CORPORATE CLASSIFIEDS

MFPE Project

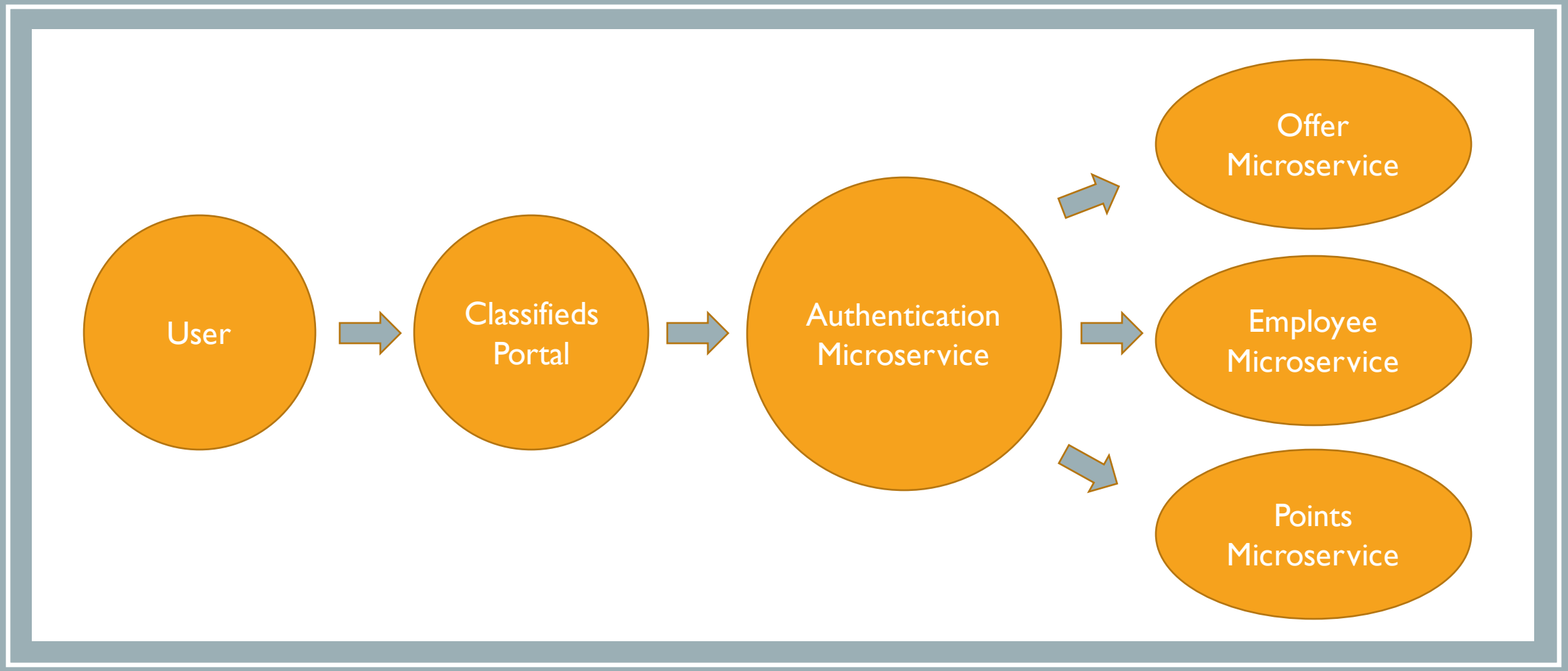
Team

NAVYA VERMA (2135238)

SOBHANA S (2134967)

SASIKUMAR J (2134957)

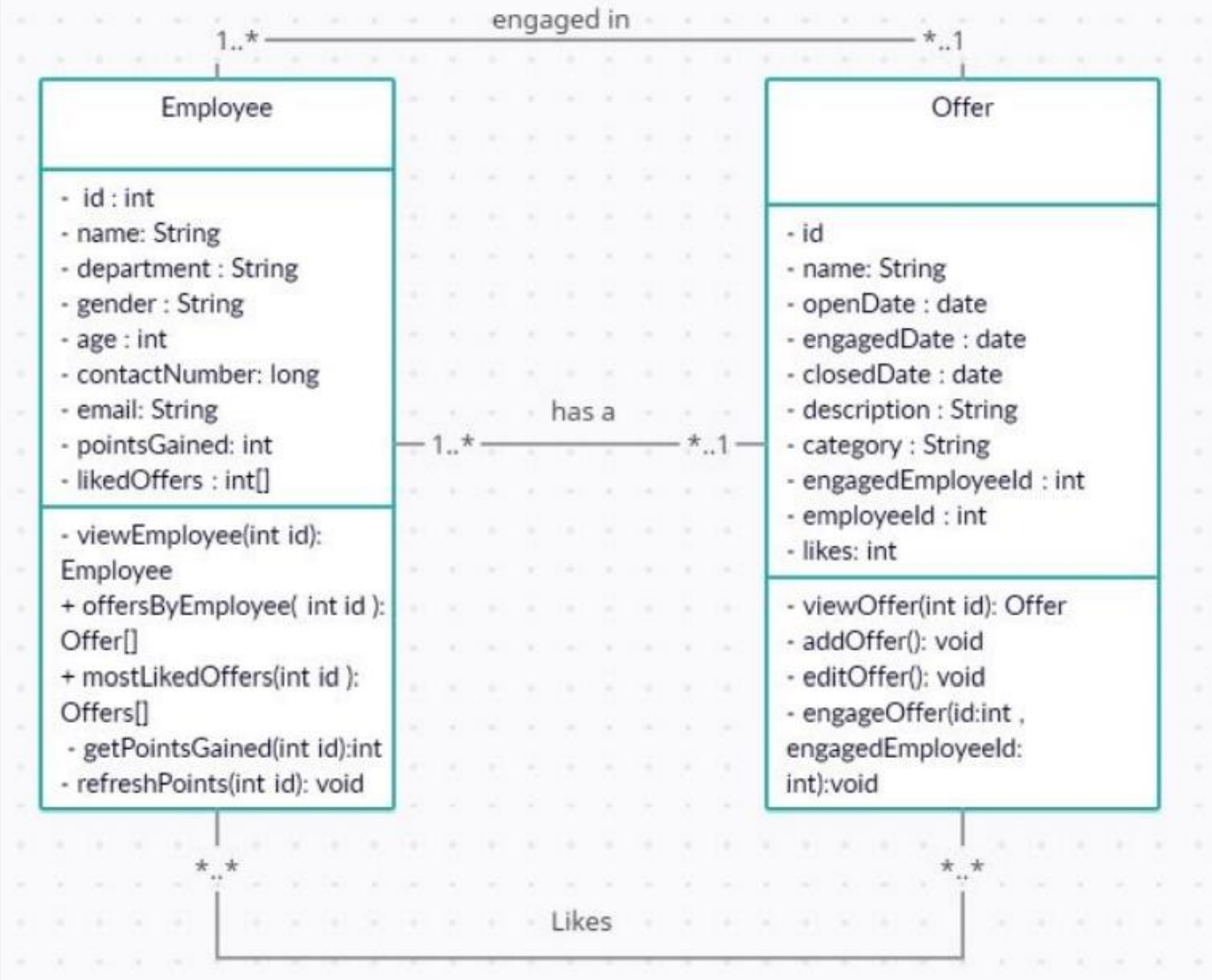
SANDHIYA M (2134656)



Outline of the Project

REQUIREMENT ANALYSIS

CLASS DIAGRAM



SQL TABLES

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
name	varchar(50)	NO		NULL	
department	varchar(50)	YES		NULL	
gender	varchar(6)	NO		NULL	
age	int	NO		NULL	
contact_number	bigint	YES		NULL	
email	varchar(100)	YES	UNI	NULL	
points_gained	int	YES		0	

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	NULL	auto_increment
name	varchar(50)	NO		NULL	
description	varchar(100)	YES		NULL	
category	varchar(50)	NO		NULL	
open_date	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED
closed_date	timestamp	YES		NULL	
engaged_date	timestamp	YES		NULL	
engaged_emp_id	int	YES	MUL	NULL	
emp_id	int	NO	MUL	NULL	
likes	int	YES		0	

Field	Type	Null	Key	Default	Extra
emp_id	int	NO	PRI	NULL	
offer_id	int	NO	PRI	NULL	
liked_on	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

DESIGN & DEVELOPMENT

MICROSERVICES

Auth Microservice: To handle authentication of the employee.

