OpenAPI definition (v0)

Download OpenAPI specification: Download

Borrow History Controller

API endpoints for managing borrowing history of bicycles and classroom keys

Return borrowed item

Marks a borrowed item (bicycle or key) as returned and updates its availability

PATH PARAMETERS

`borrowId` `required`	integer <int64> Example: `1` ID of the borrow record

Responses

— **200** Item returned successfully

— **400** Item already returned

— **404** Borrow record not found

— **500** Internal server error

```
POST   /api/history/return/{borrowId}
```

Response samples

200 | 400 | 404 | 500

Content type

/

Copy

```
Item returned successfully
```

Return item with feedback

Returns a borrowed item (bicycle or key) and optionally provides feedback

PATH PARAMETERS

`borrowId` required	integer <int64> Example: `1` ID of the borrow record

QUERY PARAMETERS

`feedback`	string Example: `feedback=Great condition` Optional feedback text
`conditionDescription`	string Example: `conditionDescription=Minor scratches` Description of item condition
`experienceRating`	integer <int32> Example: `experienceRating=4` Rating from 1 to 5

Responses

— **200** Item returned successfully with feedback

— **400** Invalid feedback data

— **404** Borrow record not found

— **409** Item already returned

POST `/api/history/return-with-feedback/{borrowId}`

Response samples

200 | 400 | 404 | 409

Content type
/

Copy

```
Item returned successfully with feedback
```

Get user's borrowing history

Retrieves complete borrowing history for a specific user

PATH PARAMETERS

`userId` required	integer <int64> Example: 1 ID of the user

Responses

> **200** Successfully retrieved user's borrowing history

— **204** No borrowing history found for user

— **404** User not found

GET /api/history/user/{userId}

Response samples

200 | 204 | 404

Content type

/

Copy


```
[
  {
    "id": 1,
    "student": {
      "id": 1
    },
    "classroomKey": {
      "id": 1,
      "keyNumber": "A101"
    },
```

```
    "borrowTime": "2023-05-15T09:00:00",
    "isReturned": true
  }
]
```


Get user's key history

Retrieves only classroom key borrowing history for a specific user

PATH PARAMETERS

`userId` required	integer <int64> Example: `1` ID of the user

Responses

> **200** Successfully retrieved key history

— **204** No key history found for user

`GET` `/api/history/user/{userId}/classroom-keys`

Response samples

200 | 204

Content type

/

Copy


```
[
  {
    "id": 2,
    "classroomKey": {
      "id": 1,
      "keyNumber": "A101"
    },
    "borrowTime": "2023-05-15T09:00:00",
    "returnTime": "2023-05-15T11:00:00"
  }
]
```

Get user's bicycle history

Retrieves only bicycle borrowing history for a specific user

PATH PARAMETERS

userId required	integer <int64> Example: 1 ID of the user

Responses

> **200** Successfully retrieved bicycle history

— **204** No bicycle history found for user

`GET` `/api/history/user/{userId}/bicycles`

Response samples

200 | **204**

Content type
/

Copy


```
[
  {
    "id": 1,
    "bicycle": {
      "id": 1,
      "qrCode": "bike123"
    },
    "borrowTime": "2023-05-15T10:00:00",
    "returnTime": "2023-05-15T12:30:00"
```

```
    }
]
```

Get user's active key borrowings

Retrieves currently active classroom key borrowings for a user

PATH PARAMETERS

`userId` required	integer <int64> Example: `1` ID of the user

Responses

> **200** Successfully retrieved active keys

— **204** No active key borrowings found

```
GET  /api/history/user/{userId}/active-keys
```

Response samples

200 204

Content type
/

Copy


```
[
  {
    "id": 4,
    "classroomKey": {
      "id": 2,
      "keyNumber": "B202"
    },
    "borrowTime": "2023-05-16T15:00:00",
    "isReturned": false
  }
]
```


Get user's active borrowings

Retrieves all currently active borrowings for a user

PATH PARAMETERS

userId required	integer <int64> Example: 1 ID of the user

Responses

> **200** Successfully retrieved active borrowings

— **204** No active borrowings found

GET /api/history/user/{userId}/active-borrowings

Response samples

200 | 204

Content type

/

Copy


```
[
  {
    "id": 3,
    "bicycle": {
      "id": 2,
      "qrCode": "bike456"
    },
    "borrowTime": "2023-05-16T14:00:00",
    "isReturned": false
  }
]
```

