# Horizon Platform - API Reference

**Version:** 1.0.0
**Base URL (Development):** http://localhost:4000
**Last Updated:** November 22, 2025

---

## Table of Contents



---

## Authentication

All API requests require authentication via Supabase JWT tokens.

**Headers Required:**

```
Authorization: Bearer <supabase_access_token>
Content-Type: application/json
```

---

## Gemini AI Endpoints

### Generate Text

Generate text using Google Gemini AI.

**Endpoint:** `POST /api/gemini/generate`

**Request Body:**

```
{
  "prompt": "Your prompt here",
  "model": "gemini-2.0-flash-exp",
  "temperature": 0.7,
  "maxTokens": 1024
}
```

**Response:**

```
{
  "text": "Generated text response",
  "model": "gemini-2.0-flash-exp",
  "duration": 1234,
```

```
  "cached": false
}
```

**Status Codes:**

- `200` : Success
- `400` : Invalid request
- `429` : Rate limit exceeded
- `500` : Server error

---

## Resume Endpoints

### Enhance Resume Content

Generate AI-enhanced resume content.

**Endpoint:** `POST /api/resume/enhance`

**Request Body:**

```
{
  "personalInfo": {
    "name": "John Doe",
    "email": "john@example.com",
    "phone": "+1234567890",
    "location": "San Francisco, CA",
    "linkedin": "linkedin.com/in/johndoe"
  },
  "summary": "Current summary text",
  "skills": ["JavaScript", "React", "Node.js"],
  "experience": [
    {
      "company": "Tech Corp",
      "position": "Software Engineer",
      "startDate": "2020-01",
      "endDate": "2023-12",
      "description": "Developed web applications",
      "achievements": ["Built X system"]
    }
  ],
  "education": [
    {
      "institution": "University Name",
      "degree": "Bachelor of Science",
      "fieldOfStudy": "Computer Science",
      "startDate": "2016",
      "endDate": "2020",
      "gpa": 3.8
    }
  ],
  "targetRole": "Senior Software Engineer",
```

```
  "jobDescription": "Job posting text here"
}
```

**Response:**

```
{
  "enhancedSummary": "AI-optimized professional summary (60-85 words)",
  "enhancedExperience": [
    {
      "company": "Tech Corp",
      "position": "Software Engineer",
      "bulletPoints": [
        "Architected microservices platform using Node.js...",
        "Led team of 5 engineers to deliver...",
        "Reduced deployment time by 75%..."
      ]
    }
  ],
  "optimizedSkills": {
    "technical": ["TypeScript", "React.js", "Node.js", "PostgreSQL"],
    "tools": ["AWS Lambda", "Docker", "Git", "Jenkins"],
    "soft": ["Agile/Scrum", "Team Leadership"]
  }
}
```

**Status Codes:**

- `200` : Success
- `400` : Invalid input
- `500` : Enhancement failed

---

## ATS Scoring Endpoints

### Score Resume Against Job Description

Analyze resume compatibility with ATS systems.

**Endpoint:** `POST /api/ats/score`

**Request Body:**

```
{
  "resumeText": "Full resume text content",
  "jobDescription": "Full job posting text"
}
```

**Response:**

```
{
  "overallScore": 78,
  "breakdown": {
```

```
    "keywordMatch": 24,
    "formatting": 18,
    "contentQuality": 20,
    "experienceRelevance": 16
  },
  "suggestions": [
    "Add missing keywords: 'Kubernetes', 'CI/CD', 'microservices'",
    "Quantify achievements: Include metrics like '40% faster' or '5M users'",
    "Strengthen technical depth: Expand on AWS experience",
    "Improve formatting: Use consistent bullet point style",
    "Add certification: Consider AWS Solutions Architect",
    "Enhance leadership indicators: Mention team size managed",
    "Update technology stack: Include version numbers (React 18)",
    "Optimize summary: Front-load critical keywords",
    "Add GitHub/portfolio: Include links to projects",
    "Strengthen action verbs: Replace 'worked on' with 'architected'"
  ],
  "missingKeywords": [
    "Kubernetes",
    "CI/CD",
    "microservices",
    "AWS Lambda",
    "Docker",
    "Python"
  ],
  "matchedKeywords": [
    "JavaScript",
    "React",
    "Node.js",
    "PostgreSQL"
  ]
}
```


**Score Breakdown Details:**

| Category | Max Points | Criteria |
|---|---|---|
| Keyword Match | 30 | % of job keywords present in resume |
| Formatting | 20 | ATS-friendly structure, no images/tables |
| Content Quality | 25 | Quantified achievements, action verbs |
| Experience | 25 | Relevance and depth of experience |

**Status Codes:**

- `200` : Success
- `400` : Invalid input
- `500` : Scoring failed

---

## Interview Endpoints

### Generate Interview Questions

Get AI-generated interview questions for a role.