Employee Microservice: To handle employee-related services.

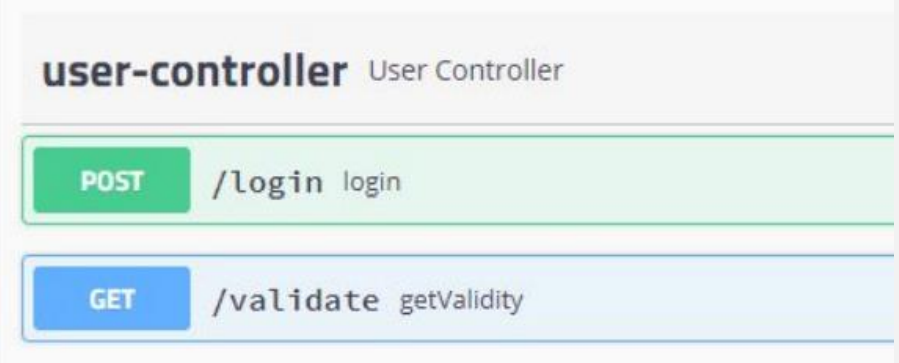
Offer Microservice: To handle offer-related services.

Points Microservice: To refresh points gained by the employee.

AUTH MICROSERVICE

REST Endpoints

- **/login :** to log in the employee and generate JWT token.
- **/validate :** to validate the JWT token.



A look at swagger.html for Auth Microservice

REST Controller

- Rest controller calls the UserServiceImpl.java where all the necessary details are implemented.

```
@PostMapping("/login")
public ResponseEntity<UserToken> login(@RequestBody UserModel user) {
    log.info("Inside Login : ");
    return new ResponseEntity<UserToken>(userServiceImpl.login(user), HttpStatus.OK);
}
```

One of the Post mappings of the rest controller (to login the user)

AUTH SERVICE

```java
// validates the JWT token
public AuthResponse getValidity(String token) {
    // retrieving the token ( removing the Bearer from the header)
    String token1 = token.substring(7);
    AuthResponse authResponse = new AuthResponse();
    // if valid
    if (jwtUtil.validateToken(token1)) {
        log.info("Token is valid");

        // extract the user name
        String username = jwtUtil.extractUsername(token1);

        // set the values for the response
        authResponse.setUsername(username);
        authResponse.setValid(true);
        authResponse.setEmpid(userRepository.findByEmpUsername(username).getEmpid());
    } else {
        authResponse.setValid(false);
        log.error("Token is invalid or expired...");
    }

    return authResponse;
}
```

One of the Post mappings of the rest controller (to login the user)

Exception Handling

- **Rest Exception Handler :** To handle all the exceptions that occurs in the microservice with proper status code.
- **UnauthorizedException :** Thrown when the user is invalid.

EMPLOYEE MICROSERVICE

REST Endpoints

- **/employee/viewProfile/{id} -** to view employee details by id parameter
- **/employee/viewEmployeeOffers/{id} -** to view all the offers posted by the employee through id parameter
- **/employee/viewMostLiked/{id} -** to view top 3 most liked offers of the employee by id parameter

A look at swagger.html for Auth Microservice

REST Controller

- **Rest controller calls the employeeServiceImpl.java where all the necessary details are implemented.**

```
@CrossOrigin(origins = "http://localhost:4200")
@GetMapping("/viewEmployeeOffers/{id}")
public ResponseEntity<?> viewEmployeeOffers(@RequestHeader(name = "Authorization", required = true)
        @PathVariable("id") int employeeId) throws InvalidUserException, MicroserviceException {
    log.info("Inside view employee offers");
    return new ResponseEntity<>(employeeService.viewEmpOffers(token, employeeId), HttpStatus.OK);
}
```

One of the Get mappings of the rest controller (to view the profile of the user)

EMPLOYEE SERVICE

```java
@Override
public List<EmployeeOffers> viewEmpOffers(String token, int id) throws MicroserviceException, InvalidUserException {
    log.info("Inside view employee offers");

    AuthResponse authResponse;
    List<EmployeeOffers> empOffers;

    // validate the user
    try {
        authResponse = authClient.getValidity(token).getBody();
    } catch (Exception e) {
        throw new MicroserviceException(e.getMessage());
    }

    // if token is valid
    if (authResponse.isValid()) {
        // verify the user id with the token id
        if (authResponse.getEmpid() != id) {
            throw new InvalidUserException("invalid token for the user");
        }
        try {
            empOffers = offerClient.getOffersById(token, id);
        } catch (Exception e) {
            throw new MicroserviceException(e.getMessage());
        }
        return empOffers;
    } else {
        log.error("Token invalid");
        throw new InvalidUserException("Invalid User");
    }
}
```

Exception Handling

- **Employee Exception handler :** to handle all the exceptions that occurs in the microservice with proper status code
- **InvalidUserException :** thrown when the user is unauthorized
- **MicroserviceException :** thrown when there is an error in the communicated microservice.

Feign Clients

- **Auth client :** to communicate with the authentication microservice
- **Offer client :** to communicate with the offer microservice

```
//to connect to authentication service
@FeignClient(url = "${auth.feign.client}", name = "${auth.feign.name}")
public interface AuthClient {

    // checks the validity of the jwt token
    @RequestMapping(path = "/validate", method = RequestMethod.GET)
    public ResponseEntity<AuthResponse> getValidity(
            @RequestHeader(name = "Authorization", required = true) String token);
}
```

OFFER MICROSERVICE

REST Endpoints

- **/offer/addOffer -** to post a new offer.
- **/offer/editOffer -** to edit an existing offer.
- **/offer/engageOffer -** to engage an offer.
- **/offer/getOfferByCategory -** to filter offers by category
- **/offer/getOfferByPostedDate -** to filter offers by posted date.
- **/offer/getOfferByTopLikes -** to retrieve top 3 most liked offers.
- **/offer/getOfferDetails/{id} -** to retrieve offer by id parameter.

REST Controller

- **Rest controller calls the offerService.java where all the necessary details are implemented**

```
@CrossOrigin(origins = "http://localhost:4200")
@GetMapping("/getOfferDetails/{offerId}")
public Offer getOfferDetails(@RequestHeader("Authorization") String token, @PathVariable("offerId") int offerId)
        throws OfferNotFoundException, InvalidTokenException, MicroserviceException {

    log.debug("inside getOfferDetails method of offer microservice");
    Offer offer = offerService.getOfferDetails(token, offerId);
    return offer;
}
```

One of the Get mappings of the rest controller (to get the details of an offer, by offer id)

OFFER SERVICE

```java
public Offer getOfferDetails(String token, int offerId)
        throws OfferNotFoundException, InvalidTokenException, MicroserviceException {
    // authenticate the user
    ResponseEntity<AuthResponse> response;
    try {
        response = authClient.verifyToken(token);
    } catch (Exception e) {
        log.info("some error in auth microservice");
        throw new MicroserviceException(e.getMessage());
    }
    // check if token is valid
    if (response.getBody().isValid()) {
        Optional<Offer> offer = offerRepository.findById(offerId);
        // if offer is not found
        if (!offer.isPresent())
            throw new OfferNotFoundException("No offer found");

        return offer.get();
    }
    // if token is invalid
    else {
        throw new InvalidTokenException("token is invalid");
    }
}
```

Exception Handling

- **Global Exception handler -** to handle all the exceptions that occurs in the microservice with proper status code.
- **EmployeeNotFoundException -** thrown when no employee is found.
- **ImproperDateException -** thrown when improper date is passed.
- **InvalidTokenException -** thrown when the jwt token is invalid.
- **MicroserviceException -** thrown when the communicated microservice is not working.
- **OfferNotFoundException -** thrown when no offers are found.

