Phase-6

Full stack java developer

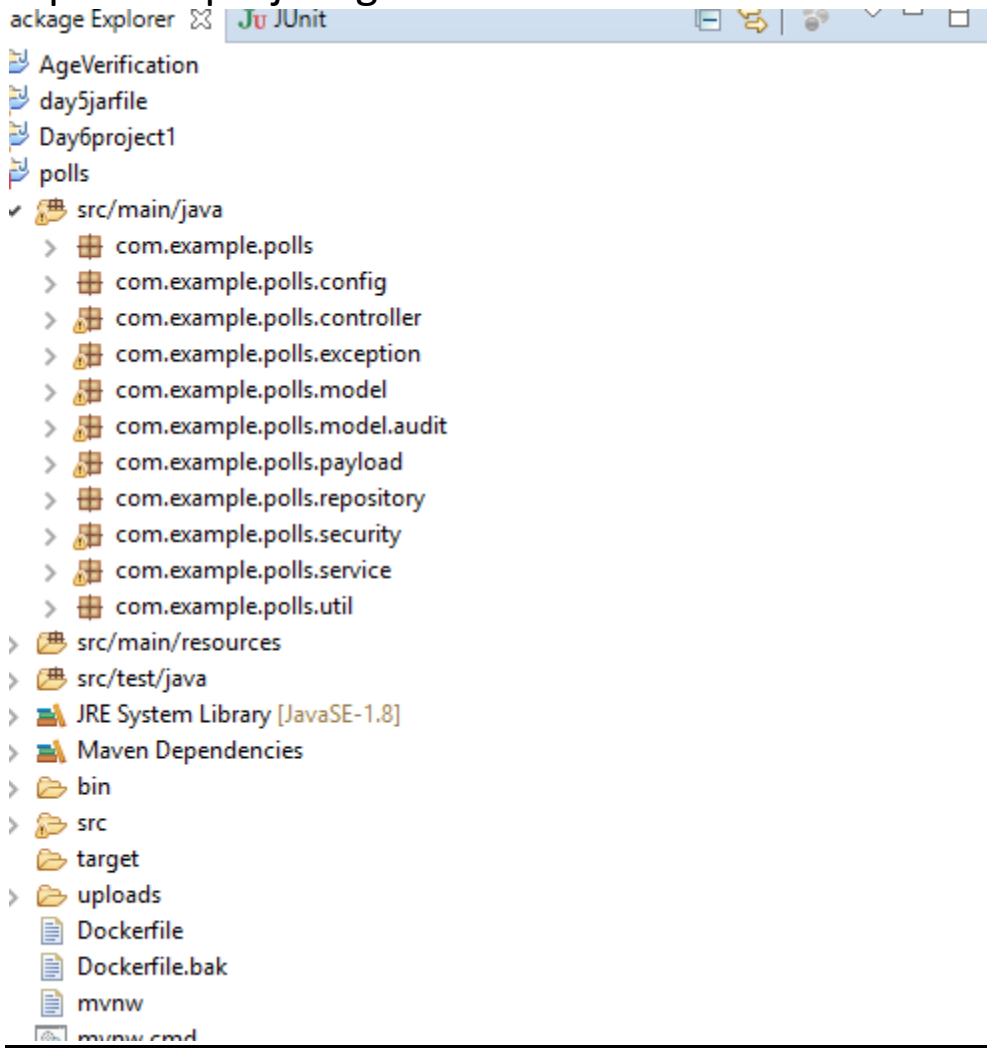
Capstone project

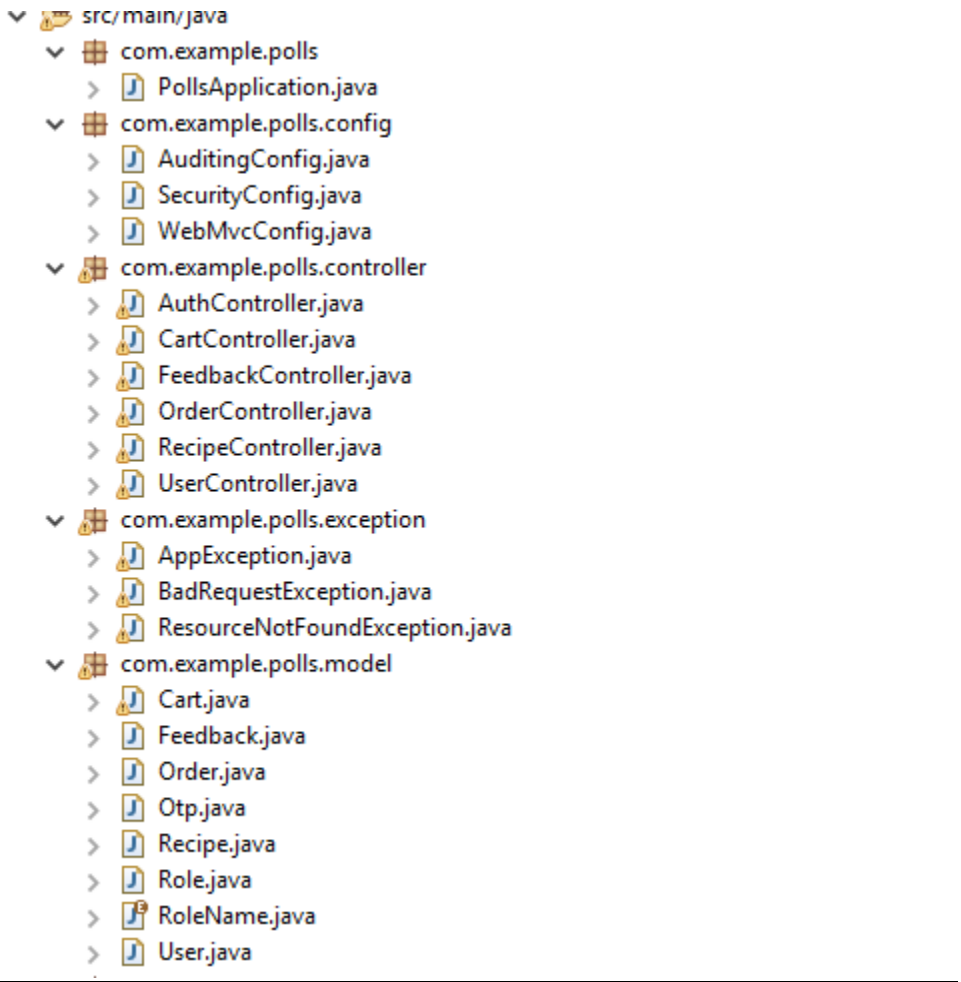
Project :-Food Box

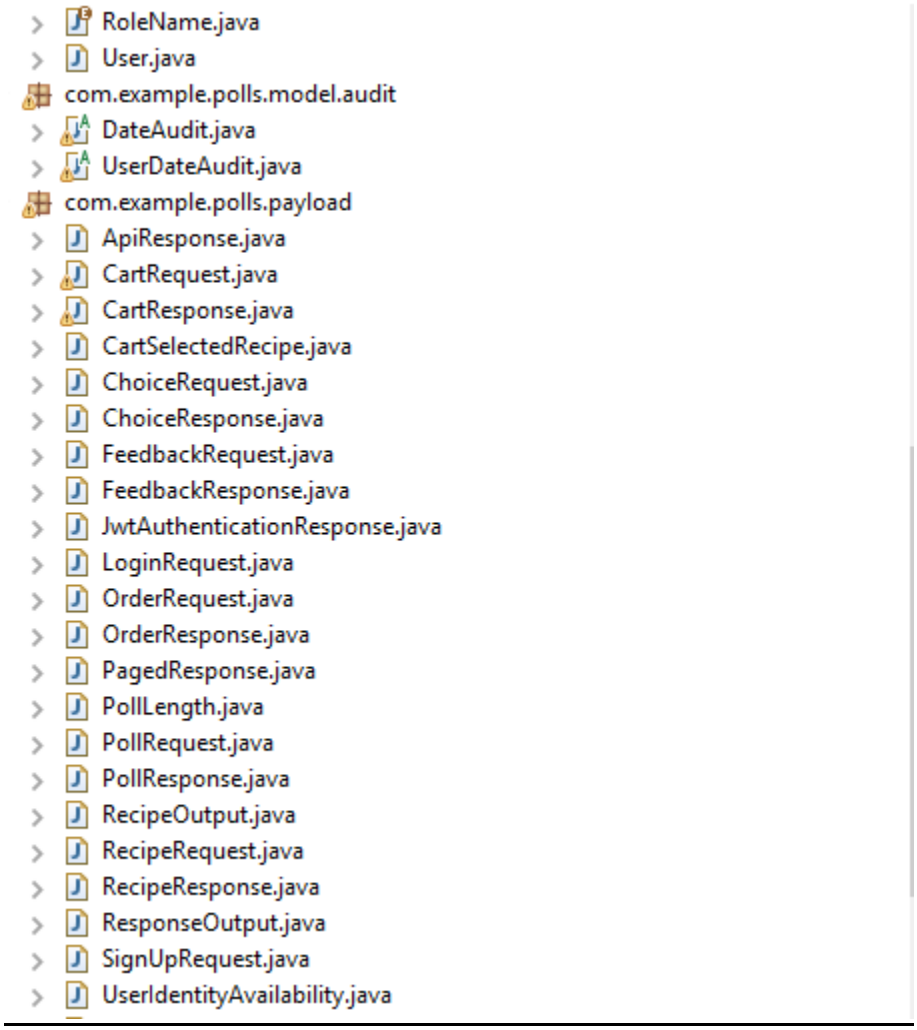
(Source Code)

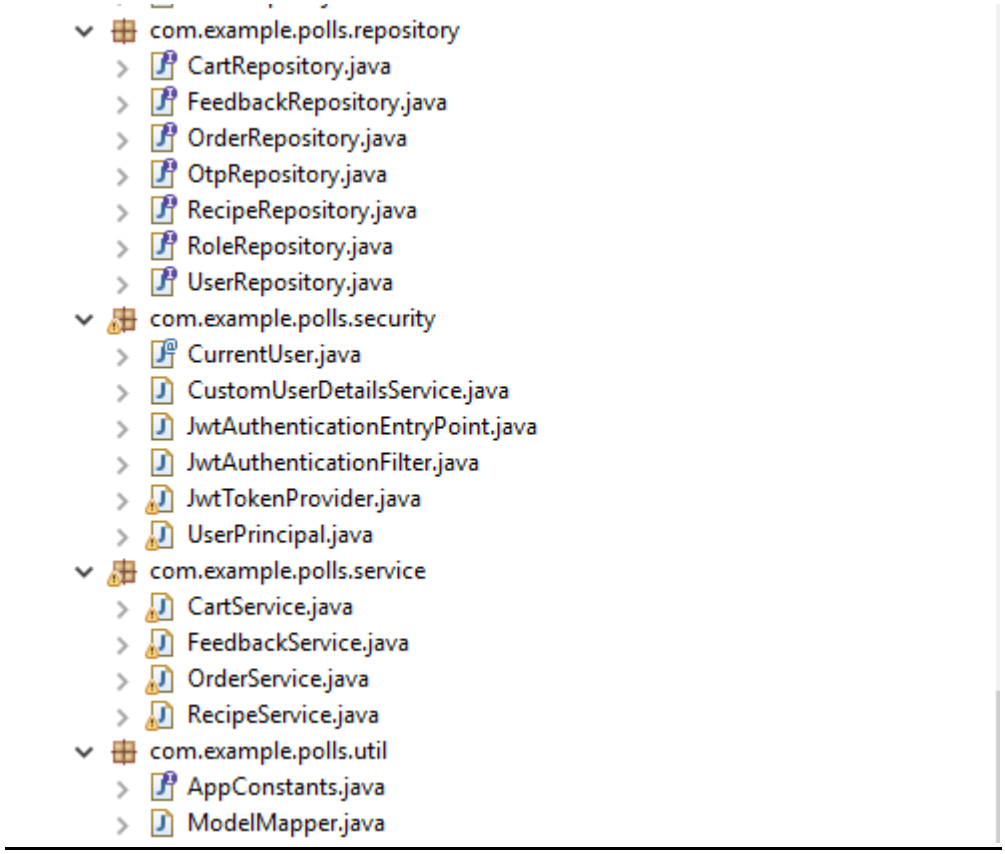
Source Code

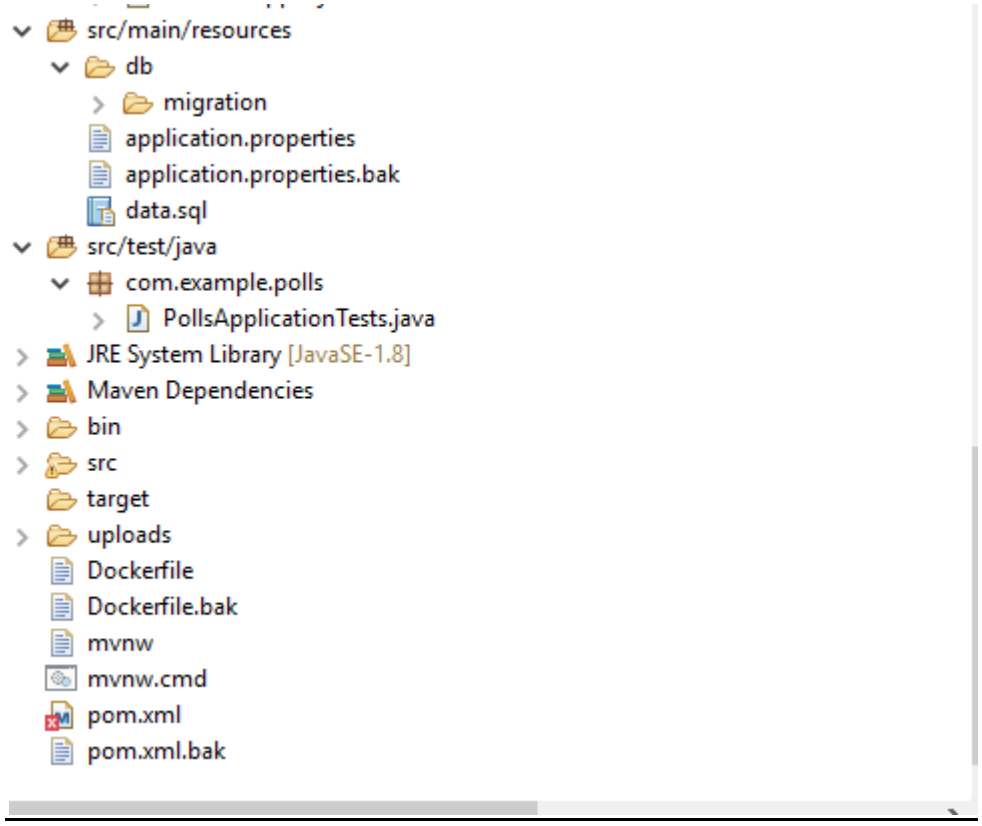
Git hub link:- https://github.com/rajashreemuduli/foodbox-caption-e-project.git




src/main/java
com.example.polls
PollsApplication.java
com.example.polls.config
AuditingConfig.java
SecurityConfig.java
WebMvcConfig.java
com.example.polls.controller
AuthController.java
CartController.java
FeedbackController.java
OrderController.java
RecipeController.java
UserController.java
com.example.polls.exception
AppException.java
BadRequestException.java
ResourceNotFoundException.java
com.example.polls.model
Cart.java
Feedback.java
Order.java
Otp.java
Recipe.java
Role.java
RoleName.java
User.java


RoleName.java
User.java
com.example.polls.model.audit
DateAudit.java
UserDateAudit.java
com.example.polls.payload
ApiResponse.java
CartRequest.java
CartResponse.java
CartSelectedRecipe.java
ChoiceRequest.java
ChoiceResponse.java
FeedbackRequest.java
FeedbackResponse.java
JwtAuthenticationResponse.java
LoginRequest.java
OrderRequest.java
OrderResponse.java
PagedResponse.java
PollLength.java
PollRequest.java
PollResponse.java
RecipeOutput.java
RecipeRequest.java
RecipeResponse.java
ResponseOutput.java
SignUpRequest.java
UserIdentityAvailability.java


com.example.polls.repository
CartRepository.java
FeedbackRepository.java
OrderRepository.java
OtpRepository.java
RecipeRepository.java
RoleRepository.java
UserRepository.java
com.example.polls.security
CurrentUser.java
CustomUserDetailsService.java
JwtAuthenticationEntryPoint.java
JwtAuthenticationFilter.java
JwtTokenProvider.java
UserPrincipal.java
com.example.polls.service
CartService.java
FeedbackService.java
OrderService.java
RecipeService.java
com.example.polls.util
AppConstants.java
ModelMapper.java


src/main/resources
db
migration
application.properties
application.properties.bak
data.sql
src/test/java
com.example.polls
PollsApplicationTests.java
JRE System Library [JavaSE-1.8]
Maven Dependencies
bin
src
target
uploads
Dockerfile
Dockerfile.bak
mvnw
mvnw.cmd
pom.xml
pom.xml.bak

Polls application.java

```
package com.example.polls;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.boot.autoconfigure.domain.EntityScan;
import org.springframework.data.jpa.convert.threeten.Jsr310JpaConverters;
import javax.annotation.PostConstruct;
import java.util.TimeZone;
@SpringBootApplication
@EntityScan(basePackageClasses = {
        PollsApplication.class,
        Jsr310JpaConverters.class
})
public class PollsApplication {
    @PostConstruct
    void init() {
        TimeZone.setDefault(TimeZone.getTimeZone("UTC"));
    }
    public static void main(String[] args) {
        SpringApplication.run(PollsApplication.class, args);
    }}
```

AuditingConfig.java

```
package com.example.polls.config;
import com.example.polls.security.UserPrincipal;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.data.domain.AuditorAware;
```

```
import org.springframework.data.jpa.repository.config.EnableJpaAuditing;
import org.springframework.security.authentication.AnonymousAuthenticationToken;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.context.SecurityContextHolder;
import java.util.Optional;
@Configuration
@EnableJpaAuditing
public class AuditingConfig {
   @Bean
   public AuditorAware<Long> auditorProvider() {
      return new SpringSecurityAuditAwareImpl();
   }}
class SpringSecurityAuditAwareImpl implements AuditorAware<Long> {
   @Override
   public Optional<Long> getCurrentAuditor() {
      Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
      if (authentication == null ||
            !authentication.isAuthenticated() ||
            authentication instanceof AnonymousAuthenticationToken) {
         return Optional.empty();
      }
      UserPrincipal userPrincipal = (UserPrincipal) authentication.getPrincipal();
      return Optional.ofNullable(userPrincipal.getId());
   }}
```

SecurityConfig.java

```
package com.example.polls.config;
import com.example.polls.security.CustomUserDetailsService;
```

```
import com.example.polls.security.JwtAuthenticationEntryPoint;

import com.example.polls.security.JwtAuthenticationFilter;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.http.HttpMethod;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.config.BeanIds;

import
org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerB
uilder;

import
org.springframework.security.config.annotation.method.configuration.EnableGlobalMethodSecuri
ty;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapte
r;

import org.springframework.security.config.http.SessionCreationPolicy;

import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter;

@Configuration

@EnableWebSecurity

@EnableGlobalMethodSecurity(

    securedEnabled = true,

    jsr250Enabled = true,

    prePostEnabled = true

)
```

```
public class SecurityConfig extends WebSecurityConfigurerAdapter {

   @Autowired

   CustomUserDetailsService customUserDetailsService;

   @Autowired

   private JwtAuthenticationEntryPoint unauthorizedHandler;

   @Bean

   public JwtAuthenticationFilter jwtAuthenticationFilter() {

      return new JwtAuthenticationFilter();

   }

   @Override

   public void configure(AuthenticationManagerBuilder authenticationManagerBuilder) throws
Exception {

      authenticationManagerBuilder

            .userDetailsService(customUserDetailsService)

            .passwordEncoder(passwordEncoder());

   }

   @Bean(BeanIds.AUTHENTICATION_MANAGER)

   @Override

   public AuthenticationManager authenticationManagerBean() throws Exception {

      return super.authenticationManagerBean();

   }

   @Bean

   public PasswordEncoder passwordEncoder() {

      return new BCryptPasswordEncoder();

   }

   @Override

   protected void configure(HttpSecurity http) throws Exception {

      http
```

```
.cors()
                .and()
            .csrf()
                .disable()
            .exceptionHandling()
                .authenticationEntryPoint(unauthorizedHandler)
                .and()
            .sessionManagement()
                .sessionCreationPolicy(SessionCreationPolicy.STATELESS)
                .and()
            .authorizeRequests()
                .antMatchers("/",
                    "/favicon.ico",
                    "/**/*.png",
                    "/**/*.gif",
                    "/**/*.svg",
                    "/**/*.jpg",
                    "/**/*.html",
                    "/**/*.css",
                    "/**/*.js")
                    .permitAll()
                .antMatchers("/api/auth/**")
                    .permitAll()
                .antMatchers("/api/user/checkUsernameAvailability",
"/api/user/checkEmailAvailability")
                    .permitAll()
                .antMatchers(HttpMethod.GET, "/api/polls/**", "/api/users/**")
                    .permitAll()
```

```
            .antMatchers(HttpMethod.POST, "/login", "/register")
               .permitAll()
            .anyRequest()
               .authenticated();

    // Add our custom JWT security filter
    http.addFilterBefore(jwtAuthenticationFilter(), UsernamePasswordAuthenticationFilter.class);
  }}
```

