependencies

The integration requires the Google Generative AI package:

```
npm install @google/generative-ai
```

This dependency has been added to the `package.json` file:

```
{
  "dependencies": {
    "@google/generative-ai": "^0.2.1",
    // ... other dependencies
  }
}
```

2. Get Google Gemini API Key

1. Visit Google AI Studio
2. Sign in with your Google account
3. Click "Create API Key"
4. Copy the generated API key (starts with `AIza...`)
5. Store it securely - you'll need it for configuration

3. Environment Configuration

Create a `.env` file in your backend root directory:

```
# Copy from .env.example
cp .env.example .env
```

Update the `.env` file with your Google Gemini API key:

```
# Google Gemini AI Configuration
GOOGLE_GEMINI_API_KEY=AIzaSyC-your-actual-api-key-here
GOOGLE_GEMINI_MODEL=gemini-pro
GOOGLE_GEMINI_MAX_TOKENS=2048
GOOGLE_GEMINI_TEMPERATURE=0.7
```

Configuration

The configuration is managed in `src/config/config.js`:

```
module.exports = {
  // ... other config
  gemini: {
    apiKey: process.env.GOOGLE_GEMINI_API_KEY,
    model: process.env.GOOGLE_GEMINI_MODEL || 'gemini-pro',
    maxTokens: parseInt(process.env.GOOGLE_GEMINI_MAX_TOKENS)
|| 2048,
    temperature:
parseFloat(process.env.GOOGLE_GEMINI_TEMPERATURE) || 0.7,
    safetySettings: [
      {
        category: 'HARM_CATEGORY_HARASSMENT',
        threshold: 'BLOCK_MEDIUM_AND_ABOVE'
      },
      {
        category: 'HARM_CATEGORY_HATE_SPEECH',
        threshold: 'BLOCK_MEDIUM_AND_ABOVE'
      }
    ]
  }
};
```

Configuration Parameters

Parameter	Description	Default	Required
`apiKey`	Google Gemini API key	-	Yes
`model`	Gemini model to use	`gemini-pro`	No
`maxTokens`	Maximum tokens per request	`2048`	No
`temperature`	Response creativity (0-1)	`0.7`	No
`safetySettings`	Content safety filters	See config	No

API Endpoints

The integration provides the following REST API endpoints:

Authentication

All endpoints require authentication. Include the JWT token in the Authorization header:

```
Authorization: Bearer <your-jwt-token>
```

Admin Endpoints

Initialize Service

```
POST /api/v1/ai/initialize
```

Initializes the Google Gemini AI service.

Response:

```
{
  "success": true,
  "message": "Google Gemini AI service initialized successfully"
}
```

Get Usage Statistics

```
GET /api/v1/ai/usage
```

Returns current usage statistics for the Gemini AI service.

Response:

```
{
  "success": true,
  "stats": {
    "isInitialized": true,
    "model": "gemini-pro",
    "requestsThisMinute": 5,
    "requestsToday": 127,
    "lastRequestAt": "2024-01-15T10:30:00Z"
  }
}
```

Health Check

```
GET /api/v1/ai/health
```

Checks the health status of the Gemini AI service.

Response:

```
{
  "success": true,
  "status": "healthy",
  "service": "Google Gemini AI",
  "isInitialized": true,
  "model": "gemini-pro",
  "timestamp": "2024-01-15T10:30:00Z"
}
```

Sync Mentors

```
POST /api/v1/ai/sync-mentors
```

Loads static mentor profiles into the database.

Response:

```
{
  "success": true,
  "message": "AI mentors synced successfully",
  "results": {
    "total": 10,
    "created": 8,
    "updated": 2,
    "failed": 0
  }
}
```

User Endpoints

Get All Mentors

```
GET /api/v1/ai/mentors?
search=javascript&specialization=web&limit=10&page=1
```

Retrieves a list of AI mentors with optional filtering.

