Student Opportunities Portal - API Documentation

Table of Contents

Version: 1.0.0
Base URL: `https://your-backend.onrender.com/api`
Documentation Date: October 28, 2025
Author: Thapelo Ndlovu

Overview

The Student Opportunities Portal API is a comprehensive REST API built with Node.js, Express.js, and MongoDB. It manages student opportunities including bursaries, internships, graduate programs, and learnerships with full user authentication, file uploads via Firebase Storage, and administrative controls.

Key Features

- **JWT Authentication** with role-based access control
- **File Storage** integration with Firebase Storage
- **Advanced Search & Filtering** for opportunities
- **Application Tracking** system
- **Profile Management** with completion tracking
- **Email Newsletter** subscription system
- **Admin Panel** functionality

Authentication

Type: Bearer Token (JWT)
Header: `Authorization: Bearer &lt;token&gt;`
Token Expiry: 7 days (configurable)

The JWT token is returned upon successful login/registration and must be included in the Authorization header for protected routes.

API Endpoints

Authentication Routes

Register New User

```
POST /api/auth/register
```

Access: Public
Description: Create a new user account with comprehensive profile data

Request Body:

```
{
  "firstName": "string (required)",
  "lastName": "string (required)",
  "email": "string (required, unique)",
  "password": "string (required, min 6 chars)",
  "phone": "string (optional)",
  "dateOfBirth": "2000-01-01 (optional)",
  "idNumber": "string (optional)",
  "gender": "Male|Female|Other (optional)",
  "race": "African|Coloured|Indian|White|Other (optional)",
  "address": {
    "street": "string",
    "city": "string",
```

```
    "province": "string",
    "postalCode": "string"
  },
  "education": {
    "institution": "string",
    "qualification": "string",
    "fieldOfStudy": "string",
    "yearOfStudy": 1,
    "graduationYear": 2025,
    "averageMarks": 75
  }
}
```

Responses:

- `201 Created`: User created successfully with JWT token
- `400 Bad Request`: User already exists or validation errors
- `500 Internal Server Error`: Server error

User Login

```
POST /api/auth/login
```

Access: Public
Description: Authenticate user and receive JWT token

Request Body:

```
{
  "email": "string (required)",
  "password": "string (required)"
}
```

Responses:

- `200 OK`: Login successful with user data and token
- `401 Unauthorized`: Invalid email or password
- `400 Bad Request`: Missing email or password

Get User Profile (Basic)

```
GET /api/auth/profile
```

Access: Protected (Bearer Token)
Description: Get current user's basic profile information

Update User Profile (Basic)

```
PUT /api/auth/profile
```

Access: Protected (Bearer Token)
Description: Update current user's basic profile information

Profile Management Routes

Get Complete Profile

```
GET /api/profile/complete
```

Access: Protected (Bearer Token)
Description: Get complete user profile with completion percentage calculation

Response:

```
{
  "_id": "string",
  "firstName": "string",
  "lastName": "string",
  "email": "string",
  "profileCompletion": 75,
  "...additionalFields"
}
```

Get Profile Statistics

```
GET /api/profile/stats
```

Access: Protected (Bearer Token)
Description: Get profile completion stats and missing fields list

Response:

```
{
  "profileCompletion": 75,
  "hasProfilePhoto": true,
  "hasResume": false,
  "hasEducation": true,
  "hasPersonalInfo": true,
  "missingFields": ["Resume", "Phone number"]
}
```

Update Detailed Profile

```
PUT /api/profile
```

Access: Protected (Bearer Token)
Description: Update detailed user profile information

Upload Profile Photo

```
POST /api/profile/photo
```

Access: Protected (Bearer Token)
Content-Type: `multipart/form-data`
Description: Upload user profile photo to Firebase Storage

File Requirements:

- **Formats:** JPEG, PNG, GIF
- **Max Size:** 5MB
- **Field Name:** `profilePhoto`

Delete Profile Photo

```
DELETE /api/profile/photo
```

Access: Protected (Bearer Token)
Description: Remove user's profile photo from Firebase Storage

Upload Resume/CV

```
POST /api/profile/resume
```

Access: Protected (Bearer Token)
Content-Type: `multipart/form-data`
Description: Upload user resume/CV to Firebase Storage

File Requirements:

- **Formats:** PDF, DOC, DOCX
- **Max Size:** 10MB
- **Field Name:** `resume`

Delete Resume/CV

```
DELETE /api/profile/resume
```

Access: Protected (Bearer Token)
Description: Remove user's resume/CV from Firebase Storage

File Upload Routes

Get User Files

```
GET /api/uploads
```

Access: Protected (Bearer Token)
Description: Get user's uploaded files (resume, profile photo, transcripts)