Feign Clients

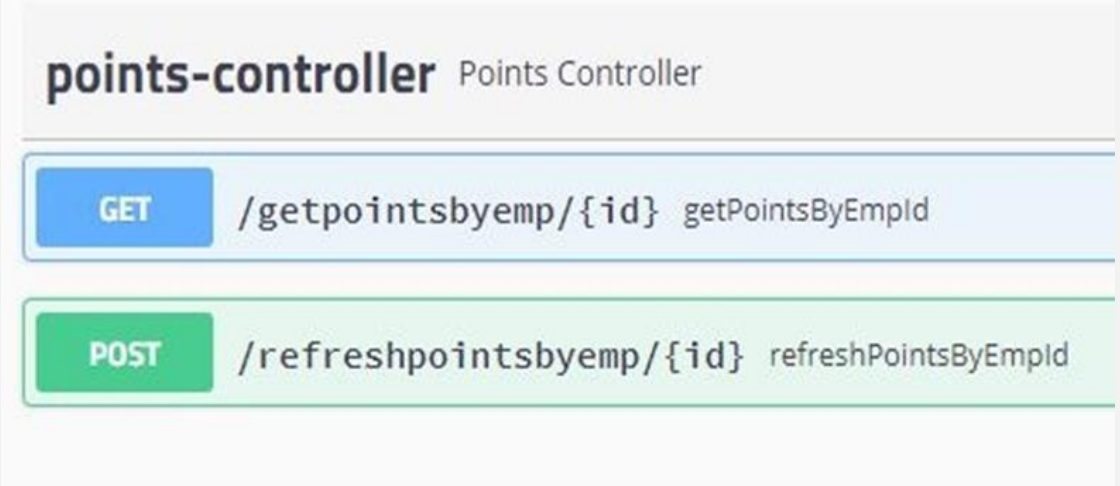
- **Auth client -** to communicate with the authentication microservice.
- **Employee client -** to communicate with the employee microservice.

```
//to connect to authentication service
@FeignClient(url = "${auth.feign.client}", name = "${auth.feign.name}")
public interface AuthClient {

    // checks the validity of the jwt token
    @RequestMapping(path = "/validate", method = RequestMethod.GET)
    public ResponseEntity<AuthResponse> getValidity(
            @RequestHeader(name = "Authorization", required = true) String token);
}
```

POINTS MICROSERVICE

REST Endpoints

- **/getpointsbyemp/{id}** - to get points gained by the employee.
- **/refreshpointsbyemp/{id}** - to refresh the points of the employee.



A look at swagger (for points controller)

REST Controller

- **RESR controller calls the pointsServiceImpl.java where all the necessary details are implemented.**

```
@GetMapping("/getpointsbyemp/{id}")
public ResponseEntity<Integer> getPointsByEmpId(@RequestHeader(name = "Authorization") String token,
        @PathVariable("id") int id) throws MicroserviceException {
    log.info("Inside getpointsbyemployeeid of points microservice");
    return new ResponseEntity<>(pointsService.getPoints(token, id), HttpStatus.OK);
}
```

One of the Get mappings of the REST controller (to get the points gained by the employee)

POINTS SERVICE

```java
public Integer getPoints(String token, int id) throws MicroserviceException, InvalidUserException {
    log.info("Inside getpoints");
    AuthResponse authResponse;
    // verify the token
    try {
        authResponse = authClient.verifyToken(token).getBody();
    } catch (Exception e) {
        throw new MicroserviceException(e.getMessage());
    }
    // validate the user
    if (authResponse.isValid()) {
        Integer points;
        // retrieve the points
        try {
            points = offerClient.getPointsById(token, id);
        } catch (Exception e) {
            throw new MicroserviceException(e.getMessage());
        }
        return points;
    } else {
        log.error("Token invalid");
        throw new InvalidUserException("Invalid User");
    }
}
```

EXCEPTION HANDLING

- **InvalidUserException -** thrown when the jwt token is invalid
- **MicroserviceException -** thrown when the communicated microservice is not working
- **PointsExceptionHandler -** to handle all the exceptions thrown by the rest controller

FEIGN CLIENTS

- **Auth client -** to communicate with the authentication microservice
- **Employee client -** to communicate with the employee microservice
- **Offer client -** to communicate with the offer microservice

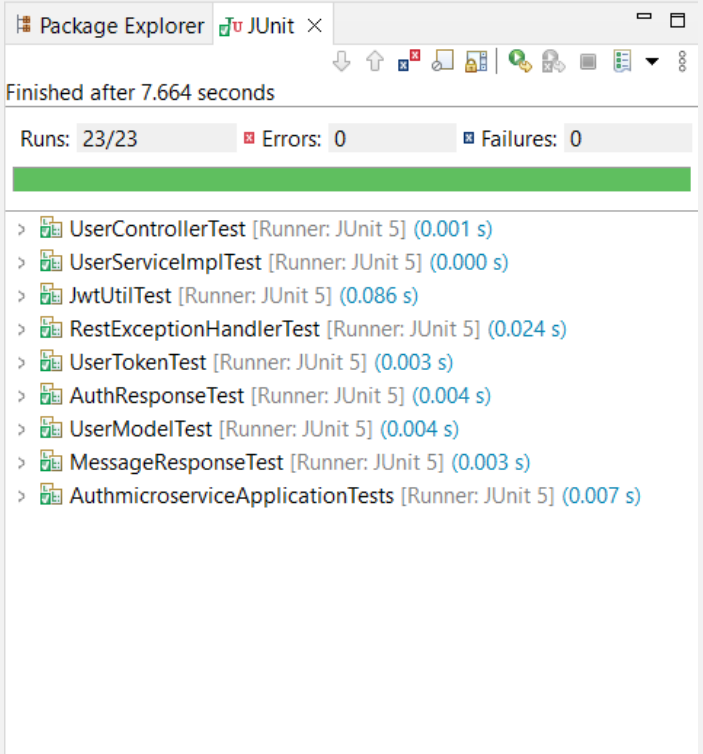
```
@FeignClient(url = "${offer.feign.client}", name = "${offer.feign.name}")
public interface OfferClient {

    @GetMapping("/getOffers/{emp_id}")
    public List<Offer> getOfferByEmpId(@RequestHeader(name = "Authorization")
            String token,
            @PathVariable("emp_id") int id);

    @GetMapping("/getPoints/{emp_id}")
    public int getPointsById(@RequestHeader("Authorization")
            String token,
            @PathVariable("emp_id") int id);
}
```

TESTING

AUTH MICROSERVICE

- UserControllerTest
- AuthMicroserviceApplicationTests
- RestExceptionHandlerTest
- JwtUtilTest
- ModelTests
- UserServiceImplTest

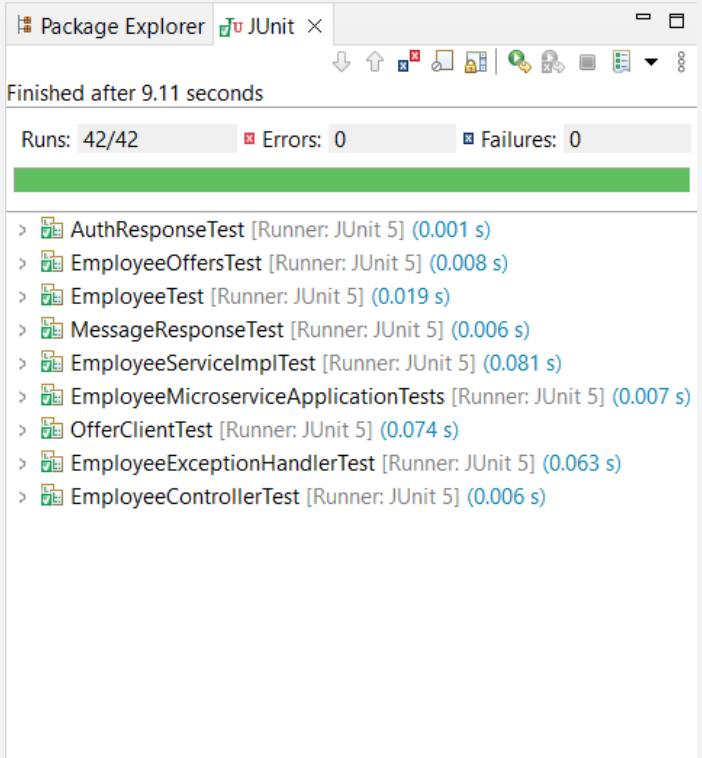
There are 23 test cases and all of them passed successfully.



