API Documentation

Base URL

```
Development: http://127.0.0.1:5000
```

Authentication

All protected endpoints require a JWT token in the Authorization header:

```
Authorization: Bearer <access_token>
```

Token Lifecycle

- **Access Token**: 15 minutes
- **Refresh Token**: 7 days

Endpoints

Authentication

Register User

```
POST /api/register
```

Request Body:

```
{
  "email": "user@example.com",
  "password": "SecurePass123!",
  "name": "John Doe"
}
```

Response (201):

```
{
  "message": "User registered successfully",
  "user_id": 1
}
```

Login

```
POST /api/login
```

Request Body:

```
{
  "email": "user@example.com",
  "password": "SecurePass123!"
}
```

Response (200):

```
{
  "access_token": "eyJ0eXAiOiJKV1QiLCJhbGc...",
  "refresh_token": "eyJ0eXAiOiJKV1QiLCJhbGc...",
  "user": {
    "id": 1,
    "email": "user@example.com",
    "name": "John Doe"
  }
}
```

Refresh Token

```
POST /api/refresh
```

Request Body:

```
{
  "refresh_token": "eyJ0eXAiOiJKV1QiLCJhbGc..."
}
```

Response (200):

```
{
  "access_token": "eyJ0eXAiOiJKV1QiLCJhbGc..."
}
```

Interview Management

Upload Resume & Start Interview

```
POST /api/upload-resume
Authorization: Bearer <token>
Content-Type: multipart/form-data
```

Request Body:

```
resume: <PDF file>
jobRole: "Software Engineer"
```

Response (200):

```
{
  "interview_id": 123,
  "questions": [
    {
      "id": 1,
      "question": "Explain the difference between REST and GraphQL APIs.",
      "expected_points": ["REST uses multiple endpoints", "GraphQL uses single endpoint
    }
  ]
}
```

Start Role-Based Interview

```
POST /api/start-role-interview
Authorization: Bearer <token>
```

Request Body:

```
{
  "jobRole": "Data Scientist",
  "jobDescription": "ML model development and deployment",
  "focusAreas": "Python, TensorFlow, AWS"
}
```

Response (200):

```
{
  "interview_id": 124,
  "questions": [...]
}
```

Submit Answer

```
POST /api/submit-answer-enhanced
Authorization: Bearer <token>
```

Request Body:

```
{
  "interviewId": 123,
  "questionId": 1,
  "answer": "REST APIs use multiple endpoints for different resources...",
  "timeSpent": 45
}
```

Response (200):

```
{
  "score": 85,
  "technical_score": 90,
  "communication_score": 80,
  "confidence_score": 85,
  "feedback": "Strong technical understanding. Consider providing more specific example
  "followup": {
    "question": "Can you explain when you would choose GraphQL over REST?",
    "questionId": 2,
    "timeLimit": 300
  }
}
```

Complete Interview

```
POST /api/complete-interview
Authorization: Bearer <token>
```

Request Body:

```
{
  "interviewId": 123
}
```

Response (200):

```
{
  "message": "Interview completed successfully",
  "evaluation_metrics": {
    "average_overall": 82.5,
    "average_technical": 85.0,
    "average_communication": 80.0,
    "average_confidence": 82.5
  },
  "improvement_plan": {...},
  "personalized_feedback": {...}
}
```

Multi-Round Interviews

Suggest Rounds

```
POST /api/suggest-rounds
Authorization: Bearer <token>
```

Request Body:

```
{
  "jobRole": "Senior Software Engineer",
  "jobDescription": "Lead backend development team"
}
```

Response (200):

```
{
  "suggested_rounds": [
    {
      "round_name": "HR Screening",
      "round_type": "hr",
      "description": "Initial screening to assess background and cultural fit",
      "duration_minutes": 20,
      "question_count": 5,
      "focus_areas": ["Background", "Motion", "Culture fit", "Expectations"]
    },
    {
      "round_name": "Technical Round",
      "round_type": "technical",
      "description": "Assess technical skills and problem-solving abilities",
      "duration_minutes": 45,
      "question_count": 5,
      "focus_areas": ["Coding", "Algorithms", "System design", "Best practices"]
    }
  ]
}
```


Start Multi-Round Interview

```
POST /api/start-multi-round-interview
Authorization: Bearer <token>
```

Request Body:

```
{
  "jobRole": "Senior Software Engineer",
  "jobDescription": "Lead backend development",
  "selectedRounds": [
    {
      "round_name": "Technical Round",
      "round_type": "technical",
      "duration_minutes": 45,
      "question_count": 5
    },
    {
      "round_name": "System Design",
      "round_type": "system_design",
      "duration_minutes": 60,
      "question_count": 2
    }
  ]
}
```

Response (200):

```
{
  "interview_id": 125,
  "round_ids": [1, 2],
  "message": "Multi-round interview created successfully"
}
```

Start Round

```
POST /api/start-round/<round_id>
Authorization: Bearer <token>
```

