SkillSwap Postman API Documentation

Base URL

```
http://localhost:8080
```

📋 Table of Contents

🔐 Authentication

1. Register a New User

POST `/register`

Body (JSON):

```json
{
  "name": "John Doe",
  "email": "john@example.com",
  "password": "Password@123"
}
```

Response (201 Created):

```json
```

```
{
  "userId": "550e8400-e29b-41d4-a716-446655440000",
  "name": "John Doe",
  "email": "john@example.com",
  "isAccountVerified": false
}
```

Notes:

- Password must be at least 8 characters with uppercase, lowercase, digit, and special character (@, #, $)

---

2. Login

POST `/login`

Body (JSON):

```
json

{
  "email": "john@example.com",
  "password": "Password@123"
}
```

Response (200 OK):

```
json

{
  "email": "john@example.com",
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
}
```

Headers (Set automatically):

```
Set-Cookie: jwt=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...; HttpOnly; Secure; SameSite=Strict
```

Note: Save the token for authenticated requests

---

3. Check Authentication Status

GET `/is-authenticated`

Headers:

```
Authorization: Bearer {token}
```

Response (200 OK):

```
true
```

---

4. Send OTP for Email Verification

POST `/send-otp`

Headers:

```
Authorization: Bearer {token}
```

Response (200 OK):

```
OTP sent to your email
```

---

5. Verify Email with OTP

POST `/verify-otp`

Headers:

```
Authorization: Bearer {token}
```

Body (JSON):

```
json

{
  "otp": "123456"
}
```

Response (200 OK):

```
Email verified successfully
```

---

6. Send Password Reset OTP

POST `/send-reset-otp?email=john@example.com`

Response (200 OK):

```
OTP sent to your email
```

---

7. Reset Password

POST `/reset-password`

Body (JSON):

```
json

{
  "email": "john@example.com",
  "otp": "123456",
  "newPassword": "NewPassword@123"
}
```

Response (200 OK):

```
Password reset successfully
```

---

8. Logout

POST `/logout`

Headers:

```
Authorization: Bearer {token}
```

Response (200 OK):

```
Logged out successfully!
```

👤 Profile Management

1. Get Your Profile

GET `/api/profile`

Headers:

```
Authorization: Bearer {token}
```

Response (200 OK):

json


```
{
  "id": 1,
  "email": "john@example.com",
  "firstName": "John",
  "lastName": "Doe",
  "bio": "Software engineer passionate about teaching",
  "dateOfBirth": "1990-05-15",
  "phoneNumber": "+919876543210",
  "location": "Mumbai, India",
  "website": "https://johndoe.com",
  "linkedinUrl": "https://linkedin.com/in/johndoe",
  "githubUrl": "https://github.com/johndoe",
  "skills": "Java, Spring Boot, React",
  "interests": "Web Development, Cloud Computing",
  "skillsToLearn": "Docker, Kubernetes",
  "timezone": "Asia/Kolkata",
  "hoursPerWeek": 10,
  "learningGoal": "JOB_PREP",
  "goalTimeline": "MEDIUM",
  "teachingMotivation": "LOVE_TEACHING",
  "teachingApproach": "STEP_BY_STEP",
  "preferredLearningMethod": "VIDEO_CALL",
  "communicationPace": "MODERATE",
  "preferredLanguage": "English",
  "domainFocus": "WEB_DEV",
  "isVerified": true,
  "createdAt": "2024-01-15T10:30:00",
  "updatedAt": "2024-01-18T15:45:00",
  "profileCompletenessScore": 85
}
```


