Applicant Tracking System

Applicant Tracking System (ATS). This includes:

- Maven dependencies (pom.xml)
- Basic CRUD operations for job applications
- RESTful API using Spring Boot
- H2DB
- Clean structure for easy resume parsing and candidate tracking

Project Overview: ATS-Friendly Spring Boot App

This is a minimal Applicant Tracking System built with Java Spring Boot. It allows recruiters to:

- Add new job applications
- View all applications
- Update application status
- Delete applications

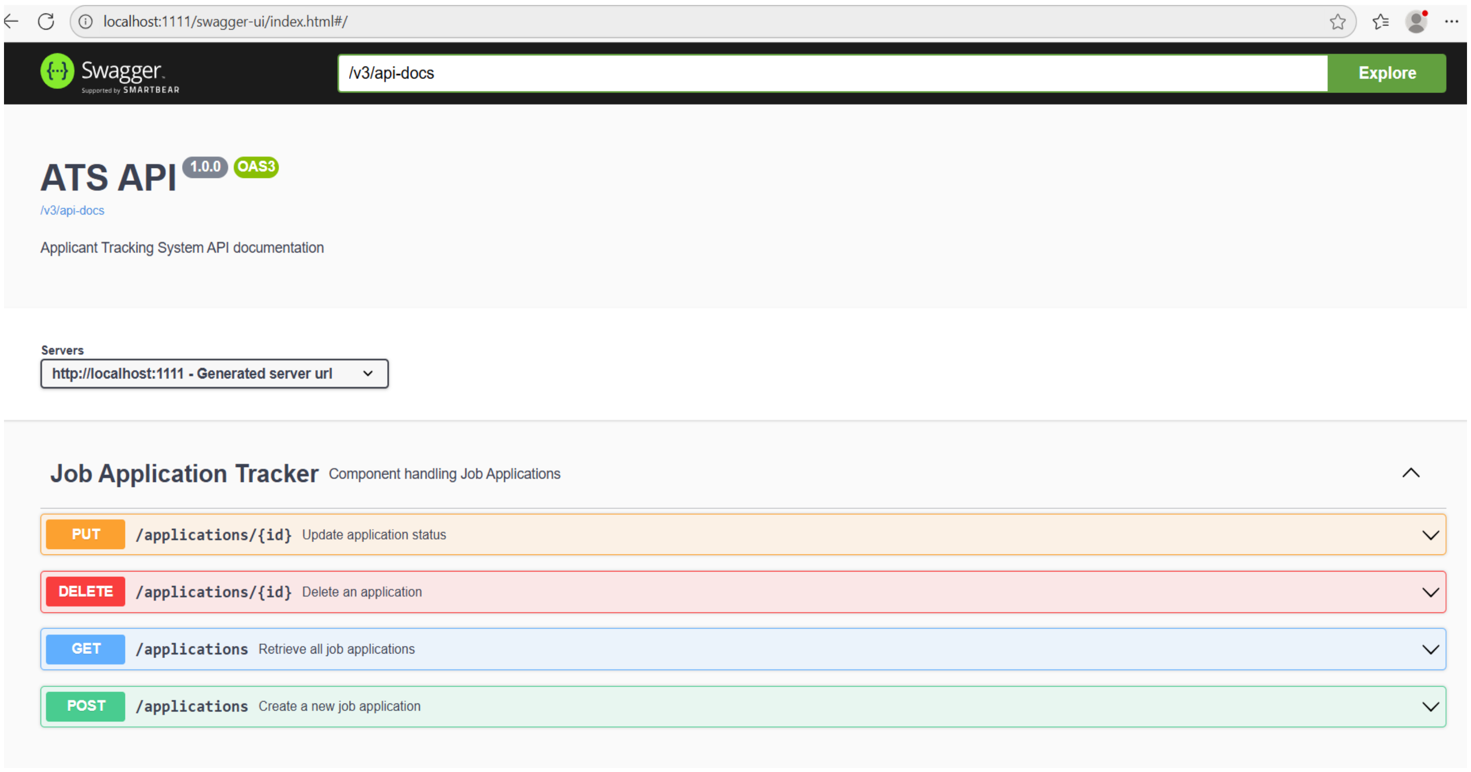
Add Swagger Dependency to `pom.xml`

- Add this to your `<dependencies>` section:
- xml
- `<!-- Springdoc OpenAPI for Swagger UI -->`
- `<dependency>`
- `    <groupId>org.springdoc</groupId>`
- `    <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>`
- `    <version>2.1.0</version>`
- `</dependency>`

http://localhost:1111/swagger-ui/index.html

You'll see a full interactive UI to test endpoints like:

- GET /applications
- POST /applications
- PUT /applications/{id}
- DELETE /applications/{id}



ATS API 1.0.0 OAS3

/v3/api-docs

Applicant Tracking System API documentation

Servers

http://localhost:1111 - Generated server url

Job Application Tracker Component handling Job Applications

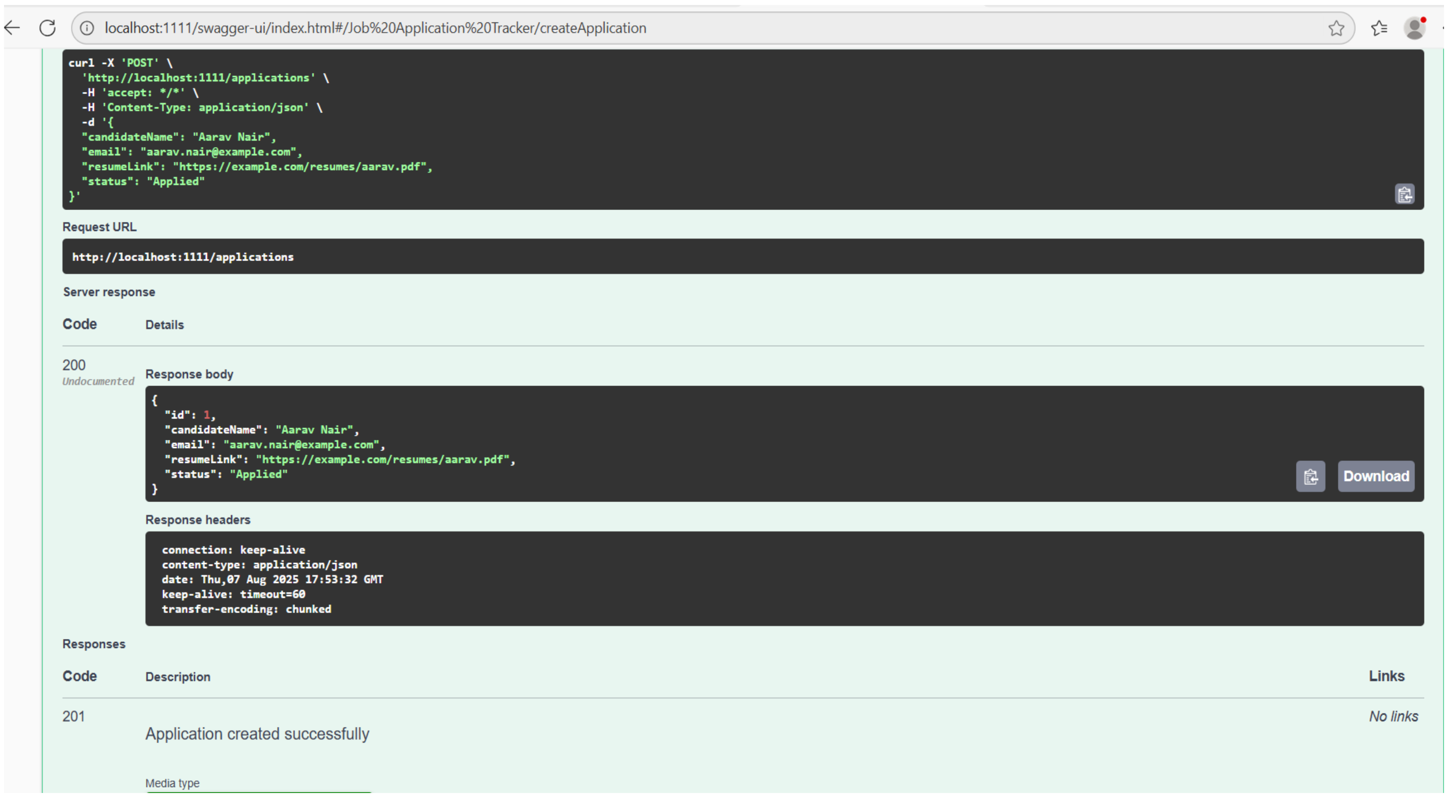
PUT /applications/{id} Update application status