WebMvcConfig.java

```
package com.example.polls.config;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.cors.CorsConfiguration;
import org.springframework.web.servlet.config.annotation.CorsRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
@Configuration
public class WebMvcConfig implements WebMvcConfigurer {
  private final long MAX_AGE_SECS = 3600;
  @Value("${app.cors.allowedOrigins}")
  private String[] allowedOrigins;
  @Override
  public void addCorsMappings(CorsRegistry registry) {
    registry.addMapping("/**")
        .allowedOrigins(CorsConfiguration.ALL)
        .allowedMethods("HEAD", "OPTIONS", "GET", "POST", "PUT", "PATCH", "DELETE")
        .maxAge(MAX_AGE_SECS);
  }}
```

Contollers

AuthController.java

```
package com.example.polls.controller;

import java.net.URI;

import java.util.ArrayList;

import java.util.Collections;

import java.util.HashSet;

import java.util.List;

import java.util.Set;

import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.security.authentication.AnonymousAuthenticationToken;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.Authentication;

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.security.core.userdetails.UsernameNotFoundException;

import org.springframework.security.crypto.password.PasswordEncoder;

import org.springframework.web.bind.annotation.DeleteMapping;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RestController;

import org.springframework.web.servlet.support.ServletUriComponentsBuilder;
```

```
import com.example.polls.exception.AppException;
import com.example.polls.model.Recipe;
import com.example.polls.model.Role;
import com.example.polls.model.RoleName;
import com.example.polls.model.User;
import com.example.polls.payload.ApiResponse;
import com.example.polls.payload.JwtAuthenticationResponse;
import com.example.polls.payload.LoginRequest;
import com.example.polls.payload.ResponseOutput;
import com.example.polls.payload.SignUpRequest;
import com.example.polls.payload.UserResponse;
import com.example.polls.repository.RoleRepository;
import com.example.polls.repository.UserRepository;
import com.example.polls.security.CurrentUser;
import com.example.polls.security.JwtTokenProvider;
import com.example.polls.security.UserPrincipal;
@RestController
// @RequestMapping("/api/auth")
public class AuthController {
        @Autowired
        AuthenticationManager authenticationManager;
        @Autowired
        UserRepository userRepository;
        @Autowired
        RoleRepository roleRepository;
        @Autowired
        PasswordEncoder passwordEncoder;
```

```
@Autowired
JwtTokenProvider tokenProvider;
@PostMapping("/login")
public ResponseEntity<?> authenticateUser(
@Valid @RequestBody LoginRequest loginRequest) {
Authentication authentication = authenticationManager
.authenticate(new UsernamePasswordAuthenticationToken(loginRequest.getEmail(),
loginRequest.getPassword()));
SecurityContextHolder.getContext().setAuthentication(authentication);
boolean admin = false;
if (!(authentication instanceof AnonymousAuthenticationToken)) {
String currentUserName = authentication.getName();
User user = userRepository.findByUsernameOrEmail(currentUserName,
currentUserName).orElseThrow(
() -> new UsernameNotFoundException(
"User not found with username or email : "
+ currentUserName));
Set<Role> roles = new HashSet<>();
for (Role role : user.getRoles()) {
if (role.getName().toString().contains("ADMIN")) {
admin = true;
} } }
String jwt = tokenProvider.generateToken(authentication);
return ResponseEntity.ok(new JwtAuthenticationResponse(jwt, admin));
}
@GetMapping("/check")
public ResponseEntity<?> validateUser() {
```

```
Authentication authentication = SecurityContextHolder.getContext()
                                .getAuthentication();
        if (!(authentication instanceof AnonymousAuthenticationToken)) {
                // String currentUserName = authentication.getName();
                return ResponseEntity.ok(new ResponseOutput("All ok"));
        }
        return (ResponseEntity<?>) ResponseEntity.badRequest();
    }
    @GetMapping("/admin/check")
    public ResponseEntity<?> validateAdminUser() {
        Authentication authentication = SecurityContextHolder.getContext()
                                .getAuthentication();
        if (!(authentication instanceof AnonymousAuthenticationToken)) {
                // String currentUserName = authentication.getName();
                return ResponseEntity.ok(new ResponseOutput("All ok"));
        }               return (ResponseEntity<?>) ResponseEntity.badRequest();
    }
    @GetMapping("/read")
    public ResponseEntity<?> getUser(@CurrentUser UserPrincipal currentUser) {
Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
        if (!(authentication instanceof AnonymousAuthenticationToken)) {
                String currentUserName = authentication.getName();
                User user = userRepository.findByUsernameOrEmail(
            currentUser.getUsername(), currentUser.getEmail()).orElseThrow(
                                () -> new UsernameNotFoundException(
                        "User not found with username or email : "currentUserName));
                UserResponse userResponse = new UserResponse();
```

```
                userResponse.setId(user.getId());
                userResponse.setName(user.getName());
                userResponse.setEmail(user.getEmail());
                userResponse.setPassword(user.getPassword());
                String roleName = "";
                Set<Role> roles = new HashSet<>();
                for (Role role : user.getRoles()) {
                        roleName = role.getName().toString();
                }
                if (roleName.contains("ADMIN")) {
                        userResponse.setRole("admin");
                }
                if (roleName.contains("USER")) {
                        userResponse.setRole("customer");
                }
                userResponse.setContact(user.getContact());
                return ResponseEntity.ok(new ResponseOutput(userResponse));
        }
        return (ResponseEntity<?>) ResponseEntity.badRequest();
}
@GetMapping("/admin/getalluser")
public ResponseEntity<?> getUsers() {
        List<UserResponse> usersList = new ArrayList<>();
        List<User> users = userRepository.findAll();
        for (User user : users) {
                UserResponse userResponse = new UserResponse();
                userResponse.setId(user.getId());
```

```
                userResponse.setName(user.getName());
                userResponse.setEmail(user.getEmail());
                userResponse.setPassword(user.getPassword());
                String roleName = "";
                Set<Role> roles = new HashSet<>();
                for (Role role : user.getRoles()) {
                        roleName = role.getName().toString();
                }
                if (roleName.contains("ADMIN")) {
                        userResponse.setRole("admin");
                }
                if (roleName.contains("USER")) {
                        userResponse.setRole("customer");
                }
                userResponse.setContact(user.getContact());
                usersList.add(userResponse);
        }
        return ResponseEntity.ok(new ResponseOutput(usersList));
}
@DeleteMapping("/admin/blockuser/{userId}")
public ResponseEntity<?> blockuser(@PathVariable String userId) {
        // Creating user's account
        User user = userRepository.getById(Long.valueOf(userId));
        user.setBlocked(Boolean.TRUE);
        userRepository.save(user);
        return ResponseEntity.ok(new ResponseOutput("All ok"));
}
```

```
@DeleteMapping("/admin/unblockuser/{userId}")
public ResponseEntity<?> unblockuser(@PathVariable String userId) {
        // Creating user's account
        User user = userRepository.getById(Long.valueOf(userId));
        user.setBlocked(Boolean.FALSE);
        userRepository.save(user);
        return ResponseEntity.ok(new ResponseOutput("All ok"));
}
@DeleteMapping("/admin/deleteuser/{userId}")
public ResponseEntity<?> deleteuser(@PathVariable String userId) {
        // Creating user's account
        User user = userRepository.getById(Long.valueOf(userId));
        // user.setBlocked(Boolean.FALSE);
        userRepository.delete(user);
        return ResponseEntity.ok(new ResponseOutput("All ok"));
}

@PostMapping("/register")
public ResponseEntity<?> registerUser(
                @Valid @RequestBody SignUpRequest signUpRequest) {
        if (!signUpRequest.getP1().equals(signUpRequest.getP2())) {
                return new ResponseEntity(new ApiResponse(false,
        "Passwords not matched!"), HttpStatus.BAD_REQUEST);
        }
        if (userRepository.existsByUsername(signUpRequest.getName())) {
                return new ResponseEntity(new ApiResponse(false,
                                "Username is already taken!"), HttpStatus.BAD_REQUEST);
```

```
                }
                if (userRepository.existsByEmail(signUpRequest.getEmail())) {
                        return new ResponseEntity(new ApiResponse(false,
                                        "Email Address already in use!"),
HttpStatus.BAD_REQUEST);
                }
                // Creating user's account
                User user = new User(signUpRequest.getName(), signUpRequest.getName(),
                                signUpRequest.getEmail(), signUpRequest.getP1(),
                                signUpRequest.getContact());
                user.setPassword(passwordEncoder.encode(user.getPassword()));
                Role userRole = roleRepository.findByName(RoleName.ROLE_USER)
                                .orElseThrow(() -> new AppException("User Role not set."));
                user.setRoles(Collections.singleton(userRole));
                User result = userRepository.save(user);
                URI location =
ServletUriComponentsBuilder.fromCurrentContextPath().path("/users/{username}").buildAndExp
and(result.getUsername()).toUri();
                return ResponseEntity.created(location).body(
                                new ApiResponse(true, "User registered successfully"));
        }}
```

CartController.java

```
package com.example.polls.controller;
import java.time.Instant;
import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;
import java.util.Optional;
import java.util.Set;
```

```
import javax.validation.Valid;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.ResponseEntity;

import org.springframework.security.authentication.AnonymousAuthenticationToken;

import org.springframework.security.core.Authentication;

import org.springframework.security.core.context.SecurityContextHolder;

import org.springframework.security.core.userdetails.UsernameNotFoundException;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RestController;

import com.example.polls.model.Cart;

import com.example.polls.model.User;

import com.example.polls.payload.CartRequest;

import com.example.polls.payload.CartResponse;

import com.example.polls.payload.ResponseOutput;

import com.example.polls.repository.CartRepository;

import com.example.polls.repository.UserRepository;

import com.example.polls.security.CurrentUser;

import com.example.polls.security.UserPrincipal;

import com.example.polls.service.CartService;

import com.fasterxml.jackson.core.JsonProcessingException;

import com.fasterxml.jackson.core.type.TypeReference;

import com.fasterxml.jackson.databind.ObjectMapper;
```

```
@RestController
//@RequestMapping("/api/polls")
public class CartController {
   @Autowired
   private CartRepository cartRepository;
     @Autowired
   private UserRepository userRepository;
   @Autowired
   private CartService cartService;
   private static final Logger logger = LoggerFactory.getLogger(CartController.class);
   //@GetMapping("/getcartitem")
      @PostMapping("/addtocart")
   public ResponseEntity<?> addtocart(@CurrentUser UserPrincipal currentUser,@Valid
@RequestBody CartRequest cartRequest) {
          Cart cart = new Cart();
                  Optional<Cart> cartItem =
cartRepository.findByWhichuser(currentUser.getUsername());
                  if(cartItem.isPresent()){
                          cart = cartItem.get();
                  }
                  double total = cart.getTotal();
                  total = total+ (cartRequest.getPrice() * cartRequest.getQty());
          cart.setTotal(total);
          cart.setCreatedAt(Instant.now());
          cart.setWhichuser(currentUser.getUsername());
          List<CartRequest> recipe = new ArrayList<>();
          recipe.add(cartRequest);
```

```
        ObjectMapper mapper = new ObjectMapper();
        try {
                String recipeJson = mapper.writeValueAsString(recipe);
                cart.setRecipeItems(recipeJson);
            } catch (JsonProcessingException e) {
                e.printStackTrace();
            }
        cartService.addtocart(cart);
        return ResponseEntity.ok(new ResponseOutput("All ok"));
    }
    @PutMapping("/updatecart")
    public ResponseEntity<?> updateFeedback(@CurrentUser UserPrincipal currentUser,@Valid
@RequestBody CartRequest cartRequest) {
        Cart cart = new Cart();
        cart.setId(cartRequest.getId());
        double total = cart.getTotal();
            total = total+ (cartRequest.getPrice() * cartRequest.getQty());
        cart.setTotal(total);
        cart.setCreatedAt(Instant.now());
        cart.setWhichuser(currentUser.getUsername());
        List<CartRequest> recipe = new ArrayList<>();
        recipe.add(cartRequest);
        cart.setRecipeItems(recipe.toString());
        cartService.updatecart(cart);
        return ResponseEntity.ok(new ResponseOutput("All ok"));
    }
```

```
	@GetMapping("/getcartitem")
	public ResponseEntity<?> getcartitem(@CurrentUser UserPrincipal currentUser) {
		CartResponse cartResponse =new CartResponse();
		Optional<Cart> cartItem =
cartRepository.findByWhichuser(currentUser.getUsername());
		if(cartItem.isPresent()){
			Cart cart = cartItem.get();
			cartResponse.setId(cart.getId());
			String recipeItems = cart.getRecipeItems();
			Set<CartRequest> recipe = new HashSet<>();
			 ObjectMapper mapper = new ObjectMapper();
		 try {
			 List<CartRequest> participantJsonList = mapper.readValue(recipeItems,
new TypeReference<List<CartRequest>>(){});
			 recipe= new HashSet<CartRequest>(participantJsonList);
				} catch (JsonProcessingException e) {
				e.printStackTrace();
			}
			cartResponse.setRecipe(recipe);
			cartResponse.setTotal(cart.getTotal());
			cartResponse.setWhichuser(cart.getWhichuser());
		}
		return ResponseEntity.ok(new ResponseOutput(cartResponse));
	}
	@GetMapping("/emptycheck")
	public ResponseEntity<?> emptycheck(@CurrentUser UserPrincipal currentUser) {
		CartResponse cartResponse =new CartResponse();
		Authentication authentication = SecurityContextHolder.getContext()
```

```
                .getAuthentication();
        String whichUser = "";
        if (!(authentication instanceof AnonymousAuthenticationToken)) {
            String currentUserName = authentication.getName();
            User user = userRepository.findByUsernameOrEmail(
                    currentUser.getUsername(), currentUser.getEmail())
                    .orElseThrow(
                            () -> new UsernameNotFoundException(
                                    "User not found with
username or email : "
                                            +
currentUserName));
            whichUser = user.getUsername();
        }
        Optional<Cart> cartItem = cartRepository.findByWhichuser(whichUser);
        if(cartItem.isPresent()){
            return ResponseEntity.ok(new ResponseOutput("not empty cart"));
        }
        return ResponseEntity.ok(new ResponseOutput(cartResponse));
    }}
```

FeedbackController.java

```
package com.example.polls.controller;
import java.util.ArrayList;
import java.util.List;
import javax.validation.Valid;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
```

```
import org.springframework.http.ResponseEntity;

import org.springframework.web.bind.annotation.DeleteMapping;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RestController;

import com.example.polls.model.Feedback;

import com.example.polls.payload.FeedbackRequest;

import com.example.polls.payload.FeedbackResponse;

import com.example.polls.payload.ResponseOutput;

import com.example.polls.repository.FeedbackRepository;

import com.example.polls.service.FeedbackService;

@RestController

//@RequestMapping("/api")

public class FeedbackController {

        @Autowired

        private FeedbackRepository feedbackRepository;

          @Autowired

   private FeedbackService feedbackService;

   private static final Logger logger = LoggerFactory.getLogger(OrderController.class);

   @PostMapping("/feedback")

   public ResponseEntity<?> saveFeedback(@Valid @RequestBody FeedbackRequest
feedbackRequest) {

      // Creating user's account

        Feedback feedback = new
Feedback(feedbackRequest.getEmail(),feedbackRequest.getName(),feedbackRequest.getMsg(),fee
dbackRequest.getWhichuser());
```

```
        feedbackService.saveFeedback(feedback);

        return ResponseEntity.ok(new ResponseOutput("All ok"));

    }

    @PutMapping("/feedback")

    public ResponseEntity<?> updateFeedback(@Valid @RequestBody FeedbackRequest
feedbackRequest) {

    // Creating user's account

        Feedback feedback = new
Feedback(feedbackRequest.getId(),feedbackRequest.getEmail(),feedbackRequest.getName(),feed
backRequest.getMsg(),feedbackRequest.getWhichuser());

        feedbackService.updateFeedback(feedback);

        return ResponseEntity.ok(new ResponseOutput("All ok"));

    }

    @DeleteMapping("/admin/deletefeedback/{feedbackId}")

    public ResponseEntity<?> deleteFeedback(@PathVariable String feedbackId) {

    // Creating user's account

        Feedback feedback =  feedbackRepository.getById(Long.valueOf(feedbackId));

        feedbackService.deleteFeedback(feedback);

        return ResponseEntity.ok(new ResponseOutput("All ok"));

    }

    @GetMapping("/feedback/{feedbackId}")

    public ResponseEntity<?> getFeedback(@PathVariable String feedbackId) {

    // Creating user's account

        Feedback feedback =  feedbackRepository.getById(Long.valueOf(feedbackId));

                return ResponseEntity.ok(new ResponseOutput(feedback));

    }   @GetMapping("/admin/getallfeedbback")

    public ResponseEntity<?> getFeedbacks() {

    // Creating user's account

        List<Feedback> feedbacks =  feedbackRepository.findAll();
```

```
        List<FeedbackResponse>feedbacksList = new ArrayList<>();
        for(Feedback feedback:feedbacks){
                FeedbackResponse feedbackResponse=new FeedbackResponse();
                feedbackResponse.setId(feedback.getId());
                feedbackResponse.setEmail(feedback.getEmail());
                feedbackResponse.setMsg(feedback.getMsg());
                feedbackResponse.setName(feedback.getName());
                feedbackResponse.setWhichuser(feedback.getWhichuser());
                feedbacksList.add(feedbackResponse);
        }                return ResponseEntity.ok(new ResponseOutput(feedbacksList));
   }}
```

OrderController.java

```
package com.example.polls.controller;
import java.util.List;
import javax.validation.Valid;
import javax.websocket.server.PathParam;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
```

```
import com.example.polls.model.Order;
import com.example.polls.payload.OrderRequest;
import com.example.polls.payload.ResponseOutput;
import com.example.polls.repository.OrderRepository;
import com.example.polls.service.OrderService;
@RestController
// @RequestMapping("/api")
public class OrderController {
	@Autowired
	private OrderRepository orderRepository;
	@Autowired
	private OrderService orderService;
	@PostMapping("/orders")
	public ResponseEntity<?> saveOrder(
	@Valid @RequestBody OrderRequest orderRequest) {
		// Creating user's account
		Order order = new Order();
		orderService.saveOrder(order);
		return ResponseEntity.ok(new ResponseOutput("All ok"));
	}	@PutMapping("/orders")
	public ResponseEntity<?> updateOrder(
			@Valid @RequestBody OrderRequest orderRequest) {
		// Creating user's account
		Order order = new Order();
		orderService.updateOrder(order);
		return ResponseEntity.ok(new ResponseOutput("All ok"));
	}
```

```
@DeleteMapping("/admin/deleteorder/{orderId}")
public ResponseEntity<?> deleteOrder(@PathVariable String orderId) {
        // Creating user's account
        Order order = orderRepository.getById(Long.getLong(orderId));
        orderService.deleteOrder(order);
        return ResponseEntity.ok(new ResponseOutput("All ok"));
}
@GetMapping("/orders/{orderId}")
public ResponseEntity<?> getOrder(@PathVariable String orderId) {
        // Creating user's account
        Order order = orderRepository.getById(Long.getLong(orderId));
        return ResponseEntity.ok(new ResponseOutput(order));
}
@GetMapping("/admin/getallorder")
public ResponseEntity<?> getOrders() {
        // Creating user's account
        List<Order> orders = orderRepository.findAll();
        return ResponseEntity.ok(new ResponseOutput(orders));
}}
```

