CEP Lost & Found API Documentation

Base URL

```
http://localhost:8000
```

Table of Contents

Response Format

All endpoints return JSON in this format:

Success Response

json

```
{
  "status": "success",
  "message": "Descriptive success message",
  "data": {
    // Response data here
  }
}
```

Error Response

json

```
{
  "status": "failed",
  "message": "Descriptive error message",
  "data": null
}
```

Authentication Routes

1. Sign Up

Endpoint: `POST /signup`

Description: Register a new user (student or admin)

Request Body:


json

```
{
  "role": "student",  // or "admin"
  "first_name": "John",
  "last_name": "Doe",
  "home_address": "123 Main St",
  "email": "john@example.com",
  "field_of_study": "Computer Science",  // Required for students, N/A for admin
  "year": 2,  // Required for students, 0 for admin
  "password": "secure_password"
}
```

Success Response (200):


json

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

Error Response (400):


json

```
{
  "status": "failed",
  "message": "Email already registered",
  "data": null
}
```

Flow:

1. Check if email exists in database
2. Hash the password
3. Create new user record
4. Return user ID

---

2. Login

Endpoint: `POST /login`

Description: Authenticate user and get user details

Request Body:


json

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

Success Response (200):


json

```
{
 "status": "success",
 "message": "Login successful",
 "data": {
  "user_id": 1,
  "role": "student",
  "email": "john@example.com",
  "first_name": "John",
  "last_name": "Doe"
 }
}
```

Error Response (401):


json

```
{
 "status": "failed",
 "message": "Invalid email or password",
 "data": null
}
```

Flow:

1. Find user by email
2. Verify password hash
3. Return user details

Item Management Routes

3. Add Item (Report Found Item)

Endpoint: `POST /items/add`

Description: Student reports a found item (requires admin approval)

Request Type: `multipart/form-data`

Form Fields:

- `user_id`: int (required)
- `item_name`: string (required)
- `item_description`: string (required)
- `location`: string (required)
- `email`: string (required)
- `item_image`: file (required) - JPG/PNG

Success Response (200):


json

```
{
 "status": "success",
 "message": "Item submitted successfully, awaiting admin approval",
 "data": {
  "item_id": 5
 }
}
```

Error Response (404):


json

```
{
 "status": "failed",
 "message": "User not found",
 "data": null
}
```

Flow:

1. Validate user exists
2. Read and store image as BLOB
3. Create item with status="pending"
4. Return item ID

---

4. Get Pending Items (Admin Only)

Endpoint: `GET /items/pending`

Description: Retrieve all items awaiting admin approval

Success Response (200):


json


```
{
 "status": "success",
 "message": "Pending items retrieved successfully",
 "data": {
  "pending_count": 3,
  "items": [
   {
    "id": 1,
    "item_name": "Red Backpack",
    "item_description": "Nike backpack with laptop compartment",
    "location": "Library Floor 2",
    "email": "finder@example.com",
    "user_id": 2,
    "date": "2025-11-02",
    "status": "pending",
    "found": false
   }
  ]
 }
}
```


Flow:

1. Query all items with status="pending"
2. Return list of items

---

5. Approve Item (Admin Only)

Endpoint: `POST /items/approve/{item_id}`

Description: Admin approves a pending item (makes it visible to all students)

URL Parameter:

- `item_id`: int

Success Response (200):

json


```
{
 "status": "success",
 "message": "Item 1 approved successfully",
 "data": {
  "item_id": 1,
  "status": "approved"
 }
}
```


Error Response (404):

json


```
{
 "status": "failed",
 "message": "Item not found",
 "data": null
}
```


Flow:

1. Find item by ID
2. Update status to "approved"
3. Item now visible to all students

---

6. Reject Item (Admin Only)

Endpoint: `POST /items/reject/{item_id}`

Description: Admin rejects a pending item

URL Parameter:

- `item_id`: int

Success Response (200):

json

```
{
  "status": "success",
  "message": "Item 1 rejected successfully",
  "data": {
    "item_id": 1,
    "status": "rejected"
  }
}
```


Flow:

1. Find item by ID
2. Update status to "rejected"
3. Item hidden from students

---

7. Get Approved Items

Endpoint: `GET /items/approved`

Description: Get all approved items (visible to all students)

Success Response (200):


json

```
{
 "status": "success",
 "message": "Approved items retrieved successfully",
 "data": {
  "approved_count": 5,
  "items": [
   {
    "id": 1,
    "item_name": "Red Backpack",
    "item_description": "Nike backpack",
    "location": "Library Floor 2",
    "email": "finder@example.com",
    "date": "2025-11-02",
    "status": "approved",
    "found": false
   }
  ]
 }
}
```


Flow:

1. Query all items with status="approved"
2. Return list of items

---

8. Get All Items (Admin Only)

Endpoint: `GET /items/all`

Description: Get all items regardless of status

Success Response (200):

json

```
{
  "status": "success",
  "message": "All items retrieved successfully",
  "data": {
    "count": 10,
    "items": [/* all items */]
  }
}
```

Flow:

1. Query all items from database
2. Return complete list

---

9. Get My Items

Endpoint: `GET /my-items/{user_id}`

Description: Get all items reported by a specific user

URL Parameter:

- `user_id`: int

Success Response (200):


json

```
{
  "status": "success",
  "message": "User items retrieved successfully",
  "data": {
    "count": 2,
    "items": [/* user's items */]
  }
}
```

Flow:

1. Query items where user_id matches
2. Return user's items

---

10. View Single Item

Endpoint: `GET /items/{item_id}`

Description: Get details of a specific item

URL Parameter:

- `item_id`: int

Success Response (200):


json


```
{
 "status": "success",
 "message": "Item retrieved successfully",
 "data": {
  "id": 1,
  "item_name": "Red Backpack",
  "item_description": "Nike backpack",
  "location": "Library Floor 2",
  "email": "finder@example.com",
  "date": "2025-11-02",
  "status": "approved",
  "found": false
 }
}
```


Flow:

1. Find item by ID
2. Return item details

---

11. Delete Item

Endpoint: `DELETE /items/{item_id}?user_id={user_id}`

Description: Delete an item (only by owner)

URL Parameters:

- `item_id`: int
- `user_id`: int (query parameter)

Success Response (200):


json

```
{
  "status": "success",
  "message": "Item 1 deleted successfully",
  "data": {
    "item_id": 1
  }
}
```

Error Response (404):


json

```
{
  "status": "failed",
  "message": "Item not found or unauthorized",
  "data": null
}
```

Flow:

1. Verify item belongs to user
2. Delete item from database
3. Return confirmation

---

12. Edit Item

Endpoint: `PUT /items/{item_id}/edit?user_id={user_id}`

Description: Edit item details (only by owner)

URL Parameters:

- `item_id`: int
- `user_id`: int (query parameter)

Request Body:


json

```
{
  "item_name": "Updated Name",
  "item_description": "Updated description",
  "location": "New location"
}
```

Success Response (200):


json

```
{
  "status": "success",
  "message": "Item 1 updated successfully",
  "data": {
    "id": 1,
    "item_name": "Updated Name",
    // ... updated item details
  }
}
```

Flow:

1. Verify item belongs to user
2. Update specified fields
3. Return updated item

---

13. Mark as Found/Returned

Endpoint: `PUT /items/{item_id}/found?user_id={user_id}`

Description: Mark item as returned to owner (only by finder)

URL Parameters:

- `item_id`: int
- `user_id`: int (query parameter)

Success Response (200):


json

```
{
  "status": "success",
  "message": "Item 1 marked as found",
  "data": {
    "id": 1,
    "found": true,
    // ... item details
  }
}
```

Error Response (400):


json

```
{
  "status": "failed",
  "message": "Only approved items can be marked as found",
  "data": null
}
```

Flow:

1. Verify item belongs to user
2. Check item status is "approved"
3. Set found=true
4. Return updated item

Claim Management Routes

14. Create Claim

Endpoint: `POST /claims`

Description: Student claims an item as theirs

Request Body:


json

```
{
  "user_id": 3,
  "item_id": 1,
  "claim_message": "This is my backpack. It has a blue keychain with my initials 'JD' and contains my CS textbook."
}
```


Success Response (200):


json


```
{
  "status": "success",
  "message": "Claim submitted successfully",
  "data": {
    "claim_id": 5
  }
}
```


Error Responses:

Item not found (404):


json


```
{
  "status": "failed",
  "message": "Item not found",
  "data": null
}
```


Item not approved (400):


json

```
{
 "status": "failed",
 "message": "Cannot claim items that are not approved",
 "data": null
}
```


Duplicate claim (400):


json


```
{
 "status": "failed",
 "message": "You have already claimed this item",
 "data": null
}
```


Flow:

1. Verify item exists and is approved
2. Check user hasn't already claimed this item
3. Create claim with status="pending"
4. Return claim ID

---

15. Get Pending Claims (Admin Only)

Endpoint: `GET /claims/pending`

Description: Get all claims awaiting admin review

Success Response (200):


json

```
{
 "status": "success",
 "message": "Pending claims retrieved successfully",
 "data": {
  "count": 3,
  "claims": [
   {
    "id": 1,
    "user_id": 3,
    "item_id": 1,
    "item_name": "Red Backpack",
    "claim_message": "This is my backpack...",
    "status": "pending",
    "created_at": "2025-11-02 10:30:00"
   }
  ]
 }
}
```