DELETE /applications/{id} Delete an application

GET /applications Retrieve all job applications

POST /applications Create a new job application

Swagger


localhost:1111/swagger-ui/index.html#/Job%20Application%20Tracker/createApplication
Download

```
curl -X 'POST' \
  'http://localhost:1111/applications' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
  "candidateName": "Aarav Nair",
  "email": "aarav.nair@example.com",
  "resumeLink": "https://example.com/resumes/aarav.pdf",
  "status": "Applied"
}'
```

Request URL

```
http://localhost:1111/applications
```

Server response

Code Details

200 *Undocumented* Response body

```
{
  "id": 1,
  "candidateName": "Aarav Nair",
  "email": "aarav.nair@example.com",
  "resumeLink": "https://example.com/resumes/aarav.pdf",
  "status": "Applied"
}
```

Response headers

```
connection: keep-alive
content-type: application/json
date: Thu,07 Aug 2025 17:53:32 GMT
keep-alive: timeout=60
transfer-encoding: chunked
```

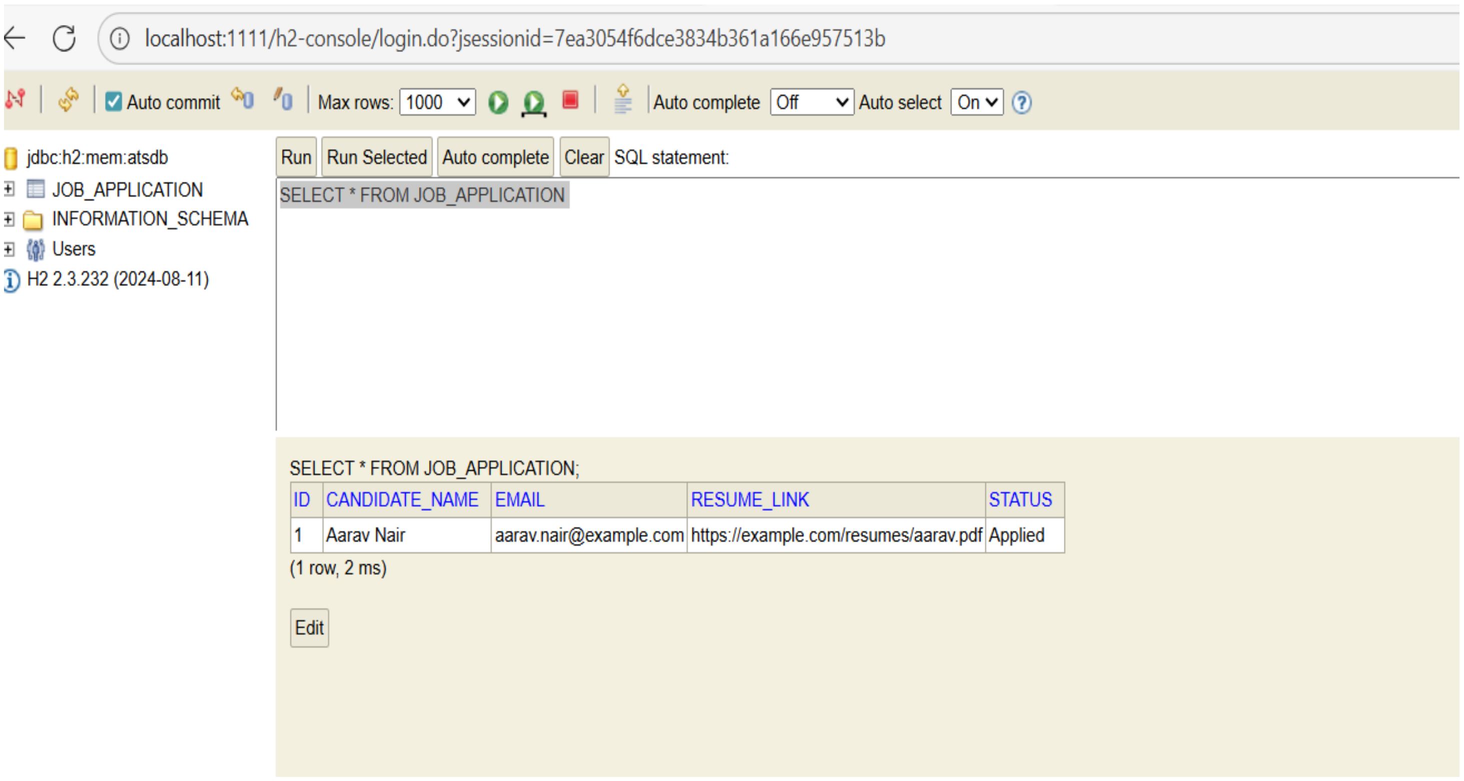
Responses

Code	Description	Links
201	Application created successfully	*No links*

Media type

H2DB

http://localhost:1111/h2-console



```
SELECT * FROM JOB_APPLICATION
```

SELECT * FROM JOB_APPLICATION;

ID	CANDIDATE_NAME	EMAIL	RESUME_LINK	STATUS
1	Aarav Nair	aarav.nair@example.com	https://example.com/resumes/aarav.pdf	Applied

(1 row, 2 ms)

Access H2 Console

Once your app is running, go to:

http://localhost:1111/h2-console

Configure application.properties for H2

# H2 Database Configuration

```
spring.datasource.url=jdbc:h2:mem:atsdb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

spring.jpa.hibernate.ddl-auto=update

spring.h2.console.enabled=true

spring.h2.console.path=/h2-console
```

Use these credentials:

JDBC URL: jdbc:h2:mem:atsdb

Username: sa

Password: (leave blank)

----

sample data and HTTP payloads for your ATS app

using the JobApplication entity. These examples will help you

test your GET, POST, and PUT endpoints using Swagger, Postman, or curl.