Get user's active bicycle borrowings

Retrieves currently active bicycle borrowings for a user

PATH PARAMETERS

userId required	integer <int64> Example: 1 ID of the user

Responses

> **200** Successfully retrieved active bicycles

— **204** No active bicycle borrowings found

GET /api/history/user/{userId}/active-bicycles

Response samples

200 | 204

Content type

/

Copy


```
[
  {
    "id": 3,
    "bicycle": {
      "id": 2,
      "qrCode": "bike456"
    },
    "borrowTime": "2023-05-16T14:00:00",
    "isReturned": false
  }
]
```


Get all key history

Retrieves complete history of all classroom key borrowings

Responses

> **200** Successfully retrieved key history

— **204** No key history found

GET /api/history/classroom-keys

Response samples

200 **204**

Content type
/

Copy


```
[
  {
    "id": 2,
    "student": {
      "id": 2
    },
    "classroomKey": {
      "id": 1
    },
    "borrowTime": "2023-05-15T09:00:00",
    "returnTime": "2023-05-15T11:00:00"
  }
]
```

Get all bicycle history

Retrieves complete history of all bicycle borrowings

Responses

> **200** Successfully retrieved bicycle history

— **204** No bicycle history found

```
GET  /api/history/bicycles
```

Response samples

200 | 204

Content type

/

Copy


```
[
  {
    "id": 1,
    "student": {
      "id": 1
```

```
    },
    "bicycle": {
      "id": 1
    },
    "borrowTime": "2023-05-15T10:00:00",
    "returnTime": "2023-05-15T12:30:00"
  }
]
```

Get all borrowing history

Retrieves complete borrowing history including both bicycles and classroom keys

Responses

> **200** Successfully retrieved borrowing history

— **204** No borrowing history found

```
GET  /api/history/all
```

Response samples

200 204

Content type

/

Copy


```
[
  {
    "id": 1,
    "student": {
      "id": 1,
      "name": "John Doe"
    },
    "bicycle": {
      "id": 1,
      "qrCode": "bike123"
    },
    "borrowTime": "2023-05-15T10:00:00",
    "isReturned": false
  }
]
```

Bicycle Controller

API endpoints for bicycle management and booking operations

Update bicycle availability

Updates the availability status of a specific bicycle

PATH PARAMETERS

`id` required	integer <int64> Example: `1` ID of the bicycle to update

QUERY PARAMETERS

`available` required	boolean New availability status

Responses

> **200** Availability updated successfully

— **404** Bicycle not found

```
PUT  /api/bicycles/{id}/availability
```

Response samples

200 404

Content type
/

Copy


```
{
  "id": 1,
  "qrCode": "bike123",
  "location": "Main Campus",
  "isAvailable": false
}
```


Book bicycle by ID

Books an available bicycle using its ID for a specific user. Marks the bicycle as unavailable and creates a borrow history record.

PATH PARAMETERS

`bicycleId`
required

integer <int64>
Example: `1`
ID of the bicycle to book

QUERY PARAMETERS

`userId`
required

integer <int64>
Example: `userId=1`
ID of the user booking the bicycle

Responses

— **200** Bicycle booked successfully

— **400** Bicycle is not available

— **404** Bicycle or user not found

— **500** Internal server error

```
POST   /api/bicycles/book/{bicycleId}
```

Response samples

200 | 400 | 500

Content type

/

Copy

```
Bicycle booked successfully
```

Book bicycle by QR code

Books an available bicycle using its QR code for a specific user. Marks the bicycle as unavailable and creates a borrow history record.

QUERY PARAMETERS

`qrCode`
required

string
Example: `qrCode=bike12345`
QR code of the bicycle to book

`userId`
required

integer <int64>
Example: `userId=1`
ID of the user booking the bicycle

Responses

— **200** Bicycle booked successfully

— **400** Bicycle is not available

— **404** Bicycle or user not found

— **500** Internal server error

POST /api/bicycles/book-by-qr

Response samples

200 400 500

Content type
/

Copy

```
Bicycle booked successfully
```

Add new bicycle

Creates a new bicycle record in the system

REQUEST BODY SCHEMA: application/json

Bicycle details to add

id	integer <int64>
qrCode	string
isAvailable	boolean
location	string

available	boolean

Responses

> **200** Bicycle added successfully

— **400** Bicycle with this QR code already exists

POST /api/bicycles/add

Request samples

Payload

Content type
application/json

Copy

```
{
    "qrCode": "newbike123",
    "location": "West Campus",
    "isAvailable": true
}
```

Response samples

200 400

Content type

/

Copy


```
{
  "id": 1,
  "qrCode": "newbike123",
  "location": "West Campus",
  "isAvailable": true
}
```

