**Project Documentation:**

***Introduction:***

The project is a Spring Boot-based Rest API for managing a Candidate Voting system.

***Requirements Addressed:***

The API provides functionalities such as entering a candidate, casting votes, counting votes, listing votes, and determining the winner.

It uses local variables instead of a database to store data.

Supports simultaneous execution by multi-users.

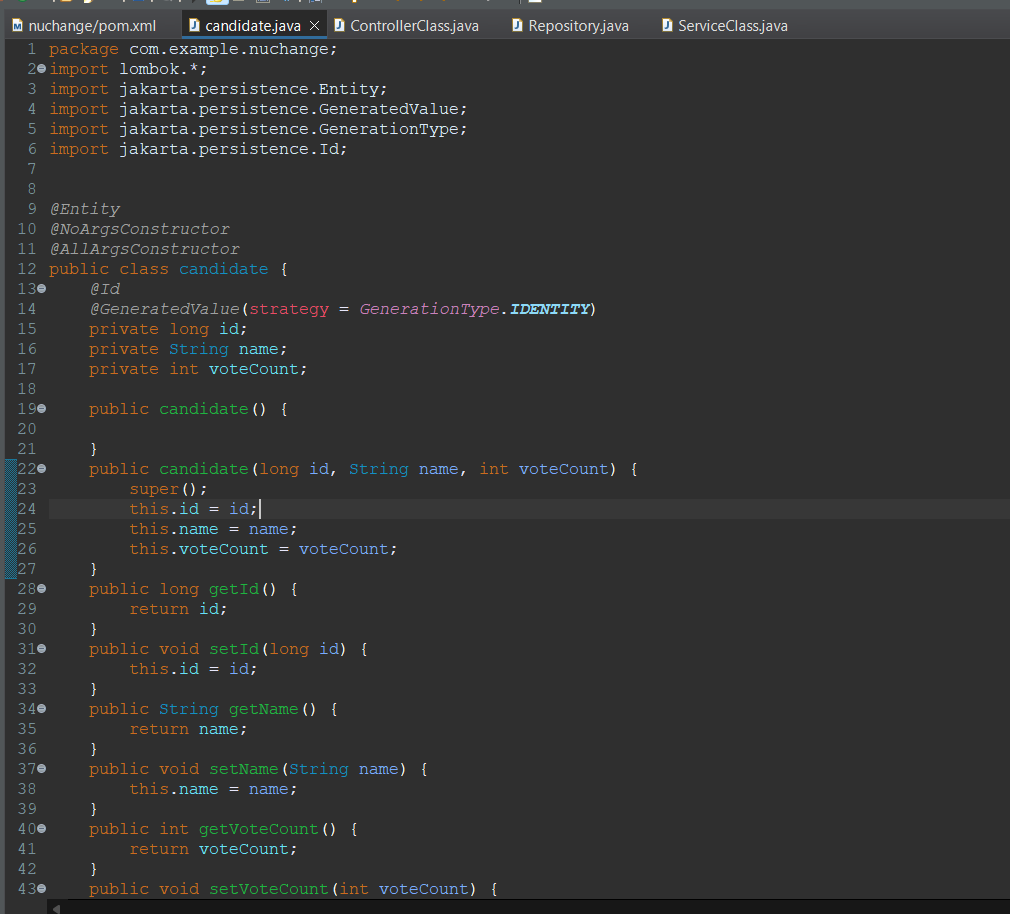
Includes unit testing.

***Implementation***:

**Entity (Candidate):**

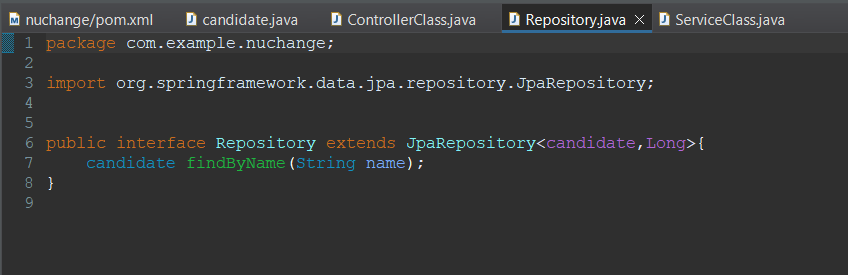
Represents a candidate with properties like id, name, and voteCount.

Utilizes JPA annotations for easy integration with the database.



**Repository (CandidateRepository):**

Extends JpaRepository for CRUD operations on the Candidate entity.