EMPLOYEE MICROSERVICE

- EmployeeMicroserviceApplicationTests
- OfferClientTest
- EmployeeControllerTest
- EmployeeExceptionHandlerTest
- ModelTests
- EmployeeServiceImplTest

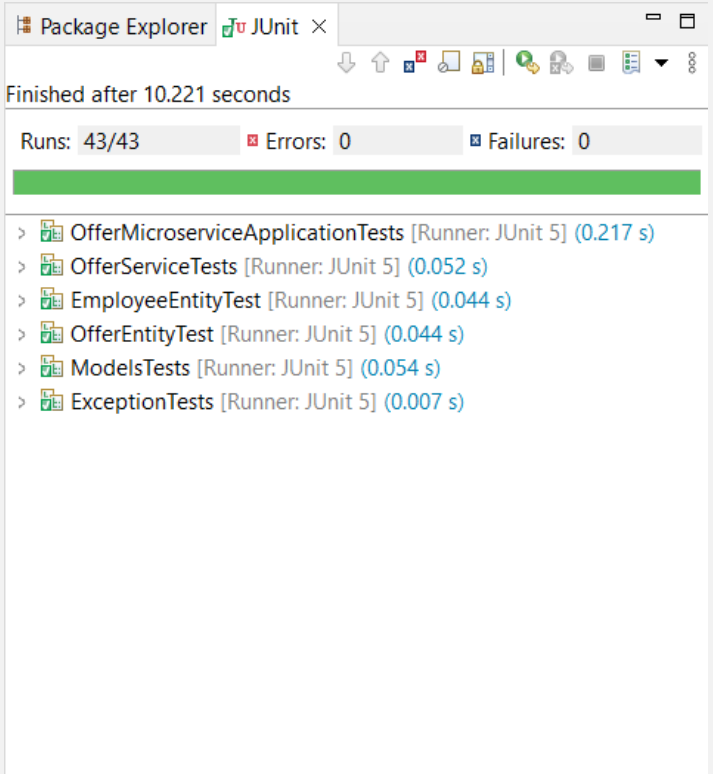
There are 42 test cases and all of them passed successfully.



OFFER MICROSERVICE

- -EmployeeEntityTests
- -ExceptionTests
- -OfferEntityTests
- -OfferServiceTests
- -ModelTests

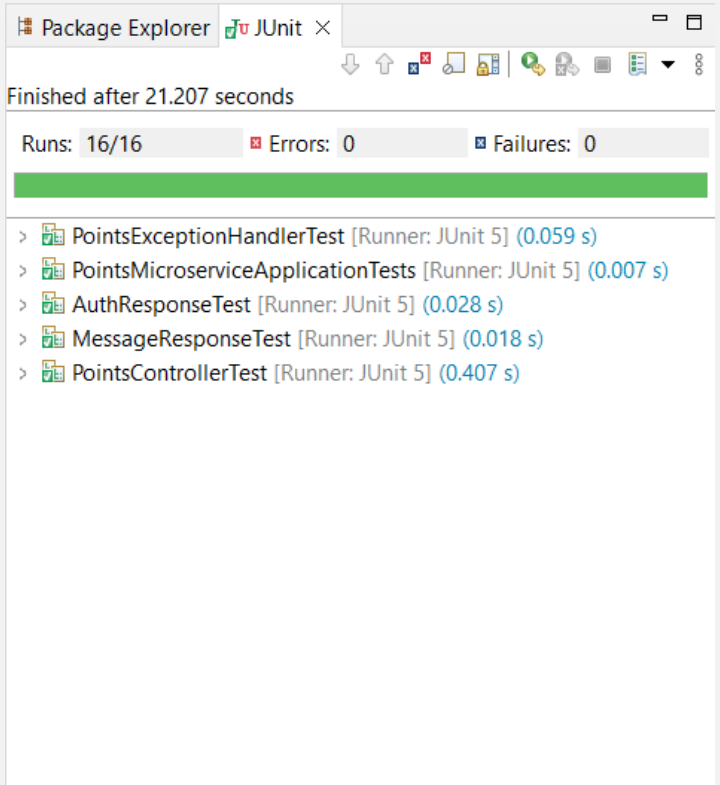
There are 43 test cases and all of them passed successfully.



POINTS MICROSERVICE

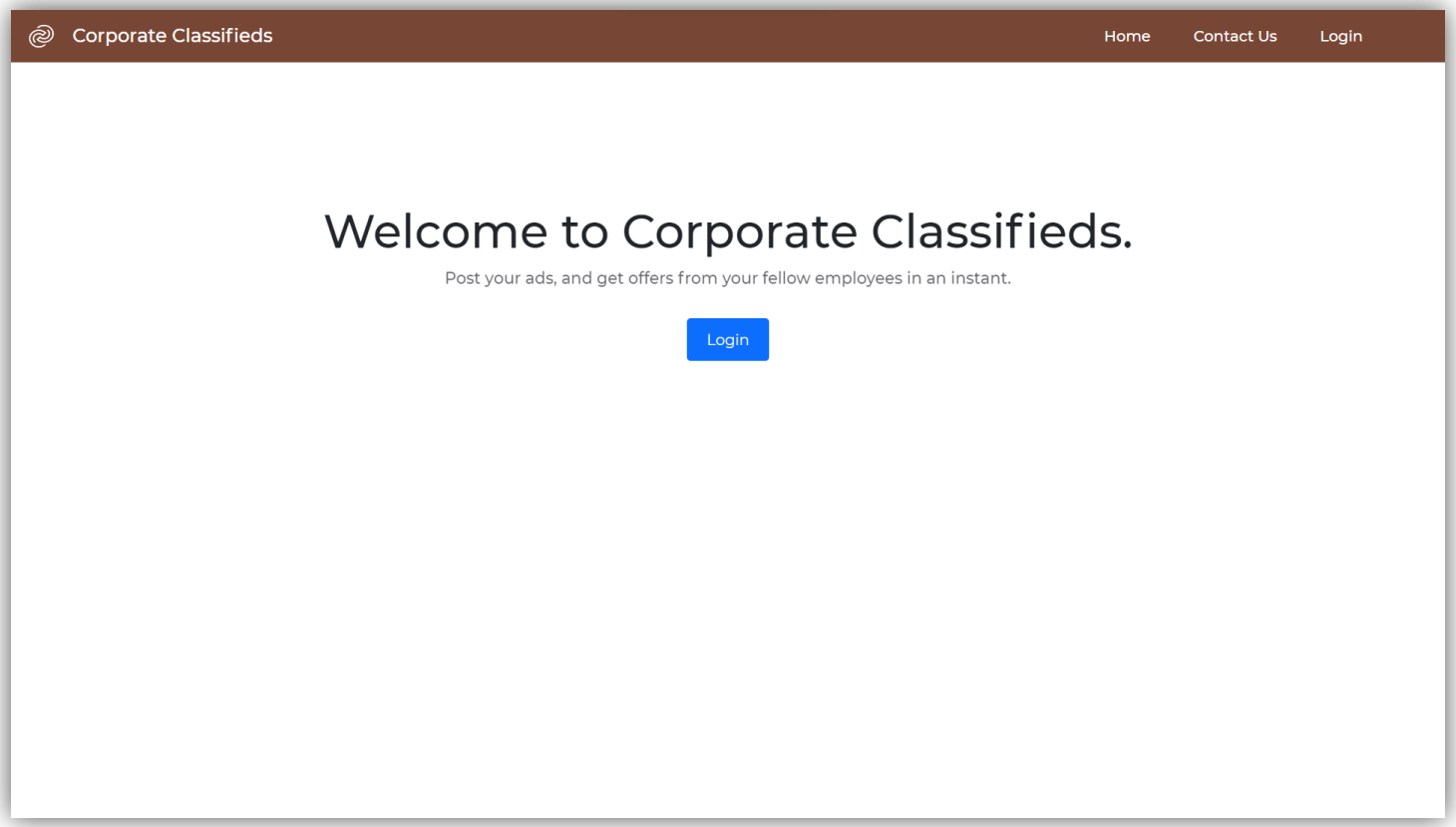
- PointsMicroserviceApplicationTest
- PointsControllerTests
- PointsExceptionHandlerTests
- ModelTests