---

2. Create Profile (First Time)

POST `/api/profile`

Headers:

```
Authorization: Bearer {token}
Content-Type: application/json
```

Body (JSON):

```json
{
  "firstName": "John",
  "lastName": "Doe",
  "bio": "Software engineer passionate about teaching",
  "dateOfBirth": "1990-05-15",
  "phoneNumber": "+919876543210",
  "location": "Mumbai, India",
  "website": "https://johndoe.com",
  "linkedinUrl": "https://linkedin.com/in/johndoe",
  "githubUrl": "https://github.com/johndoe",
  "skills": "Java, Spring Boot, React",
  "interests": "Web Development, Cloud Computing",
  "skillsToLearn": "Docker, Kubernetes",
  "timezone": "Asia/Kolkata",
  "hoursPerWeek": 10,
  "learningGoal": "JOB_PREP",
  "goalTimeline": "MEDIUM",
  "teachingMotivation": "LOVE_TEACHING",
  "teachingApproach": "STEP_BY_STEP",
  "preferredLearningMethod": "VIDEO_CALL",
  "communicationPace": "MODERATE",
  "preferredLanguage": "English",
  "domainFocus": "WEB_DEV"
}
```

Response (201 Created):

```json
```

```
{
  "id": 1,
  "email": "john@example.com",
  "firstName": "John",
  "lastName": "Doe",
  "profileCompletenessScore": 85
}
```

---

3. Update Profile

PUT `/api/profile`

Headers:

```
Authorization: Bearer {token}
Content-Type: application/json
```

Body: (Same as Create - send only fields you want to update)

Response (200 OK): (Updated profile data)

---

4. Delete Profile

DELETE `/api/profile`

Headers:

```
Authorization: Bearer {token}
```

Response (200 OK):

```
json

{
  "message": "Profile deleted successfully"
}
```

---

5. Search Profiles

GET `/api/profile/search?keyword=java&page=0&size=10&sortBy=firstName`

Headers:

```
Authorization: Bearer {token}
```

Query Parameters:

- `keyword`: Search term (name, skills)
- `page`: Page number (default: 0)
- `size`: Results per page (default: 10)
- `sortBy`: Field to sort by (default: firstName)

Response (200 OK):

```
json

{
  "content": [
    {
      "id": 2,
      "firstName": "Jane",
      "lastName": "Smith",
      "email": "jane@example.com",
      "skills": "Java, Python, JavaScript",
      "profileCompletenessScore": 78
    }
  ],
  "totalElements": 5,
  "totalPages": 1,
  "currentPage": 0
}
```

🎓 User Skills Management

1. Add/Update Skill Level

POST `/api/profile/skills`

Headers:

```
Authorization: Bearer {token}
Content-Type: application/json
```

Body (JSON):

```json
{
  "skillName": "Spring Boot",
  "proficiencyLevel": "ADVANCED",
  "yearsOfExperience": 5,
  "selfRating": 4,
  "projectsCompleted": 12,
  "lastUsedDate": "2024-01-15",
  "willingToTeach": true
}
```

Proficiency Levels: BEGINNER, INTERMEDIATE, ADVANCED, EXPERT

Response (201 Created):

```json
{
  "id": 1,
  "skillName": "Spring Boot",
  "proficiencyLevel": "ADVANCED",
  "yearsOfExperience": 5,
  "selfRating": 4,
  "projectsCompleted": 12,
  "lastUsedDate": "2024-01-15",
  "willingToTeach": true
}
```

---

2. Get All Your Skills

GET `/api/profile/skills`

Headers:

```
Authorization: Bearer {token}
```

Response (200 OK):

```json
```

```
[
  {
    "id": 1,
    "skillName": "Spring Boot",
    "proficiencyLevel": "ADVANCED",
    "yearsOfExperience": 5,
    "selfRating": 4,
    "projectsCompleted": 12,
    "willingToTeach": true
  },
  {
    "id": 2,
    "skillName": "React",
    "proficiencyLevel": "INTERMEDIATE",
    "yearsOfExperience": 3,
    "selfRating": 3,
    "projectsCompleted": 8,
    "willingToTeach": false
  }
]
```