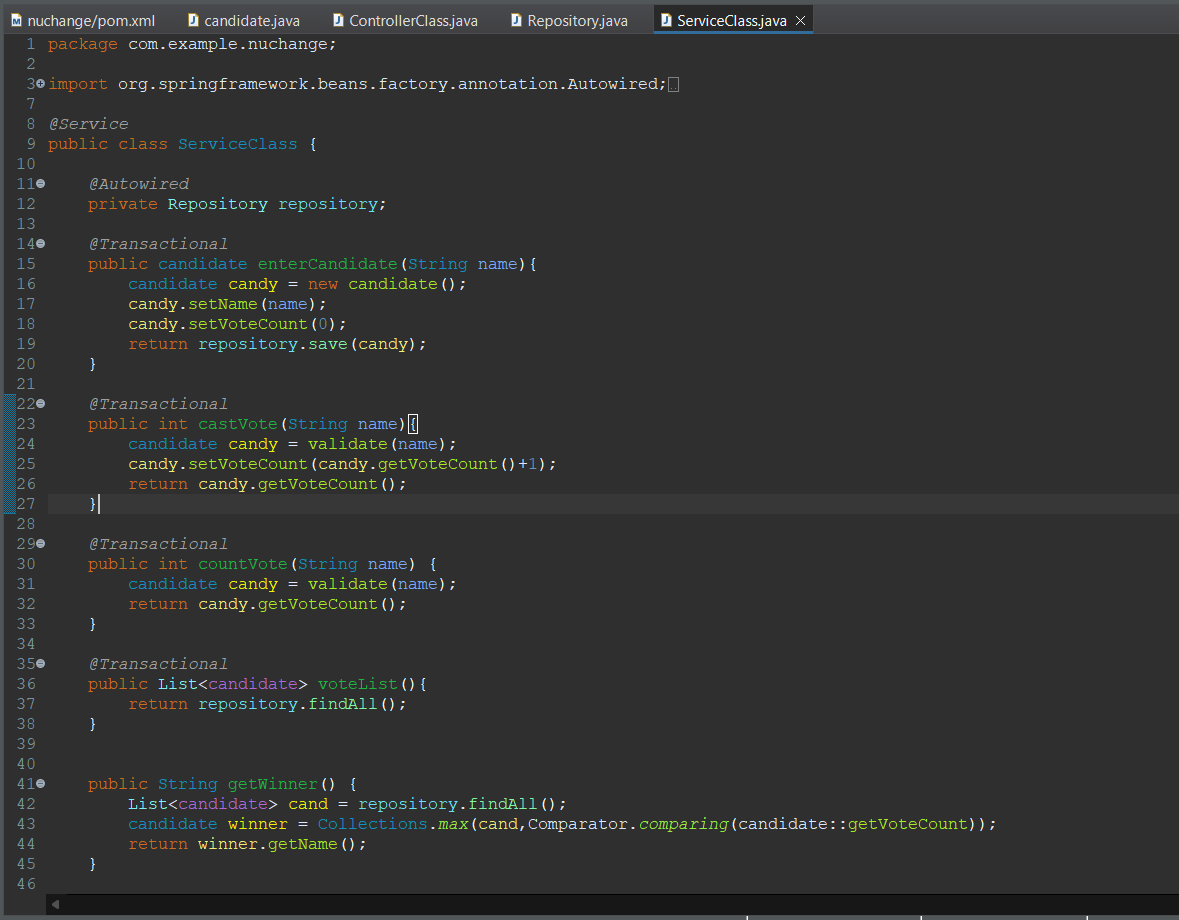


**Service (CandidateService):**

Manages business logic and data manipulation.

Implements methods for entering a candidate, casting votes, counting votes, listing votes, and determining the winner.

Uses transactions to ensure data consistency.



**Controller (CandidateController):**

Handles HTTP requests and interacts with the CandidateService.

Defines endpoints for entering a candidate, casting votes, counting votes, listing votes, and determining the winner.

**Usage Guide:**

***Enter Candidate:***

Endpoint: POST /entercandidate

Parameter: name (Candidate's name)

Returns: Candidate entity.

***Cast Vote:***

Endpoint: POST /castvote

Parameter: name (Candidate's name)

Returns: Updated vote count.

***Count Vote:***

Endpoint: GET /countvote

Parameter: name (Candidate's name)

Returns: Vote count for the specified candidate.

***List Votes:***

Endpoint: GET /listvote

Returns: List of all candidates and their vote counts in JSON format.

***Get Winner:***

Endpoint: GET /getwinner

Returns: Name of the candidate with the highest vote count.

***Architecture:***

Entity-Repository-Service-Controller Pattern:

Follows a layered architecture for separation of concerns.