List available bicycles

Returns a list of all bicycles that are currently available for booking

Responses

> **200** Successfully retrieved available bicycles

GET /api/bicycles/available

Response samples

200

Content type

/

Copy

```
[
  {
    "id": 1,
    "qrCode": "bike123",
    "location": "Main Campus",
    "isAvailable": true
  }
]
```

Find available bicycles at location

Returns available bicycles at the specified location

QUERY PARAMETERS

location
required

string
Example: `location=North Campus`
Location to search for bicycles

Responses

> **200** Successfully retrieved bicycles at location

— **404** No bicycles found at the specified location

`GET` `/api/bicycles/available-at-location`

Response samples

200 **404**

Content type

/

Copy


```
[
  {
    "id": 1,
    "qrCode": "bike123",
    "location": "North Campus",
    "isAvailable": true
  }
]
```

Find all bicycles at location

Returns all bicycles at the specified location regardless of availability

QUERY PARAMETERS

`location`
required

string
Example: `location=North Campus`
Location to search for bicycles

Responses

> **200** Successfully retrieved bicycles at location

— **404** No bicycles found at the specified location

GET /api/bicycles/at-location

Response samples

200 | 404

Content type
/

Copy


```json
[
  {
    "id": 1,
    "qrCode": "bike123",
    "location": "North Campus",
    "isAvailable": false
  }
]
```

List all bicycles

Returns a list of all bicycles regardless of their availability status

Responses

> **200** Successfully retrieved all bicycles

GET /api/bicycles/all

Response samples

200

Content type

/

Copy


```
[
  {
    "id": 1,
    "qrCode": "bike123",
    "location": "Main Campus",
    "isAvailable": false
  }
]
```


Admin Controller

API endpoints for administrative operations

Mark key as borrowed

Update key status to borrowed

PATH PARAMETERS

keyId required	integer <int64> Example: 1 ID of the key to mark as borrowed

Responses

— **200** Key marked as borrowed

— **404** Key not found

— **500** Internal server error

PUT /api/admin/mark-key-borrowed/{keyId}

Response samples

200 | 404 | 500

Content type

/

Copy

```
Key marked as borrowed successfully
```

Mark key as available

Update key status to available

PATH PARAMETERS

keyId required	integer <int64> Example: 1 ID of the key to mark as available

Responses

— **200** Key marked as available

— **400** Invalid key ID

— **404** Key not found

— **500** Internal server error

```
PUT  /api/admin/mark-key-available/{keyId}
```

Response samples

200 400 404 500

Content type

/

Copy

```
Key marked as available successfully
```

Mark bicycle as borrowed

Update bicycle status to borrowed

PATH PARAMETERS

`bicycleId`
required

integer <int64>
Example: 1
ID of the bicycle to mark

Responses

— **200** Bicycle marked as borrowed

— **404** Bicycle not found

— **500** Internal server error

```
PUT   /api/admin/mark-bicycle-borrowed/{bicycleId}
```

Response samples

200 | 404 | 500

Content type
/

Copy

```
Bicycle marked as borrowed successfully
```

Mark bicycle as available

Update bicycle status to available

PATH PARAMETERS

`bicycleId` required	integer <int64> Example: 1 ID of the bicycle to mark

Responses

— **200** Bicycle marked as available

— **404** Bicycle not found

— **500** Internal server error

```
PUT  /api/admin/mark-bicycle-available/{bicycleId}
```

Response samples

200 | 404 | 500

Content type

/

Copy

```
Bicycle marked as available successfully
```

Change user role

Toggle user role between CR and NON_CR

PATH PARAMETERS

userId required	integer <int64> Example: 3 ID of the user to modify

Responses

— **200** Role changed successfully

— **400** Invalid role change

— **404** User not found

PUT /api/admin/change-role/{userId}

Response samples

200 | 400 | 404

Content type
/

Copy

```
User role updated to CR
```

Alternative admin signup

Alternative endpoint for admin registration

REQUEST BODY SCHEMA: application/json

id	integer <int64>
name	string
email	string
userName	string
picture	string
role	string
password	string

Responses

— **200** Admin registered successfully

— **400** Bad request

POST `/api/admin/signup`

Request samples

Payload

Content type
application/json

Copy

```
{
    "id": 0,
    "name": "string",
    "email": "string",
    "userName": "string",
    "picture": "string",
    "role": "string",
    "password": "string"
}
```

