CUSTOMER MANAGEMENT CRUD APPLICATION

STUDENT NAME – VIKAS KUMAR SINGH

UNIVERSITY ROLL NUMBER – 2400291530210

STUDENT ROLL NUMBER - 202401100400210

COURSE – B.TECH CSE-AIML

SECTION – C

GITHUB LINK - https://github.com/VikasSinghKIET/CRUD_Customer_Management.git

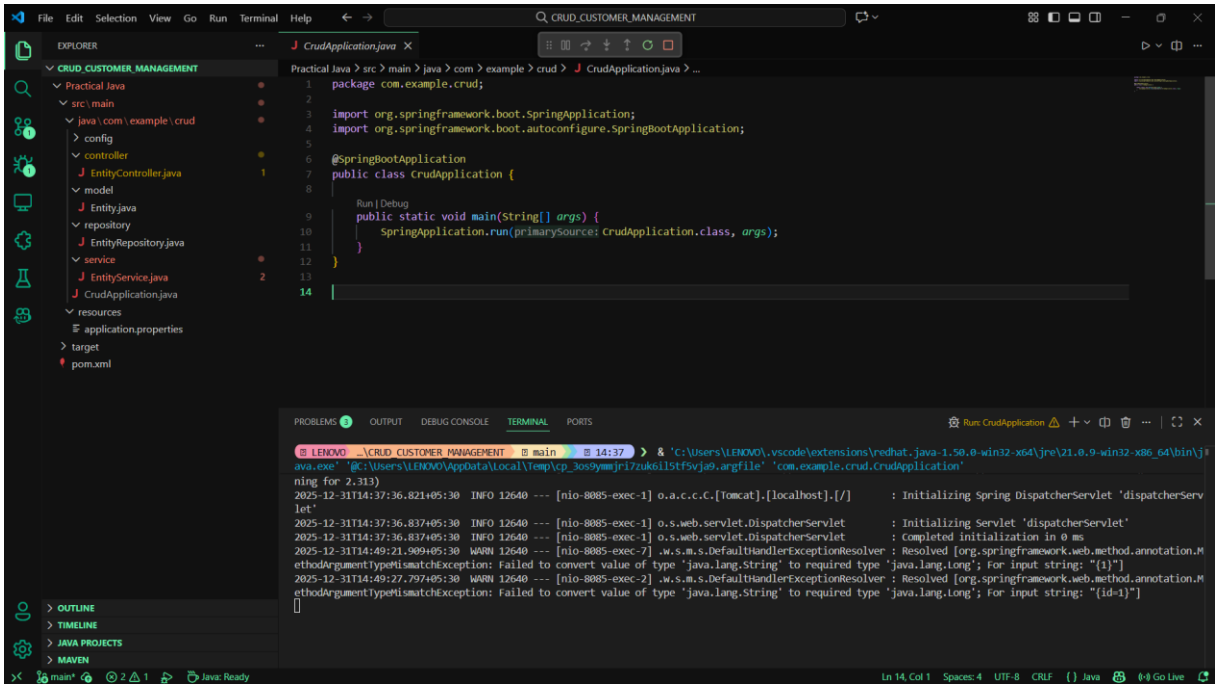
PROJECT DESCRIPTION

This project is a **Customer Management CRUD Application** developed using RESTful API architecture. It provides a backend service to manage customer data, offering features to **Create, Read, Update, and Delete (CRUD)** customer records efficiently. The system allows storing and retrieving essential details such as customer name and email, supporting seamless interaction with client applications.