There are 16 test cases and all of them passed successfully.

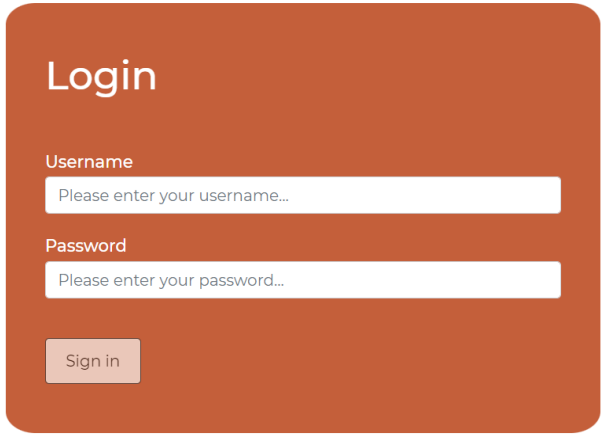


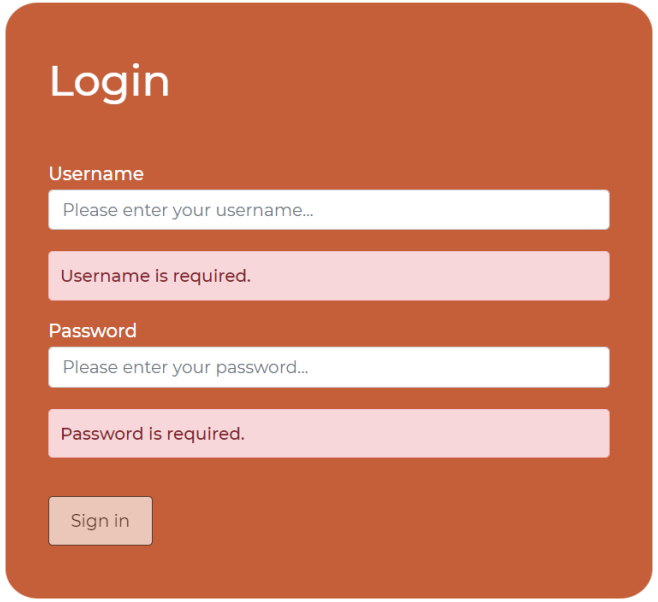
USER INTERFACE

LANDING PAGE



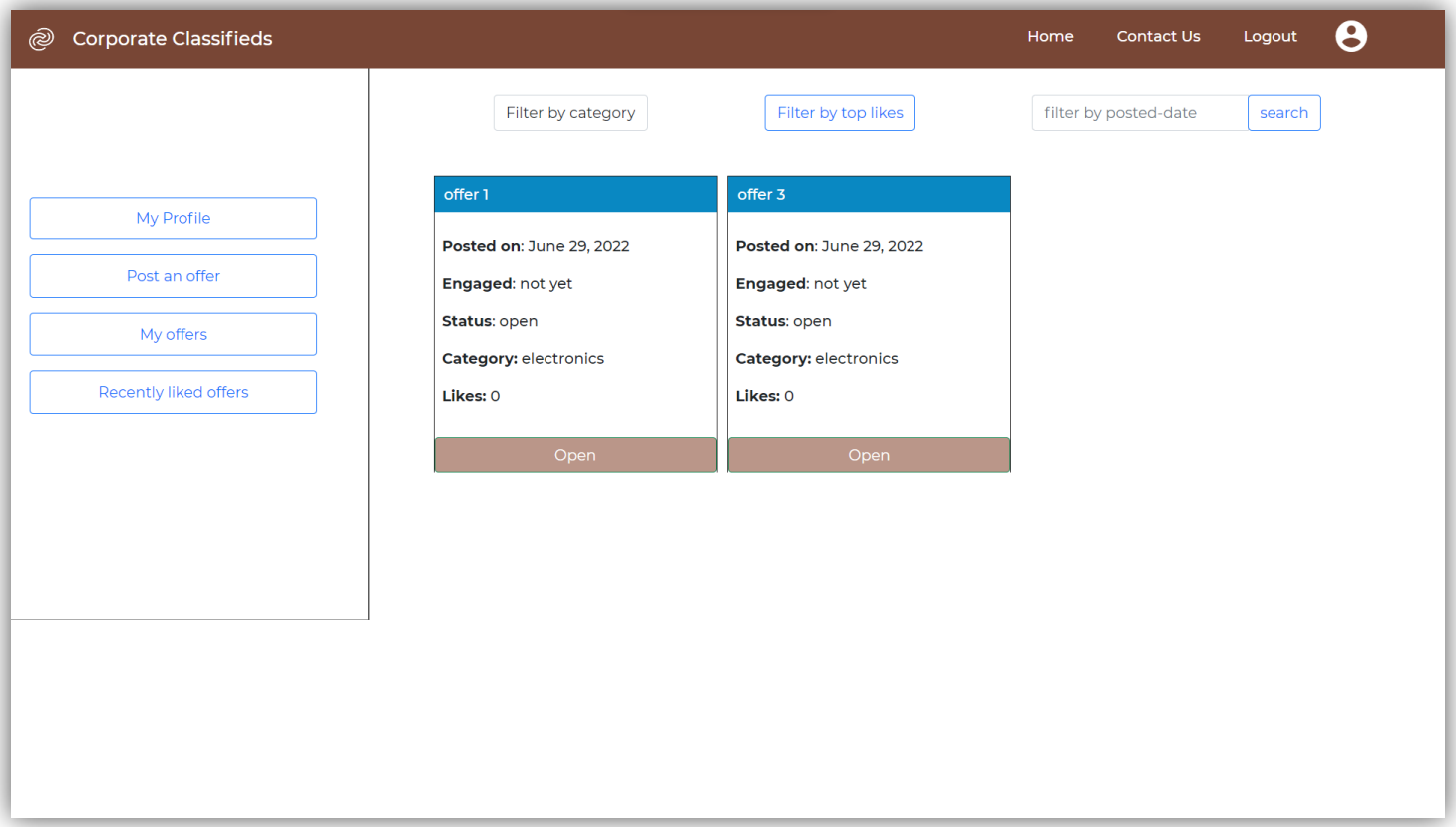
LOGIN PAGE



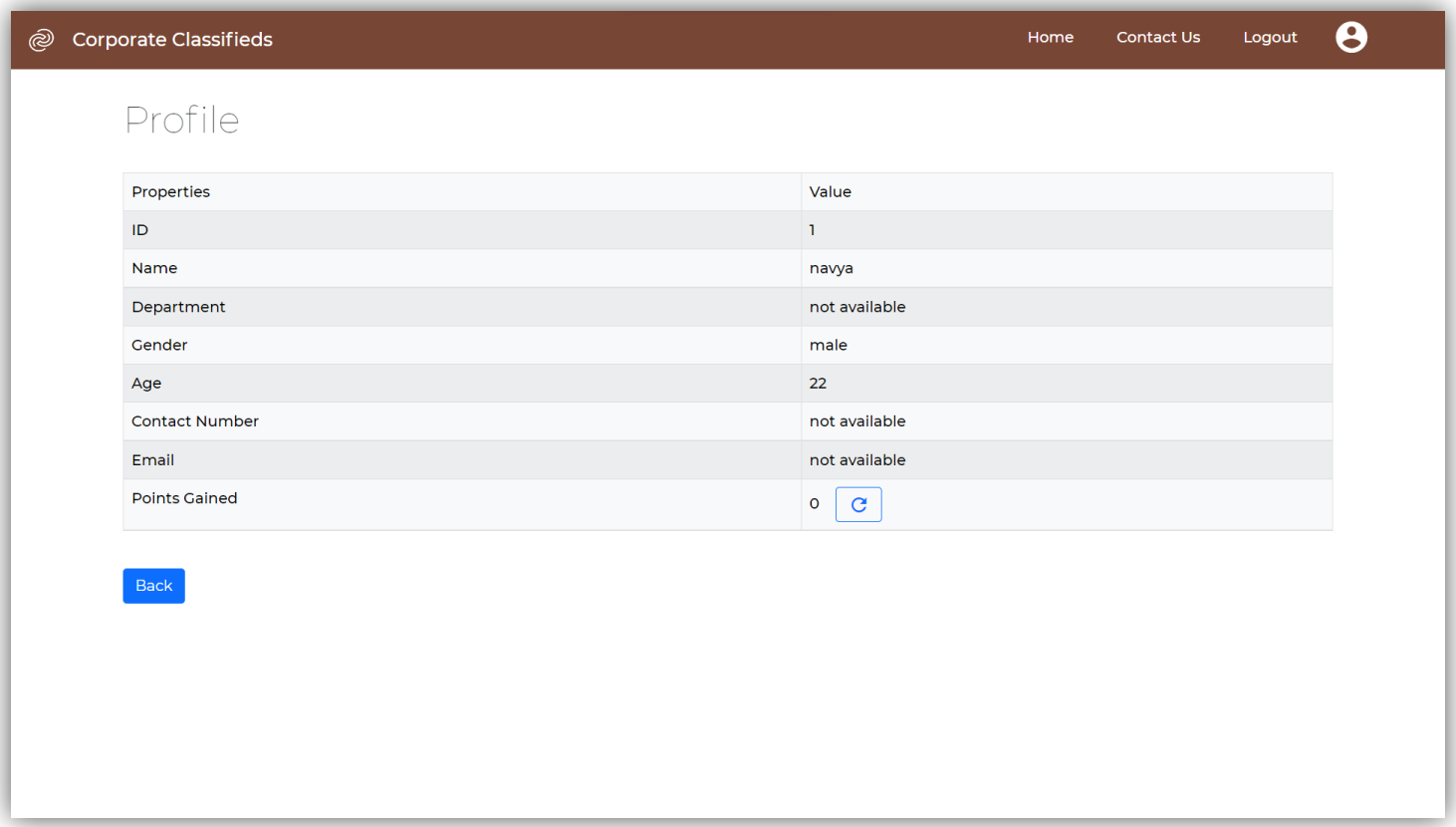


Proper validation for each field

MAIN PAGE

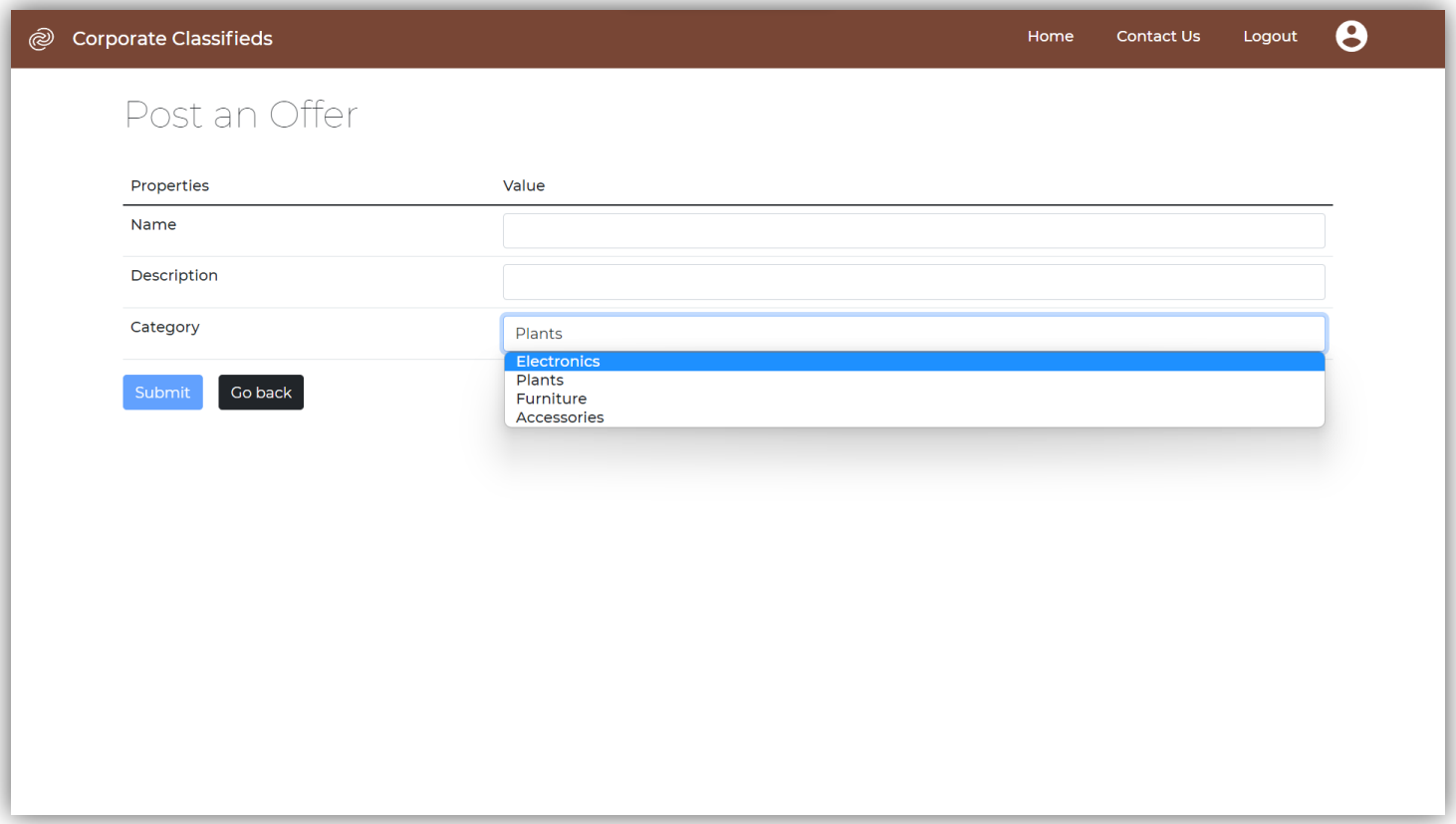

Corporate Classifieds
Home
Contact Us
Logout
Filter by category
Filter by top likes
filter by posted-date
search
My Profile
Post an offer
My offers
Recently liked offers
offer 1
Posted on: June 29, 2022
Engaged: not yet
Status: open
Category: electronics
Likes: 0
Open
offer 3
Posted on: June 29, 2022
Engaged: not yet
Status: open
Category: electronics
Likes: 0
Open