Response samples

200 **400**

Content type

/

Copy

```
Admin registered successfully
```

Admin signup

Register a new admin user

REQUEST BODY SCHEMA: application/json

Admin user details

id	integer <int64>
name	string
email	string
userName	string
picture	string
role	string
password	string

Responses

— **200** Admin registered successfully

— **400** Username already in use

POST `/api/admin/signup2`

Request samples

Payload

Content type
application/json

Copy

```
{
    "userName": "newadmin",
    "password": "secure123",
    "email": "admin@example.com"
}
```

Response samples

200 **400**

Content type

/

Copy

```
Admin registered successfully
```

Alternative admin login

Alternative endpoint for admin authentication

REQUEST BODY SCHEMA: application/json

id	integer <int64>
name	string
email	string
userName	string
picture	string
role	string
password	string

Responses

— **200** Login successful

— **401** Unauthorized

POST `/api/admin/login`

Request samples

Payload

Content type
application/json

Copy

```
{
    "id": 0,
    "name": "string",
    "email": "string",
    "userName": "string",
    "picture": "string",
    "role": "string",
    "password": "string"
}
```

Response samples

200 401

Content type
/

Copy

```
Login successful
```

Admin login

Authenticate admin user

REQUEST BODY SCHEMA: application/json

Login credentials

property name*
additional property — string

Responses

— **200** Login successful

— **401** Invalid credentials

POST /api/admin/login2

Request samples

Payload

Content type
application/json

Copy

```
{
    "userName": "admin",
    "password": "password123"
}
```

Response samples

200 **401**

Content type
/

Copy

```
{
  "message": "Login successful",
  "role": "ADMIN"
}
```

Book bicycle

Book a bicycle for a user

PATH PARAMETERS

`bicycleId`
required

integer <int64>
Example: `1`
ID of the bicycle to book

QUERY PARAMETERS

`userId`
required

integer <int64>
Example: `userId=3`
ID of the user booking the bicycle

Responses

— **200** Bicycle booked successfully

— **400** Bad request

— **404** Bicycle not found

POST /api/admin/bookbicycle/{bicycleId}

Response samples

200 400 404

Content type
/

Copy

```
Bicycle booked successfully by user ID: 3
```

Add classroom key

Add a new classroom key to the system

REQUEST BODY SCHEMA: application/json

Classroom key details

id	integer <int64>
classroomName	string
blockName	string
isAvailable	integer <int32>

floor string

Responses

— **200** Classroom added successfully

— **400** Classroom name cannot be empty

— **409** Classroom already exists

POST /api/admin/addclassrooms

Request samples

Payload

Content type
application/json

Copy

```
{
    "classroomName": "A101",
    "blockName": "Block A",
    "isAvailable": 1
}
```

Response samples

200 | 400 | 409

Content type

/

Copy

```
Classroom added successfully
```

Add bicycle

Add a new bicycle to the system

REQUEST BODY SCHEMA: application/json

Bicycle details

id	integer <int64>
qrCode	string
isAvailable	boolean
location	string
available	boolean

Responses

> **201** Bicycle added successfully

— **409** Bicycle with QR code already exists

— **500** Internal server error

POST `/api/admin/addbicycle`

Request samples

Payload

Content type
application/json

Copy

```
{
    "qrCode": "BIKE123",
    "available": true
}
```

Response samples

201 **409** **500**

Content type

/

Copy


```
{
  "id": 1,
  "qrCode": "BIKE123",
  "available": true
}
```

Get recently added keys

Get keys added most recently

Responses

> **200** Successfully retrieved recent keys

```
GET   /api/admin/recently-added-keys
```

Response samples

200

Content type

/

Copy


```
[
  {
    "id": 5,
    "classroomName": "B202",
    "blockName": "Block B",
    "isAvailable": 1
  }
]
```


Get recently added bicycles

Get bicycles added most recently

Responses

> **200** Successfully retrieved recent bicycles

GET /api/admin/recently-added-bicycle

Response samples

200

Content type
/

Copy


```
[
  {
    "id": 5,
    "qrCode": "BIKE555",
    "available": true
  }
]
```

Get key history

Get borrowing history for all classroom keys

Responses

> **200** Successfully retrieved key history

`GET` `/api/admin/key-history`

Response samples

200

Content type
/

Copy


```
[
  {
    "id": 1,
    "userId": 3,
    "classroomKeyId": 1,
    "borrowTime": "2023-05-15T10:30:00"
  }
]
```

Get borrowed keys

Get currently borrowed classroom keys

Responses

> **200** Successfully retrieved borrowed keys

```
GET  /api/admin/borrowed-keys
```