# Request Payload

----POST /applications — Create a New Job Application--------

```
{
  "candidateName": "Aarav Nair",
  "email": "aarav.nair@example.com",
  "resumeLink": "https://example.com/resumes/aarav.pdf",
  "status": "Applied"
}
```

---Sample Response--------

```
{
  "id": 1,
  "candidateName": "Aarav Nair",
  "email": "aarav.nair@example.com",
  "resumeLink": "https://example.com/resumes/aarav.pdf",
  "status": "Applied"
}
```

--------GET /applications — Retrieve All Applications----------------------------------

```
[
  {
    "id": 1,
    "candidateName": "Aarav Nair",
    "email": "aarav.nair@example.com",
    "resumeLink": "https://example.com/resumes/aarav.pdf",
    "status": "Applied"
  },
  {
    "id": 2,
    "candidateName": "Meera Thomas",
    "email": "meera.thomas@example.com",
    "resumeLink": "https://example.com/resumes/meera.pdf",
    "status": "Interviewing"
  }
]
```

--------PUT /applications/{id} — Update Application Status----------------------------------

Request Payload (e.g., update status of ID 1)

```
{
  "status": "Interview Scheduled"
}
```

----Sample Response---

```
{
  "id": 1,
  "candidateName": "Aarav Nair",
  "email": "aarav.nair@example.com",
  "resumeLink": "https://example.com/resumes/aarav.pdf",
  "status": "Interview Scheduled"
}
```

DELETE /applications/{id} — Remove an Application

No payload needed. Just send a DELETE request to /applications/1.

------Sample data.sql for H2 Initialization------

------If you want to auto-load sample data when the app starts:--------

INSERT INTO job_application (candidate_name, email, resume_link, status)

VALUES ('Aarav Nair', 'aarav.nair@example.com', 'https://example.com/resumes/aarav.pdf', 'Applied');

INSERT INTO job_application (candidate_name, email, resume_link, status)

VALUES ('Meera Thomas', 'meera.thomas@example.com', 'https://example.com/resumes/meera.pdf', 'Interviewing');

------------------------***--***-------------------- -------***--------------