Query Parameters: - `search` (optional): Search term for name, specialization, bio, or skills - `specialization` (optional): Filter by specialization - `experienceLevel` (optional): Filter by experience level - `teachingStyle` (optional): Filter by teaching style - `limit` (optional): Number of results per page (default: 20) - `page` (optional): Page number (default: 1)

Response:


```
{
  "success": true,
  "mentors": [
    {
      "_id": "mentor_id_here",
      "name": "Alex Chen",
      "specialization": "Full-Stack Development",
      "bio": "Experienced full-stack developer...",
      "profileImage": "/assets/mentors/alex-chen.jpg",
      "experienceLevel": "expert",
      "skills": ["JavaScript", "React", "Node.js"],
      "teachingStyle": "practical",
      "communicationStyle": "supportive",
      "rating": 4.8,
      "reviewCount": 156,
      "geminiMentorId": "mentor_001"
    }
  ],
  "pagination": {
    "total": 25,
    "page": 1,
    "limit": 10,
    "pages": 3
  }
}
```


Get Specific Mentor

```
GET /api/v1/ai/mentors/:id
```

Retrieves details for a specific AI mentor.

Response:


```
{
  "success": true,
```

```
  "mentor": {
    "_id": "mentor_id_here",
    "name": "Alex Chen",
    "specialization": "Full-Stack Development",
    // ... full mentor details
  }
}
```

Generate Career Advice

```
POST /api/v1/ai/generate/career-advice
```

Generates personalized career advice using Gemini AI.

Request Body:

```
{
  "userProfile": {
    "currentRole": "Junior Developer",
    "experienceLevel": "beginner",
    "skills": ["JavaScript", "HTML", "CSS"],
    "careerGoals": "Become a full-stack developer",
    "industry": "Technology",
    "education": "Computer Science Degree"
  }
}
```

Response:

```
{
  "success": true,
  "advice": {
    "content": "Based on your profile as a Junior Developer...",
    "recommendations": [
      "Focus on learning React or Vue.js for frontend",
      "Learn Node.js and Express for backend development",
      "Practice with databases like MongoDB or PostgreSQL"
    ],
    "timeline": "6-12 months",
    "nextSteps": [
      "Build a portfolio project using full-stack technologies",
      "Contribute to open-source projects",
      "Consider taking advanced courses in cloud technologies"
    ],
    "generatedAt": "2024-01-15T10:30:00Z"
```

```
  }
}
```

Generate Learning Plan

```
POST /api/v1/ai/generate/learning-plan
```

Creates a personalized learning plan using Gemini AI.

Request Body:

```
{
  "userId": "user_123",
  "careerGoals": ["Full-stack development", "Cloud computing"],
  "currentSkills": ["JavaScript", "React", "Node.js"]
}
```

Response:

```
{
  "success": true,
  "plan": {
    "content": "Your personalized learning plan...",
    "phases": [
      {
        "title": "Foundation Phase (Weeks 1-4)",
        "description": "Strengthen core skills",
        "topics": ["Advanced JavaScript", "ES6+ Features",
"Async Programming"],
        "resources": ["MDN Documentation", "JavaScript.info",
"Practice projects"]
      },
      {
        "title": "Backend Development (Weeks 5-8)",
        "description": "Learn server-side development",
        "topics": ["Express.js", "Database Design", "API
Development"],
        "resources": ["Express documentation", "MongoDB
University", "REST API tutorials"]
      }
    ],
    "estimatedDuration": "12 weeks",
    "difficulty": "intermediate",
    "generatedAt": "2024-01-15T10:30:00Z"
  }
}
```

Create AI Mentor Session

```
POST /api/v1/ai/sessions
```

Creates a new mentoring session with an AI mentor.

Request Body:

```
{
  "mentorId": "mentor_id_here",
  "sessionData": {
    "topic": "Career guidance",
    "goals": ["Understand career path", "Get skill
recommendations"]
  }
}
```

Response:

```
{
  "success": true,
  "session": {
    "id": "session_123",
    "mentorId": "mentor_id_here",
    "userId": "user_123",
    "topic": "Career guidance",
    "status": "active",
    "createdAt": "2024-01-15T10:30:00Z"
  }
}
```

Send Message in Session

```
POST /api/v1/ai/sessions/:sessionId/messages
```

Sends a message to an AI mentor in an active session.