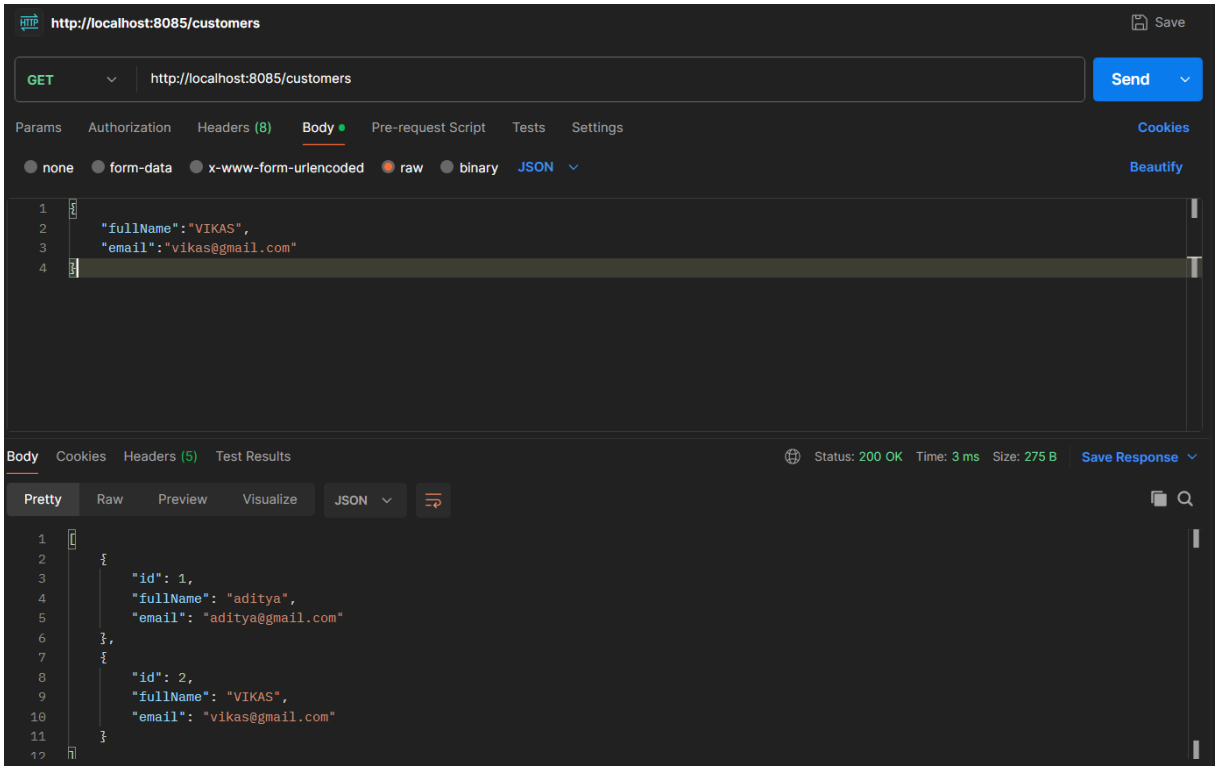
API Endpoints

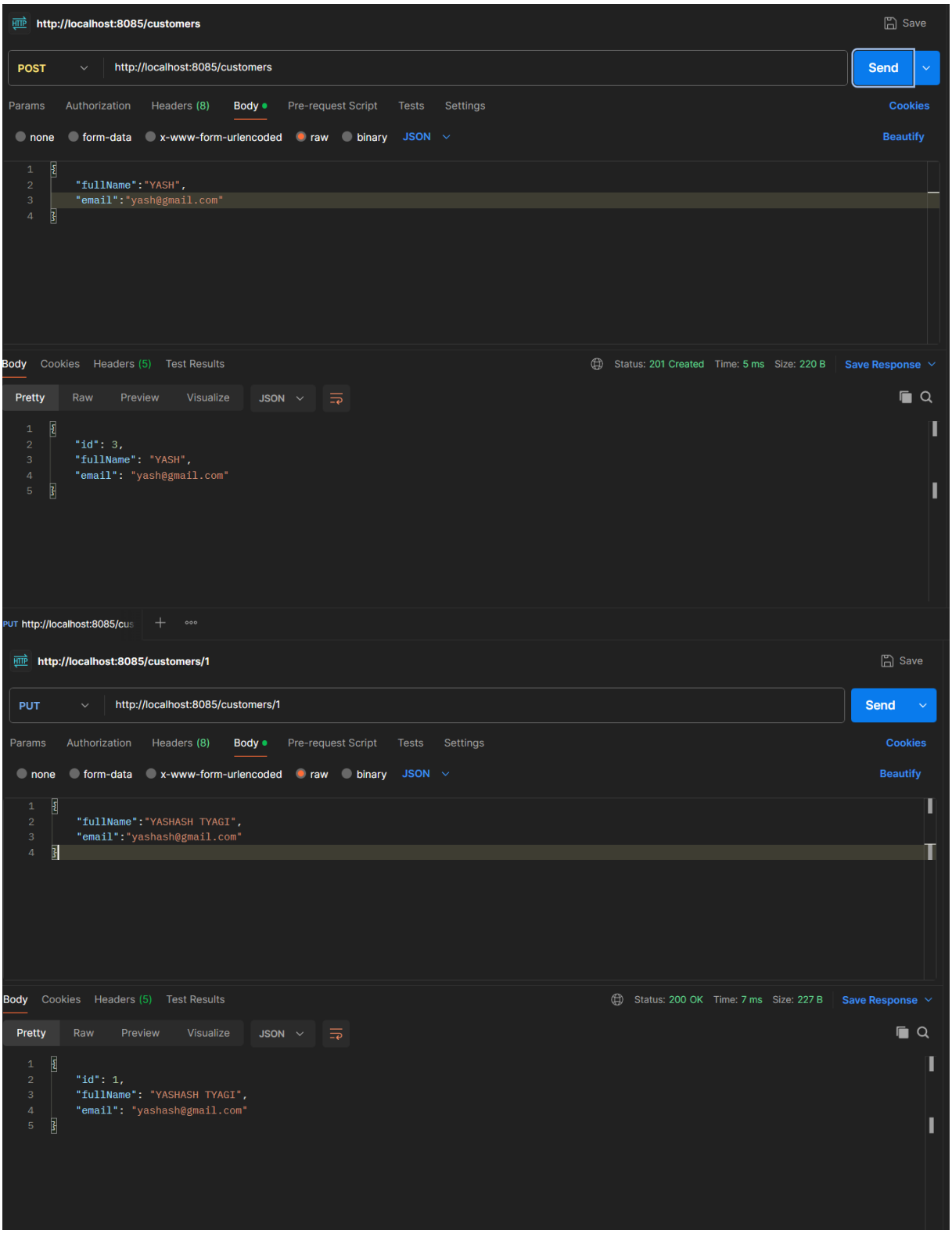
Operation	Endpoint	Description
Create Student	POST /customers	Add a new customer record
Retrieve Students	GET /customers	Fetch all customer details
Update Student	PUT /customers/{id}	Modify existing customer information
Delete Student	DELETE /customers/{id}	Remove customer record by ID