Entity represents the data model.

Repository handles database interactions.

Service contains business logic.

Controller handles HTTP requests and responses.

***Spring Boot:***

Utilizes Spring Boot for rapid development and easy configuration.

Supports embedded Tomcat for running the application.

JPA (Java Persistence API):

Uses JPA for easy database access and manipulation.

Entities are annotated for mapping to database tables.

***RESTful API:***

Adheres to RESTful principles with clear endpoints for different functionalities.

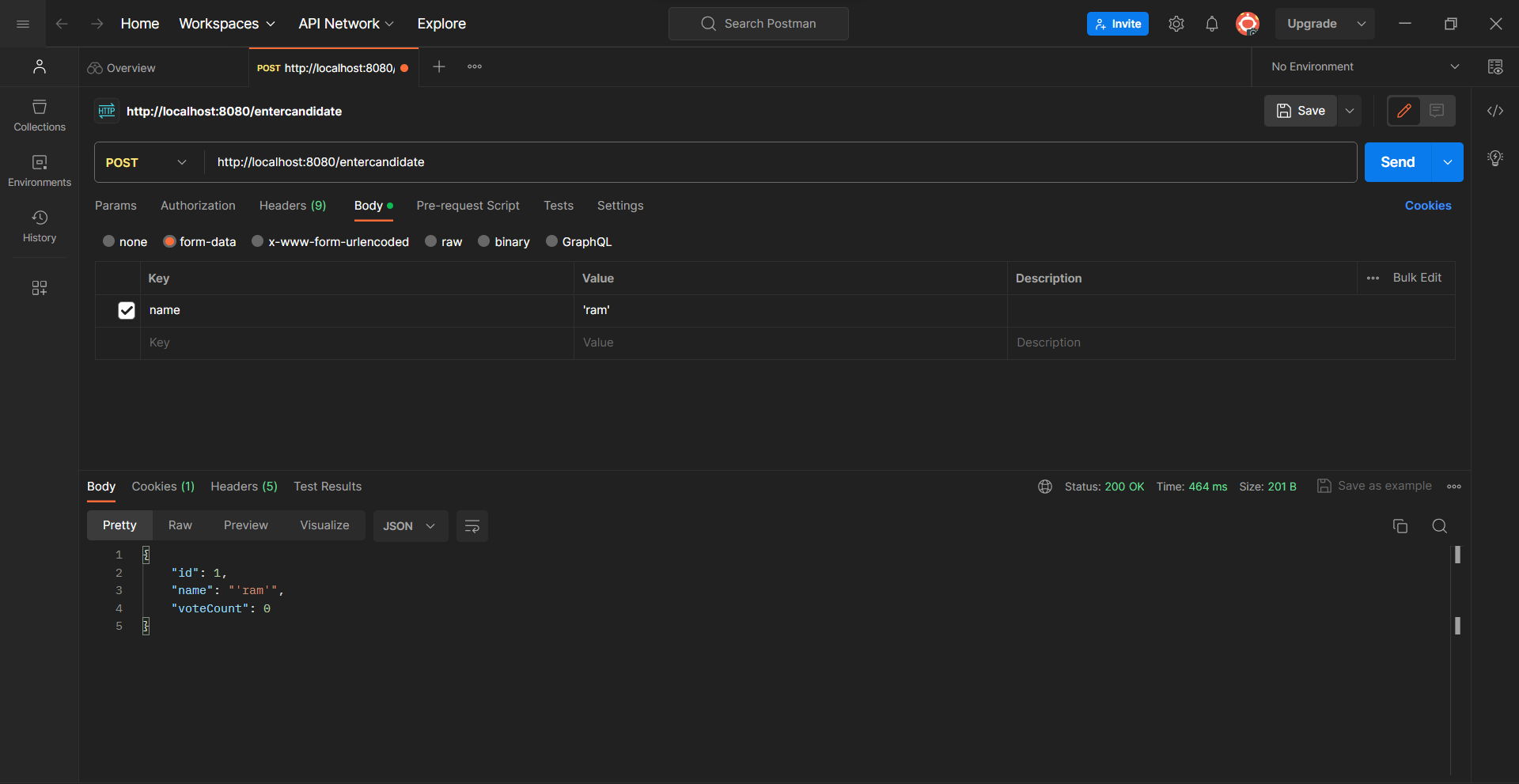
Follows standard HTTP methods (POST, GET).

***Scalability:***

Designed to handle simultaneous execution by multiple users.

Utilizes transactions to maintain data integrity.

***Testing the APIs using the Postman API tool:***



**Database schema:**