Response samples

200

Content type
/

Copy


```
[
  {
    "id": 1,
    "userId": 3,
    "classroomKeyId": 1,
    "borrowTime": "2023-05-15T10:30:00"
  }
]
```

Get borrowed bicycles

Get currently borrowed bicycles

Responses

> **200** Successfully retrieved borrowed bicycles

`GET` `/api/admin/borrowed-bicycles`

Response samples

200

Content type
/

Copy


```
[
  {
    "id": 1,
    "userId": 3,
    "bicycleId": 1,
    "borrowTime": "2023-05-15T10:30:00"
  }
]
```

Get bicycle history

Get borrowing history for all bicycles

Responses

> **200** Successfully retrieved bicycle history

`GET` `/api/admin/bicycle-history`

Response samples

200

Content type

/

Copy


```
[
  {
    "id": 1,
    "userId": 3,
    "bicycleId": 1,
    "borrowTime": "2023-05-15T10:30:00"
```

```
  }
]
```

List available keys

Get all currently available classroom keys

Responses

> **200** Successfully retrieved available keys

```
GET  /api/admin/available-keys
```

Response samples

200

Content type

/

Copy

```
[
  {
```

```
    "id": 1,
    "classroomName": "A101",
    "blockName": "Block A",
    "isAvailable": 1
  }
]
```

List available bicycles

Get all currently available bicycles

Responses

> **200** Successfully retrieved available bicycles

`GET` `/api/admin/available-bicycles`

Response samples

200

Content type

/

Copy


```
[
  {
    "id": 1,
    "qrCode": "BIKE123",
    "available": true
  }
]
```

Get current authenticated user

Returns details of currently logged in user

Responses

> **200** Successfully retrieved user

> **401** Unauthorized

`GET` `/api/admin/api/user`

Response samples

200

Content type

/

Copy


```
{
  "id": 1,
  "userName": "admin",
  "email": "admin@example.com",
  "role": "ADMIN"
}
```

List all users

Get all users in the system

Responses

> **200** Successfully retrieved all users

GET /api/admin/all-users

Response samples

200

Content type

/

Copy


```
[
  {
    "id": 1,
    "userName": "admin",
    "email": "admin@example.com",
    "role": "ADMIN"
  }
]
```

List all keys

Get all classroom keys regardless of availability

Responses

> **200** Successfully retrieved all keys

GET /api/admin/all-keys

Response samples

200

Content type
/

Copy


```
[
  {
    "id": 1,
    "classroomName": "A101",
    "blockName": "Block A",
    "isAvailable": 0
  }
]
```

List all bicycles

Get all bicycles regardless of availability

Responses

> **200** Successfully retrieved all bicycles

GET /api/admin/all-bicycles

Response samples

200

Content type
/

Copy

```
[
  {
    "id": 1,
    "qrCode": "BIKE123",
    "available": false
  }
]
```

Delete user

Remove a user from the system

PATH PARAMETERS

userId required	integer <int64> Example: 3 ID of the user to delete

Responses

— **200** User deleted successfully

— **500** Internal server error

DELETE /api/admin/delete-user/{userId}

Response samples

200 500

Content type
/

Copy

```
User deleted successfully
```

Delete classroom key

Remove a key from the system

PATH PARAMETERS

`keyId` `required`	integer <int64> Example: `1` ID of the key to delete

Responses

— **200** Key deleted successfully

— **400** Invalid key ID

— **404** Key not found

— **500** Internal server error

`DELETE` `/api/admin/delete-key/{keyId}`

Response samples

200 400 404 500

Content type

/

Copy

```
Key deleted successfully
```

Delete bicycle

Remove a bicycle from the system

PATH PARAMETERS

`bicycleId` required	integer <int64> Example: 1 ID of the bicycle to delete

Responses

— **200** Bicycle deleted successfully

— **404** Bicycle not found

— **500** Internal server error

```
DELETE   /api/admin/delete-bicycle/{bicycleId}
```

Response samples

200 | 404 | 500

Content type
/

Copy

```
Bicycle deleted successfully
```

Student Controller

API endpoints for student operations including key and bicycle management

Student signup

Registers a new student account

REQUEST BODY SCHEMA: application/json

Student details to register

id	integer <int64>
name	string
email	string
userName	string
picture	string
role	string
password	string

Responses

— **200** Student registered successfully

— **400** Registration failed

POST /api/student/signup

Request samples

Payload

Content type
application/json

Copy

```
{
    "name": "John Doe",
    "email": "john@example.com",
    "password": "secure123"
}
```