RecipeController.java

```
package com.example.polls.controller;
import java.util.ArrayList;
import java.util.List;
import javax.validation.Valid;
import javax.websocket.server.PathParam;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.DeleteMapping;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import com.example.polls.model.Recipe;
import com.example.polls.payload.RecipeOutput;
import com.example.polls.payload.RecipeRequest;
import com.example.polls.payload.RecipeResponse;
import com.example.polls.payload.ResponseOutput;
import com.example.polls.repository.RecipeRepository;
import com.example.polls.repository.UserRepository;
import com.example.polls.service.RecipeService;
@RestController
public class RecipeController {
        @Autowired
   private RecipeRepository recipeRepository;
     @Autowired
```

```
    private UserRepository userRepository;

    @Autowired

    private RecipeService recipeService;

    private static final Logger logger = LoggerFactory.getLogger(CartController.class);

    //@PostMapping("/admin/addrecipe")

    @RequestMapping(path = "/admin/addrecipe",

            consumes = MediaType.MULTIPART_FORM_DATA_VALUE, produces =
MediaType.APPLICATION_JSON_VALUE,

           method = { RequestMethod.POST})

    public ResponseEntity<?> saverecipe(@Valid @ModelAttribute RecipeRequest recipeRequest) {
                // Creating user's account

            //         recipeService.saveRecipe(recipeRequest);

            return ResponseEntity.status(HttpStatus.CREATED).body(new RecipeOutput());

    }

      // @PostMapping("/admin/editrecipewithoutimage")

    @RequestMapping(path = "/admin/editrecipewithoutimage",

                        //headers = "Accept= application/json, text/plain, */*, Content-
type=application/json",

           method = { RequestMethod.GET})

    public ResponseEntity<?> updaterecipe(@RequestParam String id,

                        @RequestParam String recipename,

                        @RequestParam String quantity,

                        @RequestParam String price) {

            Recipe recipe1 =  recipeRepository.getById(Long.valueOf(id));

            Recipe recipe = new Recipe(Long.valueOf(id),recipename,quantity
,Double.valueOf(price),recipe1.getDishimage());

            recipe.setCreatedAt(recipe1.getCreatedAt());


        // Creating user's account
```

```
            recipeService.updateRecipe(recipe);


            return ResponseEntity.status(HttpStatus.CREATED).body(new RecipeOutput());
    }


   // @PostMapping("/admin/editrecipewithimage")
    @RequestMapping(path = "/admin/editrecipewithimage",
                    consumes = MediaType.MULTIPART_FORM_DATA_VALUE, produces =
MediaType.APPLICATION_JSON_VALUE,
          method = { RequestMethod.POST})
    public ResponseEntity<?> updaterecipe(@Valid @ModelAttribute RecipeRequest
recipeRequest) {
       // Creating user's account
                    recipeService.updateRecipe(recipeRequest);
                    return ResponseEntity.status(HttpStatus.CREATED).body(new RecipeOutput());
    }
    @DeleteMapping("/admin/deleterecipe/{recipeId}")
    public ResponseEntity<?> deleteRecipe(@PathVariable String  recipeId) {
       // Creating user's account
            Recipe recipe =  recipeRepository.getById(Long.valueOf(recipeId));
            recipeService.deleteRecipe(recipe);
            return ResponseEntity.ok(new ResponseOutput("All ok"));
    }
       @GetMapping("/admin/recipe/{recipeId}")
    public ResponseEntity<?> getrecipe(@PathVariable String recipeId) {
       // Creating user's account
            Recipe recipe =  recipeRepository.getById(Long.valueOf(recipeId));
```

```
        return ResponseEntity.ok(new ResponseOutput(recipe));
}    @GetMapping("/admin/getallrecipe")
public ResponseEntity<?> getrecipes() {
   // Creating user's account
        List<Recipe> recipes =  recipeRepository.findAll();
        List<RecipeResponse> recipesList = new ArrayList<>();
        for(Recipe recipe:recipes){
                RecipeResponse recipeResponse=new RecipeResponse();
                recipeResponse.setId(recipe.getId());
                recipeResponse.setDishimage(recipe.getDishimage());
                recipeResponse.setDishname(recipe.getDishname());
                recipeResponse.setPrice(recipe.getPrice());
                recipeResponse.setQuantity(recipe.getQuantity());
                recipesList.add(recipeResponse);
        }          return ResponseEntity.ok(new ResponseOutput(recipesList));
}
@GetMapping("/getallrecipe")
public ResponseEntity<?> getallrecipe() {
   // Creating user's account
        List<Recipe> recipes =  recipeRepository.findAll();
        List<RecipeResponse> recipesList = new ArrayList<>();
        for(Recipe recipe:recipes){
                RecipeResponse recipeResponse=new RecipeResponse();
                recipeResponse.setId(recipe.getId());
                recipeResponse.setDishimage(recipe.getDishimage());
                recipeResponse.setDishname(recipe.getDishname());
                recipeResponse.setPrice(recipe.getPrice());
```

```
                recipeResponse.setQuantity(recipe.getQuantity());
                recipesList.add(recipeResponse);
        }       return ResponseEntity.ok(new ResponseOutput(recipesList));
  }}
```

UserController.java

```
package com.example.polls.controller;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import com.example.polls.payload.UserIdentityAvailability;
import com.example.polls.payload.UserSummary;
import com.example.polls.repository.UserRepository;
import com.example.polls.security.CurrentUser;
import com.example.polls.security.UserPrincipal;
@RestController
@RequestMapping("/api")
public class UserController {
  @Autowired
  private UserRepository userRepository;
  private static final Logger logger = LoggerFactory.getLogger(UserController.class);
  @GetMapping("/user/me")
  @PreAuthorize("hasRole('USER')")
```

```
    public UserSummary getCurrentUser(@CurrentUser UserPrincipal currentUser) {

        UserSummary userSummary = new UserSummary(currentUser.getId(),
currentUser.getUsername(), currentUser.getName());

        return userSummary;

    }    @GetMapping("/user/checkUsernameAvailability")

    public UserIdentityAvailability checkUsernameAvailability(@RequestParam(value =
"username") String username) {

        Boolean isAvailable = !userRepository.existsByUsername(username);

        return new UserIdentityAvailability(isAvailable);

    }    @GetMapping("/user/checkEmailAvailability")

    public UserIdentityAvailability checkEmailAvailability(@RequestParam(value = "email") String
email) {

        Boolean isAvailable = !userRepository.existsByEmail(email);

        return new UserIdentityAvailability(isAvailable);

    }}
```

Exception

AppException.java

```
package com.example.polls.exception;

import org.springframework.http.HttpStatus;

import org.springframework.web.bind.annotation.ResponseStatus;

@ResponseStatus(HttpStatus.INTERNAL_SERVER_ERROR)

public class AppException extends RuntimeException {

    public AppException(String message) {

        super(message);

    }

    public AppException(String message, Throwable cause) {

        super(message, cause);

    }}
```

BadRequestException.java

```
package com.example.polls.exception;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;
@ResponseStatus(HttpStatus.BAD_REQUEST)
public class BadRequestException extends RuntimeException {
   public BadRequestException(String message) {
      super(message);
   }   public BadRequestException(String message, Throwable cause) {
      super(message, cause);
   }}
```

ResourceNotFoundException.java

```
package com.example.polls.exception;
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;
@ResponseStatus(HttpStatus.NOT_FOUND)
public class ResourceNotFoundException extends RuntimeException {
   private String resourceName;
   private String fieldName;
   private Object fieldValue;
   public ResourceNotFoundException( String resourceName, String fieldName, Object fieldValue) {
      super(String.format("%s not found with %s : '%s'", resourceName, fieldName, fieldValue));
      this.resourceName = resourceName;
      this.fieldName = fieldName;
      this.fieldValue = fieldValue;
   }   public String getResourceName() {
      return resourceName;
   }
```

```
    public String getFieldName() {
        return fieldName;
    }    public Object getFieldValue() {
        return fieldValue;
    }}
```

Model

Cart.java

```
package com.example.polls.model;
import java.util.HashSet;
import java.util.Set;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.Table;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.Size;
import com.example.polls.model.audit.DateAudit;
@Entity
@Table(name = "cart")
public class Cart extends DateAudit {
```

```
        private static final long serialVersionUID = 1L;
        @Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;
        @NotBlank
@Size(max = 40)
private String whichuser;
/* @ManyToMany(fetch = FetchType.LAZY)
@JoinTable(name = "cart_recipes",
      joinColumns = @JoinColumn(name = "cart_id"),
      inverseJoinColumns = @JoinColumn(name = "recipe_id"))
        private Set<Recipe> recipe = new HashSet<>();
*/
        //@NotBlank
//@Size(max = 40)
        private String recipeItems;
           private double total = 0;
        public Cart() {
                super();
        }       public String getRecipeItems() {
                return recipeItems;     }
        public void setRecipeItems(String recipeItems) {
                this.recipeItems = recipeItems;
        }       public Long getId() {
                return id;}
        public void setId(Long id) {
                this.id = id;       }
```

```
        public String getWhichuser() {
                return whichuser;
        }       public void setWhichuser(String whichuser) {
                this.whichuser = whichuser;
        }       public double getTotal() {
                return total;
        }       public void setTotal(double total) {
                this.total = total;
        }}
```

Feedback.java

```
package com.example.polls.model;
//import java.util.HashSet;
//import java.util.Set;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
//import javax.persistence.UniqueConstraint;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.Size;
import com.example.polls.model.audit.DateAudit;
@Entity
@Table(name = "feedback")
public class Feedback extends DateAudit {
   /**
         *
```

```
	 */
	private static final long serialVersionUID = 1L;
	@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;
	@NotBlank
@Size(max = 40)
private String whichuser;
		@NotBlank
@Size(max = 40)
private String email;
	@NotBlank
@Size(max = 40)
private String name;
	@NotBlank
@Size(max = 40)
private String msg;
	public Feedback() {
		super();
	}
	public Feedback(
			@NotBlank @Size(max = 40) String email,
			@NotBlank @Size(max = 40) String name,
			@NotBlank @Size(max = 40) String msg,
			@NotBlank @Size(max = 40) String whichuser) {
		super();
		this.whichuser = whichuser;
```

```
		this.email = email;
		this.name = name;
		this.msg = msg;
	}	public Feedback(Long id,
			@NotBlank @Size(max = 40) String email,
			@NotBlank @Size(max = 40) String name,
			@NotBlank @Size(max = 40) String msg, @NotBlank @Size(max = 40)
String whichuser) {
		super();
		this.id = id;
		this.whichuser = whichuser;
		this.email = email;
		this.name = name;
		this.msg = msg;
	}public Long getId() {
		return id;
	}public void setId(Long id) {
		this.id = id;
	}public String getWhichuser() {
		return whichuser;
	}	public void setWhichuser(String whichuser) {
		this.whichuser = whichuser;
	}	public String getEmail() {
		return email;
	}	public void setEmail(String email) {
		this.email = email;
	}	public String getName() {
		return name;
```

```
    }       public void setName(String name) {
            this.name = name;
    }       public String getMsg() {
            return msg;
    }       public void setMsg(String msg) {
            this.msg = msg;
    }}
```

Order.java

```
package com.example.polls.model;
import java.util.HashSet;
import java.util.Set;
import javax.persistence.Entity;
import javax.persistence.FetchType;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.Table;
//import javax.persistence.UniqueConstraint;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.Size;
import com.example.polls.model.audit.DateAudit;
@Entity
@Table(name = "order_details")
public class Order extends DateAudit {
```

```
        private static final long serialVersionUID = 1L;
@Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;
        @NotBlank
   @Size(max = 40)
   private String whichuser;
   @ManyToMany(fetch = FetchType.LAZY)
   @JoinTable(name = "order_recipes",
       joinColumns = @JoinColumn(name = "order_details_id"),
       inverseJoinColumns = @JoinColumn(name = "recipe_id"))
         private Set<Recipe> recipe = new HashSet<>();
                  @NotBlank
   @Size(max = 40)
   private double total;
                  public String getWhichuser() {
                  return whichuser;
         }        public void setWhichuser(String whichuser) {
                  this.whichuser = whichuser;
         }        public double getTotal() {
                  return total;
         }public void setTotal(double total) {
                  this.total = total;
         }public Long getId() {
                  return id;
         }public void setId(Long id) {
                  this.id = id;
```

```
        }public Set<Recipe> getRecipe() {
                return recipe;
        }public void setRecipe(Set<Recipe> recipe) {
                this.recipe = recipe;
        }}
```

Otp.java

```
package com.example.polls.model;
//import java.util.HashSet;
//import java.util.Set;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
//import javax.persistence.UniqueConstraint;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.Size;
import com.example.polls.model.audit.DateAudit;
@Entity
@Table(name = "otp")
public class Otp extends DateAudit {
   private static final long serialVersionUID = 1L;
@Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;
public double getOtp() {
                return otp;
```

```
        }       public void setOtp(double otp) {
                this.otp = otp;
        }       public String getEmail() {
                return email;
        }       public void setEmail(String email) {
                this.email = email;
        }@NotBlank
    @Size(max = 40)
    private double otp;
        @NotBlank
    @Size(max = 40)
    private String email;}
```

Recipe.java

```
package com.example.polls.model;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.Table;
//import javax.persistence.UniqueConstraint;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.Size;
import com.example.polls.model.audit.DateAudit;
@Entity
@Table(name = "recipe")
public class Recipe extends DateAudit {
        private static final long serialVersionUID = 1L;
```

```
        @Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;
        @NotBlank
@Size(max = 40)
private String dishname;
@NotBlank
@Size(max = 40)
private String quantity;
   private double price;
  @NotBlank
@Size(max = 40)
private String dishimage;
   public Recipe(Long id, @NotBlank @Size(max = 40) String dishname,
                        @NotBlank @Size(max = 40) String quantity,
                        @NotBlank @Size(max = 40) double price) {
                super();
                this.id = id;
                this.dishname = dishname;
                this.quantity = quantity;
                this.price = price;
        }       public Recipe() {
                super();
        }       public Recipe(@NotBlank @Size(max = 40) String dishname,
                        @NotBlank @Size(max = 40) String quantity,
                        @NotBlank @Size(max = 40) double price,
```

```
                @NotBlank @Size(max = 40) String dishimage) {
        super();
        this.dishname = dishname;
        this.quantity = quantity;
        this.price = price;
        this.dishimage = dishimage;
}public Recipe(Long id, @NotBlank @Size(max = 40) String dishname,
                @NotBlank @Size(max = 40) String quantity,
                @NotBlank @Size(max = 40) double price,
                @NotBlank @Size(max = 40) String dishimage) {
        super();
        this.id = id;
        this.dishname = dishname;
        this.quantity = quantity;
        this.price = price;
        this.dishimage = dishimage;
}public String getDishname() {
        return dishname;
}       public void setDishname(String dishname) {
        this.dishname = dishname;
}public Long getId() {
        return id;
}public void setId(Long id) {
        this.id = id;
}       public String getQuantity() {
        return quantity;
}
```

```
        public void setQuantity(String quantity) {
                this.quantity = quantity;
        }public double getPrice() {
                return price;
        }       public void setPrice(double price) {
                this.price = price;
        }       public String getDishimage() {
                return dishimage;
        }       public void setDishimage(String dishimage) {
                this.dishimage = dishimage;
        } }
```

Role.java

```
package com.example.polls.model;
import org.hibernate.annotations.NaturalId;
import javax.persistence.*;
@Entity
@Table(name = "roles")
public class Role {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;
   @Enumerated(EnumType.STRING)
   @NaturalId
   @Column(length = 60)
   private RoleName name;
   public Role() {
```

```
    }
    public Role(RoleName name) {
        this.name = name;    }
    public Long getId() {
        return id;    }
    public void setId(Long id) {
        this.id = id;    }
    public RoleName getName() {
        return name;    }
    public void setName(RoleName name) {
        this.name = name;    }}
```

RoleName.java

```
package com.example.polls.model;

public enum  RoleName {
    ROLE_USER,
    ROLE_ADMIN
}
```

User.java

```
package com.example.polls.model;

import java.util.HashSet;

import java.util.Set;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.JoinTable;

import javax.persistence.ManyToMany;
```

```
import javax.persistence.Table;
import javax.persistence.UniqueConstraint;
import javax.validation.constraints.Email;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.Size;
import org.hibernate.annotations.NaturalId;
import com.example.polls.model.Role;
import com.example.polls.model.audit.DateAudit;
@Entity
@Table(name = "users", uniqueConstraints = {
      @UniqueConstraint(columnNames = {
         "username"       }),
      @UniqueConstraint(columnNames = {
         "email"       })})
public class User extends DateAudit {
   private static final long serialVersionUID = 1L;
@Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;
   @NotBlank
   @Size(max = 40)
   private String name;
      @NotBlank
   @Size(max = 40)
   private String contact;
   @NotBlank
   @Size(max = 15)
```

```
private String username;
@NaturalId
@NotBlank
@Size(max = 40)
@Email
private String email;
@NotBlank
@Size(max = 100)
private String password;
  private Boolean blocked;
@ManyToMany(fetch = FetchType.LAZY)
@JoinTable(name = "user_roles",
      joinColumns = @JoinColumn(name = "user_id"),
      inverseJoinColumns = @JoinColumn(name = "role_id"))
private Set<Role> roles = new HashSet<>();
public User() {   }
public User(String name, String username, String email, String password) {
   this.name = name;
   this.username = username;
   this.email = email;
   this.password = password;   }
   public User(@NotBlank @Size(max = 40) String name,
                        @NotBlank @Size(max = 15) String username,
                        @NotBlank @Size(max = 40) @Email String email,
                        @NotBlank @Size(max = 100) String password,
                        @NotBlank @Size(max = 40) String contact) {
                  super();
```

```
                this.name = name;
                this.contact = contact;
                this.username = username;
                this.email = email;
                this.password = password;
        }public Long getId() {
    return id;    }
public void setId(Long id) {
    this.id = id;   }
public String getUsername() {
    return username;    }
 public void setUsername(String username) {
    this.username = username;    }
 public String getName() {
    return name;    }
 public void setName(String name) {
    this.name = name;    }
 public String getEmail() {
    return email;    }
 public void setEmail(String email) {
    this.email = email;    }
 public String getPassword() {
    return password;    }
 public void setPassword(String password) {
    this.password = password;    }
 public Set<Role> getRoles() {
    return roles;    }
```

```
    public void setRoles(Set<Role> roles) {
        this.roles = roles;    }
            public String getContact() {
                    return contact; }
            public void setContact(String contact) {
                    this.contact = contact;   }
            public Boolean getBlocked() {
                    return blocked; }
            public void setBlocked(Boolean blocked) {
                    this.blocked = blocked;}}
```

Audit

DataAudit.java

```
package com.example.polls.model.audit;
import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
import org.springframework.data.annotation.CreatedDate;
import org.springframework.data.annotation.LastModifiedDate;
import org.springframework.data.jpa.domain.support.AuditingEntityListener;
import javax.persistence.EntityListeners;
import javax.persistence.MappedSuperclass;
import java.io.Serializable;
import java.time.Instant;
@MappedSuperclass
@EntityListeners(AuditingEntityListener.class)
@JsonIgnoreProperties(
        value = {"createdAt", "updatedAt"},
        allowGetters = true
)
```

```
public abstract class DateAudit implements Serializable {
   @CreatedDate
   private Instant createdAt;
   @LastModifiedDate
   private Instant updatedAt;
   public Instant getCreatedAt() {
      return createdAt;
   }   public void setCreatedAt(Instant createdAt) {
      this.createdAt = createdAt;   }
   public Instant getUpdatedAt() {
      return updatedAt;   }
   public void setUpdatedAt(Instant updatedAt) {
      this.updatedAt = updatedAt;  }}
```

UserDateAudit.java

```
package com.example.polls.model.audit;
import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
import org.springframework.data.annotation.CreatedBy;
import org.springframework.data.annotation.LastModifiedBy;
import javax.persistence.MappedSuperclass;
@MappedSuperclass
@JsonIgnoreProperties(
      value = {"createdBy", "updatedBy"},
      allowGetters = true)
public abstract class UserDateAudit extends DateAudit {
   @CreatedBy
   private Long createdBy;
   @LastModifiedBy
```

```
    private Long updatedBy;

    public Long getCreatedBy() {

        return createdBy;

    }    public void setCreatedBy(Long createdBy) {

        this.createdBy = createdBy;    }

    public Long getUpdatedBy() {

        return updatedBy;    }

    public void setUpdatedBy(Long updatedBy) {

        this.updatedBy = updatedBy;   }}
```

Payload---ApiResponse.java

```
package com.example.polls.payload;

public class ApiResponse {
    private Boolean success;
    private String message;

    public ApiResponse(Boolean success, String message) {
        this.success = success;
        this.message = message;    }

    public Boolean getSuccess() {
        return success;    }

    public void setSuccess(Boolean success) {
        this.success = success;    }

    public String getMessage() {
        return message;    }
    public void setMessage(String message) {
        this.message = message;    }}
```