**Endpoint:** `POST /api/interview/questions`

**Request Body:**

```
{
  "role": "Software Engineer",
  "difficulty": "mid",
  "count": 10,
  "topics": ["algorithms", "system design", "behavioral"]
}
```

**Response:**

```
{
  "questions": [
    {
      "id": "q1",
      "question": "Explain how you would design a URL shortener service",
      "type": "system-design",
      "difficulty": "mid",
      "expectedTime": 300
    },
    {
      "id": "q2",
      "question": "Tell me about a time when you had to resolve a conflict",
      "type": "behavioral",
      "difficulty": "mid",
      "expectedTime": 180
    }
  ]
}
```

### Evaluate Interview Response

Get AI feedback on interview answer.

**Endpoint:** `POST /api/interview/evaluate`

**Request Body:**

```
{
  "question": "Explain how you would design a URL shortener",
  "response": "User's answer text",
  "role": "Software Engineer",
```

```
  "evaluationCriteria": ["technical accuracy", "communication", "depth"]
}
```

**Response:**

```
{
  "score": 85,
  "strengths": [
    "Clear explanation of hashing algorithm",
    "Considered scalability requirements",
    "Good use of concrete examples"
  ],
  "improvements": [
    "Could elaborate on database sharding strategy",
    "Missing discussion of caching layer"
  ],
  "followUpQuestions": [
    "How would you handle millions of requests per second?",
    "What database would you choose and why?"
  ]
}
```

**Status Codes:**

- `200` : Success
- `400` : Invalid input
- `500` : Evaluation failed

---

# Roadmap Endpoints

## Generate Learning Roadmap

Create personalized learning path.

**Endpoint:** `POST /api/roadmap/generate`

**Request Body:**

```
{
  "targetRole": "Senior Full-Stack Engineer",
  "currentRole": "Junior Developer",
  "currentSkills": ["JavaScript", "React", "HTML/CSS"],
  "timeframe": "6 months",
  "preferredLearningStyle": "project-based"
}
```

**Response:**

```
{
  "roadmap": {
    "title": "Junior Developer → Senior Full-Stack Engineer",
```

```
    "duration": "6 months",
    "phases": [
      {
        "id": "phase1",
        "title": "Foundation Strengthening",
        "duration": "1 month",
        "topics": [
          "Advanced JavaScript (ES6+)",
          "TypeScript fundamentals",
          "Testing with Jest"
        ],
        "resources": [
          {
            "title": "JavaScript: The Definitive Guide",
            "type": "book",
            "url": "https://..."
          },
          {
            "title": "TypeScript Crash Course",
            "type": "video",
            "url": "https://youtube.com/..."
          }
        ],
        "projects": [
          "Build a TypeScript library",
          "Create comprehensive test suite"
        ]
      },
      {
        "id": "phase2",
        "title": "Backend Development",
        "duration": "2 months",
        "topics": [
          "Node.js & Express",
          "Database design (SQL & NoSQL)",
          "RESTful API design",
          "Authentication & Security"
        ],
        "resources": [...],
        "projects": [...]
      }
    ]
  }
}
```


**Status Codes:**

- `200` : Success
- `400` : Invalid input
- `500` : Generation failed

---

## Cache Management

### Get Cache Statistics

View current cache status.

**Endpoint:** `GET /api/cache/stats`

**Response:**

```
{
  "size": 1024000,
  "entries": 42,
  "hitRate": 0.73,
  "ttl": 600000
}
```

### Clear Cache

Clear all cached responses.