Response samples

200 **400**

Content type
/

Copy

```
Student registered successfully
```

Student login

Authenticates a student and returns a token

REQUEST BODY SCHEMA: application/json

Student login credentials

id	integer <int64>
name	string
email	string
userName	string
picture	string
role	string
password	string

Responses

— **200** Login successful

— **401** Unauthorized

POST /api/student/login

Request samples

Payload

Content type
application/json

Copy

```
{
    "email": "john@example.com",
    "password": "secure123"
}
```

Response samples

200 **401**

Content type
/

Copy

```
Login successful
```

Book bicycle

Books an available bicycle for a student

PATH PARAMETERS

`bicycleId`	integer <int64>

required

Example: `1`

ID of the bicycle to book

QUERY PARAMETERS

userId
required

integer <int64>

Example: `userId=1`

ID of the user booking the bicycle

Responses

— **200** Bicycle booked successfully

— **400** Booking failed

POST /api/student/bookbicycle/{bicycleId}

Response samples

200

Content type

/

Copy

```
Bicycle booked successfully
```

Book classroom key

Books a classroom key for a student

PATH PARAMETERS

`keyId` required	integer <int64> Example: `1` ID of the key to book

QUERY PARAMETERS

`userId` required	integer <int64> Example: `userId=1` ID of the user booking the key

Responses

— **200** Key booked successfully

— **400** Booking failed

POST /api/student/book-classroom-key/{keyId}

Response samples

200

Content type
/

Copy

```
Key successfully booked by John Doe
```

Get user by ID

Retrieves student details by user ID

PATH PARAMETERS

userId required	integer <int64> Example: 1 ID of the user to retrieve

Responses

> **200** Successfully retrieved user

— **404** User not found

```
GET  /api/student/{userId}
```

Response samples

200 | 404

Content type
/

Copy

```
{
  "id": 1,
  "name": "John Doe",
  "email": "john@example.com"
}
```

Check key availability

Checks the availability status of a specific classroom key

PATH PARAMETERS

keyId
required

integer <int64>
Example: 1
ID of the key to check

Responses

— **200** Key availability status

— **400** Key not found

GET /api/student/check-key-availability/{keyId}

Response samples

200 400

Content type
/

Copy

```
1
```

Get all available rooms

Retrieves list of all available classroom keys with additional details

Responses

> **200** Successfully retrieved available rooms

`GET` `/api/student/available-rooms`

Response samples

200

Content type

/

Copy


```
[
  {
    "classroomName": "A101",
    "blockName": "A",
    "floor": "1",
    "status": "Available"
```

```
    }
]
```

Get available rooms by block and floor

Retrieves list of available classroom keys for a specific block and floor

PATH PARAMETERS

`blockName` required	string Example: `A` Name of the building block
`floor` required	string Example: `1` Floor number

Responses

> **200** Successfully retrieved available rooms

— **204** No available rooms found

GET /api/student/available-rooms/{blockName}/{floor}

Response samples

200 | 204

Content type
/

Copy


```
[
  {
    "id": 1,
    "classroomName": "A101",
    "blockName": "A",
    "floor": "1",
    "isAvailable": 1
  }
]
```

List available bicycles

Retrieves list of all available bicycles

Responses

> **200** Successfully retrieved available bicycles

GET /api/student/available-bicycles

Response samples

200

Content type

/

Copy


```
[
  {
    "id": 1,
    "qrCode": "bike123",
    "location": "Main Gate",
    "isAvailable": true
  }
]
```

Get all keys by block and floor

Retrieves all classroom keys for a specific block and floor regardless of availability

PATH PARAMETERS

`blockName` required	string Example: `A` Name of the building block
`floor` required	string Example: `1` Floor number

Responses

> **200** Successfully retrieved keys

— **204** No keys found

GET `/api/student/all-keys/{blockName}/{floor}`

Response samples

200 **204**

Content type

/

Copy

```
[
  {
    "id": 1,
    "classroomName": "A101",
    "blockName": "A",
    "floor": "1",
    "isAvailable": 0
  }
]
```

List all bicycles

Retrieves list of all bicycles regardless of availability

Responses

> **200** Successfully retrieved all bicycles

```
GET  /api/student/all-bicycles
```

Response samples

200

Content type

/

Copy


```
[
  {
    "id": 1,
    "qrCode": "bike123",
    "location": "Main Gate",
    "isAvailable": false
  }
]
```