Upload Profile Photo (Alternative)

```
POST /api/uploads/profile-photo
```

Access: Protected (Bearer Token)
Content-Type: `multipart/form-data`

Upload Resume (Alternative)

```
POST /api/uploads/resume
```

Access: Protected (Bearer Token)
Content-Type: `multipart/form-data`

Upload Academic Transcript

```
POST /api/uploads/transcript
```

Access: Protected (Bearer Token)
Content-Type: `multipart/form-data`
Description: Upload academic transcript with optional description

Request Body:

```
{
  "transcript": "file",
```

```
  "description": "High School Transcript (optional)"
}
```

Upload Application Document

```
POST /api/uploads/documents
```

Access: Protected (Bearer Token)
Content-Type: `multipart/form-data`
Description: Upload document for specific opportunity application

Request Body:

```
{
  "document": "file",
  "opportunityId": "string (required)"
}
```

Delete Transcript

```
DELETE /api/uploads/transcript/:transcriptId
```

Access: Protected (Bearer Token)
Description: Delete specific academic transcript

Opportunities Routes

Get All Opportunities

```
GET /api/opportunities
```

Access: Public
Description: Get all active opportunities with filtering and pagination

Query Parameters:

- `category`: Filter by type (bursary, internship, graduate, learnership)
- `search`: Search in title, description, provider
- `page`: Page number (default: 1)
- `limit`: Items per page (default: 10)

Response:

```
{
  "opportunities": [...],
  "totalPages": 5,
  "currentPage": 1,
  "total": 47
}
```

Get Upcoming Deadlines

```
GET /api/opportunities/upcoming-deadlines
```

Access: Public
Description: Get opportunities with deadlines in the next 30 days

Response:

```
{
  "opportunities": [
    {
      "title": "Engineering Bursary 2025",
      "provider": "Company ABC",
      "category": "bursary",
      "applicationDeadline": "2025-11-15T00:00:00.000Z",
      "daysLeft": 18
    }
  ]
}
```

Get Single Opportunity

```
GET /api/opportunities/:id
```

Access: Public
Description: Get detailed opportunity information (increments view count)

Create Opportunity

```
POST /api/opportunities
```

Access: Protected (Admin Only)
Description: Create new opportunity/bursary listing

Request Body:

```
{
  "title": "string (required)",
  "description": "string (required)",
```

```
  "category": "bursary|internship|graduate|learnership (required)",
  "field": "Engineering (required)",
  "provider": "string (required)",
  "eligibility": {
    "minAge": 18,
    "maxAge": 25,
    "requiredEducation": "Matric",
    "requiredFields": ["Engineering", "Science"],
    "minimumAverage": "70%",
    "citizenship": ["South African"],
    "yearOfStudy": [1, 2, 3],
    "otherRequirements": "string"
  },
  "funding": {
    "tuition": "Full tuition covered",
    "accommodation": "R50,000 per year",
    "allowance": "R5,000 per month"
  },
  "applicationDeadline": "2025-12-31T00:00:00.000Z (required)",
  "applicationProcess": "string",
  "applyMethod": {
    "type": "site|redirect",
    "url": "string"
  },
  "documentsRequired": ["CV", "Academic Transcript", "Motivational Letter"],
  "contactInfo": {
    "email": "string",
    "phone": "string",
    "website": "string"
  },
  "location": "string (required)"
}
```

Update Opportunity

```
PUT /api/opportunities/:id
```

Access: Protected (Admin Only)
Description: Update existing opportunity

Delete Opportunity

```
DELETE /api/opportunities/:id
```

Access: Protected (Admin Only)
Description: Remove opportunity from system

Application Routes

Get My Applications

```
GET /api/applications/my-applications
```

Access: Protected (Bearer Token)
Description: Get current user's applications with opportunity details

Get Applications (Admin/User)

```
GET /api/applications
```

Access: Protected (Bearer Token)
Description: Get applications (users see own applications, admins see all)

Query Parameters:

- `status`: Filter by status (Pending, Under Review, Shortlisted, Rejected, Accepted)
- `page`: Page number (default: 1)
- `limit`: Items per page (default: 10)

Get Single Application

```
GET /api/applications/:id
```

Access: Protected (Owner or Admin)
Description: Get detailed application information

Submit Application

```
POST /api/applications
```

Access: Protected (Bearer Token)
Content-Type: `multipart/form-data` (if files included)
Description: Submit application for specific opportunity

Request Body:

```
{
  "opportunityId": "string (required)",
  "answers": [
    {
      "question": "Why are you applying?",
      "answer": "string"
```

```
    }
  ],
  "files": "array of files (optional)"
}
```

Business Rules:

- Users can only apply once per opportunity
- Applications not allowed after deadline
- Validates opportunity exists and is active