---

3. Get Specific Skill

GET `/api/profile/skills/Spring%20Boot`

Headers:

```
Authorization: Bearer {token}
```

Response (200 OK):

json


```
{
  "id": 1,
  "skillName": "Spring Boot",
  "proficiencyLevel": "ADVANCED",
  "yearsOfExperience": 5,
  "selfRating": 4,
  "projectsCompleted": 12,
  "willingToTeach": true
}
```


---

4. Delete Skill

DELETE `/api/profile/skills/Spring%20Boot`

Headers:

```
Authorization: Bearer {token}
```

Response (200 OK):

```json
{
  "message": "Skill level deleted successfully"
}
```

📄 Resume Portal

Certifications

1. Get All Certifications

GET `/api/resume/certifications`

Headers:

```
Authorization: Bearer {token}
```

Response (200 OK):

```json
[
  {
    "id": 1,
    "skillName": "Spring Boot",
    "certificationName": "Spring Professional Developer",
    "issuingOrganization": "Pivotal Software",
    "issueDate": "Jan 2023",
    "expirationDate": "Jan 2026",
    "proofUrl": "https://example.com/cert1.pdf",
    "credibilityScore": 0.9
  }
]
```

---

2. Add Certification

POST `/api/resume/certifications`

Headers:

```
Authorization: Bearer {token}
Content-Type: application/json
```

Body (JSON):

```json
{
  "skillName": "Spring Boot",
  "certificationName": "Spring Professional Developer",
  "issuingOrganization": "Pivotal Software",
  "issueDate": "Jan 2023",
  "expirationDate": "Jan 2026",
  "proofUrl": "https://example.com/cert1.pdf"
}
```

Response (201 Created): (Certification data with credibilityScore)

---

3. Update Certification

PUT `/api/resume/certifications/{id}`

Headers:

```
Authorization: Bearer {token}
Content-Type: application/json
```

Body: (Same as Add - send fields to update)

Response (200 OK): (Updated certification data)

---

4. Delete Certification

DELETE `/api/resume/certifications/{id}`

Headers:

```
Authorization: Bearer {token}
```

Response (204 No Content)

Coding Stats

1. Get Coding Stats

GET `/api/resume/coding-stats`

Headers:

```
Authorization: Bearer {token}
```

Response (200 OK):

```json
[
  {
    "id": 1,
    "platformName": "LeetCode",
    "profileUrl": "https://leetcode.com/johndoe/",
    "totalProblemsSolved": 450,
    "easySolved": 150,
    "mediumSolved": 200,
    "hardSolved": 100,
    "proofUrl": "https://example.com/leetcode.png"
  }
]
```

---

2. Add Coding Stats

POST `/api/resume/coding-stats`

Headers:

```
Authorization: Bearer {token}
Content-Type: application/json
```

Body (JSON):

```json
```

```
{
  "platformName": "LeetCode",
  "profileUrl": "https://leetcode.com/johndoe/",
  "totalProblemsSolved": 450,
  "easySolved": 150,
  "mediumSolved": 200,
  "hardSolved": 100,
  "proofUrl": "https://example.com/leetcode.png"
}
```


Response (201 Created): (Coding stats data)

---

3. Update Coding Stats

PUT `/api/resume/coding-stats/{id}`

Headers:

```
Authorization: Bearer {token}
```

Response (200 OK): (Updated stats)

---

4. Delete Coding Stats

DELETE `/api/resume/coding-stats/{id}`

Response (204 No Content)

Experience

1. Get All Experience

GET `/api/resume/experience`

Headers:

```
Authorization: Bearer {token}
```

Response (200 OK):

```
json
```

```
[
  {
    "id": 1,
    "jobTitle": "Senior Software Engineer",
    "companyName": "Google",
    "skillName": "Spring Boot",
    "startDate": "Jan 2021",
    "endDate": "Dec 2023",
    "description": "Led microservices architecture development",
    "proofUrl": "https://example.com/offer.pdf"
  }
]
```