CartRequest.java

```
package com.example.polls.payload;

import java.util.HashSet;

import java.util.Set;

import com.example.polls.model.Recipe;

import com.fasterxml.jackson.annotation.JsonProperty;

public class CartRequest {@JsonProperty("_id")

        private Long id;
```

```
private String dishname;
private String quantity;
private double price;
private String dishimage;
private int qty;
        public Long getId() {
        return id;        }
public void setId(Long id) {
        this.id = id;       }
public String getDishname() {
        return dishname;        }
public void setDishname(String dishname) {
        this.dishname = dishname;       }
public String getQuantity() {
        return quantity;        }
public void setQuantity(String quantity) {
        this.quantity = quantity;        }
public double getPrice() {
        return price;     }
public void setPrice(double price) {
        this.price = price;        }
public String getDishimage() {
        return dishimage;        }
public void setDishimage(String dishimage) {
        this.dishimage = dishimage;      }
public int getQty() {
        return qty;       }
```

```
	public void setQty(int qty) {
		this.qty = qty;	}
	/*//@NotBlank
//@Size(max = 140)
	private Long id;
	private String whichuser;
	private Set<Recipe> recipe = new HashSet<>();
	private double total;
	public Long getId() {
		return id;
	}
	public void setId(Long id) {
		this.id = id;
	}
	public String getWhichuser() {
		return whichuser;
	}
	public void setWhichuser(String whichuser) {
		this.whichuser = whichuser;
	}
	public Set<Recipe> getRecipe() {
		return recipe;
	}
	public void setRecipe(Set<Recipe> recipe) {
		this.recipe = recipe;
	}
	public double getTotal() {
```

```
		return total;
	}
	public void setTotal(double total) {
		this.total = total;
	}*/}
```

CartResponse.java

```
package com.example.polls.payload;

import java.util.HashSet;
import java.util.Set;

import com.example.polls.model.Recipe;
import com.fasterxml.jackson.annotation.JsonProperty;

public class CartResponse {

	@JsonProperty("_id")
	 private Long id;
	 private String whichuser;
	 private Set<CartRequest> recipe = new HashSet<>();
	 private double total;

	 public Long getId() {
		return id;
	}
	public void setId(Long id) {
```

```
		this.id = id;
	}
	public String getWhichuser() {
		return whichuser;
	}
	public void setWhichuser(String whichuser) {
		this.whichuser = whichuser;
	}
	public Set<CartRequest> getRecipe() {
		return recipe;
	}
	public void setRecipe(Set<CartRequest> recipe) {
		this.recipe = recipe;
	}
	public double getTotal() {
		return total;
	}
	public void setTotal(double total) {
		this.total = total;
	}

}
```

ChoiceRequest.java

```
package com.example.polls.payload;
import javax.validation.constraints.NotBlank;
```

```
import javax.validation.constraints.Size;

public class ChoiceRequest {

   @NotBlank

   @Size(max = 40)

   private String text;

   public String getText() {

      return text;

   }

   public void setText(String text) {

      this.text = text;

   }

}
```

ChoiceResponse.java

```
package com.example.polls.payload;

public class ChoiceResponse {
    private long id;
    private String text;
    private long voteCount;

    public long getId() {
        return id;
    }

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

    public String getText() {
        return text;
    }

    public void setText(String text) {
        this.text = text;
    }

    public long getVoteCount() {
        return voteCount;
    }

    public void setVoteCount(long voteCount) {
        this.voteCount = voteCount;
    }
```

```
}
```

FeedBackRequest.java

```
package com.example.polls.payload;

public class FeedbackRequest {

	private Long id;
	private String whichuser;
	private String email;
	private String name;
	private String msg;

	public Long getId() {
		return id;
	}
	public void setId(Long id) {
		this.id = id;
	}
	public String getWhichuser() {
		return whichuser;
	}
	public void setWhichuser(String whichuser) {
		this.whichuser = whichuser;
	}
	public String getEmail() {
		return email;
	}
	public void setEmail(String email) {
		this.email = email;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public String getMsg() {
		return msg;
	}
	public void setMsg(String msg) {
		this.msg = msg;
	}

}
```

FeedBackResponse.java

```
package com.example.polls.payload;

import com.fasterxml.jackson.annotation.JsonProperty;

public class FeedbackResponse {
	@JsonProperty("_id")
	private Long id;
	private String whichuser;
	private String email;
	private String name;
```

```
        private String msg;

        public Long getId() {
               return id;
        }
        public void setId(Long id) {
               this.id = id;
        }
        public String getWhichuser() {
               return whichuser;
        }
        public void setWhichuser(String whichuser) {
               this.whichuser = whichuser;
        }
        public String getEmail() {
               return email;
        }
        public void setEmail(String email) {
               this.email = email;
        }
        public String getName() {
               return name;
        }
        public void setName(String name) {
               this.name = name;
        }
        public String getMsg() {
               return msg;
        }
        public void setMsg(String msg) {
               this.msg = msg;
        }

}
```

jwtAuthenticationResponse.java

```
package com.example.polls.payload;

public class JwtAuthenticationResponse {
    private String token;
    private boolean admin = false;

    public JwtAuthenticationResponse(String token,boolean admin) {
        this.token = token;
        this.admin = admin;
    }
    public String getToken() {
               return token;
        }

        public void setToken(String token) {
               this.token = token;
        }
        public boolean isAdmin() {
               return admin;
        }
        public void setAdmin(boolean admin) {
               this.admin = admin;
        }
```

```
}
```

LoginRequest.java

```
package com.example.polls.payload;

import javax.validation.constraints.NotBlank;

public class LoginRequest {
    @NotBlank
    private String email;

    @NotBlank
    private String password;



    public String getEmail() {
            return email;
       }

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

       public String getPassword() {
         return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

OrderRequest.java

```
package com.example.polls.payload;



import java.util.HashSet;

import java.util.Set;



import com.example.polls.model.Recipe;



public class OrderRequest

{

        private Long id;

        private String whichuser;
```

```
 private Set<Recipe> recipe = new HashSet<>();
 private double total;

 public Long getId() {
        return id;
}
public void setId(Long id) {
        this.id = id;
}
public String getWhichuser() {
        return whichuser;
}
public void setWhichuser(String whichuser) {
        this.whichuser = whichuser;
}
public Set<Recipe> getRecipe() {
        return recipe;
}
public void setRecipe(Set<Recipe> recipe) {
        this.recipe = recipe;
}
public double getTotal() {
        return total;
}
public void setTotal(double total) {
        this.total = total;
}
```

```
	}
```

OrderResponse.java

```
package com.example.polls.payload;


import java.util.HashSet;
import java.util.Set;


import com.example.polls.model.Recipe;
import com.fasterxml.jackson.annotation.JsonProperty;


public class OrderResponse {
	@JsonProperty("_id")
	 private Long id;
	 private String whichuser;
	 private Set<Recipe> recipe = new HashSet<>();
	 private double total;


	 public Long getId() {
		return id;
	}
	public void setId(Long id) {
		this.id = id;
	}
	public String getWhichuser() {
		return whichuser;
	}
	public void setWhichuser(String whichuser) {
```

```
            this.whichuser = whichuser;

        }

        public Set<Recipe> getRecipe() {

            return recipe;

        }

        public void setRecipe(Set<Recipe> recipe) {

            this.recipe = recipe;

        }

        public double getTotal() {

            return total;

        }

        public void setTotal(double total) {

            this.total = total;

        }       }
```

PagedResponse.java

```
package com.example.polls.payload;

import java.util.List;

public class PagedResponse<T> {

    private List<T> content;
    private int page;
    private int size;
    private long totalElements;
    private int totalPages;
    private boolean last;

    public PagedResponse() {

    }

    public PagedResponse(List<T> content, int page, int size, long totalElements,
int totalPages, boolean last) {
        this.content = content;
        this.page = page;
        this.size = size;
        this.totalElements = totalElements;
        this.totalPages = totalPages;
        this.last = last;
    }
```

```
    public List<T> getContent() {
        return content;
    }

    public void setContent(List<T> content) {
        this.content = content;
    }

    public int getPage() {
        return page;
    }

    public void setPage(int page) {
        this.page = page;
    }

    public int getSize() {
        return size;
    }

    public void setSize(int size) {
        this.size = size;
    }

    public long getTotalElements() {
        return totalElements;
    }

    public void setTotalElements(long totalElements) {
        this.totalElements = totalElements;
    }

    public int getTotalPages() {
        return totalPages;
    }

    public void setTotalPages(int totalPages) {
        this.totalPages = totalPages;
    }

    public boolean isLast() {
        return last;
    }

    public void setLast(boolean last) {
        this.last = last;
    }
}
```

PollLength.java

```
package com.example.polls.payload;



import javax.validation.constraints.Max;

import javax.validation.constraints.NotNull;
```

```
public class PollLength {
    @NotNull
    @Max(7)
    private Integer days;
    @NotNull
    @Max(23)
    private Integer hours;
    public int getDays() {
        return days;
    }
    public void setDays(int days) {
        this.days = days;
    }
    public int getHours() {
        return hours;
    }
    public void setHours(int hours) {
        this.hours = hours;
    }
}
```

PollRequest.java

```
package com.example.polls.payload;
import javax.validation.Valid;
import javax.validation.constraints.NotBlank;
import javax.validation.constraints.NotNull;
import javax.validation.constraints.Size;
import java.util.List;
```

```
public class PollRequest {
    @NotBlank
    @Size(max = 140)
    private String question;
    @NotNull
    @Size(min = 2, max = 6)
    @Valid
    private List<ChoiceRequest> choices;
    @NotNull
    @Valid
    private PollLength pollLength;
    public String getQuestion() {
        return question;
    }
    public void setQuestion(String question) {
        this.question = question;
    }
    public List<ChoiceRequest> getChoices() {
        return choices;
    }
    public void setChoices(List<ChoiceRequest> choices) {
        this.choices = choices;
    }
    public PollLength getPollLength() {
        return pollLength;
    }
    public void setPollLength(PollLength pollLength) {
```

```
        this.pollLength = pollLength;
    }
}
```

PollResponse.java

```
package com.example.polls.payload;
import com.fasterxml.jackson.annotation.JsonInclude;
import java.time.Instant;
import java.util.List;
public class PollResponse {
    private Long id;
    private String question;
    private List<ChoiceResponse> choices;
    private UserSummary createdBy;
    private Instant creationDateTime;
    private Instant expirationDateTime;
    private Boolean isExpired;
    @JsonInclude(JsonInclude.Include.NON_NULL)
    private Long selectedChoice;
    private Long totalVotes;
    public Long getId() {
        return id;
    }

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

```
    return question;
}
public void setQuestion(String question) {
    this.question = question;
}
public List<ChoiceResponse> getChoices() {
    return choices;
}
public void setChoices(List<ChoiceResponse> choices) {
    this.choices = choices;
}
public UserSummary getCreatedBy() {
    return createdBy;
}
public void setCreatedBy(UserSummary createdBy) {
    this.createdBy = createdBy;
}
public Instant getCreationDateTime() {
    return creationDateTime;
}
public void setCreationDateTime(Instant creationDateTime) {
    this.creationDateTime = creationDateTime;
}
public Instant getExpirationDateTime() {
    return expirationDateTime;
}
public void setExpirationDateTime(Instant expirationDateTime) {
```

```
        this.expirationDateTime = expirationDateTime;
    }
    public Boolean getExpired() {
        return isExpired;
    }
    public void setExpired(Boolean expired) {
        isExpired = expired;
    }
    public Long getSelectedChoice() {
        return selectedChoice;
    }
    public void setSelectedChoice(Long selectedChoice) {
        this.selectedChoice = selectedChoice;
    }
    public Long getTotalVotes() {
        return totalVotes;
    }
    public void setTotalVotes(Long totalVotes) {
        this.totalVotes = totalVotes;
    }
}
```

RecipeOutput.java

```
package com.example.polls.payload;

public class RecipeOutput {

	int n =1;
	int nModified=1;
	int ok=1;
	public int getN() {
		return n;
```

```
	}
	public void setN(int n) {
		this.n = n;
	}
	public int getnModified() {
		return nModified;
	}
	public void setnModified(int nModified) {
		this.nModified = nModified;
	}
	public int getOk() {
		return ok;
	}
	public void setOk(int ok) {
		this.ok = ok;
	}



}
```

RecipeRequest.java

```
package com.example.polls.payload;

import org.springframework.web.multipart.MultipartFile;

public class RecipeRequest {
	private Long id;
	private String dishname;
	private String quantity;
	private double price;
	private String dishimage;
	private MultipartFile file;

	public Long getId() {
		return id;
	}
	public void setId(Long id) {
		this.id = id;
	}
	public String getDishname() {
		return dishname;
	}
	public void setDishname(String dishname) {
		this.dishname = dishname;
	}

	public String getQuantity() {
		return quantity;
	}
	public void setQuantity(String quantity) {
		this.quantity = quantity;
	}
	public double getPrice() {
		return price;
	}
	public void setPrice(double price) {
```

```
		this.price = price;
	}
	public String getDishimage() {
		return dishimage;
	}
	public void setDishimage(String dishimage) {
		this.dishimage = dishimage;
	}
	public MultipartFile getFile() {
		return file;
	}
	public void setFile(MultipartFile file) {
		this.file = file;
	}



}
```

RecipeResponse.java

```
package com.example.polls.payload;

import com.fasterxml.jackson.annotation.JsonProperty;

public class RecipeResponse {
	@JsonProperty("_id")
	private Long id;
	private String dishname;
	private String quantity;
	private double price;
	private String dishimage;



	public Long getId() {
		return id;
	}
	public void setId(Long id) {
		this.id = id;
	}
	public String getDishname() {
		return dishname;
	}
	public void setDishname(String dishname) {
		this.dishname = dishname;
	}

	public String getQuantity() {
		return quantity;
	}
	public void setQuantity(String quantity) {
		this.quantity = quantity;
	}
	public double getPrice() {
		return price;
	}
	public void setPrice(double price) {
```

```
		this.price = price;
	}
	public String getDishimage() {
		return dishimage;
	}
	public void setDishimage(String dishimage) {
		this.dishimage = dishimage;
	}



}
```

ResponseOutput.java

```
package com.example.polls.payload;

public class ResponseOutput {

	private Object msg;



	public ResponseOutput() {
		super();
	}

	public ResponseOutput(Object msg) {
		super();
		this.msg = msg;
	}

	public Object getMsg() {
		return msg;
	}

	public void setMsg(Object msg) {
		this.msg = msg;
	}

}
```

SignupRequest.java

```
package com.example.polls.payload;

import javax.validation.constraints.*;

public class SignUpRequest {
    @NotBlank
    @Size(min = 4, max = 40)
    private String name;


    @NotBlank
    @Size(max = 40)
    @Email
    private String email;

    @NotBlank
    @Size(min = 6, max = 20)
```

```
    private String p1;

    @NotBlank
    @Size(min = 6, max = 20)
    private String p2;

    @Size(max = 12)
    private String contact;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }



    public String getEmail() {
        return email;
    }

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



      public String getP1() {
            return p1;
      }

      public void setP1(String p1) {
            this.p1 = p1;
      }

      public String getP2() {
            return p2;
      }

      public void setP2(String p2) {
            this.p2 = p2;
      }

      public String getContact() {
            return contact;
      }

      public void setContact(String contact) {
            this.contact = contact;
      }


}
```

UserIdentityAvailability.java

```
package com.example.polls.payload;
```

```
public class UserIdentityAvailability {
    private Boolean available;

    public UserIdentityAvailability(Boolean available) {
        this.available = available;
    }

    public Boolean getAvailable() {
        return available;
    }

    public void setAvailable(Boolean available) {
        this.available = available;
    }
}
```

UserProfile.java

```
package com.example.polls.payload;

import java.time.Instant;

public class UserProfile {
    private Long id;
    private String username;
    private String name;
    private Instant joinedAt;
    private Long pollCount;
    private Long voteCount;

    public UserProfile(Long id, String username, String name, Instant joinedAt,
Long pollCount, Long voteCount) {
        this.id = id;
        this.username = username;
        this.name = name;
        this.joinedAt = joinedAt;
        this.pollCount = pollCount;
        this.voteCount = voteCount;
    }

    public Long getId() {
        return id;
    }

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

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getName() {
        return name;
    }
```

```
    public void setName(String name) {
        this.name = name;
    }

    public Instant getJoinedAt() {
        return joinedAt;
    }

    public void setJoinedAt(Instant joinedAt) {
        this.joinedAt = joinedAt;
    }

    public Long getPollCount() {
        return pollCount;
    }

    public void setPollCount(Long pollCount) {
        this.pollCount = pollCount;
    }

    public Long getVoteCount() {
        return voteCount;
    }

    public void setVoteCount(Long voteCount) {
        this.voteCount = voteCount;
    }
}
```

UserResponse.java

```
package com.example.polls.payload;


import com.fasterxml.jackson.annotation.JsonIgnore;

import com.fasterxml.jackson.annotation.JsonProperty;


public class UserResponse {


        @JsonProperty("_id")

   private Long id;


   private String name;


   private String contact;
```

```
//@JsonIgnore
private String email;

@JsonIgnore
private String password;

private boolean blocked;

private String role;

    public Long getId() {
        return id;
    }

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

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
```

```
public String getContact() {
        return contact;
}

public void setContact(String contact) {
        this.contact = contact;
}

public String getEmail() {
        return email;
}

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

public String getPassword() {
        return password;
}

public void setPassword(String password) {
        this.password = password;
}

public boolean isBlocked() {
        return blocked;
```

```
        }

        public void setBlocked(boolean blocked) {
                this.blocked = blocked;
        }

        public String getRole() {
                return role;
        }

        public void setRole(String role) {
                this.role = role;
        }

}
```

UserSummary.java

```
package com.example.polls.payload;

public class UserSummary {
    private Long id;
    private String username;
    private String name;

    public UserSummary(Long id, String username, String name) {
        this.id = id;
        this.username = username;
        this.name = name;
    }

    public Long getId() {
        return id;
    }

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

    public String getUsername() {
```

```
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

VoteRequest.java

```
package com.example.polls.payload;
import javax.validation.constraints.NotNull;

public class VoteRequest {
    @NotNull
    private Long choiceId;

    public Long getChoiceId() {
        return choiceId;
    }

    public void setChoiceId(Long choiceId) {
        this.choiceId = choiceId;
    }
}
```

Repository

Cart Repository.java

```
package com.example.polls.repository;

import java.util.Optional;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;



import com.example.polls.model.Cart;

@Repository

public interface CartRepository extends JpaRepository<Cart, Long>{



        Optional<Cart> findByWhichuser(String whichuser);
```

```
}
```

Feedback Repository.java

```
package com.example.polls.repository;


import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;


import com.example.polls.model.Feedback;


@Repository
public interface FeedbackRepository extends JpaRepository<Feedback, Long>
{
  //  Optional<Role> findByName(RoleName roleName);
}
```

Order Repository.java

```
package com.example.polls.repository;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
import com.example.polls.model.Order;


@Repository
public interface OrderRepository extends JpaRepository<Order, Long>
{
  //  Optional<Role> findByName(RoleName roleName);
}
```

Otp Repository.java

```
package com.example.polls.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import com.example.polls.model.Role;

@Repository

public interface OtpRepository extends JpaRepository<Role, Long>

{

  // Optional<Role> findByName(RoleName roleName);

}
```

Recipe Repository.java

```
package com.example.polls.repository;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import com.example.polls.model.Recipe;

@Repository

public interface RecipeRepository extends JpaRepository<Recipe, Long>

{

  // Optional<Role> findByName(RoleName roleName);

}
```

Role Repository.java

```
package com.example.polls.repository;

import com.example.polls.model.Role;

import com.example.polls.model.RoleName;

import org.springframework.data.jpa.repository.JpaRepository;

import org.springframework.stereotype.Repository;

import java.util.Optional;
```

```
@Repository
public interface RoleRepository extends JpaRepository<Role, Long> {
   Optional<Role> findByName(RoleName roleName);
}
```

user Repository.java

```
package com.example.polls.repository;
import com.example.polls.model.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

import java.util.List;
import java.util.Optional;
@Repository
public interface UserRepository extends JpaRepository<User, Long> {
   Optional<User> findByEmail(String email);
   Optional<User> findByUsernameOrEmail(String username, String email);
   List<User> findByIdIn(List<Long> userIds);
   Optional<User> findByUsername(String username);
   Boolean existsByUsername(String username);
   Boolean existsByEmail(String email);
}
```

Security

CurrentUser.java

```
package com.example.polls.security;
import org.springframework.security.core.annotation.AuthenticationPrincipal;
import java.lang.annotation.*;
@Target({ElementType.PARAMETER, ElementType.TYPE})
```

```
@Retention(RetentionPolicy.RUNTIME)
@Documented
@AuthenticationPrincipal
public @interface CurrentUser {
}
```

CustomUserDetailsService.java

```
package com.example.polls.security;

import com.example.polls.exception.ResourceNotFoundException;
import com.example.polls.model.User;
import com.example.polls.repository.UserRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
@Service
public class CustomUserDetailsService implements UserDetailsService {
    @Autowired
    UserRepository userRepository;

    @Override
    @Transactional
    public UserDetails loadUserByUsername(String usernameOrEmail)
            throws UsernameNotFoundException {
        // Let people login with either username or email
```