Update Application Status

```
PUT /api/applications/:id/status
```

Access: Protected (Admin Only)
Description: Update application review status

Request Body:

```
{
  "status": "Under Review|Shortlisted|Rejected|Accepted"
}
```

User Management Routes (Admin)

Get All Users

```
GET /api/users
```

Access: Protected (Admin Only)
Description: Get paginated list of all users

Query Parameters:

- `search`: Search in firstName, lastName, email
- `page`: Page number (default: 1)
- `limit`: Items per page (default: 10)

Get Single User

```
GET /api/users/:id
```

Access: Protected (Admin Only)
Description: Get specific user details (password excluded)

Get User Applications

```
GET /api/users/:id/applications
```

Access: Protected (Owner or Admin)
Description: Get specific user's applications with opportunity details

Update User

```
PUT /api/users/:id
```

Access: Protected (Admin Only)
Description: Update user information (admin function)

Delete User

```
DELETE /api/users/:id
```

Access: Protected (Admin Only)
Description: Delete user and all their applications

Newsletter Routes

Subscribe to Newsletter

```
POST /api/newsletter/subscribe
```

Access: Public
Description: Add email to newsletter subscription list

Request Body:

```
{
  "email": "user@example.com (required)"
}
```

Responses:

- `201 Created`: New subscription created
- `200 OK`: Resubscribed existing email
- `400 Bad Request`: Email already subscribed or validation error

Unsubscribe from Newsletter

```
POST /api/newsletter/unsubscribe
```

Access: Public
Description: Remove email from newsletter subscriptions

Request Body:

```
{
  "email": "user@example.com (required)"
}
```

Data Models

User Model

Collection: `users`

Core Fields:

- `firstName`, `lastName`: String (required, max 50 chars)
- `email`: String (required, unique, lowercase, validated)
- `password`: String (required, min 6 chars, hashed with bcryptjs)
- `phone`: String (optional)
- `dateOfBirth`: Date
- `idNumber`: String (South African ID)
- `gender`: Enum (Male, Female, Other)
- `race`: Enum (African, Coloured, Indian, White, Other)

Address Object:

```
{
  "street": "string",
  "city": "string",
  "province": "string",
  "postalCode": "string"
}
```

Education Object:

```
{
  "institution": "string",
  "qualification": "string",
  "fieldOfStudy": "string",
  "yearOfStudy": "number",
  "graduationYear": "number",
  "averageMarks": "number"
}
```

File Objects:

- `resume`: Firebase file object
- `profilePhoto`: Firebase file object
- `transcripts`: Array of Firebase file objects

System Fields:

- `isAdmin`: Boolean (default: false)
- `emailVerified`: Boolean (default: false)
- `createdAt`, `updatedAt`: Auto-generated timestamps

Methods:

- `matchPassword(password)`: Compare plain password with hashed password

Opportunity Model

Collection: `opportunities`

Core Fields:

- `title`: String (required)
- `description`: String (required)
- `category`: Enum (bursary, internship, graduate, learnership)
- `field`: String (e.g., "Engineering", "Medicine")
- `provider`: String (organization name)
- `location`: String (required)

Eligibility Object:

```
{
  "minAge": "number",
  "maxAge": "number",
  "requiredEducation": "string",
  "requiredFields": ["array of strings"],
  "minimumAverage": "string (e.g., '70%')",
  "citizenship": ["array of strings"],
```

```
  "yearOfStudy": [1, 2, 3],
  "otherRequirements": "string"
}
```

Funding Object:

```
{
  "tuition": "string",
  "accommodation": "string",
  "allowance": "string"
}
```

Application Method:

```
{
  "type": "site|redirect",
  "url": "string"
}
```

System Fields:

- `applicationDeadline`: Date (required)
- `rating`: Number (0-5)
- `applicationsCount`: Number (default: 0)
- `views`: Number (default: 0)
- `isActive`: Boolean (default: true)
- `applications`: Array of Application ObjectIds

Application Model

Collection: `applications`

Core Fields:

- `applicant`: ObjectId -> User (required)
- `opportunity`: ObjectId -> Opportunity (required)
- `status`: Enum (Pending, Under Review, Shortlisted, Rejected, Accepted)
- `applicationDate`: Date (auto-set)

Answers Array:

```
[
  {
    "question": "string",
    "answer": "string"
```

```
  }
]
```

Documents Array:

```
[
  {
    "name": "string",
    "firebaseName": "string",
    "downloadURL": "string",
    "uploadedAt": "date"
  }
]
```

Unique Index: `applicant + opportunity` (prevents duplicate applications)

Newsletter Model

Collection: `newsletters`

Fields:

- `email`: String (required, unique, lowercase, validated)
- `isSubscribed`: Boolean (default: true)
- `subscribedAt`: Date (auto-set)