Feedback Controller

API endpoints for managing and retrieving feedback on bicycle usage

Get feedback provided by a user

Retrieves paginated feedback entries provided by a specific user

PATH PARAMETERS

`userId`	integer <int64>

required

Example: 1

ID of the user

QUERY PARAMETERS

pageable >
required

object (Pageable)

Pagination and sorting parameters

Responses

> **200** Successfully retrieved user feedback

— **204** No feedback found for user

— **404** User not found

GET /api/feedback/user/{userId}

Response samples

200 | 204 | 404

Content type

/

Copy

```
{
  "content": [
    {
      "id": 1,
      "conditionDescription": "Brakes were squeaky"
    }
  ],
  "pageable": {
    "sort": {
      "sorted": false
    }
  }
}
```


Get feedback for specific borrow record

Retrieves feedback for a specific bicycle borrowing instance

PATH PARAMETERS

`borrowId` required	integer <int64> Example: `1` ID of the borrow record

Responses

> **200** Successfully retrieved feedback

— **404** Borrow record not found or no feedback available

GET /api/feedback/borrow/{borrowId}

Response samples

200 | 404

Content type
/

Copy

```
{
  "id": 1,
  "conditionDescription": "Chain needed lubrication",
  "experienceRating": 3,
  "feedback": "Ride was okay but maintenance needed"
}
```

Get all bicycle feedback

Retrieves paginated feedback for all bicycles in the system

QUERY PARAMETERS

pageable >
required

object (Pageable)
Pagination and sorting parameters

Responses

> **200** Successfully retrieved bicycle feedback

— **204** No feedback available

GET /api/feedback/bicycles

Response samples

200 | 204

Content type
/

Copy


```
{
  "content": [
    {
      "id": 1,
      "conditionDescription": "Tires needed air"
    }
```

```
  ],
  "pageable": {
    "sort": {
      "sorted": false
    }
  }
}
```

Get bicycle feedback summary

Retrieves aggregated statistics about bicycle feedback

Responses

— **200** Successfully retrieved feedback summary

— **204** No feedback available for summary

```
GET  /api/feedback/bicycles/summary
```

Response samples

200 | 204

Content type

/

Copy


```
{
  "totalFeedback": 15,
  "averageRating": 4.2,
  "commonIssues": [
    "Brakes",
    "Tires"
  ]
}
```


Get feedback for a bicycle

Retrieves all feedback entries for a specific bicycle, ordered by most recent

PATH PARAMETERS

`bicycleId` required	integer <int64> Example: `1` ID of the bicycle

Responses

> **200** Successfully retrieved bicycle feedback

— **404** Bicycle not found or no feedback available

GET /api/feedback/bicycle/{bicycleId}

Response samples

200 | 404

Content type

/

Copy


```
[
  {
    "id": 1,
    "conditionDescription": "Minor scratches",
    "experienceRating": 4,
    "feedback": "Good condition overall"
  }
]
```

Key Request Controller

API endpoints for managing classroom key requests and transfers

Initiate key transfer

Initiates the transfer process for an approved key request

PATH PARAMETERS

`requestId` `required`	integer <int64> Example: `1` ID of the key request

Responses

— **200** Transfer initiated successfully

— **400** Invalid transfer request

PUT /api/key-requests/initiate-transfer/{requestId}

Response samples

200

Content type
/

Copy

```
Transfer initiated. Waiting for requester to accept
```

Decline key request

Declines a pending key request

PATH PARAMETERS

requestId required	integer <int64> Example: 1 ID of the key request

Responses

— **200** Key request declined successfully

— **400** Invalid decline request

```
PUT  /api/key-requests/decline/{requestId}
```

Response samples

200

Content type
/

Copy

```
Key request declined successfully
```

Complete key transfer

Completes the key transfer process initiated by the current holder

PATH PARAMETERS

`requestId` required	integer <int64> Example: `1`

ID of the key request

Responses

— **200** Key transferred successfully

— **400** Invalid transfer completion

`PUT /api/key-requests/complete-transfer/{requestId}`

Response samples

200

Content type
/

Copy

```
Key successfully transferred
```

Cancel key request

Cancels a pending or approved key request

PATH PARAMETERS

`requestId` `required`	integer <int64> Example: `1` ID of the key request

Responses

— **200** Request cancelled successfully

— **400** Request cannot be canceled

— **404** Request not found

— **500** Internal server error

`PUT` `/api/key-requests/cancel/{requestId}`

Response samples

200 | 400 | 404 | 500

Content type

/

Copy

```
{
  "message": "Request cancelled successfully"
}
```

Approve key request

Approves a pending key request

PATH PARAMETERS

`requestId` `required`	integer <int64> Example: `1` ID of the key request

Responses

— **200** Key request approved successfully

— **400** Invalid approval request

```
PUT  /api/key-requests/approve/{requestId}
```