```
        User user = userRepository.findByUsernameOrEmail(usernameOrEmail, usernameOrEmail)
                .orElseThrow(() ->
                        new UsernameNotFoundException("User not found with username or email : " +
usernameOrEmail)
        );
        return UserPrincipal.create(user);
    }
    @Transactional
    public UserDetails loadUserById(Long id) {
        User user = userRepository.findById(id).orElseThrow(
            () -> new ResourceNotFoundException("User", "id", id)
        );
        return UserPrincipal.create(user);
    }
}
```

JwtAuthenticationEntryPoint.java

```
package com.example.polls.security;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.security.core.AuthenticationException;
import org.springframework.security.web.AuthenticationEntryPoint;
import org.springframework.stereotype.Component;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
@Component
```

```
public class JwtAuthenticationEntryPoint implements AuthenticationEntryPoint {
   private static final Logger logger = LoggerFactory.getLogger(JwtAuthenticationEntryPoint.class);
   @Override
   public void commence(HttpServletRequest httpServletRequest,
                        HttpServletResponse httpServletResponse,
                        AuthenticationException e) throws IOException, ServletException {
      logger.error("Responding with unauthorized error. Message - {}", e.getMessage());
      httpServletResponse.sendError(HttpServletResponse.SC_UNAUTHORIZED, e.getMessage());
   }
}
```

JwtAuthenticationFilter.java

```
package com.example.polls.security;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.web.authentication.WebAuthenticationDetailsSource;
import org.springframework.util.StringUtils;
import org.springframework.web.filter.OncePerRequestFilter;

import javax.servlet.FilterChain;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
```

```
public class JwtAuthenticationFilter extends OncePerRequestFilter {

    @Autowired
    private JwtTokenProvider tokenProvider;

    @Autowired
    private CustomUserDetailsService customUserDetailsService;

    private static final Logger logger = LoggerFactory.getLogger(JwtAuthenticationFilter.class);

    @Override
    protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response,
FilterChain filterChain) throws ServletException, IOException {
        try {
            String jwt = getJwtFromRequest(request);

            if (StringUtils.hasText(jwt) && tokenProvider.validateToken(jwt)) {
                Long userId = tokenProvider.getUserIdFromJWT(jwt);

                /*
                    Note that you could also encode the user's username and roles inside JWT claims
                    and create the UserDetails object by parsing those claims from the JWT.
                    That would avoid the following database hit. It's completely up to you.
                 */
                UserDetails userDetails = customUserDetailsService.loadUserById(userId);

                UsernamePasswordAuthenticationToken authentication = new
UsernamePasswordAuthenticationToken(userDetails, null, userDetails.getAuthorities());
                authentication.setDetails(new WebAuthenticationDetailsSource().buildDetails(request));


                SecurityContextHolder.getContext().setAuthentication(authentication);
            }
        } catch (Exception ex) {
            logger.error("Could not set user authentication in security context", ex);
        }
```

```
        filterChain.doFilter(request, response);
    }
    private String getJwtFromRequest(HttpServletRequest request) {
        String bearerToken = request.getHeader("Authorization");
        if (StringUtils.hasText(bearerToken) && bearerToken.startsWith("Bearer ")) {
            return bearerToken.substring(7, bearerToken.length());
        }
        return null;
    }
}
```

JwtTokenProvider.java

```
package com.example.polls.security;

import io.jsonwebtoken.*;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.security.core.Authentication;
import org.springframework.stereotype.Component;

import java.util.Date;

@Component
public class JwtTokenProvider {

    private static final Logger logger = LoggerFactory.getLogger(JwtTokenProvider.class);
```

```
@Value("${app.jwtSecret}")
private String jwtSecret;

@Value("${app.jwtExpirationInMs}")
private int jwtExpirationInMs;

public String generateToken(Authentication authentication) {

   UserPrincipal userPrincipal = (UserPrincipal) authentication.getPrincipal();

   Date now = new Date();
   Date expiryDate = new Date(now.getTime() + jwtExpirationInMs);

   return Jwts.builder()
         .setSubject(Long.toString(userPrincipal.getId()))
         .setIssuedAt(new Date())
         .setExpiration(expiryDate)
         .signWith(SignatureAlgorithm.HS256, jwtSecret)
         .compact();
}

public Long getUserIdFromJWT(String token) {
   Claims claims = Jwts.parser()
         .setSigningKey(jwtSecret)
         .parseClaimsJws(token)
         .getBody();
```

```
        return Long.parseLong(claims.getSubject());
    }

    public boolean validateToken(String authToken) {
        try {
            Jwts.parser().setSigningKey(jwtSecret).parseClaimsJws(authToken);
            return true;
        } catch (SignatureException ex) {
            logger.error("Invalid JWT signature");
        } catch (MalformedJwtException ex) {
            logger.error("Invalid JWT token");
        } catch (ExpiredJwtException ex) {
            logger.error("Expired JWT token");
        } catch (UnsupportedJwtException ex) {
            logger.error("Unsupported JWT token");
        } catch (IllegalArgumentException ex) {
            logger.error("JWT claims string is empty.");
        }
        return false;
    }
}
```

UserPrincipal.java

```
package com.example.polls.security;

import com.example.polls.model.User;
import com.fasterxml.jackson.annotation.JsonIgnore;
import org.springframework.security.core.GrantedAuthority;
```

```
import org.springframework.security.core.authority.SimpleGrantedAuthority;

import org.springframework.security.core.userdetails.UserDetails;


import java.util.Collection;

import java.util.List;

import java.util.Objects;

import java.util.stream.Collectors;


public class UserPrincipal implements UserDetails {

   private Long id;


   private String name;


   private String username;


   @JsonIgnore

   private String email;


   @JsonIgnore

   private String password;


   private Collection<? extends GrantedAuthority> authorities;


   public UserPrincipal(Long id, String name, String username, String email, String password,
Collection<? extends GrantedAuthority> authorities) {

      this.id = id;

      this.name = name;

      this.username = username;
```

```
        this.email = email;
        this.password = password;
        this.authorities = authorities;
    }

    public static UserPrincipal create(User user) {
        List<GrantedAuthority> authorities = user.getRoles().stream().map(role ->
                new SimpleGrantedAuthority(role.getName().name())
        ).collect(Collectors.toList());

        return new UserPrincipal(
                user.getId(),
                user.getName(),
                user.getUsername(),
                user.getEmail(),
                user.getPassword(),
                authorities
        );
    }

    public Long getId() {
        return id;
    }

    public String getName() {
        return name;
    }
```

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

@Override
public String getUsername() {
   return username;
}

@Override
public String getPassword() {
   return password;
}

@Override
public Collection<? extends GrantedAuthority> getAuthorities() {
   return authorities;
}

@Override
public boolean isAccountNonExpired() {
   return true;
}

@Override
public boolean isAccountNonLocked() {
```

```
        return true;
    }

    @Override
    public boolean isCredentialsNonExpired() {
        return true;
    }

    @Override
    public boolean isEnabled() {
        return true;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        UserPrincipal that = (UserPrincipal) o;
        return Objects.equals(id, that.id);
    }

    @Override
    public int hashCode() {

        return Objects.hash(id);
    }
}
```

Service

CartService.java

```
package com.example.polls.service;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.example.polls.model.Cart;
import com.example.polls.repository.CartRepository;
import com.example.polls.repository.UserRepository;
@Service
public class CartService {
	@Autowired
	private CartRepository cartRepository;
	@Autowired
	private UserRepository userRepository;
	private static final Logger LOGGER = LoggerFactory
			.getLogger(CartService.class);
	public void addtocart(Cart cart) {
		LOGGER.info("addtocart");
		cartRepository.save(cart);
	}
	public void updatecart(Cart cart) {
		LOGGER.info("updatecart");
		cartRepository.save(cart);
	}}
```

FeedbackService.java

```
 package com.example.polls.service;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.example.polls.model.Feedback;
import com.example.polls.repository.FeedbackRepository;
@Service
public class FeedbackService {
	@Autowired
	private FeedbackRepository feedbackRepository;
	private static final Logger logger = LoggerFactory
			.getLogger(FeedbackService.class);
	public void saveFeedback(Feedback feedback) {
		feedbackRepository.save(feedback);
	}
	public void updateFeedback(Feedback feedback) {
		feedbackRepository.save(feedback);
	}

	public void deleteFeedback(Feedback feedback) {
		feedbackRepository.delete(feedback);
	}}
```

OrderService.java

```
package com.example.polls.service;
import org.slf4j.Logger;
```

```
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.example.polls.model.Order;
import com.example.polls.repository.OrderRepository;
import com.example.polls.repository.UserRepository;
@Service
public class OrderService {
	@Autowired
	private OrderRepository orderRepository;
	@Autowired
	private UserRepository userRepository;
	private static final Logger LOGGER = LoggerFactory
			.getLogger(OrderService.class);
	public void saveOrder(Order order) {
		orderRepository.save(order);
	}
		public void updateOrder(Order order) {
		orderRepository.save(order);
	}

	public void deleteOrder(Order order) {
		orderRepository.delete(order);
	}}
```

RecipeService.java

```
package com.example.polls.service;
import java.io.IOException;
```

```
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.example.polls.model.Recipe;
import com.example.polls.payload.RecipeRequest;
import com.example.polls.repository.RecipeRepository;
import com.example.polls.repository.UserRepository;
@Service
public class RecipeService {
		private final Path root = Paths.get("uploads");
		@Autowired
	private RecipeRepository recipeRepository;
	@Autowired
	private UserRepository userRepository;
	private static final Logger LOGGER = LoggerFactory
			.getLogger(RecipeService.class);


	public void saveRecipe(RecipeRequest recipeRequest) {
		LOGGER.info("Inside save recipe");
		try {
		    Files.createDirectory(root);
		  } catch (IOException e) {
```

```
                    //throw new RuntimeException("Could not initialize folder for upload!");
                  }
                 try {
                    Files.copy(recipeRequest.getFile().getInputStream(),
this.root.resolve(recipeRequest.getFile().getOriginalFilename()));
                  } catch (Exception e) {
                    //throw new RuntimeException("Could not store the file. Error: " +
e.getMessage());
                  }
                Recipe recipe = new
Recipe(recipeRequest.getDishname(),recipeRequest.getQuantity(),recipeRequest.getPrice(),recipe
Request.getFile().getOriginalFilename());
                recipeRepository.save(recipe);
        }

        public void updateRecipe(RecipeRequest recipeRequest) {
                LOGGER.info("Inside save recipe");
                try {
                    Files.createDirectory(root);
                  } catch (IOException e) {
                    //throw new RuntimeException("Could not initialize folder for upload!");
                  }
                 try {
                    Files.copy(recipeRequest.getFile().getInputStream(),
this.root.resolve(recipeRequest.getFile().getOriginalFilename()));
                  } catch (Exception e) {
                    //throw new RuntimeException("Could not store the file. Error: " +
e.getMessage());
                  }
```

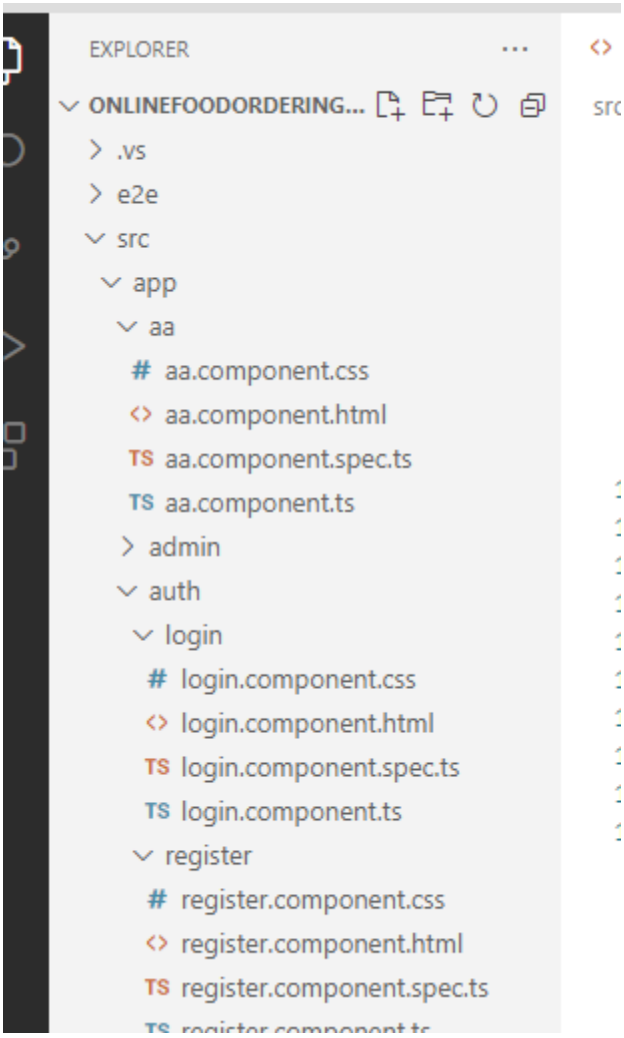
```
            Recipe recipe = new
Recipe(recipeRequest.getId(),recipeRequest.getDishname(),recipeRequest.getQuantity(),recipeRe
quest.getPrice(),recipeRequest.getFile().getOriginalFilename());


                recipeRepository.save(recipe);

        }


        public void updateRecipe(Recipe recipe) {

                LOGGER.info("Inside save recipe");

                recipeRepository.save(recipe);

        }


        public void deleteRecipe(Recipe recipe) {

                recipeRepository.delete(recipe);

        }}
```

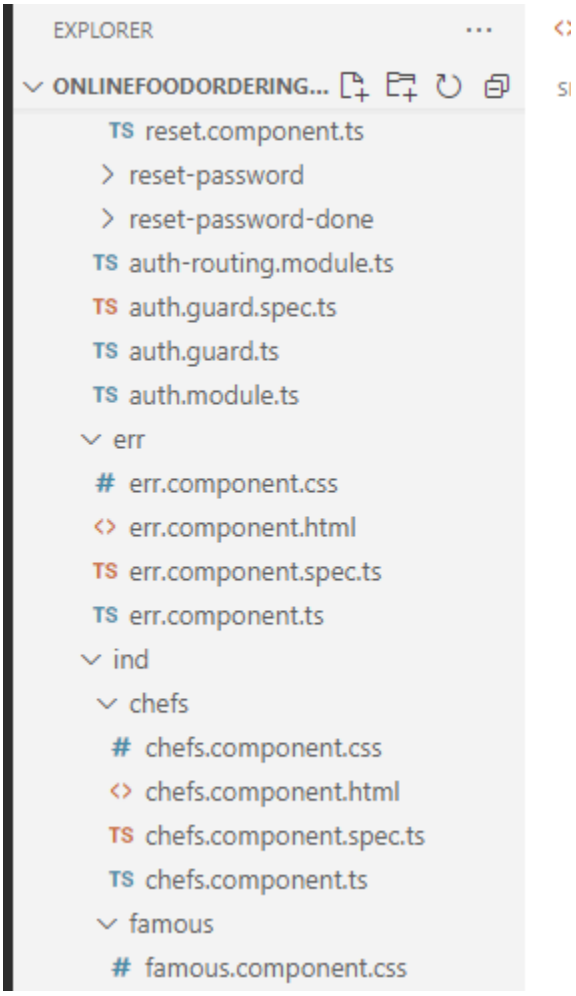
ApplicationProperties

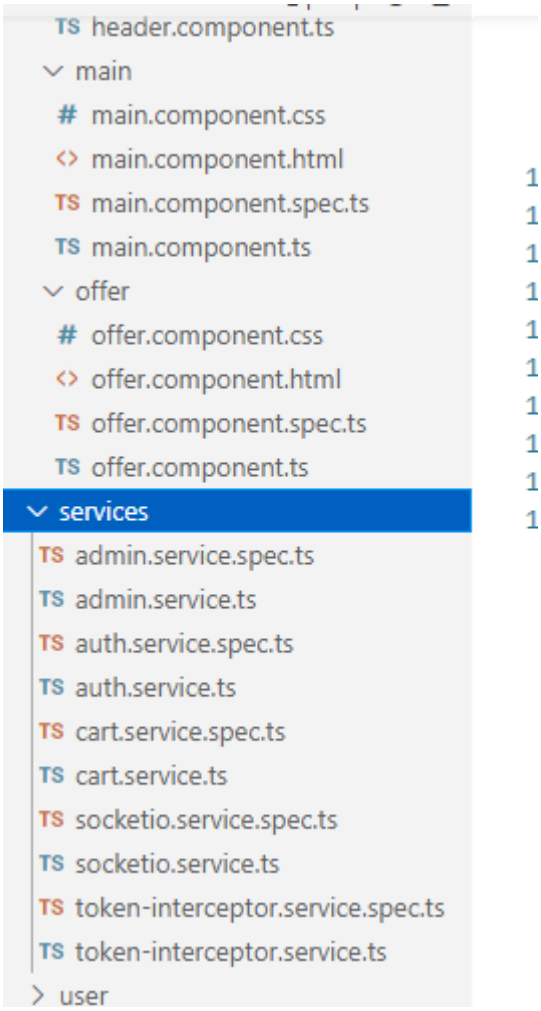
```
## Server Properties
server.port= 3000
server.compression.enabled=true

## Spring DATASOURCE (DataSourceAutoConfiguration & DataSourceProperties)
spring.datasource.url=
jdbc:mysql://db4free.net:3306/capstoneswathi12?allowPublicKe
8yRetrieval=true&useSSL=false&serverTimezone=UTC&useLegacyDatetimeCode=false
spring.datasource.username= capstoneswathi12
spring.datasource.password= swathi123


## Hibernate Properties
# The SQL dialect makes Hibernate generate better SQL for the chosen database
spring.jpa.properties.hibernate.dialect =
org.hibernate.dialect.MySQL5InnoDBDialect
spring.jpa.hibernate.ddl-auto = update

## Hibernate Logging
logging.level.org.hibernate.SQL= DEBUG

# Initialize the datasource with available DDL and DML scripts
spring.datasource.initialization-mode=always
spring.jpa.defer-datasource-initialization= true

## Jackson Properties
```

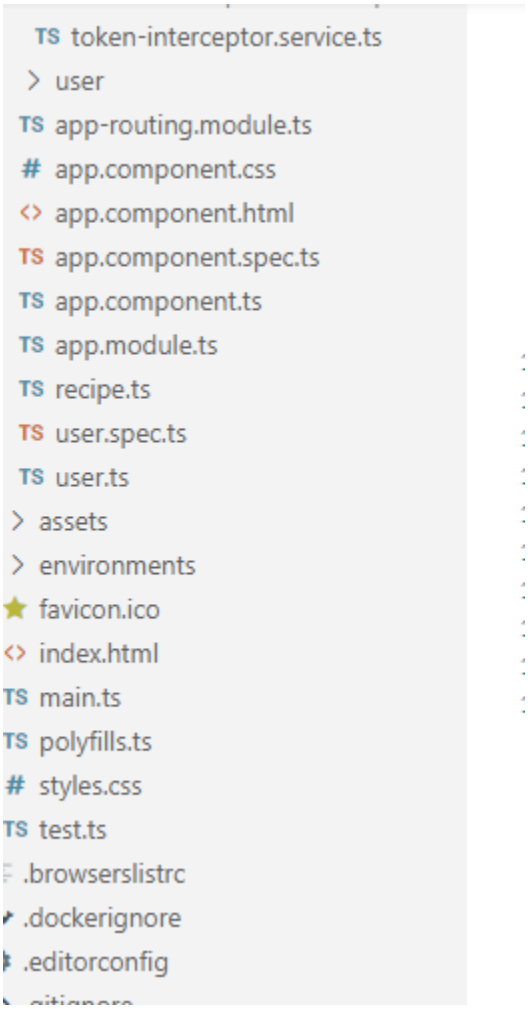
```
spring.jackson.serialization.WRITE_DATES_AS_TIMESTAMPS= false
spring.jackson.time-zone= UTC

## App Properties
app.jwtSecret=
9a02115a835ee03d5fb83cd8a468ea33e4090aaaec87f53c9fa54512bbef4db8dc656c82a315fa0c78
5c08b0134716b81ddcd0153d2a7556f2e154912cf5675f
app.jwtExpirationInMs = 604800000

# Comma separated list of allowed origins
app.cors.allowedOrigins = *

## Spring Profiles
# spring.profiles.active=prod
```