File Storage

Service: Firebase Storage
Authentication: Service account credentials

Storage Folders

- `profile-photos/{userId}/`: User profile photos
- `resumes/{userId}/`: User resumes/CVs
- `transcripts/`: Academic transcripts
- `applications/{userId}/{opportunityId}/`: Application documents

File Restrictions

Image Files (Profile Photos):

- **Formats:** JPEG, JPG, PNG, GIF
- **Max Size:** 5MB

Document Files (Resumes, Transcripts, Applications):

- **Formats:** PDF, DOC, DOCX

- **Max Size:** 10MB

File Object Structure

```
{
  "filename": "original-filename.pdf",
  "firebaseName": "unique-firebase-path",
  "downloadURL": "https://firebase-download-url",
  "uploadedAt": "2025-10-28T14:30:00.000Z"
}
```

⚠ Error Codes

Code	Status	Description
200	OK	Request successful
201	Created	Resource created successfully
400	Bad Request	Validation error or malformed request
401	Unauthorized	Invalid credentials or missing auth
403	Forbidden	Insufficient permissions
404	Not Found	Resource doesn't exist
500	Internal Server Error	Server-side error

Common Error Response Format

```
{
  "message": "Error description",
  "error": "Additional error details (development only)"
}
```

Environment Configuration

Required Environment Variables

```
# Database<a></a>
MONGODB_URI=mongodb+srv://...

# JWT Authentication  <a></a>
JWT_SECRET=your-secret-key
JWT_EXPIRE=7d

# Email Service<a></a>
EMAIL_HOST=smtp.gmail.com
EMAIL_PORT=587
```

```
EMAIL_USER=your-email@gmail.com
EMAIL_PASS=your-app-password

# Frontend URL<a></a>
FRONTEND_URL=https://your-frontend.vercel.app

# Firebase Configuration<a></a>
VITE_FIREBASE_API_KEY=...
VITE_FIREBASE_AUTH_DOMAIN=...
VITE_FIREBASE_PROJECT_ID=...
VITE_FIREBASE_STORAGE_BUCKET=...
VITE_FIREBASE_MESSAGING_SENDER_ID=...
VITE_FIREBASE_APP_ID=...
VITE_FIREBASE_MEASUREMENT_ID=...

# Server<a></a>
PORT=5000
```

Performance & Scaling

Database Indexes

- **User:** email (unique)
- **Opportunity:** category + isActive, applicationDeadline, rating
- **Application:** applicant + opportunity (unique compound)

Caching Strategy

- Opportunity listings cached for faster browsing
- User sessions maintained with JWT tokens
- Static file serving via Firebase CDN

Rate Limiting

- API rate limiting implemented to prevent abuse
- File upload size restrictions for storage optimization

Testing

Test Environment Setup

1. Set up test MongoDB database
2. Configure test environment variables
3. Mock Firebase Storage for file uploads
4. Set up test user accounts (regular + admin)

API Testing Checklist

- [ ] Authentication flows (register, login, profile)
- [ ] File upload functionality (all file types)
- [ ] Opportunity CRUD operations
- [ ] Application submission and status updates
- [ ] User management (admin functions)
- [ ] Newsletter subscription/unsubscription
- [ ] Search and filtering functionality
- [ ] Pagination across all list endpoints
- [ ] Authorization checks (admin vs regular user)
- [ ] Error handling and validation

Deployment

Backend Hosting

Platform: Render.com
Build Command: `npm install`
Start Command: `npm start`

Database

Service: MongoDB Atlas
Connection: Via MONGODB_URI environment variable

File Storage

Service: Firebase Storage
Authentication: Service account JSON credentials

Environment Setup

1. Configure all environment variables in Render dashboard
2. Set up MongoDB Atlas whitelist for Render IP addresses
3. Configure Firebase project with proper security rules
4. Set up custom domain (optional)

Support & Contact

Developer: Thapelo Ndlovu
GitHub: @Heisenburg-z
Project: Student Opportunities Portal

API Version History

- **v1.0.0** (October 2025): Initial release with full CRUD functionality
- Firebase Storage integration
- JWT authentication
- Admin role system
- Application tracking
- Profile completion system

Last Updated: October 28, 2025
[1] [2] [3] [4] [5] [6] [7] [8] [9] [10] [11] [12] [13] [14] [15] [16] [17]

⁂

1. newsletter.js
2. userController.js
3. authController.js
4. uploadController.js
5. profileController.js
6. Application.js
7. Opportunity.js
8. User.js
9. Newsletter.js
10. users.js
11. applications.js
12. opportunities.js
13. profile.js
14. uploads.js
15. auth.js
16. applicationController.js
17. opportunityController.js