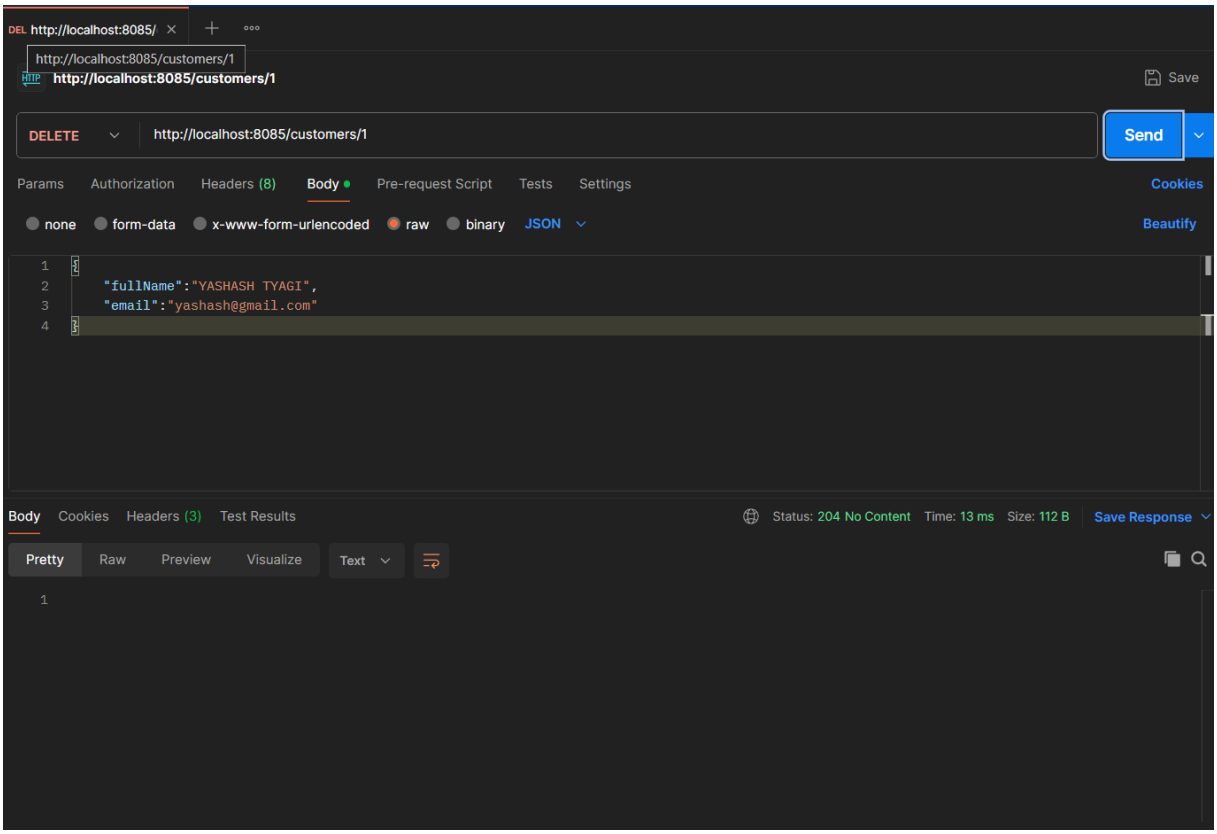
PROJECT STRUCTURE AND RUNNING



API ENDPOINTS




http://localhost:8085/customers
Save
POST
http://localhost:8085/customers
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
JSON
Beautify
{
"fullName":"YASH",
"email":"yash@gmail.com"
}
Body
Cookies
Headers (5)
Test Results
Status: 201 Created
Time: 5 ms
Size: 220 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"id": 3,
"fullName": "YASH",
"email": "yash@gmail.com"
}
PUT http://localhost:8085/cus
http://localhost:8085/customers/1
Save
PUT
http://localhost:8085/customers/1
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
JSON
Beautify
{
"fullName":"YASHASH TYAGI",
"email":"yashash@gmail.com"
}
Body
Cookies
Headers (5)
Test Results
Status: 200 OK
Time: 7 ms
Size: 227 B
Save Response
Pretty
Raw
Preview
Visualize
JSON
{
"id": 1,
"fullName": "YASHASH TYAGI",
"email": "yashash@gmail.com"
}


DEL http://localhost:8085/
http://localhost:8085/customers/1
http://localhost:8085/customers/1
Save
DELETE
http://localhost:8085/customers/1
Send
Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
JSON
Beautify
1
2
3
4
{
"fullName":"YASHASH TYAGI",
"email":"yashash@gmail.com"
}
Body
Cookies
Headers (3)
Test Results
Status: 204 No Content
Time: 13 ms
Size: 112 B
Save Response
Pretty
Raw
Preview
Visualize
Text
1

CODE

CRUD APPLICATION

```
package com.example.crud;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class CrudApplication {

   public static void main(String[] args) {
      SpringApplication.run(CrudApplication.class,  args);
   }
}
```

CONFIG

```
package com.example.crud.config;

import org.springframework.boot.context.properties.ConfigurationProperties;
import org.springframework.context.annotation.Configuration;

@Configuration
@ConfigurationProperties(prefix = "entity")
public class EntityConfig {

   private String entityName = "customer";
   private String field1Name = "fullName";
   private String field2Name = "email";

   public String getEntityName() {
      return entityName;
   }
```

```
public void setEntityName(String entityName) {
    this.entityName = entityName;
```

```
}

    public String getField1Name() {
        return field1Name;
    }

    public void setField1Name(String field1Name) {
        this.field1Name = field1Name;
    }

    public String getField2Name() {
        return field2Name;
    }

    public void setField2Name(String field2Name) {
        this.field2Name = field2Name;
    }

    public String getEntityNamePlural() {
        return entityName + "s";
    }

    public String getEntityNameCapitalized() {
        return entityName.substring(0, 1).toUpperCase() + entityName.substring(1);
    }
}
```