---

2. Add Experience

POST `/api/resume/experience`

Headers:

```
Authorization: Bearer {token}
```

Body (JSON):

```json
{
  "jobTitle": "Senior Software Engineer",
  "companyName": "Google",
  "skillName": "Spring Boot",
  "startDate": "Jan 2021",
  "endDate": "Dec 2023",
  "description": "Led microservices architecture development",
  "proofUrl": "https://example.com/offer.pdf"
}
```

Response (201 Created): (Experience data)

---

3. Update Experience

PUT `/api/resume/experience/{id}`

Response (200 OK): (Updated experience)

---

4. Delete Experience

DELETE `/api/resume/experience/{id}`

Response (204 No Content)

Education

1. Get All Education

GET `/api/resume/education`

Headers:

```
Authorization: Bearer {token}
```

Response (200 OK):

```json
[
  {
    "id": 1,
    "educationLevel": "Bachelor's Degree",
    "institutionName": "Indian Institute of Technology",
    "boardOrUniversity": "IIT Bombay",
    "fieldOfStudy": "Computer Science",
    "passingYear": "2020",
    "scoreDetails": "9.2 CGPA",
    "proofUrl": "https://example.com/degree.pdf"
  }
]
```

---

2. Add Education

POST `/api/resume/education`

Headers:

```
Authorization: Bearer {token}
```

Body (JSON):

```json
{
  "educationLevel": "Bachelor's Degree",
  "institutionName": "Indian Institute of Technology",
  "boardOrUniversity": "IIT Bombay",
  "fieldOfStudy": "Computer Science",
  "passingYear": "2020",
  "scoreDetails": "9.2 CGPA",
  "proofUrl": "https://example.com/degree.pdf"
}
```

Response (201 Created): (Education data)

---

3. Update Education

PUT `/api/resume/education/{id}`

Response (200 OK): (Updated education)

---

4. Delete Education

DELETE `/api/resume/education/{id}`

Response (204 No Content)

🧪 Test Portal

1. Generate Test for a Skill

POST `/api/test/generate`

Headers:

```
Authorization: Bearer {token}
Content-Type: application/json
```

Body (JSON):

```json
```

```
{
  "skillName": "Java"
}
```

Supported Skills: Java, Spring Boot

Response (201 Created):

```
json

{
  "testId": 1,
  "skillName": "Java",
  "totalQuestions": 15,
  "passingScore": 10,
  "questions": [
    {
      "questionNumber": 1,
      "question": "Which statement about Java memory model is TRUE regarding 'volatile'?",
      "options": [
        "A. It prevents instruction reordering and guarantees atomicity",
        "B. It guarantees visibility across threads but not atomicity",
        "C. It is equivalent to 'synchronized'",
        "D. It only affects writes, not reads"
      ]
    }
  ],
  "expiresAt": 1705606200000,
  "testStatus": "PENDING"
}
```

Note: Test expires in 30 minutes

---

2. Submit Test

POST `/api/test/submit`

Headers:

```
Authorization: Bearer {token}
Content-Type: application/json
```

Body (JSON):

```
json
```

```
{
  "testId": 1,
  "answers": [
    {
      "questionNumber": 1,
      "selectedAnswer": "B"
    },
    {
      "questionNumber": 2,
      "selectedAnswer": "B"
    }
  ]
}
```

Response (200 OK):

```
json

{
  "testId": 1,
  "skillName": "Java",
  "score": 12,
  "totalQuestions": 15,
  "passingScore": 10,
  "isPassed": true,
  "questionResults": [
    {
      "questionNumber": 1,
      "question": "Which statement about Java memory model...",
      "correctAnswer": "B",
      "userAnswer": "B",
      "isCorrect": true
    }
  ]
}
```