Response (200):

```
{
  "round_id": 1,
  "round_name": "Technical Round",
  "round_type": "technical",
  "questions": [
    {
      "id": 10,
      "question": "Implement a function to reverse a linked list.",
      "expected_points": ["Iterative approach", "Recursive approach", "Time complexity
    }
  ]
}
```


Complete Round

```
POST /api/complete-round/<round_id>
Authorization: Bearer <token>
```

Response (200):


```
{
  "round_score": 78.5,
  "message": "Round completed successfully",
  "next_round": {
    "id": 2,
    "name": "System Design",
    "type": "system_design"
  }
}
```


Feedback & Results

Get Personalized Feedback

```
GET /api/personalized-feedback/<interview_id>
Authorization: Bearer <token>
```

Response (200):

```
{
  "strengths": [
    "Strong technical knowledge in data structures",
    "Clear and articulate communication",
    "Consistent performance across questions"
  ],
  "weaknesses": [
    "Needs improvement in algorithm complexity analysis",
    "Could provide more specific real-world examples"
  ],
  "roadmap": {
    "immediate": [
      "Practice mock interviews daily",
      "Review fundamental algorithms",
      "Prepare 5 STAR-method examples"
    ],
    "short_term": [
      "Complete data structures course",
      "Solve 50 LeetCode problems",
      "Join study group"
    ],
    "long_term": [
      "Master advanced algorithms",
      "Contribute to open-source projects",
      "Mentor junior developers"
    ]
  },
  "resources": [
    {
      "title": "Algorithms Specialization",
      "type": "course",
      "description": "Comprehensive algorithms course by Stanford",
      "url": "https://coursera.org/...",
      "priority": "high"
    }
  ],
  "generated_at": "2024-01-15T10:30:00Z"
}
```

Get Interview Results

```
GET /api/interview-results/<interview_id>
Authorization: Bearer <token>
```

Response (200):

```
{
  "interview": {
    "id": 123,
    "job_role": "Software Engineer",
    "score": 82.5,
    "status": "completed",
    "created_at": "2024-01-15T10:00:00Z"
  },
  "questions": [...],
  "evaluation_metrics": {...},
  "improvement_plan": {...}
}
```

Error Responses

400 Bad Request

```
{
  "error": "Missing required fields"
}
```

401 Unauthorized

```
{
  "error": "Invalid or expired token"
}
```

403 Forbidden

```
{
  "error": "Insufficient permissions"
}
```

404 Not Found

```
{
  "error": "Interview not found or unauthorized"
}
```

429 Too Many Requests

```
{
  "error": "Rate limit exceeded. Please try again later."
}
```

500 Internal Server Error

```
{
  "error": "An unexpected error occurred"
}
```

Rate Limits

Endpoint Type	Limit
Authentication	5 requests/minute
Interview Start	10 requests/minute
Answer Submission	20 requests/minute
Feedback Retrieval	30 requests/minute

Example Usage (cURL)

Complete Interview Flow

```
# 1. Register
curl -X POST http://127.0.0.1:5000/api/register \
  -H "Content-Type: application/json" \
  -d '{"email":"test@example.com","password":"Test123!","name":"Test User"}'

# 2. Login
TOKEN=$(curl -X POST http://127.0.0.1:5000/api/login \
  -H "Content-Type: application/json" \
  -d '{"email":"test@example.com","password":"Test123!"}' \
  | jq -r '.access_token')

# 3. Start Interview
INTERVIEW_ID=$(curl -X POST http://127.0.0.1:5000/api/start-role-interview \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $TOKEN" \
  -d '{"jobRole":"Software Engineer","jobDescription":"Full-stack development"}' \
  | jq -r '.interview_id')

# 4. Submit Answer
curl -X POST http://127.0.0.1:5000/api/submit-answer-enhanced \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $TOKEN" \
  -d "{\"interviewId\":$INTERVIEW_ID,\"questionId\":1,\"answer\":\"My answer here\",\"t

# 5. Complete Interview
curl -X POST http://127.0.0.1:5000/api/complete-interview \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer $TOKEN" \
  -d "{\"interviewId\":$INTERVIEW_ID}"

# 6. Get Feedback
curl -X GET http://127.0.0.1:5000/api/personalized-feedback/$INTERVIEW_ID \
  -H "Authorization: Bearer $TOKEN"
```

Postman Collection

Import this collection for easy API testing:


```
{
  "info": {
    "name": "AI Mock Interview API",
    "schema": "https://schema.getpostman.com/json/collection/v2.1.0/collection.json"
  },
  "item": [
    {
      "name": "Authentication",
      "item": [
        {
          "name": "Register",
          "request": {
            "method": "POST",
            "url": "{{base_url}}/api/register",
            "body": {
              "mode": "raw",
              "raw": "{\"email\":\"test@example.com\",\"password\":\"Test123!\",\"name\
            }
          }
        }
      ]
    }
  ],
  "variable": [
    {
      "key": "base_url",
      "value": "http://127.0.0.1:5000"
    }
  ]
}
```