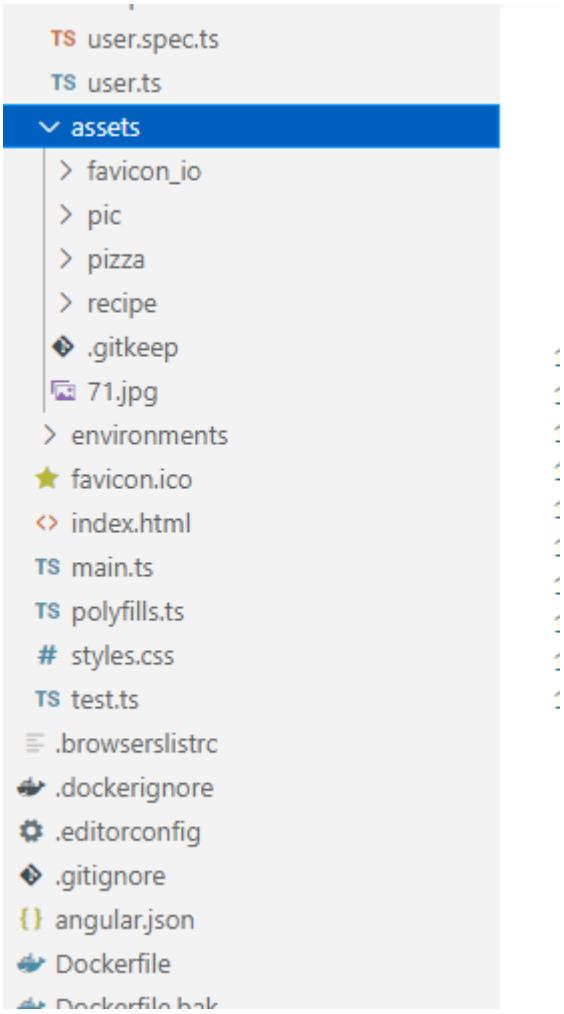
Angula code explorer




EXPLORER
ONLINEFOODORDERING...
reset.component.ts
reset-password
reset-password-done
auth-routing.module.ts
auth.guard.spec.ts
auth.guard.ts
auth.module.ts
err
err.component.css
err.component.html
err.component.spec.ts
err.component.ts
ind
chefs
chefs.component.css
chefs.component.html
chefs.component.spec.ts
chefs.component.ts
famous
famous.component.css


header.component.ts
main
main.component.css
main.component.html
main.component.spec.ts
main.component.ts
offer
offer.component.css
offer.component.html
offer.component.spec.ts
offer.component.ts
services
admin.service.spec.ts
admin.service.ts
auth.service.spec.ts
auth.service.ts
cart.service.spec.ts
cart.service.ts
socketio.service.spec.ts
socketio.service.ts
token-interceptor.service.spec.ts
token-interceptor.service.ts
user


token-interceptor.service.ts
user
app-routing.module.ts
app.component.css
app.component.html
app.component.spec.ts
app.component.ts
app.module.ts
recipe.ts
user.spec.ts
user.ts
assets
environments
favicon.ico
index.html
main.ts
polyfills.ts
styles.css
test.ts
.browserslistrc
.dockerignore
.editorconfig



Githublink is:- https://github.com/swathikanduri7/foodboxCapstoneProject.git

This project is developed by SwathiKanduri

Angular code

Source Code

Admin->Add Recipe->Component.css

```
*{
   margin: 0;
   padding: 0;
   box-sizing: border-box;
   font-family: sans-serif;
```

```
   /* max-height: vh; */
}
body{
  width:100%;
  min-height: 100vh;
  background-image:url("../../../assets/pic/49.jpg");
  background-repeat: no-repeat;
  background-size: 100% 100%;
  position: relative;
  background-attachment: fixed;
  margin-top: 0px;
  margin-bottom: 0px;
  /* margin-top: 10px; */
}
.contact{
    position: relative;
    min-height: 100vh;
    padding: 50px 100px;
    display: flex;
    justify-content: center;
    align-items: center;
    flex-direction:column;

}
.contact .content
{
    max-width: 800px;
```

```
    text-align: center;
}

.contact .content h2{
    font-size: 36px;
    font-weight: 500;
    color: #fff;
}
.contact .content p{

    font-weight: 300;
    color: #fff;
}

.container{
    width: 100%;
    display: flex;
    justify-content: center;
    align-items: center;
    margin-top: 30px;
}

.container .contactInfo{
    width: 50%;
    display: flex;
    flex-direction: column;
}
```

```
.container .contactInfo .box{
    position: relative;
    padding: 20px 0;
    display: flex;
}
.container .contactInfo .box .icon{
    min-width: 60px;
    height: 60px;
    background: #fff;
    display: flex;
    justify-content: center;
    align-items: center;
    border-radius: 50%;
    font-size: 22px;
}



.container .contactInfo .box .text{
    display: flex;
    margin-left: 20px;
    font-size: 16px;
    color: #fff;
    flex-direction: column;
    font-weight: 300;
}

.container .contactInfo .box .text h3{
```

```
    font-weight: 500;
    color:#00bcd4;
}

.contactFrom{
    /* margin-left: 20%; */
    width: 40%;
    padding: 40px;
    background: #fff;
}

.contactFrom h2{
    font-size: 30px;
    color: #333;
    font-weight: 500;
}

.contactFrom .inputBox{
    position: relative;
    width: 100%;
    margin-top: 10px;
}
.contactFrom .inputBox input,.contactFrom .inputBox textarea,.contactFrom .inputBox select{
    width: 100%;
    padding: 5px 0;
    font-size: 16px;
    margin: 10px 0;
```

```
    border: none;
    border-bottom: 2px solid #333;
    outline: none;
    resize: none;
}
.contactFrom .inputBox span{
    position: absolute;
    left: 0;
    padding: 5px 0;
    font-size: 16px;
    margin: 10px 0;
    pointer-events: none;
    transition: 0.5s;
    color: #666;
}

.contactFrom .inputBox select:focus~span,
.contactFrom .inputBox select:valid~span,
.contactFrom .inputBox input:focus~span,
.contactFrom .inputBox input:valid~span
{
    color: #e91e63;
    font-size: 12px;
    transform: translateY(-20px);
}

.contactFrom .inputBox input[type="submit"]
```

```
{
    width: 100px;
    background: #00bcd4;
    border: none;
    cursor: pointer;
    padding: 10px;
    font-size: 18px;

}

@media(max-width:991px)
{
    body{
        background: url('../../../assets/pic/49.jpg');
    }
    .contact{
        padding: 50px;
    }
    .container{
        flex-direction: column;
    }
    .container .contactInfo,.contactFrom{
        margin-bottom: 40px;
        width: 100%;
    }
}
```

Component.html

```
<!DOCTYPE html>
<html>

<head>
  <meta charset='utf-8'>
  <meta http-equiv='X-UA-Compatible' content='IE=edge'>

  <meta name='viewport' content='width=device-width, initial-scale=1'>
</head>

<body>
  <app-ind></app-ind>
  <!-- <section class="contact"> -->
  <div class="content">
    <h2>.</h2>
  </div>
  <div class="container">
    <div class="contactFrom">
      <form #f="ngForm" (ngSubmit)="onSubmit(f)">

        <h4>Add recipe(Fodd Item) for sell</h4>
        <div style="color: red;" *ngIf="avail">
          <h6> {{msg}}</h6>
        </div>
```

```
      <div class="inputBox">
       <input type="text" name="dishname"  ngModel required #dishname="ngModel">
       <span>Dish name</span>
      </div>
      <div class="inputBox" >
       <select  name="quantity" ngModel required #quantity="ngModel">
        <option value="small">Single</option>
        <option value="medium">Full</option>
        <option value="large">Jumbo</option>
       </select>
       <span>Choose quantity</span>

      </div>
      <div class="inputBox">
       <input type="number" name="price"  min="1" ngModel required #price="ngModel">
       <span>Recipe price</span>
      </div>
       <div class="inputBox">
       <input type="file" name="recipepic" title="choose recipe image"
(change)="selectImage($event)" ngModel required #recipepic="ngModel">
       <!-- <span>Choose recipe pic</span> -->
      </div>
      <div class="inputBox">
       <input type="submit" [disabled]="f.invalid" value="submit">

      </div>
     </form>
```

```
  </div>
 </div>
 <!-- </section> -->
</body>


</html>
```

Component.spec.ts

```
import { ComponentFixture, TestBed } from '@angular/core/testing';

import { AddrecipeComponent } from './addrecipe.component';

describe('AddrecipeComponent', () => {
 let component: AddrecipeComponent;
 let fixture: ComponentFixture<AddrecipeComponent>;

 beforeEach(async () => {
  await TestBed.configureTestingModule({
   declarations: [ AddrecipeComponent ]
  })
  .compileComponents();
 });

 beforeEach(() => {
  fixture = TestBed.createComponent(AddrecipeComponent);
  component = fixture.componentInstance;
```

```
    fixture.detectChanges();
  });


  it('should create', () => {
    expect(component).toBeTruthy();
  });
});



Component.ts
import { HttpClient, HttpErrorResponse } from '@angular/common/http';
import { Component, OnInit } from '@angular/core';
import { NgForm, FormGroup, FormControl, Validators } from '@angular/forms';
import { Router } from '@angular/router';
import { AdminService } from 'src/app/services/admin.service';


@Component({
  selector: 'app-addrecipe',
  templateUrl: './addrecipe.component.html',
  styleUrls: ['./addrecipe.component.css']
})
export class AddrecipeComponent implements OnInit {
  msg: any = [];
  avail: boolean;
  onerecipe: any;
  image;
  constructor(private http: HttpClient, private router: Router, private adminService: AdminService)
```

```
{ }


 ngOnInit(): void {
   this.check()
   this.onerecipe = this.adminService.temp;
 }
 check() {
   this.adminService.check().subscribe(
     data => {
       console.log(data);
     },
     (error) => {

       if (error instanceof HttpErrorResponse) {


         this.router.navigate(['/login'])


       }
       console.log(error);
     }
   )
 }



 onSubmit(f: NgForm) {
   if (!f.valid) {
     this.msg = "something went  wrong!!";
```

```
  this.avail = true;
  return;
 }
 const formData = new FormData();
 formData.append('file', this.image);
 formData.append('dishname', f.controls.dishname.value);
 formData.append('quantity', f.controls.quantity.value);
 formData.append('price', f.controls.price.value);
 this.http.post<any>('http://localhost:3000/admin/addrecipe', formData).subscribe(
  (res) => {
   this.adminService.avail = true;
   this.adminService.msg = "Successfully Added a recipe!!!"
   this.router.navigate(['/admin']);
   // console.log(res)
  }
  ,
  (error) => {

   if (error instanceof HttpErrorResponse) {

      this.router.navigate(['/login'])

   }
   console.log(error);
  }
 );
```

```
  }
  selectImage(event) {
    console.log("image selected");
    if (event.target.files.length > 0) {
      const file = event.target.files[0];
      this.image = file;
    }
  }
}
```

AdminHome .component.css

```
html{
  scroll-behavior: smooth;
}
*{
  margin: 0;
  padding: 0;

}
.header{
  width:100%;
  min-height: 50vh;
  background-image:url("../../../assets/pic/40.jpg");
  background-repeat: no-repeat;
```

```
  background-size: 100% 100%;
  position: relative;
  background-attachment: fixed;
  margin-top: 0px;
  margin-bottom: 0px;
  /* clip-path: polygon(100% 0%,100% 75%,50% 100%,0% 75%,0% 0%); */
}

.table{
  margin-top: 30px;
  margin-bottom: 250px;
  margin-left: auto;
  margin-right: auto;
}
th,td{
text-align: center;
}
html,body{

width: 100%;
height: 100%;
background-attachment: fixed;
background-repeat: no-repeat;
}

button{
  outline: none;
```

```
}



AdminHome .Component.html


<html>


<head>

 <link


href="https://fonts.googleapis.com/css?family=Poppins:100,100i,200,200i,300,300i,400,400i,500,
500i,600,600i,700,700i,800,800i,900,900i"

   rel="stylesheet">

 <link href="https://fonts.googleapis.com/css?family=Audiowide&display=swap"
rel="stylesheet">

 <link
href="https://fonts.googleapis.com/css?family=Concert+One|Oswald|Quicksand&display=swap"
rel="stylesheet">

 <link href="https://fonts.googleapis.com/css2?family=Noto+Sans+JP:wght@500&display=swap"
rel="stylesheet">




 <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css">

 <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">

 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
```

```
 <meta name="viewport" content="width=device-width, initial-scale=1">
</head>

<body>
 <app-ind></app-ind>
 <div class="header">

  <div class="alert alert-success alert-dismissible" *ngIf="avail">
   <button type="button" class="close" data-dismiss="alert">&times;</button>
   {{msg}}
  </div>
 </div>
 <h1 class="text-center">Orders</h1>
 <table class="table table-hover w-50"
 style="background-color: white;  box-shadow: 0px 0px 19px 5px rgba(0, 0, 0, 0.5);">
 <thead>
  <tr>
   <th scope="col">#</th>
   <th scope="col">Order Id</th>
   <th scope="col">Recipes</th>
   <th scope="col">Bill Amount</th>
   <th scope="col">Delete</th>
  </tr>
 </thead>
 <tbody>
  <tr *ngFor="let order of orders;index as i">
   <th scope="row">{{i+1}}</th>
```

```
    <td class="text-danger" (click)="viewuser(order.whichuser)" >{{order.whichuser}}</td>

    <td><button class="btn btn-link" (click)="seeorder(order)" ><i class="fa fa-eye" aria-
hidden="true"></i></button></td>

    <td >{{order.total}}</td>

    <td><button class="btn btn-link" (click)="delete(order)"><i class="fa fa-trash" aria-
hidden="true"></i></button>

    </td>




   </tr>


  </tbody>

</table>

</body>

</html>


AdminHome.component.spec.ts


import { ComponentFixture, TestBed } from '@angular/core/testing';


import { AdminhomeComponent } from './adminhome.component';


describe('AdminhomeComponent', () => {

  let component: AdminhomeComponent;

  let fixture: ComponentFixture<AdminhomeComponent>;


  beforeEach(async () => {
```

```
  await TestBed.configureTestingModule({
    declarations: [ AdminhomeComponent ]
  })
  .compileComponents();
 });

 beforeEach(() => {
  fixture = TestBed.createComponent(AdminhomeComponent);
  component = fixture.componentInstance;
  fixture.detectChanges();
 });

 it('should create', () => {
  expect(component).toBeTruthy();
 });
});
```

AdminHome.component.ts

```
import { HttpErrorResponse } from '@angular/common/http';
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { AdminService } from 'src/app/services/admin.service';

@Component({
 selector: 'app-adminhome',
 templateUrl: './adminhome.component.html',
```

```
  styleUrls: ['./adminhome.component.css']
})
export class AdminhomeComponent implements OnInit {
  msg : any =[];
  avail:boolean;
  orders : any =[];
  constructor(private adminService: AdminService,private router: Router) { }

  ngOnInit(): void {
    this.check()
    this.Order()
    if(this.adminService.avail)
    {
      this.avail=true;
      this.msg = this.adminService.msg;
    }
    this.change();

  }
  change()
  {
    // console.log("aa");
    this.adminService.avail=false;
    this.adminService.msg="";
  }

  check() {
```

```
  this.adminService.check().subscribe(
    data => {
      console.log(data);
    },
    (error) => {
      if (error instanceof HttpErrorResponse) {
          this.router.navigate(['/login'])
      }
      console.log(error);
    }
  )
}



Order()
{
  this.adminService.getorder().subscribe(
    data => {
      this.orders=data['msg'];
    },
    (error) => {
      console.log(error);
    }
  )
}
```

```
delete(order)
{
  this.adminService.deleteorder(order._id).subscribe(
    data => {

      if(data['msg']=="yes deleted order by admin")
      {
        alert("successfully deleted order")
        window.location.reload();
      }
    },
    (error) => {
      console.log(error);
    }
  )
}

seeorder(order)
{
  this.adminService.setOrder(order._id)
  this.router.navigate(['admin/viewoneorder'])
}

viewuser(userid)
{
  this.adminService.setOrder(userid)
  this.router.navigate(['admin/viewoneuser'])
```

```
  }
}

EditRecipe.Component.css

*{
  margin: 0;
  padding: 0;
  box-sizing: border-box;
  font-family: sans-serif;
  /* max-height: vh; */
}
body{
width:100%;
min-height: 100vh;
background-image:url("../../../assets/pic/49.jpg");
background-repeat: no-repeat;
background-size: 100% 100%;
position: relative;
background-attachment: fixed;
margin-top: 0px;
margin-bottom: 0px;
/* margin-top: 10px; */
}
.contact{
  position: relative;
  min-height: 100vh;
```

```
  padding: 50px 100px;
  display: flex;
  justify-content: center;
  align-items: center;
  flex-direction:column;

}
.contact .content
{
  max-width: 800px;
  text-align: center;
}

.contact .content h2{
  font-size: 36px;
  font-weight: 500;
  color: #fff;
}
.contact .content p{

  font-weight: 300;
  color: #fff;
}

.container{
  width: 100%;
  display: flex;
```

```
  justify-content: center;
  align-items: center;
  margin-top: 30px;
}

.container .contactInfo{
  width: 50%;
  display: flex;
  flex-direction: column;
}
.container .contactInfo .box{
  position: relative;
  padding: 20px 0;
  display: flex;
}
.container .contactInfo .box .icon{
  min-width: 60px;
  height: 60px;
  background: #fff;
  display: flex;
  justify-content: center;
  align-items: center;
  border-radius: 50%;
  font-size: 22px;
}
```

```
.container .contactInfo .box .text{
  display: flex;
  margin-left: 20px;
  font-size: 16px;
  color: #fff;
  flex-direction: column;
  font-weight: 300;
}

.container .contactInfo .box .text h3{
  font-weight: 500;
  color:#00bcd4;
}

.contactFrom{
  /* margin-left: 20%; */
  width: 40%;
  padding: 40px;
  background: #fff;
}

.contactFrom h2{
  font-size: 30px;
  color: #333;
  font-weight: 500;
}
```

```
.contactFrom .inputBox{
  position: relative;
  width: 100%;
  margin-top: 10px;
}
.contactFrom .inputBox input,.contactFrom .inputBox textarea,.contactFrom .inputBox select{
  width: 100%;
  padding: 5px 0;
  font-size: 16px;
  margin: 10px 0;
  border: none;
  border-bottom: 2px solid #333;
  outline: none;
  resize: none;
}
.contactFrom .inputBox span{
  position: absolute;
  left: 0;
  padding: 5px 0;
  font-size: 16px;
  margin: 10px 0;
  pointer-events: none;
  transition: 0.5s;
  color: #666;
}

.contactFrom .inputBox select:focus~span,
```

```
.contactFrom .inputBox select:valid~span,
.contactFrom .inputBox input:focus~span,
.contactFrom .inputBox input:valid~span
{
  color: #e91e63;
  font-size: 12px;
  transform: translateY(-20px);
}

.contactFrom .inputBox input[type="submit"]
{
  width: 100px;
  background: #00bcd4;
  border: none;
  cursor: pointer;
  padding: 10px;
  font-size: 18px;

}

@media(max-width:991px)
{
  body{
     background: url('../../../assets/pic/49.jpg');
  }
  .contact{
     padding: 50px;
```

```
 }
 .container{
    flex-direction: column;
 }
 .container .contactInfo,.contactFrom{
    margin-bottom: 40px;
    width: 100%;
 }
}
```

EditRecipe.Component.html

```
<!DOCTYPE html>
<html>

<head>
 <meta charset='utf-8'>
 <meta http-equiv='X-UA-Compatible' content='IE=edge'>

 <meta name='viewport' content='width=device-width, initial-scale=1'>
</head>

<body>
 <app-ind></app-ind>
 <!-- <section class="contact"> -->
 <div class="content">
  <h2>.</h2>
```

```
</div>
<div class="container">
 <div class="contactFrom">
  <form #f="ngForm" (ngSubmit)="onSubmit(f)">


   <h4>Edit recipe </h4>









   <div style="color: red;">
    <h6>Dishname = {{onerecipe.dishname}}</h6>
    <h6>Quantity = {{onerecipe.quantity}}</h6>
    <h6>Price = {{onerecipe.price}}</h6>
    <h6>If any change then write in below form</h6>
   </div>
   <div class="inputBox">
    <input type="text" name="dishname" ngModel  #dishname="ngModel">
    <span>Dish name</span>
   </div>
   <div class="inputBox">
    <select name="quantity" ngModel  #quantity="ngModel">
     <option value="small">Single</option>
     <option value="medium">Full</option>
     <option value="large">Jumbo</option>
    </select>
    <span>Choose dish quantity</span>
```

```
      </div>
      <div class="inputBox">
        <input type="number" name="price" min="1" ngModel  #price="ngModel">
        <span>Dish price</span>
      </div>
      <div class="inputBox">
        <input type="file" name="recipepic" title="choose recipe image"
(change)="selectImage($event)" ngModel
          #recipepic="ngModel">
        <!-- <span>Choose recipe pic</span> -->
      </div>
      <div class="inputBox">
        <input type="submit" [disabled]="f.invalid" value="submit">

      </div>
    </form>
   </div>
  </div>
  <!-- </section> -->
</body>

</html>
```

EditRecipe.Component.spec.ts

```
import { ComponentFixture, TestBed } from '@angular/core/testing';
```

```
import { EditrecipeComponent } from './editrecipe.component';


describe('EditrecipeComponent', () => {
  let component: EditrecipeComponent;
  let fixture: ComponentFixture<EditrecipeComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ EditrecipeComponent ]
    })
    .compileComponents();
  });

  beforeEach(() => {
    fixture = TestBed.createComponent(EditrecipeComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

EditRecipe.Component.ts