Request Body:

```
{
  "message": "I'm struggling with choosing between React and
Vue.js. What would you recommend?",
  "context": {
    "userProfile": {
      "experienceLevel": "beginner",
```

```
      "currentSkills": ["JavaScript", "HTML", "CSS"]
    }
  }
}
```

Response:

```
{
  "success": true,
  "message": {
    "content": "Great question! Both React and Vue.js are
excellent choices...",
    "sender": "ai_mentor",
    "timestamp": "2024-01-15T10:30:00Z",
    "sessionId": "session_123"
  }
}
```

Service Architecture

The Google Gemini integration follows a modular architecture:

Core Components

1. **GeminiAIService** (`src/services/geminiAIService.js`)
2. Main service class handling all Gemini AI interactions
3. Manages API calls, rate limiting, and error handling
4. Provides methods for content generation and mentor management
5. **AI Controller** (`src/controllers/aiController.js`)
6. Express.js controller handling HTTP requests
7. Validates input and formats responses
8. Integrates with the GeminiAIService
9. **AI Routes** (`src/routes/aiRoutes.js`)
10. Defines REST API endpoints
11. Includes input validation middleware
12. Handles authentication and authorization
13. **AIMentor Model** (`src/models/AIMentor.js`)

14. MongoDB schema for AI mentor data
15. Includes methods for statistics and reviews
16. Supports search and filtering

Data Flow

```
 Client Request → AI Routes → AI Controller → GeminiAIService →
Google Gemini API
                                ↓
Database ← AIMentor Model ← Response Processing ← AI Response
```

Static Mentor Profiles

The service includes predefined mentor profiles covering various specializations:

- **Software Development**: Full-stack, Frontend, Backend, Mobile
- **Data Science**: Machine Learning, Data Analysis, AI/ML Engineering
- **Design**: UX/UI, Product Design, Graphic Design
- **Business**: Product Management, Marketing, Entrepreneurship
- **Engineering**: DevOps, Cloud Architecture, Cybersecurity

Each mentor profile includes: - Detailed biography and expertise - Teaching and communication styles - Skill compatibility ratings - Experience level and specializations

Testing

Running Tests

Execute the comprehensive test suite:

```
 # Run the integration tests
node test-gemini-integration.js
```

The test script validates: - Service initialization - Mentor data retrieval - Career advice generation - Learning plan creation - Session management - Message handling - Rate limiting - Usage statistics

Test Results

Test results are saved to `test-results.json` with detailed information:

```
{
  "timestamp": "2024-01-15T10:30:00Z",
  "duration": 15.2,
  "summary": {
    "total": 9,
    "passed": 9,
    "failed": 0,
    "successRate": "100.0"
  },
  "tests": [
    {
      "name": "Service Initialization",
      "passed": true,
      "error": null,
      "timestamp": "2024-01-15T10:30:01Z"
    }
  ]
}
```


Manual Testing

You can also test individual endpoints using curl or Postman:

```
# Test health check
curl -X GET http://localhost:3000/api/v1/ai/health \
  -H "Authorization: Bearer YOUR_JWT_TOKEN"

# Test career advice generation
curl -X POST http://localhost:3000/api/v1/ai/generate/career-
advice \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer YOUR_JWT_TOKEN" \
  -d '{
    "userProfile": {
      "currentRole": "Junior Developer",
      "experienceLevel": "beginner",
      "skills": ["JavaScript", "HTML", "CSS"],
      "careerGoals": "Become a full-stack developer"
    }
  }'
```

Deployment

Environment Variables

Ensure these environment variables are set in production:

```
NODE_ENV=production
GOOGLE_GEMINI_API_KEY=your_production_api_key
GOOGLE_GEMINI_MODEL=gemini-pro
GOOGLE_GEMINI_MAX_TOKENS=2048
GOOGLE_GEMINI_TEMPERATURE=0.7
```