**Endpoint:** `DELETE /api/cache/clear`

**Response:**

```
{
  "message": "Cache cleared successfully",
  "itemsCleared": 42
}
```

**Status Codes:**

- `200` : Success
- `500` : Operation failed

---

## Error Handling

### Standard Error Response

```
{
  "error": {
    "code": "INVALID_INPUT",
    "message": "Resume text is required",
    "details": {
      "field": "resumeText",
      "reason": "Field cannot be empty"
    }
  }
}
```

### Error Codes

| Code | HTTP Status | Description |
|---|---|---|
| `INVALID_INPUT` | 400 | Request validation failed |
| `UNAUTHORIZED` | 401 | Authentication required |
| `FORBIDDEN` | 403 | Insufficient permissions |
| `NOT_FOUND` | 404 | Resource not found |
| `RATE_LIMIT_EXCEEDED` | 429 | Too many requests |
| `SERVER_ERROR` | 500 | Internal server error |
| `AI_SERVICE_ERROR` | 503 | Gemini API unavailable |

---

## Rate Limiting

**Limits:**

- 60 requests per minute per IP
- 1000 requests per hour per user
- 10,000 requests per day per account

**Response Headers:**

```
X-RateLimit-Limit: 60
X-RateLimit-Remaining: 45
X-RateLimit-Reset: 1637712000
```

**Rate Limit Exceeded Response:**

```
{
  "error": {
    "code": "RATE_LIMIT_EXCEEDED",
    "message": "Too many requests",
    "retryAfter": 30
  }
}
```

---

## Webhooks (Coming Soon)

### Resume Analysis Complete

Webhook fired when resume analysis finishes.

**Endpoint:** Your configured URL

**Payload:**

```
{
  "event": "resume.analysis.complete",
  "timestamp": "2025-11-22T12:00:00Z",
  "data": {
    "userId": "user-uuid",
    "resumeId": "resume-uuid",
    "score": 85,
    "suggestionsCount": 12
  }
}
```

---

## SDK Examples

### JavaScript/TypeScript

```
import { HorizonAPI } from '@horizon/sdk';

const client = new HorizonAPI({
  apiKey: process.env.HORIZON_API_KEY,
  baseURL: 'http://localhost:4000'
});

// Score resume
const result = await client.ats.score({
  resumeText: resumeContent,
  jobDescription: jobPosting
});

console.log(`ATS Score: ${result.overallScore}/100`);
```

### Python

```
from horizon_sdk import HorizonClient

client = HorizonClient(
    api_key=os.getenv('HORIZON_API_KEY'),
    base_url='http://localhost:4000'
)

# Generate roadmap
roadmap = client.roadmap.generate(
    target_role='Data Scientist',
    current_skills=['Python', 'SQL'],
    timeframe='6 months'
)

print(f"Roadmap has {len(roadmap.phases)} phases")
```

---

## Testing

### Health Check

```
curl http://localhost:4000/health
```

**Response:**

```
{
  "status": "ok",
  "timestamp": "2025-11-22T12:00:00Z",
  "uptime": 86400,
  "version": "1.0.0"
}
```

### Test ATS Scoring

```
curl -X POST http://localhost:4000/api/ats/score \
  -H "Content-Type: application/json" \
  -d '{
    "resumeText": "Software Engineer with 5 years experience...",
    "jobDescription": "Looking for Senior Engineer with React..."
  }'
```

---

## Best Practices

### 1. Use Caching

Enable caching for frequently requested data:

```
{
  "useCache": true,
  "cacheTTL": 600
}
```

### 2. Batch Requests

Combine multiple operations when possible:

```
{
  "operations": [
    { "type": "enhance", "section": "summary" },
    { "type": "enhance", "section": "experience" },
    { "type": "optimize", "section": "skills" }
  ]
}
```

### 3. Handle Errors Gracefully

Always implement retry logic with exponential backoff:

```
async function callAPIWithRetry(fn, maxRetries = 3) {
  for (let i = 0; i < maxRetries; i++) {
    try {
      return await fn();
    } catch (error) {
      if (i === maxRetries - 1) throw error;
      await sleep(Math.pow(2, i) * 1000);
    }
  }
}
```

### 4. Monitor Rate Limits

Check rate limit headers in every response and implement backoff.

---

## Changelog

### Version 1.0.0 (November 2025)

- Initial API release
- Gemini AI integration
- ATS scoring with 10+ suggestions
- Interview simulation endpoints
- Learning roadmap generation
- Cache management

---

## Support

- **Documentation**: See `DOCUMENTATION.pdf`
- **Issues**: https://github.com/Vaasu08/merged-app/issues
- **API Status**: https://status.horizon-platform.com (coming soon)

---

**API Version:** 1.0.0
**Last Updated:** November 22, 2025