```
import { HttpClient, HttpErrorResponse } from '@angular/common/http';
import { Component, OnInit } from '@angular/core';
import { NgForm } from '@angular/forms';
import { Router } from '@angular/router';
import { AdminService } from 'src/app/services/admin.service';


@Component({
  selector: 'app-editrecipe',
  templateUrl: './editrecipe.component.html',
  styleUrls: ['./editrecipe.component.css']
})
export class EditrecipeComponent implements OnInit {
  msg: any = [];
  avail: boolean;
  onerecipe: any;
  dishname: any;
  quantity: any;
  price: any;
  pn: any;
  ps: any;
  pp: any;
  id: any;
  image;
  constructor(private http: HttpClient, private router: Router, private adminService: AdminService)
{ }


  ngOnInit(): void {
```

```
  this.check()
  this.onerecipe = this.adminService.temp;


  this.dishname = this.onerecipe.dishname;
  this.quantity = this.onerecipe.quantity;
  this.price = this.onerecipe.price;
  this.id = this.onerecipe._id;
}


check() {
  this.adminService.check().subscribe(
    data => {
      console.log(data);
    },
    (error) => {


      if (error instanceof HttpErrorResponse) {


          this.router.navigate(['/login'])


      }
      console.log(error);
    }
  )
  // console.log();
}
```

```
onSubmit(f: NgForm) {
 if (!f.valid) {
  this.msg = "something went  wrong!!";
  this.avail = true;
  return;
 }
 const formData = new FormData();
 formData.append('id', this.id);

 if (f.controls.dishname.value) {
  // console.log("yes name");
  formData.append('dishname', f.controls.dishname.value);
  this.pn = f.controls.dishname.value;
 }
 else {
  // console.log("no name");
  formData.append('dishname', this.dishname);
  this.pn = this.dishname;
 }
 if (f.controls.quantity.value) {
  // console.log("yes size");
  formData.append('quantity', f.controls.quantity.value);
  this.ps = f.controls.quantity.value;
 }
 else {
  // console.log("no size");
  formData.append('quantity', this.quantity);
```

```
  this.ps = this.quantity;
 }


 if (f.controls.price.value) {
  // console.log("yes price");
  formData.append('price', f.controls.price.value);
  this.pp = f.controls.price.value;
 }
 else {
  // console.log("no price");
  formData.append('price', this.price);
  this.pp = this.price;
 }



 if (f.controls.recipepic.value) {
  // console.log("yes image");
  formData.append('file', this.image);


  // *************
  this.http.post<any>('http://localhost:3000/admin/editrecipewithimage', formData).subscribe(
   (res) => {
    this.adminService.avail = true;
    this.adminService.msg = "Successfully Edited a recipe!!!"
    this.router.navigate(['/admin']);
    console.log(res)
   }
```

```
    ,
    (error) =>{

      if(error instanceof HttpErrorResponse)
      {

        this.router.navigate(['/login'])

      }
      console.log(error);
    }
   );

  }
  else {

   this.http.get<any>('http://localhost:3000/admin/editrecipewithoutimage?id=' + this.id +
'&recipename=' + this.pn + '&quantity=' + this.ps + '&price=' + this.pp
   ).subscribe(
    (res) => {
      this.adminService.avail = true;
      this.adminService.msg = "Successfully Edited a recipe!!!"
      this.router.navigate(['/admin']);
      console.log(res)
    }
    ,
    (error) =>{
```

```
        if(error instanceof HttpErrorResponse)
        {
           this.router.navigate(['/login'])

        }
        console.log(error);
      }
     );
   }


 }


 selectImage(event) {
   if (event.target.files.length > 0) {
     const file = event.target.files[0];
     this.image = file;
   }
 }
}

Ind.component.css
html{
 scroll-behavior: smooth;
}
```

```
*{
  margin: 0;
  padding: 0;

}

/*------------------------------------------------------------- navbar ----------------------------------*/
a{
  font-family: 'Audiowide' ,cursive;
}
#colarr{
  color: white;
}
#colarr:hover{
  color: tomato;
  transition: .8s;
}
#xy{
  position: relative;
  display: flex;
  /* justify-content: center; */
  align-items: center;
  flex-direction: row;
  flex-wrap: wrap;
  /* background-color: #090d00; */
  width: 100%;
  /* height: 90px
```

```
  border : #2c3e50;

box-shadow: 0 0px 80px 40px rgba(0,0 , 0, 0.719);

  z-index: 1; */

}


ul li a{

  position: relative;

  padding: 16px 0 5px;

  margin: 0 25px;

  color: rgb(255, 255, 255);

  text-decoration: none;

  text-transform: uppercase;

  font-family: 'Audiowide' ,cursive;

  font-size: 16px;

}
```

Ind.component.html

```
<html>


<head>

  <link


href="https://fonts.googleapis.com/css?family=Poppins:100,100i,200,200i,300,300i,400,400i,500,
500i,600,600i,700,700i,800,800i,900,900i"

    rel="stylesheet">

  <link href="https://fonts.googleapis.com/css?family=Audiowide&display=swap"
rel="stylesheet">

  <link
href="https://fonts.googleapis.com/css?family=Concert+One|Oswald|Quicksand&display=swap"
```

```
rel="stylesheet">

 <link href="https://fonts.googleapis.com/css2?family=Noto+Sans+JP:wght@500&display=swap"
rel="stylesheet">




 <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css">

 <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">

 <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>







 <meta name="viewport" content="width=device-width, initial-scale=1">

</head>



<body>

 <nav class="navbar navbar-expand-lg bg-dark navbar-dark bb" id="xy">

  <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarSupportedContent"

   aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle
navigation">

   <span class="navbar-toggler-icon "></span>

  </button>




  <div class="collapse navbar-collapse" id="navbarSupportedContent">

   <ul class="navbar-nav mr-auto">

    <li class="nav-item"><a class="nav-link" [routerLink]="['/admin']" id="colarr">Home</a></li>
```

```
      <li class="nav-item"><a class="nav-link" [routerLink]="['/admin/viewuser']" id="colarr">view
Users</a></li>

      <li class="nav-item"><a class="nav-link" [routerLink]="['/admin/viewrecipe']"
id="colarr">View recipe</a></li>

      <li class="nav-item"><a class="nav-link" [routerLink]="['/admin/addrecipe']" id="colarr">ADD
recipe</a></li>

      <li class="nav-item"><a class="nav-link" [routerLink]="['/admin/viewfeedback']"
id="colarr">View Feedback</a></li>

      <li class="nav-item"><a class="nav-link" [routerLink]="['/']" (click)="logoutadmin()"
id="colarr">Log Out</a></li>

     </ul>

    </div>

  </nav>


</body>


</html>


Ind.component.spec.ts

import { ComponentFixture, TestBed } from '@angular/core/testing';


import { IndComponent } from './ind.component';


describe('IndComponent', () => {

  let component: IndComponent;

  let fixture: ComponentFixture<IndComponent>;


  beforeEach(async () => {

    await TestBed.configureTestingModule({
```

```
    declarations: [ IndComponent ]
   })
   .compileComponents();
 });

 beforeEach(() => {
   fixture = TestBed.createComponent(IndComponent);
   component = fixture.componentInstance;
   fixture.detectChanges();
 });

 it('should create', () => {
   expect(component).toBeTruthy();
 });
});
```

Ind.component.ts

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';

@Component({
 selector: 'app-ind',
 templateUrl: './ind.component.html',
 styleUrls: ['./ind.component.css']
})
export class IndComponent implements OnInit {

 constructor(private router: Router) { }
```

```
 ngOnInit(): void {
 }

 logoutadmin()
 {
   console.log("yes in admin logout");
   localStorage.removeItem('token');
   localStorage.removeItem('userid')
   localStorage.removeItem('N1@#I2@#M3@#D4@#A6');
   this.router.navigate(['/']);
 }
}
```

Viewfeedback.component.css

```
.table{
 margin-top: 150px;
 margin-bottom: 200px;
 margin-left: auto;
 margin-right: auto;
}
th,td{
text-align: center;
}
html,body{
width: 100%;
```

```
height: 100%;

background-attachment: fixed;

background-repeat: no-repeat;

}


Viewfeedback.component.html

<!doctype html>

<html lang="en">


<head>

  <!-- Required meta tags -->

  <meta charset="utf-8">

  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">


  <!-- Bootstrap CSS -->




  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"

    integrity="sha384-
9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKGj7Sk"
crossorigin="anonymous">


  <!-- fafa -->

  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">


</head>
```

```
<body>
 <app-ind></app-ind>
 <table class="table table-hover w-50"
  style="background-color: white; margin-top: 100px;  box-shadow: 0px 0px 19px 5px rgba(0, 0, 0, 0.5);">
  <thead>
   <tr>
    <th>Index</th>
    <th scope="col">User's Name</th>
    <th scope="col">Feedback <i class="fas fa-comment-dots" aria-hidden="true"></i></th>
    <th scope="col">Delete</th>
   </tr>
  </thead>
  <tbody>



   <tr *ngFor="let a of arr;index as i">
    <th scope="row">{{i+1}}</th>
    <td>{{a.name}}</td>
    <td>{{a.msg}}</td>
    <td><button class="btn btn-primary"><i class="fa fa-trash" aria-hidden="true"
(click)="delete(a)"></i></button></td>
   </tr>
```

```
  </tbody>
 </table>
</body>

</html>
```

Viewfeedback.component.spec.ts

```
import { ComponentFixture, TestBed } from '@angular/core/testing';

import { ViewfeedbackComponent } from './viewfeedback.component';

describe('ViewfeedbackComponent', () => {
 let component: ViewfeedbackComponent;
 let fixture: ComponentFixture<ViewfeedbackComponent>;

 beforeEach(async () => {
  await TestBed.configureTestingModule({
   declarations: [ ViewfeedbackComponent ]
  })
  .compileComponents();
 });

 beforeEach(() => {
  fixture = TestBed.createComponent(ViewfeedbackComponent);
  component = fixture.componentInstance;
```

```
    fixture.detectChanges();
  });


  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

Viewfeedback.component.ts

```
import { Component, OnInit } from '@angular/core';
import { HttpErrorResponse } from '@angular/common/http';
import { Router } from '@angular/router';
import { AdminService } from 'src/app/services/admin.service';


@Component({
  selector: 'app-viewfeedback',
  templateUrl: './viewfeedback.component.html',
  styleUrls: ['./viewfeedback.component.css']
})
export class ViewfeedbackComponent implements OnInit {
  arr: any[];
  constructor(private adminService: AdminService, private router: Router) { }


  ngOnInit(): void {
    this.check()
    this.getFeed()
```

```
}


check() {
  this.adminService.check().subscribe(
    data => {
      console.log(data);
    },
    (error) => {

      if (error instanceof HttpErrorResponse) {

          this.router.navigate(['/login'])

      }
      console.log(error);
    }
  )
  // console.log();
}



getFeed() {
  this.adminService.getAllFeedback().subscribe(
    data => {
      console.log(data);
      this.arr=data['msg'];
    },
```

```
    (error) => {


      if (error instanceof HttpErrorResponse) {


         this.router.navigate(['/login'])


      }
      console.log(error);
    }
  )
}


delete(a)
{
  var id = a._id;



  this.adminService.deletefeedback(id).subscribe(
    data => {
      // console.log(data);
      this.adminService.avail = true;
      this.adminService.msg = "Successfully Deleted a feedback!!!";
      this.router.navigate(['/admin']);
    },
    (error) => {
      // console.log("yoyo err");
      if (error instanceof HttpErrorResponse) {
```

```
        this.router.navigate(['/login'])


      }
      console.log(error);
    }
  )
 }
}
```

ViewEncoder.component.css

```
/* hello */


*{
 margin: 0;
 padding: 0;
 box-sizing: border-box;
}
body{
 /* background: #e7e7e7; */
 font-family: sans-serif;
}
.column {
 float: left;
 width: 100%;
 padding: 10px;
 height: 10%;
```

```
}



.parent{
  background: #fff;
  border-radius: 10px;
  box-shadow: 0px 10px 10px rgba(0, 0, 0,0.2) ;
  display: flex;
  max-width: 100%;
  overflow: hidden;
  width: 700px;
  margin-left: auto;
  margin-right: auto;
  position: relative;
  transition: 0.4s ease;
}


.parent:hover{
  transform: translateY(-10px);
  box-shadow: 0 0 20px 0 rgba(0,0,0,0.7);
}
.parent h6{
  margin-top:5px ;
  margin-bottom:5px ;
  opacity: 0.6;
  /* color: gray; */
```

```
  letter-spacing: 1px;
  text-transform: uppercase;
}
.parent h2{
  letter-spacing: 1px;
  margin:  10px 0;


}
.form-group{
  padding-top: 10px;
  display: flex;
}
input[type=number] {
  /* -moz-appearance: textfield; */
  width: 50px;
}
/* input::-webkit-outer-spin-button,
input::-webkit-inner-spin-button {
  -webkit-appearance: none;
  margin: 0;
}
*/


.preview{
  color: #fff;
  padding: 30px;
  width: 250px;
```

```
  position: relative;
}

.preview a{
  color: #fff;
  font-size: 12px;
  opacity: 0.6;
  margin-top: 30px;
  text-decoration: none;
}

.info{
  padding: 30px;
  position: relative;
  width: 100%;
}
.info p{
  font-size: 12px;
  font-weight: bold;
  text-transform: uppercase;
  opacity: 0.6;
}

.btnn{
  background: #e40046;
  border: 1px solid transparent;
  box-shadow: 0px 5px 10px rgba(0, 0, 0,0.6);
```

```
  color: #fff;
  font-size: 16px;
  padding: 10px 20px;
  position: absolute;
  bottom: 30px;
  right: 30px;
  letter-spacing: 1px;
  cursor: pointer;
  transition: all 0.4s ease-in-out;
  font-weight: bold;
  border-radius: 20px;
}
.it{
  padding: 10px 20px;
  position: absolute;
  top: 30px;
  right: 20px;
  letter-spacing: 1px;
}

.btnn:hover{
  background: #fff;
  color: #e40046;
  border: 1px solid #e40046;
  transform: scale(1.05);
}
.btnn:focus{
```

```
  outline: none;
}




@media(max-width:768px)
{
  .parent{
     flex-direction: column;
     width: 90%;
  }
  .preview
  {
     width: 100%;
     padding-top: 10px;
     padding-bottom: 0;
  }
  .preview h2{
     margin: 10px 0 0;
  }
  .preview a{
     margin-top: 10px;
  }
  .info h2{
     margin-top: 20px;
  }
```

```
 .info p{
    margin-bottom: 50px;
 }
 .btnn{
    padding: 10px 15px ;
    font-size: 14px;
 }
}



@media(max-width:320px)
{


 .preview
 {
    width: 100%;
    padding-top: 10px;
    padding-bottom: 0;
 }



 .info p{
    margin-bottom: 50px;
 }
 .btnn{
    padding: 10px 15px ;
    font-size: 14px;
```

```
    bottom: 37px;
  }
  #last{
    bottom: 20px;
  }

}



img.img-responsive {
  margin-left: auto;
  margin-right: auto;
  width: 125px;
  height: 125px;
}



#itemprice{

  padding-top: 10px;
  text-align: center;
  color: rgb(68, 52, 52);
  font-weight: 300;
  letter-spacing: 1px;

}
```

```
#lastcol{
  margin-bottom: 50px;
  box-shadow: none;
}

#lastcol p {
  color: black;
  font-size: medium;
}
```

ViewEncoder.component.html

```
<html>
  <head>
    <meta charset='utf-8'>
    <meta http-equiv='X-UA-Compatible' content='IE=edge'>
    <meta name='viewport' content='width=device-width, initial-scale=1'>
    <!-- <link rel='stylesheet' type='text/css' media='screen' href='uhome.css'> -->
    <!-- nav font -->
    <link href="https://fonts.googleapis.com/css?family=Audiowide&display=swap"
rel="stylesheet">




    <!-- Bootstrap CSS -->
    <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
```

```
    crossorigin="anonymous">



  <!-- fafa -->

  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">



 </head>

 <body>

  <app-ind></app-ind>

  <div class="container" id="container1">

   <div class="row">


    <div class="column" *ngFor="let recipe of recipes">

     <div class="parent">

      <div class="preview">

       <!-- <center> -->

       <img src="../../../assets/recipe/{{recipe.dishimage}}" alt="" class="img-responsive">

       <!-- </center> -->

      </div>


      <div class="info">


       <table>

        <tr>

         <td colspan="2">
```

```
    <h6>{{recipe.dishname}}</h6>
   </td>
  </tr>
  <tr>
   <td>
    <h6>Quantity:-</h6>
   </td>
   <td>
    <h6>{{recipe.quantity}}</h6>
   </td>
  </tr>
  <tr>
   <td>
    <h6>Price:-</h6>
   </td>

   <td>
    <h6><span class="product-price">{{recipe.price}}</span>RS</h6>
   </td>

  </tr>



  <tr>
   <td>
    <h6>Quantity:-</h6>
   </td>
```

```
            <td>
              <h6>{{recipe.qty}}</h6>
            </td>


          </tr>


        </table>


        <!-- <div class="product-quantity">
          <input type="number" id="qty_input4" class="ab form-control form-control-sm"
min="1"
            title="no of recipe"   (change)="update(recipe)">
        </div> -->
        <!-- <button class="btnn" (click)="deletefromcart(recipe)"> <i class="fa fa-trash" aria-
hidden="true"></i></button> -->
       </div>


      </div>
    </div>


      <!-----------------------------------------------total section-----------------------------------------------------------
-------  -->


    <div class="parent" style="background-color:rgb(235, 243, 251);margin-top: 15px;"
id="lastcol">


      <div class="info">
```

```
        <p style="margin-top: 5px;margin-bottom: 5px;">total = <b><span
class="tot">{{total}}</span>Rs</b></p>

        <!-- <p style="margin-top: 5px;margin-bottom: 5px;">delivery charge = <b><span
class="dc">50</span>Rs</b></p> -->

        <p style="margin-top: 5px;margin-bottom: 5px;">grand total =<b><span
class="gt">{{total}}</span>Rs</b></p>

        <!-- <button class="btn btn-success" id="last" style="margin-top: 5px;margin-bottom: 5px;"
(click)="checkout()">Check out</button> -->

       </div>

      </div>




    </div>


   </div>


 </body>

</html>
```

ViewEncoder.component.spec.ts

```
import { ComponentFixture, TestBed } from '@angular/core/testing';


import { ViewoneorderComponent } from './viewoneorder.component';


describe('ViewoneorderComponent', () => {
```

```
  let component: ViewoneorderComponent;
  let fixture: ComponentFixture<ViewoneorderComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ ViewoneorderComponent ]
    })
    .compileComponents();
  });

  beforeEach(() => {
    fixture = TestBed.createComponent(ViewoneorderComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

ViewEncoder.component.ts

```
import { HttpErrorResponse } from '@angular/common/http';

import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { AdminService } from 'src/app/services/admin.service';
```

```
@Component({
 selector: 'app-viewoneorder',
 templateUrl: './viewoneorder.component.html',
 styleUrls: ['./viewoneorder.component.css']
})
export class ViewoneorderComponent implements OnInit {

 constructor(private router: Router, private adminService: AdminService) { }
 oneorderid:any;
 arr: any[];
 recipes: any[];
 total:any;


 ngOnInit(): void {
  this.check()
  this.oneorderid =this.adminService.getoneOrder();
  this.view()
 }

 check() {
  this.adminService.check().subscribe(
   data => {
    console.log(data);
   },
   (error) => {
```

```
    if (error instanceof HttpErrorResponse) {


       this.router.navigate(['/login'])


    }
    console.log(error);
   }
 )
}
view()
{
 if(this.oneorderid==undefined)
 {
  this.router.navigate(['/login'])
 }
 else
 {
 this.adminService.getOneCartItem(this.oneorderid).subscribe(
  data => {


   this.arr = data[0];
   this.total=this.arr['total'];
   this.recipes=this.arr['recipe']
  },
  error => {
   console.log(error);
```

```
    }
   )
   }
 }
}

Viewoneuser.component.css
.table{
  margin-top: 150px;
  margin-left: auto;
  margin-right: auto;
}
th,td{
text-align: center;
}
html,body{
width: 100%;
height: 100%;
background-attachment: fixed;
background-repeat: no-repeat;
}
Viewoneuser.component.html
<!doctype html>
<html lang="en">
 <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">


    <!-- Bootstrap CSS -->

    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
integrity="sha384-
9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKGj7Sk"
crossorigin="anonymous">


 </head>

 <body>

   <app-ind></app-ind>

   <table class="table table-hover w-50 h-30" style="background-color: white;   box-shadow: 0px
0px 19px 5px rgba(0, 0, 0, 0.5);">


    <tbody>

     <tr>

       <th scope="row">Name</th>


       <td>{{name}}</td>


     </tr>

     <tr>


       <th scope="row">Email</th>

       <td >{{email}}</td>


     </tr>

     <tr>
```

```
        <th scope="row">Phone no</th>
        <td >{{contact}}</td>


      </tr>


    </tbody>
  </table>


 </body>
</html>


Viewoneuser.component.spec.ts
import { ComponentFixture, TestBed } from '@angular/core/testing';


import { ViewoneuserComponent } from './viewoneuser.component';


describe('ViewoneuserComponent', () => {
 let component: ViewoneuserComponent;
 let fixture: ComponentFixture<ViewoneuserComponent>;


 beforeEach(async () => {
  await TestBed.configureTestingModule({
   declarations: [ ViewoneuserComponent ]
  })
  .compileComponents();
 });
```

```
  beforeEach(() => {
    fixture = TestBed.createComponent(ViewoneuserComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

Viewoneuser.component.ts