PROFILE PAGE

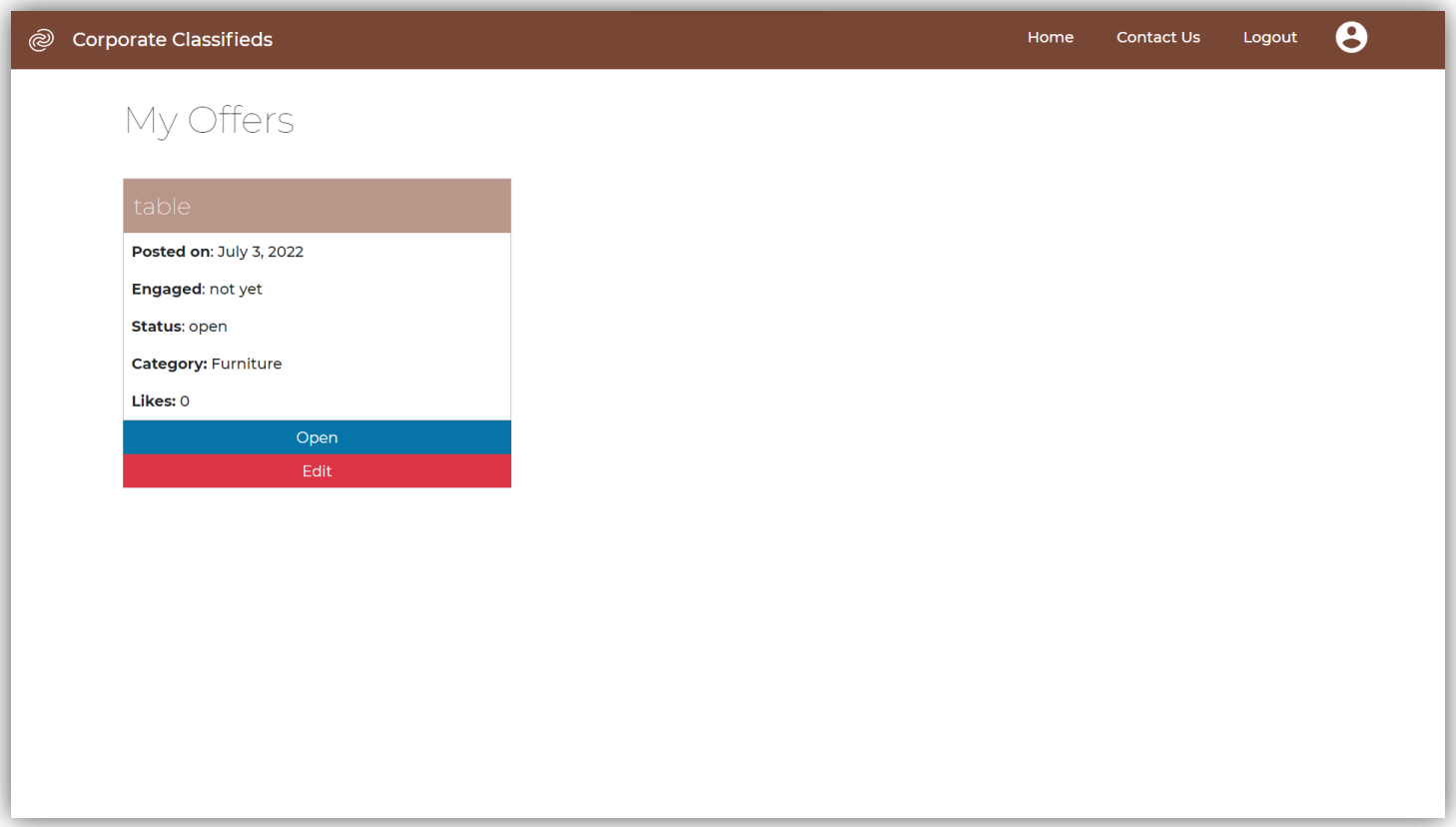


Properties	Value
ID	1
Name	navya
Department	not available
Gender	male
Age	22
Contact Number	not available
Email	not available
Points Gained	0