---

3. Get Test History

GET `/api/test/history`

Headers:

```
Authorization: Bearer {token}
```

Response (200 OK):

```
json

[
  {
    "testId": 1,
    "skillName": "Java",
    "score": 12,
    "totalQuestions": 15,
    "isPassed": true,
    "testStatus": "COMPLETED",
    "createdAt": "2024-01-15T10:30:00"
  }
]
```

---

4. Get Test Result

GET `/api/test/result/{testId}`

Headers:

```
Authorization: Bearer {token}
```

Response (200 OK): (Same as submit test response)

---

5. Check Qualification Status

GET `/api/test/qualification-status?skill=Java`

Headers:

```
Authorization: Bearer {token}
```

Response (200 OK):

```
json

{
  "skillName": "Java",
  "isQualified": true
}
```

Note: Returns true only if user passed test (score >= 10/15)

🎯 Skill Matching

1. Find Skill Swap Matches

GET `/api/matches/swap?skillToLearn=Java&skillToTeach=Spring%20Boot`

Headers:

```
Authorization: Bearer {token}
```

Query Parameters:

- `skillToLearn`: Skill you want to learn
- `skillToTeach`: Skill you want to teach (must have passed test)

Response (200 OK):

```json
```

```
[
  {
    "partnerId": 2,
    "partnerName": "Jane Smith",
    "partnerEmail": "jane@example.com",
    "profileImageUrl": "https://example.com/jane.jpg",
    "location": "Bangalore, India",
    "timezone": "Asia/Kolkata",
    "skillILearn": "Java",
    "skillITeach": "Spring Boot",
    "matchScore": 0.92,
    "swapType": "PERFECT_SWAP",
    "mutualMatch": true,
    "skillLevelGapScore": 0.9,
    "goalAlignmentScore": 0.85,
    "availabilityScore": 0.88,
    "timeCommitmentScore": 0.9,
    "learningStyleScore": 0.8,
    "theirTestScore": 0.93,
    "theirCertScore": 0.9,
    "theirReputationScore": 0.95,
    "myTestScore": 0.87,
    "myCertScore": 0.85,
    "myReputationScore": 0.9,
    "theirSkillLevel": "EXPERT",
    "mySkillLevel": "BEGINNER",
    "theirGoal": "REINFORCE_KNOWLEDGE",
    "myGoal": "JOB_PREP",
    "availabilityOverlapHours": 8,
    "compatibilityReason": "Perfect bidirectional match! You both want to learn from each other. They are expert level...",
    "partnerBio": "Passionate about teaching Spring Boot",
    "partnerSkills": "Java, Spring Boot, Docker",
    "partnerSkillsToLearn": "Kubernetes, React",
    "partnerHoursPerWeek": 12,
    "partnerLearningMethod": "VIDEO_CALL",
    "partnerDomainFocus": "WEB_DEV"
  }
]
```


Match Types:

- `PERFECT_SWAP`: Both want to learn from each other (higher score)
- `PARTIAL_MATCH`: One-way learning opportunity (lower score)

⭐ Feedback & Reputation

1. Submit Feedback

POST `/api/feedback/submit`

Headers:

```
Authorization: Bearer {token}
Content-Type: application/json
```

Body (JSON):

```json
{
  "revieweeId": 2,
  "sessionId": 1,
  "rating": 5,
  "comments": "Excellent teacher! Very clear explanations.",
  "dimensions": "Communication:5,Punctuality:5,Knowledge:4"
}
```

Rating: 1-5 (required)

Response (200 OK):

```json
{
  "message": "Feedback submitted successfully"
}
```

💬 Sessions

1. Request Learning Session

POST `/api/sessions/request?teacherId=2&skill=Spring%20Boot`

Headers:

```
Authorization: Bearer {token}
```

Response (200 OK):

```json
{
  "sessionId": 1,
  "skillName": "Spring Boot",
  "status": "REQUESTED",
  "partnerId": 2,
  "partnerName": "Jane Smith",
  "role": "LEARNER",
  "createdAt": "2024-01-15T10:30:00"
}
```