Flow:

1. Query all claims with status="pending"
2. Join with item and user tables for details
3. Return list of claims

---

16. Approve Claim (Admin Only)

Endpoint: `PUT /claims/{claim_id}/approve`

Description: Admin approves a claim and sends email to finder

URL Parameter:

- `claim_id`: int

Success Response (200):


json

```
{
  "status": "success",
  "message": "Claim 1 approved and finder notified by email",
  "data": {
    "claim_id": 1,
    "status": "approved"
  }
}
```


Error Response (404):

json


```
{
  "status": "failed",
  "message": "Claim not found",
  "data": null
}
```


Flow:

1. Find claim by ID
2. Update status to "approved"
3. Send email to item finder with:
    - Claimer's name
    - Claim message
    - Contact information
4. Return confirmation

Email Template:

Subject: Your found item '[Item Name]' has been claimed

Hello [Finder Name],

Your found item [Item Name] has been claimed by [Claimer Name].

Message from claimer: [Claim Message]

Please contact the claimer to arrange item return.

Regards,
CEP Lost & Found Team

---

17. Reject Claim (Admin Only)

Endpoint: `PUT /claims/{claim_id}/reject`

Description: Admin rejects a claim

URL Parameter:

- `claim_id`: int

Success Response (200):


json


```
{
 "status": "success",
 "message": "Claim 1 rejected successfully",
 "data": {
  "claim_id": 1,
  "status": "rejected"
 }
}
```


Flow:

1. Find claim by ID
2. Update status to "rejected"
3. Return confirmation

Complete User Flow Examples

Flow 1: Student Finds an Item

1. POST /signup (role: "student")
   → Gets user_id

2. POST /login
   → Confirms account

3. POST /items/add (with image)
   → Item created with status="pending"
   → Gets item_id

4. Admin reviews via GET /items/pending

5. Admin approves via POST /items/approve/{item_id}
   → Item now visible to all students

Flow 2: Student Claims Their Lost Item

1. GET /items/approved
   → Sees list of found items

2. POST /claims
   → Submits claim with proof message
   → Claim status="pending"

3. Admin reviews via GET /claims/pending

4. Admin approves via PUT /claims/{claim_id}/approve
   → Email sent to finder
   → Finder and claimer coordinate return

5. Finder marks item via PUT /items/{item_id}/found
   → Item.found = true

Flow 3: Admin Workflow

1. POST /signup (role: "admin")
   → Gets admin account

2. POST /login
   → Access admin panel

3. GET /items/pending
   → Review reported items

4. POST /items/approve/{item_id} OR POST /items/reject/{item_id}
   → Approve or reject items

5. GET /claims/pending
   → Review ownership claims

6. PUT /claims/{claim_id}/approve OR PUT /claims/{claim_id}/reject
   → Approve or reject claims

Error Codes Summary

Code	Meaning	When It Happens
200	Success	Request processed successfully
400	Bad Request	Invalid data or validation failed
401	Unauthorized	Wrong email/password
404	Not Found	Resource doesn't exist
500	Server Error	Database or system error

Notes

1. **Authentication**: Currently no JWT/token system. User ID is passed in requests.
2. **Image Storage**: Images stored as BLOB in database (not recommended for production).
3. **Email**: Requires email configuration in `email_helper.py`.
4. **Roles**: Only two roles supported - "student" and "admin".
5. **Rate Limiting**: Not implemented.
6. **Pagination**: Not implemented for list endpoints.

Testing with cURL

Sign Up

bash

```
curl -X POST http://localhost:8000/signup \
  -H "Content-Type: application/json" \
  -d '{
    "role": "student",
    "first_name": "John",
    "last_name": "Doe",
    "home_address": "123 Main St",
    "email": "john@example.com",
    "field_of_study": "CS",
    "year": 2,
    "password": "pass123"
  }'
```

Login

bash

```bash
curl -X POST http://localhost:8000/login \
  -H "Content-Type: application/json" \
  -d '{
    "email": "john@example.com",
    "password": "pass123"
  }'
```

Add Item


bash

```bash
curl -X POST http://localhost:8000/items/add \
  -F "user_id=1" \
  -F "item_name=Red Backpack" \
  -F "item_description=Nike backpack" \
  -F "location=Library" \
  -F "email=john@example.com" \
  -F "item_image=@/path/to/image.jpg"
```

API Health Check

Endpoint: `GET /`

Response:


json

```json
{
  "status": "success",
  "message": "CEP Lost & Found API is running!",
  "data": {
    "version": "1.0",
    "endpoints": "/docs"
  }
}
```

Visit `http://localhost:8000/docs` for interactive Swagger documentation.