Response samples

200

Content type

/

Copy

```
Key request approved successfully
```

Request a classroom key

Creates a new request for a classroom key with specified time range and purpose

QUERY PARAMETERS

`studentId`
required

integer <int64>
Example: `studentId=1`
ID of the student making the request

`classroomKeyId`
required

integer <int64>
Example: `classroomKeyId=1`
ID of the classroom key being requested

`startTime`
required

string
Example: `startTime=2023-05-20T09:00`
Start time for key usage (format: yyyy-MM-dd HH:mm:ss or yyyy-MM-ddTHH:mm)

`endTime`

string

required

Example: `endTime=2023-05-20T11:00`

End time for key usage (format: yyyy-MM-dd HH:mm:ss or yyyy-MM-ddTHH:mm)

---

purpose
required

string

Example: `purpose=Class presentation`

Purpose for key request

Responses

— **200** Key request submitted successfully

— **400** Invalid request parameters

POST /api/key-requests/request

Response samples

200

Content type

/

Copy

```
Key request submitted successfully
```

Get sent key requests

Retrieves all key requests sent by a user with detailed information

PATH PARAMETERS

`userId` `required`	integer <int64> Example: `1` ID of the user

Responses

> **200** Successfully retrieved sent requests

— **404** User not found

— **500** Internal server error

```
GET  /api/key-requests/sent-requests/{userId}
```

Response samples

200 | 404 | 500

Content type

/

Copy


```
[
  {
    "id": 1,
    "status": "PENDING",
    "classroomKey": {
      "classroomName": "A101"
    },
    "holderAtRequestTime": {
      "name": "John Doe"
    }
  }
]
```


Get key request details

Retrieves detailed information about a classroom key's current status and requests

PATH PARAMETERS

`classroomKeyId` required	integer <int64> Example: `1` ID of the classroom key

Responses

> **200** Successfully retrieved key details

— **400** Classroom key not found

```
GET  /api/key-requests/request-details/{classroomKeyId}
```

Response samples

200 **400**

Content type

/

Copy


```
{
  "keyStatus": "Unavailable",
  "currentHolder": {
    "id": 1,
    "name": "John Doe"
  },
  "requester": {
    "id": 2,
    "name": "Jane Smith"
  }
}
```

Get received key requests

Retrieves all key requests received by a user for keys they currently hold

PATH PARAMETERS

`userId` `required`	integer <int64> Example: `1` ID of the user

Responses

> **200** Successfully retrieved received requests

— **400** User not found

```
GET  /api/key-requests/received-requests/{userId}
```

Response samples

200 | 400

Content type

/

Copy


```
[
  {
    "id": 1,
    "status": "PENDING",
    "student": {
      "name": "John Doe"
    },
    "classroomKey": {
      "classroomName": "A101"
    }
  }
]
```


Check pending requests

Checks if a user has any pending or approved requests for a specific key

PATH PARAMETERS

`userId` required	integer <int64> Example: `1` ID of the user
`keyId` required	integer <int64> Example: `1`

ID of the classroom key

Responses

> **200** Successfully checked pending requests

— **500** Internal server error

```
GET  /api/key-requests/check-pending/{userId}/{keyId}
```

Response samples

200 | 500

Content type
/

Copy


```
{
  "hasPendingRequest": true
}
```

auth-controller

logout

Responses

> **200** OK

POST `/api/logout`

---

getCurrentUser

Responses

> **200** OK

GET /api/user

---

user-controller

signup_2

REQUEST BODY SCHEMA: application/json

id	integer <int64>
name	string
email	string
picture	string
role	string

password	string

Responses

> **200** OK

POST /api/admin/signup1

Request samples

Payload

Content type
application/json

Copy

```
{
    "id": 0,
    "name": "string",
    "email": "string",
    "picture": "string",
    "role": "string",
    "password": "string"
}
```


---

login_2

REQUEST BODY SCHEMA: application/json

id	integer <int64>
name	string
email	string
picture	string
role	string
password	string

Responses

> **200** OK

POST `/api/admin/login3`

Request samples

Payload

Content type
application/json

Copy


```
{
    "id": 0,
    "name": "string",
    "email": "string",
    "picture": "string",
    "role": "string",
    "password": "string"
}
```