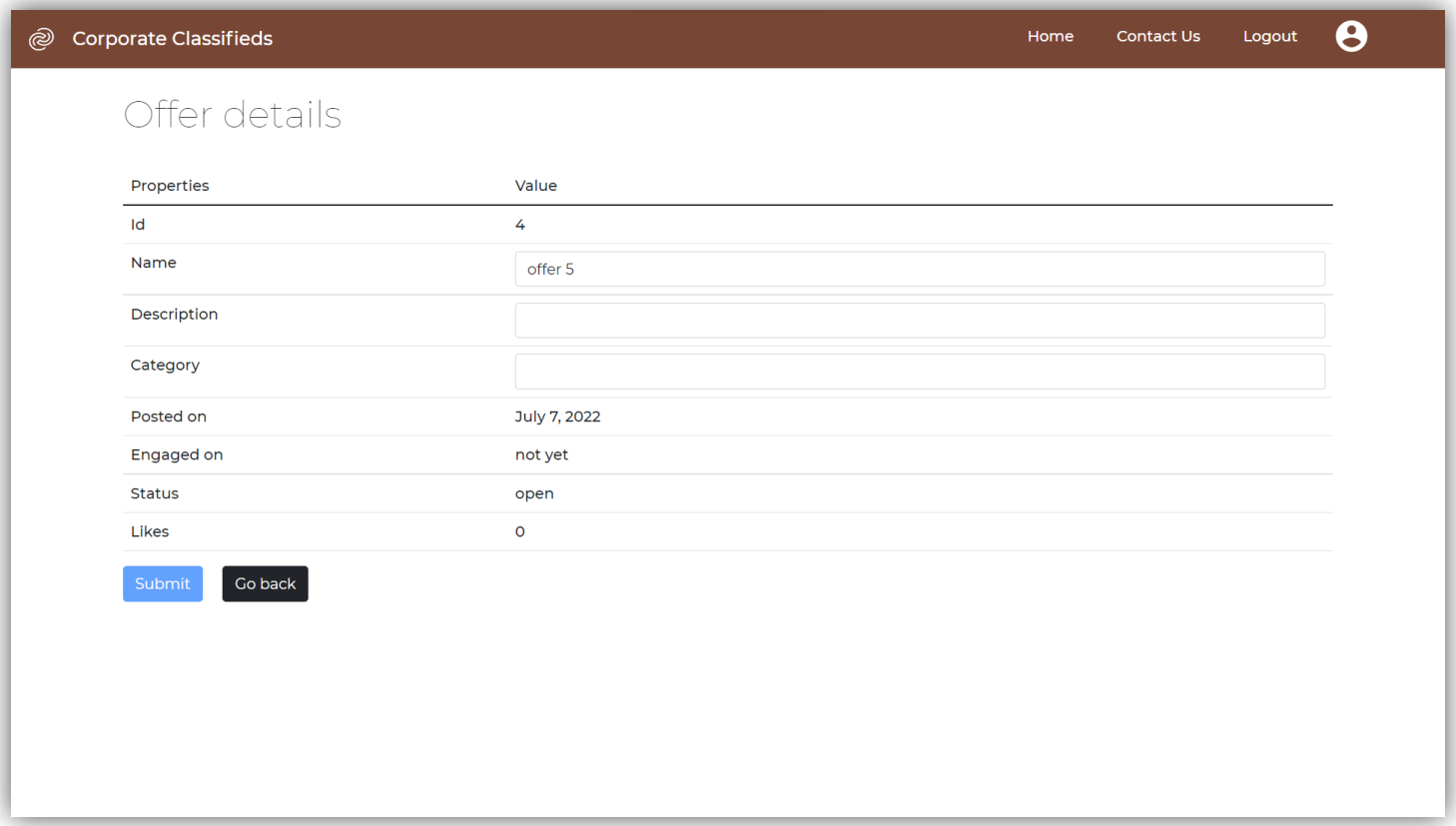
POST OFFER PAGE



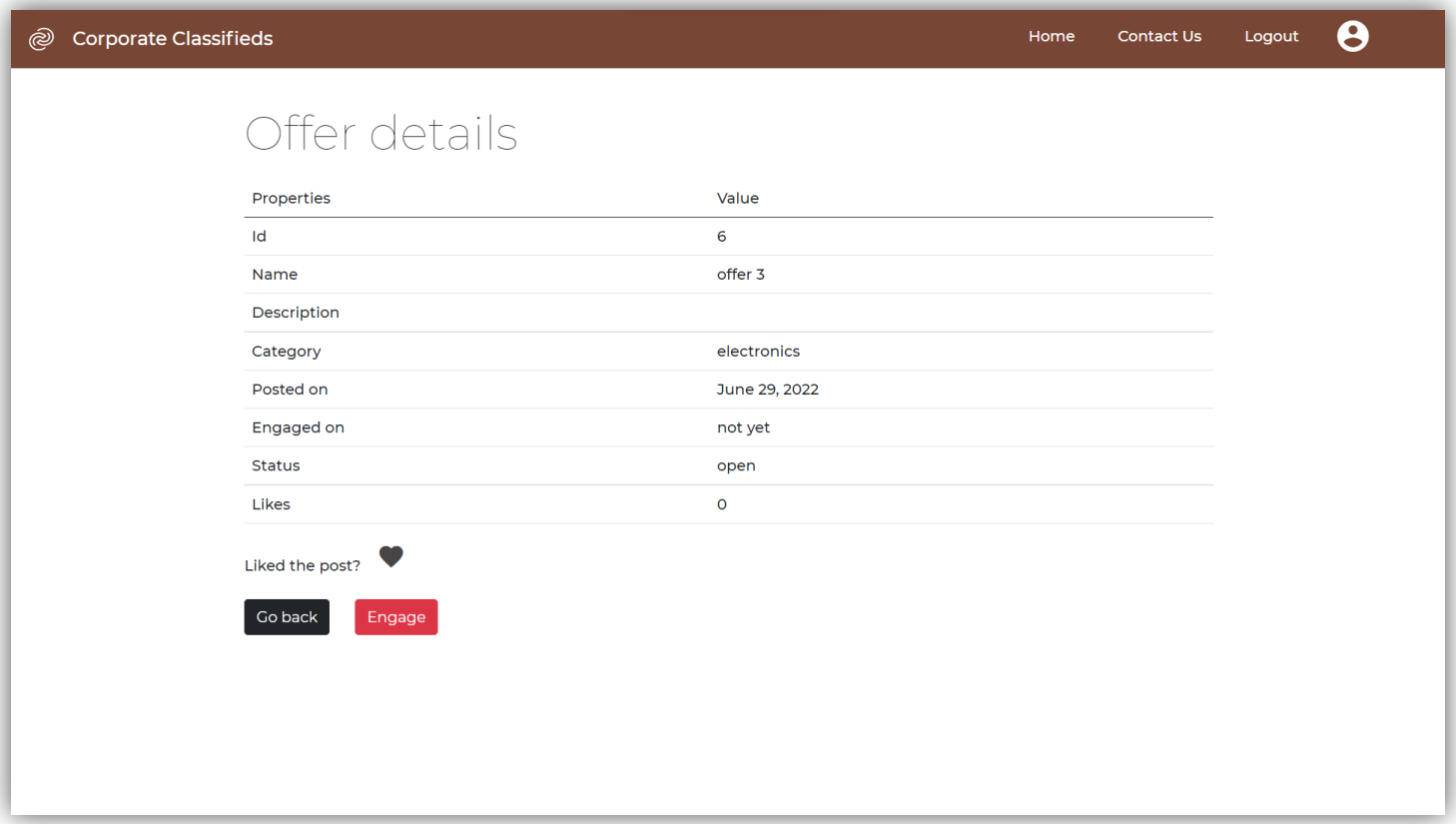
MY OFFERS PAGE



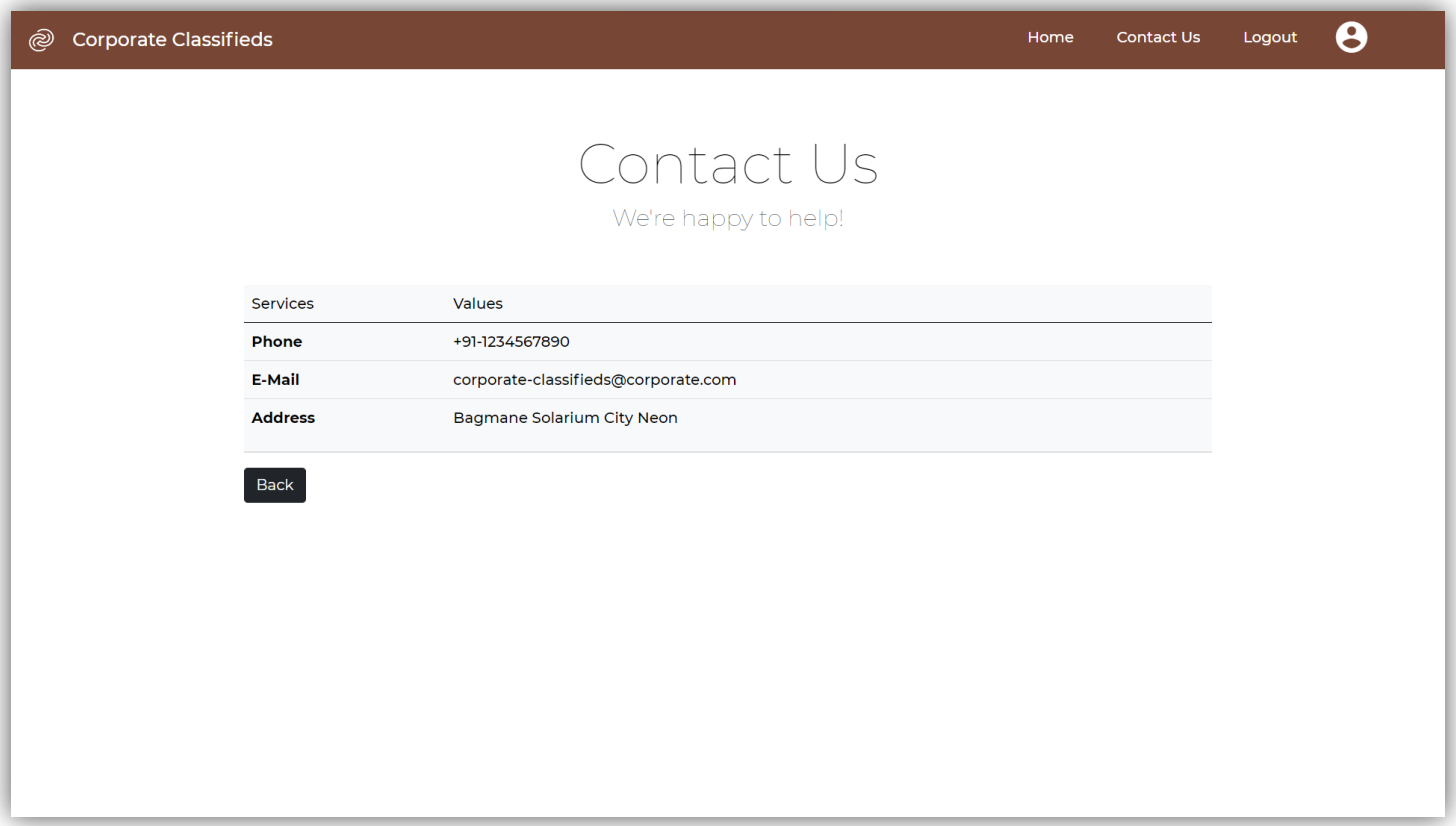
EDIT OFFER PAGE



OFFER DETAILS PAGE



CONTACT US PAGE



THANK YOU