Production Considerations

1. **API Key Security**
2. Store API keys in secure environment variables
3. Use different keys for development and production
4. Rotate keys regularly
5. **Rate Limiting**
6. Monitor API usage to avoid quota limits
7. Implement client-side rate limiting
8. Cache responses when appropriate
9. **Error Handling**
10. Log all API errors for monitoring
11. Implement graceful fallbacks
12. Set up alerts for service failures
13. **Performance**
14. Use connection pooling for database
15. Implement response caching
16. Monitor response times

Docker Deployment

If using Docker, update your Dockerfile to include the new dependencies:

```
FROM node:16-alpine

WORKDIR /app

COPY package*.json ./
RUN npm ci --only=production

COPY . .
```

```
EXPOSE 3000

CMD ["npm", "start"]
```

Troubleshooting

Common Issues

1. API Key Not Working

Error: `Invalid API key`

Solution: - Verify the API key is correct - Check that the key has proper permissions - Ensure the key is not expired

2. Rate Limit Exceeded

Error: `Rate limit exceeded`

Solution: - Implement request throttling - Cache responses to reduce API calls - Upgrade to higher quota if needed

3. Model Not Found

Error: `Model not found: gemini-pro`

Solution: - Check the model name in configuration - Verify the model is available in your region - Try using `gemini-pro-vision` for multimodal content

4. Content Blocked by Safety Filters

Error: `Content blocked by safety filters`

Solution: - Review and adjust safety settings - Modify prompts to be less ambiguous - Implement content preprocessing

Debugging

Enable debug logging by setting:

```
 DEBUG=gemini:*
NODE_ENV=development
```

This will provide detailed logs of all Gemini AI interactions.

Monitoring

Monitor these metrics in production:

- API response times
- Error rates
- Token usage
- Request volume
- Cache hit rates

Best Practices

1. Prompt Engineering

- Use clear, specific prompts
- Include context and examples
- Structure prompts consistently
- Test prompts thoroughly

2. Error Handling

- Always handle API errors gracefully
- Provide meaningful error messages to users
- Log errors for debugging
- Implement retry logic for transient failures

3. Performance Optimization

- Cache frequently requested content
- Use appropriate token limits
- Implement request batching where possible
- Monitor and optimize response times

4. Security

- Validate all user inputs
- Sanitize prompts to prevent injection
- Use HTTPS for all communications
- Regularly update dependencies

5. User Experience

- Provide loading indicators for AI operations
- Stream responses for long content
- Allow users to regenerate responses
- Implement feedback mechanisms

Migration from Manus AI

This integration replaces the previously planned Manus AI integration. Key changes:

Removed Components

- `ManusAIConfig` model
- `manusAIClient` service
- Manus AI configuration endpoints

Updated Components

- `AIMentor` model: `manusAIMentorId` → `geminiMentorId`
- AI controller: Updated to use Gemini service
- AI routes: Simplified endpoint structure
- Configuration: Google Gemini specific settings

Migration Steps

1. **Update Environment Variables** ```bash # Remove old Manus AI variables unset MANUS_AI_API_KEY unset MANUS_AI_ENDPOINT

Add new Gemini variables export GOOGLE_GEMINI_API_KEY=your_key_here ```

1. **Update Database** `javascript // MongoDB migration script db.aimentors.updateMany( { manusAIMentorId: { $exists: true } }, { $rename: { manusAIMentorId: "geminiMentorId" }, $set: { aiProvider: "gemini" } } );`

2. **Update Frontend Code**

3. Update API endpoint calls
4. Handle new response formats
5. Update error handling

Benefits of Migration

- **Real AI Service**: Google Gemini is a real, production-ready AI service
- **Better Performance**: Faster response times and higher reliability
- **Rich Features**: Advanced language understanding and generation
- **Scalability**: Google's infrastructure handles scaling automatically
- **Cost Effective**: Competitive pricing with generous free tier

Conclusion

The Google Gemini AI integration provides a robust, scalable solution for AI-powered mentoring in the TimeTravelers application. By following this guide, you can successfully implement and deploy the integration, providing users with intelligent career guidance and personalized learning experiences.

For additional support or questions, refer to the Google AI documentation or contact the development team.