CONTROLLER

```
package com.example.crud.controller;

import com.example.crud.config.EntityConfig;
import com.example.crud.model.Entity;
import com.example.crud.service.EntityService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/${entity.entity-name}s")
public class EntityController {

    @Autowired
    private EntityService service;

    @Autowired
    private EntityConfig config;

    @PostMapping
    public ResponseEntity<Entity> create(@RequestBody Entity entity) {
        Entity created = service.create(entity);
        return new ResponseEntity<>(created, HttpStatus.CREATED);
    }

    @GetMapping
    public ResponseEntity<List<Entity>> getAll() {
        List<Entity> entities = service.getAll();
        return new ResponseEntity<>(entities, HttpStatus.OK);
    }

    @GetMapping("/{id}")
    public ResponseEntity<Entity> getById(@PathVariable Long id) {
        return service.getById(id)
                .map(entity -> new ResponseEntity<>(entity, HttpStatus.OK))
                .orElse(new ResponseEntity<>(HttpStatus.NOT_FOUND));
    }

    @PutMapping("/{id}")
    public ResponseEntity<Entity> update(@PathVariable Long id, @RequestBody Entity entity) {
        try {
```

```
            Entity updated = service.update(id, entity);
            return new ResponseEntity<>(updated, HttpStatus.OK);
        } catch (RuntimeException e) {
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
        }
    }

    @DeleteMapping("/{id}")
    public ResponseEntity<Void> delete(@PathVariable Long id) {
        try {
            service.delete(id);
            return new ResponseEntity<>(HttpStatus.NO_CONTENT);
        } catch (RuntimeException e) {
            return new ResponseEntity<>(HttpStatus.NOT_FOUND);
        }
    }
}
```

MODEL

```
package com.example.crud.model;

public class Entity {

    private Long id;
    private String fullName;
    private String email;

    public Entity() {
    }

    public Entity(String fullName, String email) {
        this.fullName = fullName;
        this.email = email;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getFullName() {
        return fullName;
    }

    public void setFullName(String fullName) {
        this.fullName = fullName;
    }
```

```
    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    @Override
    public String toString() {
        return "Entity{" +
                "id=" + id +
                ", fullName='" + fullName + '\'' +
                ", email='" + email + '\'' +
                '}';
    }
}
```

REPOSITORY

```
package com.example.crud.repository;

import com.example.crud.model.Entity;
import org.springframework.stereotype.Repository;

import java.util.ArrayList;
import java.util.List;
import java.util.Optional;
import java.util.concurrent.atomic.AtomicLong;

@Repository
public class EntityRepository {

    private final List<Entity> entities = new ArrayList<>();
    private final AtomicLong idCounter = new AtomicLong(1);
```

```
    public Entity save(Entity entity) {
        if (entity.getId() == null) {
            entity.setId(idCounter.getAndIncrement());
            entities.add(entity);
        } else {
            Optional<Entity> existing = findById(entity.getId());
            if (existing.isPresent()) {
                int index = entities.indexOf(existing.get());
                entities.set(index, entity);
            } else {
                entities.add(entity);
            }
        }
        return entity;
    }

    public List<Entity> findAll() {
        return new ArrayList<>(entities);
    }

    public Optional<Entity> findById(Long id) {
        return entities.stream()
                .filter(e -> e.getId().equals(id))
                .findFirst();
    }

    public void delete(Entity entity) {
        entities.remove(entity);
    }

    public void deleteById(Long id) {
        entities.removeIf(e -> e.getId().equals(id));
    }
}
```

SERVICE

```
package com.example.crud.service;

import com.example.crud.model.Entity;
import com.example.crud.repository.EntityRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
```

```
import java.util.List;
import java.util.Optional;

@Service
public class EntityService {

    @Autowired
    private EntityRepository repository;

    public Entity create(Entity entity) {
        return repository.save(entity);
    }

    public List<Entity> getAll() {
        return repository.findAll();
    }

    public Optional<Entity> getById(Long id) {
        return repository.findById(id);
    }

    public Entity update(Long id, Entity entityDetails) {
        Entity entity = repository.findById(id)
                .orElseThrow(() -> new RuntimeException("Entity not found with id: " + id));

        entity.setName(entityDetails.getName());
        entity.setField2(entityDetails.getField2());

        return repository.save(entity);
    }

    public void delete(Long id) {
        Entity entity = repository.findById(id)
                .orElseThrow(() -> new RuntimeException("Entity not found with id: " + id));
        repository.delete(entity);
    }
}
```