---

2. Accept/Reject Session (Teacher Only)

PUT `/api/sessions/{sessionId}/status?status=ACCEPTED`

Headers:

```
Authorization: Bearer {token}
```

Status Options: ACCEPTED, REJECTED, COMPLETED, CANCELLED

Response (200 OK):

```json
{
  "sessionId": 1,
  "skillName": "Spring Boot",
  "status": "ACCEPTED",
  "partnerId": 1,
  "partnerName": "John Doe",
  "role": "TEACHER",
  "createdAt": "2024-01-15T10:30:00"
}
```

---

3. Get My Sessions

GET `/api/sessions/my-sessions`

Headers:

```
Authorization: Bearer {token}
```

Response (200 OK):

```json
[
  {
    "sessionId": 1,
    "skillName": "Spring Boot",
    "status": "ACCEPTED",
    "partnerId": 2,
    "partnerName": "Jane Smith",
    "role": "LEARNER",
    "createdAt": "2024-01-15T10:30:00",
    "scheduledTime": null
  },
  {
    "sessionId": 2,
    "skillName": "React",
    "status": "REQUESTED",
    "partnerId": 3,
    "partnerName": "Bob Johnson",
    "role": "TEACHER",
    "createdAt": "2024-01-16T14:20:00",
    "scheduledTime": null
  }
]
```

📝 Common Response Codes

Code	Meaning
200	Success
201	Resource Created
204	No Content (Deleted)
400	Bad Request
401	Unauthorized (Missing/Invalid Token)
403	Forbidden (Not Allowed)
404	Not Found

Code	Meaning
409	Conflict (Already Exists)
429	Too Many Requests (Rate Limited)
500	Server Error

🔑 Authentication Headers Template

For any **authenticated endpoint**, use:

```
Authorization: Bearer {your_token_here}
```

Or let the system use the **JWT cookie** automatically (if set).

🚀 Quick Start Workflow

Step 1: Register & Login

```
POST /register → Create account
POST /login → Get JWT token
```

Step 2: Setup Profile

```
POST /api/profile → Create profile
POST /api/profile/skills → Add skills
```

Step 3: Build Resume

```
POST /api/resume/certifications → Add certs
POST /api/resume/education → Add education
POST /api/resume/experience → Add experience
```

Step 4: Get Qualified

```
POST /api/test/generate → Generate test
POST /api/test/submit → Submit answers
GET /api/test/qualification-status → Check if qualified
```

Step 5: Find Matches

```
GET /api/matches/swap?skillToLearn=X&skillToTeach=Y → Find partners
POST /api/sessions/request → Request session
```

Step 6: Learn & Teach

```
PUT /api/sessions/{id}/status → Accept/Reject sessions
POST /api/feedback/submit → Give feedback after session
```

✅ Validation Rules

Password

- Minimum 8 characters
- At least 1 uppercase letter (A-Z)
- At least 1 lowercase letter (a-z)
- At least 1 digit (0-9)
- At least 1 special character (@, #, $)

Proficiency Levels

- BEGINNER
- INTERMEDIATE
- ADVANCED
- EXPERT

Learning Goals

- JOB_PREP
- CAREER_SWITCH
- PERSONAL_PROJECT
- EXPLORATION

Teaching Motivation

- LOVE_TEACHING
- REINFORCE_KNOWLEDGE
- BUILD_REPUTATION

- NETWORKING

Testing

- Total Questions: 15
- Passing Score: 10/15 (67%)
- Test Duration: 30 minutes
- Supported Skills: Java, Spring Boot

Matching Algorithm Weights

Component	Weight
Skill Level Gap	20%
Goal Alignment	10%
Availability	10%
Time Commitment	5%
Learning Style	5%
Test Score	30%
Certification Score	10%
Reputation Score	10%

Note: Test score is REQUIRED (must pass to teach)