```
import { HttpErrorResponse } from '@angular/common/http';
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { AdminService } from 'src/app/services/admin.service';
import { User } from 'src/app/user';

@Component({
  selector: 'app-viewoneuser',
  templateUrl: './viewoneuser.component.html',
  styleUrls: ['./viewoneuser.component.css']
})
export class ViewoneuserComponent implements OnInit {
  userid: any;
  public user:User;
  public name: any;
```

```
public email: any;

public contact: any;

constructor(private router: Router, private adminService: AdminService) { }


ngOnInit(): void {
  this.check()
  this.userid = this.adminService.getoneOrder();
  this.view()
}


check() {
  this.adminService.check().subscribe(
    data => {
      console.log(data);
    },
    (error) => {

      if (error instanceof HttpErrorResponse) {


        this.router.navigate(['/login'])


      }
      console.log(error);
    }
  )
}
```

```
 view() {
  if (this.userid == undefined) {
   this.router.navigate(['/login'])
  }
  else {
   this.adminService.getOneCartItemUser(this.userid).subscribe(
     data => {
      console.log(data);
      this.user=data['user']
      this.name = this.user['name']
      this.email = this.user['email']
      this.contact = this.user['contact']
     },
     error => {
      console.log(error);
     }
    )
  }
 }



}
Viewrecipe.component.css
.table{
 margin-top: 50px;
 margin-bottom: 200px;
 margin-left: auto;
```

```
  margin-right: auto;
}
th,td{
text-align: center;
}
html,body{

width: 100%;
height: 100%;
background-attachment: fixed;
background-repeat: no-repeat;
}
Viewrecipe.component.html
<!doctype html>
<html lang="en">

<head>
  <!-- Required meta tags -->
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

  <!-- Bootstrap CSS -->

  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"
    integrity="sha384-
9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKGj7Sk"
crossorigin="anonymous">
```

```
</head>


<body>
 <app-ind></app-ind>
 <table class="table table-hover w-50"
  style="background-color: white;   box-shadow: 0px 0px 19px 5px rgba(0, 0, 0, 0.5);">
  <thead>
   <tr>
    <th>Index</th>
    <th scope="col">Item Image</th>
    <th scope="col">Dish Name</th>
    <th scope="col">Update</th>
    <th scope="col">Delete</th>
   </tr>
  </thead>
  <tbody>


   <tr *ngFor="let recipe of recipes;index as i">


    <th scope="row">{{i+1}}</th>
    <td><img src="../../../assets/recipe/{{recipe.dishimage}}" alt="" height=100 width=100
/></td>
    <td>{{recipe.dishname}}</td>
    <td><button class="btn btn-link" href="#" (click)="editrecipe(recipe)"><i class="fas fa-edit"
aria-hidden="true"></i></button></td>
    <td><button class="btn btn-link" (click)="deleterecipe(recipe)"><i class="fa fa-trash" aria-
hidden="true"></i></button></td>
```

```
    </tr>


   </tbody>
  </table>


</body>


</html>
```

Viewrecipe.component.spec.ts

```
import { ComponentFixture, TestBed } from '@angular/core/testing';


import { ViewrecipeComponent } from './viewrecipe.component';


describe('ViewrecipeComponent', () => {
  let component: ViewrecipeComponent;
  let fixture: ComponentFixture<ViewrecipeComponent>;


  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ ViewrecipeComponent ]
    })
    .compileComponents();
  });


  beforeEach(() => {
    fixture = TestBed.createComponent(ViewrecipeComponent);
    component = fixture.componentInstance;
```

```
    fixture.detectChanges();
  });


  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

Viewrecipe.component.ts

```
import { HttpErrorResponse } from '@angular/common/http';
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { AdminService } from 'src/app/services/admin.service';
import { Recipe } from '../../recipe'
@Component({
  selector: 'app-viewrecipe',
  templateUrl: './viewrecipe.component.html',
  styleUrls: ['./viewrecipe.component.css']
})
export class ViewrecipeComponent implements OnInit {
  public recipes: Recipe[];
  avail: boolean;
  arr: any[];

  constructor(private router: Router, private adminService: AdminService) { }

  ngOnInit(): void {
    this.check()
```

```
  this.readRecipe()
}

check() {
  this.adminService.check().subscribe(
    data => {
      console.log(data);
    },
    (error) => {

      if (error instanceof HttpErrorResponse) {

          this.router.navigate(['/login'])

      }
      console.log(error);
    }
  )

}

readRecipe() {
  this.adminService.getAllRecipe().subscribe(
    data => {
      this.arr = data['msg'];
      this.recipes = data['msg'];
```

```
    },
    (error) => {

      if (error instanceof HttpErrorResponse) {

         this.router.navigate(['/login'])

      }
      console.log(error);
    }
  )
  // console.log();
}

deleterecipe(recipe) {


  var recipeid = recipe._id;
  this.adminService.deleterecipe(recipeid).subscribe(
    data => {
      // console.log(data);
      this.adminService.avail = true;
      this.adminService.msg = "Successfully Deleted a Recipe!!!";
      this.router.navigate(['/admin']);
    },
    (error) => {
```

```
      if (error instanceof HttpErrorResponse) {


         this.router.navigate(['/login'])


      }
      console.log(error);
     }
   )
 }
 editrecipe(recipe) {
   this.adminService.temp = recipe;
   this.router.navigate(['/admin/editrecipe']);
 }
}
Viewuser.component.css
.table{
 margin-top: 150px;
 margin-bottom: 250px;
 margin-left: auto;
 margin-right: auto;
}
th,td{
text-align: center;
}
html,body{


width: 100%;
```

```
height: 100%;

background-attachment: fixed;

background-repeat: no-repeat;

}


button{

  outline: none;

}

Viewuser.component.html

<html>


<head>

  <!-- Required meta tags -->

  <meta charset="utf-8">

  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">


  <!-- Bootstrap CSS -->

  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css"

    integrity="sha384-
9aIt2nRpC12Uk9gS9baDl411NQApFmC26EwAOH8WgZl5MYYxFfc+NcPb1dKGj7Sk"
crossorigin="anonymous">




</head>


<body>

  <app-ind></app-ind>
```

```
 <table class="table table-hover w-50"
  style="background-color: white; margin-top: 100px;  box-shadow: 0px 0px 19px 5px rgba(0, 0, 0, 0.5);">
  <thead>
   <tr>
    <th scope="col">#</th>
    <th scope="col">Name</th>
    <th scope="col">E-mail</th>
    <th scope="col">Block</th>
    <th scope="col">Unblock</th>
    <th scope="col">Delete</th>
   </tr>
  </thead>
  <tbody>
   <tr *ngFor="let user of users;index as i">
    <th scope="row">{{i+1}}</th>
    <td>{{user.name}}</td>
    <td>{{user.email}}</td>
    <td>
     <button class="btn btn-link" [disabled]="user.blocked" (click)="block(user)"><i class="fa fa-ban"
       aria-hidden="true"></i></button></td>
    <td><button class="btn btn-link" [disabled]="!user.blocked" (click)="unblock(user)"><i class="fa fa-unlock"
       aria-hidden="true"></i></button></td>
    <td><button class="btn btn-link" (click)="delete(user)"><i class="fa fa-trash" aria-hidden="true"></i></button>
    </td>
```

```
    </tr>


   </tbody>
  </table>
</body>


</html>
```

Viewuser.component.spec.ts

```
import { ComponentFixture, TestBed } from '@angular/core/testing';


import { ViewuserComponent } from './viewuser.component';


describe('ViewuserComponent', () => {
  let component: ViewuserComponent;
  let fixture: ComponentFixture<ViewuserComponent>;


  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ ViewuserComponent ]
    })
    .compileComponents();
  });


  beforeEach(() => {
    fixture = TestBed.createComponent(ViewuserComponent);
    component = fixture.componentInstance;
```

```
    fixture.detectChanges();
  });


  it('should create', () => {
    expect(component).toBeTruthy();
  });
Viewuser.component.ts
import { HttpErrorResponse } from '@angular/common/http';
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { AdminService } from 'src/app/services/admin.service';
import { User } from 'src/app/user';


@Component({
  selector: 'app-viewuser',
  templateUrl: './viewuser.component.html',
  styleUrls: ['./viewuser.component.css']
})
export class ViewuserComponent implements OnInit {
  msg: any = [];
  public users: User[];
  avail: boolean;
  arr: any[];
  constructor(private router: Router, private adminService: AdminService) { }


  ngOnInit(): void {
    this.check()
```

```
  this.readUser()
}

check() {
  this.adminService.check().subscribe(
    data => {
      console.log(data);
    },
    (error) => {

      if (error instanceof HttpErrorResponse) {

        this.router.navigate(['/login'])

      }
      console.log(error);
    }
  )

}

readUser() {
  this.adminService.getAllUser().subscribe(
    data => {
      this.arr = data['msg'];
      this.users = data['msg'];
      // console.log(data);
```

```
    },
    (error) => {

      if (error instanceof HttpErrorResponse) {

        this.router.navigate(['/login'])

      }
      console.log(error);
    }
  )
  // console.log();
}
block(user) {

  var userid = user._id;

  this.adminService.blockuser(userid).subscribe(
    data => {
      // console.log(data);
      this.adminService.avail = true;
      this.adminService.msg = "Successfully Blocked User!!!";
      this.router.navigate(['/admin']);
    },
    (error) => {

      if (error instanceof HttpErrorResponse) {
```

```
      this.router.navigate(['/login'])

    }
    console.log(error);
  }
 )
}

unblock(user) {

 var userid = user._id;

 this.adminService.unblockuser(userid).subscribe(
  data => {
   // console.log(data);
   this.adminService.avail = true;
   this.adminService.msg = "Successfully Unblocked User!!!";
   this.router.navigate(['/admin']);
  },
  (error) => {

   if (error instanceof HttpErrorResponse) {

     this.router.navigate(['/login'])

    }
```

```
      console.log(error);
    }
  )
 }

 delete(user) {

  var userid = user._id;
  this.adminService.deleteuser(userid).subscribe(
   data => {
    // console.log(data);
    this.adminService.avail = true;
    this.adminService.msg = "Successfully Deleted User!!!";
    this.router.navigate(['/admin']);
   },
   (error) => {

    if (error instanceof HttpErrorResponse) {

      this.router.navigate(['/login'])

    }
    console.log(error);
   }
  )
 }
}
```

Auth->login

Login.component.css

```
/* font */
.font-poppins {
  font-family: "Poppins", "Arial", "Helvetica Neue", sans-serif;
}



html {
  /* -webkit-box-sizing: border-box;
  -moz-box-sizing: border-box;
  box-sizing: border-box;
  width: 100%;
  height: 100%;
  background: linear-gradient(to top, #fbc2eb 0%, #a18cd1 100%);
  background: -webkit-gradient(linear, left bottom, left top, from(#fbc2eb), to(#a18cd1));
  background: -webkit-linear-gradient(bottom, #fbc2eb 0%, #a18cd1 100%);
  background: -moz-linear-gradient(bottom, #fbc2eb 0%, #a18cd1 100%);
  background: -o-linear-gradient(bottom, #fbc2eb 0%, #a18cd1 100%);
```

```
 background-attachment: fixed;
 background-repeat: no-repeat; */
 width:100%;
  height: 100vh;
  overflow-x: hidden;
  background-image:url("../../../assets/pic/71.jpg");
  background-repeat: no-repeat;
  background-size: 100% 100%;
  position: relative;
  background-attachment: fixed;

}

*, *:before, *:after {
 -webkit-box-sizing: inherit;
 -moz-box-sizing: inherit;
 box-sizing: inherit;
}

body{
 margin: 0;
 padding: 0;
}

/* fieldset {
```

```
  min-width: 0;


  border: 0;
}
 */
button {
  outline: none;
  background: none;
  border: none;
}

/* ==========================================================================
   #PAGE WRAPPER
   ========================================================================== */
.page-wrapper {
  min-height: 100vh;
}

body {
  font-family: "Poppins", "Arial", "Helvetica Neue", sans-serif;
  font-weight: 400;
  font-size: 14px;
}
```

```
/* ========================================================================
   #SPACING
   ======================================================================== */




.p-t-180 {
  padding-top: 180px;
}
.p-t-10 {
  padding-top: 10px;
}
.p-b-100 {
  padding-bottom: 100px;
}


/* ========================================================================
   #WRAPPER
   ======================================================================== */
.wrapper {
  margin: 0 auto;
}



.wrapper--w780 {
  max-width: 780px;
}
```

```
/* ========================================================================
  #BUTTON
  ======================================================================== */
.btn {
  display: inline-block;
  line-height: 40px;
  padding: 0 33px;
  font-family: Poppins, Arial, "Helvetica Neue", sans-serif;
  cursor: pointer;
  color: #fff;
  -webkit-transition: all 0.4s ease;
  -o-transition: all 0.4s ease;
  -moz-transition: all 0.4s ease;
  transition: all 0.4s ease;
  font-size: 18px;
}

.btn--radius {
  -webkit-border-radius: 3px;
  -moz-border-radius: 3px;
  border-radius: 3px;
}

.btn--pill {
  -webkit-border-radius: 20px;
```

```
  -moz-border-radius: 20px;
  border-radius: 20px;
}

.btn--green {
  background: #57b846;
}

.btn--green:hover {
  background: #318b21;
}

/* ========================================================================
   #FORM
   ======================================================================== */
input {
  outline: none;
  margin: 0;
  border: none;
  -webkit-box-shadow: none;
  -moz-box-shadow: none;
  box-shadow: none;
  width: 100%;
  background: transparent;
  font-size: 14px;
  font-family: inherit;
}
```

```
.input-group {
  position: relative;
  margin-bottom: 33px;
  background: transparent;
  border-bottom: 1px solid rgba(255, 255, 255, 0.2);
  display: flex;
}

.fa{
  padding-right: 10px;
  margin-bottom: 0;
  padding-top: 10px;
}




.input--style-3 {
  padding: 5px 0;
  font-size: 16px;
  color: #ccc;
  background: transparent;
}

.input--style-3::-webkit-input-placeholder {
  /* WebKit, Blink, Edge */
  color: #ccc;
```

```
}

.input--style-3:-moz-placeholder {
  /* Mozilla Firefox 4 to 18 */
  color: #ccc;
  opacity: 1;
}

.input--style-3::-moz-placeholder {
  /* Mozilla Firefox 19+ */
  color: #ccc;
  opacity: 1;
}

.input--style-3:-ms-input-placeholder {
  /* Internet Explorer 10-11 */
  color: #ccc;
}

.input--style-3:-ms-input-placeholder {
  /* Microsoft Edge */
  color: #ccc;
}

/* ==========================================================================
   #TITLE
   ========================================================================== */
```

```
.title {
  font-size: 24px;
  color: #fff;
  font-weight: 400;
  margin-bottom: 36px;
}

/* ========================================================================
   #CARD
   ======================================================================== */
.card {
  overflow: hidden;
  -webkit-border-radius: 3px;
  -moz-border-radius: 3px;
  border-radius: 3px;
  background: #fff;
  /* animation: expand 1s ease forwards; */
}

.iforgot{
  animation: forgot 1s ease forwards;
}

.ichangepassword{
animation: changepassword 1s ease forwards;
}
```

```
.ilogin{

animation: login 1s ease forwards;

}


.iregister{

animation: register 1s ease forwards;

}


@keyframes login{


0%  { transform: translate(0%, -100%);}

100%{ transform: translate(0, 0);}


}

@keyframes register{




  0%  { transform: translate(0%,-100%);}

  100%{ transform: translate(0, 0);}


}




@keyframes forgot{
```

```
0%  { transform: translate(-100%, -100%);}

100%{ transform: translate(0, 0);}



}


@keyframes changepassword{


0%  { transform: translate(-100%, -100%);}

100%{ transform: translate(0, 0);}



}









.card-3 {

 opacity: 0.89;

 background: #000;

 -webkit-border-radius: 10px;

 -moz-border-radius: 10px;

 border-radius: 10px;

 -webkit-box-shadow: 0px 8px 20px 0px rgba(0, 0, 0, 0.15);

 -moz-box-shadow: 0px 8px 20px 0px rgba(0, 0, 0, 0.15);

 box-shadow: 0px 8px 20px 0px rgba(0, 0, 0, 0.15);

 width: 100%;
```

```
  display: table;
}
.card-3 .card-body {
  padding: 57px 65px;
  padding-bottom: 65px;
  display: table-cell;
}

/* .container {

  animation: expand 1s ease forwards;



} */



.niche{
padding-top: 10px;
padding-left: 10px;
font-family: "Poppins", "Arial", "Helvetica Neue", sans-serif;
}
a{
text-decoration: none;
color: steelblue;
padding-left: 10px;
}
```

```
#message {
  font-family: "Poppins", "Arial", "Helvetica Neue", sans-serif;
  text-align: center;
  align-items: center;
  display:none;
  background:transparent;
  color: #000;
  position: relative;
  padding: 20px;
  margin-top: 10px;

}




#message p {
  padding: 10px 35px;
  font-size: 18px;
}

/* Add a green text color and a checkmark when the requirements are right */
.valid {
  color:#57b846;
}

.valid:before {
  position: relative;
```

```
  left: -35px;
  content: " ✓";
}

/* Add a red text color and an "x" icon when the requirements are wrong */
.invalid {
  color: rgb(238, 36, 36);
}

.invalid:before {
  position: relative;
  left: -35px;
  content: " ✗";
}
```

Login.component.html

```
<!DOCTYPE html>
<html lang="en">

<head>
  <link

href="https://fonts.googleapis.com/css?family=Poppins:100,100i,200,200i,300,300i,400,400i,500,500i,600,600i,700,700i,800,800i,900,900i"
    rel="stylesheet">

  <!-- fafa -->
  <!-- <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css"> -->
```

```
</head>


<body>
 <div class="container">
  <div class="page-wrapper bg-gra-01 p-t-180 p-b-100 font-poppins">
   <div class="wrapper wrapper--w780">
    <div class="card card-3 ilogin">
     <div class="card-body">


      <h2 class="title">Login Info</h2>


      <div  style="color: red;" *ngIf="avail" >
       <h3>  {{msg}}</h3>
      </div>




      <form #f="ngForm" (ngSubmit)="onSubmit(f)">
       <div class="input-group">
        <i class="fa fa-user" aria-hidden="true" style="color: white;"></i>
        <input class="input--style-3" type="email" pattern="[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-
z]{2,}$" placeholder="Enter Your Email" name="email" email ngModel required
#email="ngModel"
         autocomplete="off">
       </div>
       <div class="input-group">
        <i class="fa fa-key" aria-hidden="true" style="color: white;"></i>
        <input class="input--style-3" type="password" placeholder="Enter password"
```

```
name="password" autocomplete="off" pattern="(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{8,}"

            title="Plaese enter valid and strong password" ngModel required #password="ngModel">

          </div>

          <div class="p-t-10">

            <!-- <center> -->

            <button class="btn btn--pill btn--green" type="submit"
[disabled]="f.invalid">Login</button>


            <button class="btn btn--pill btn--green" style="margin-left: 10px;"
type="reset">Reset</button>

            <!-- </center> -->

          </div>

        </form>

        <div class="niche">

          <!-- <center> -->

          <a [routerLink]="['/reset']" id="fp">Forgot password?</a>

          <a [routerLink]="['/register']" routerLinkActive="active" id="fp">Don't have account?</a>

          <a [routerLink]="['/']"><i class="fa fa-arrow-circle-left" aria-hidden="true"

             style="color:white;">Goback</i></a>

          <!-- </center> -->

        </div>

      </div>


    </div>


  </div>
```

```
 </div>



</body>


</html>
```

Login.component.spec.ts

```
import { ComponentFixture, TestBed } from '@angular/core/testing';


import { LoginComponent } from './login.component';


describe('LoginComponent', () => {
 let component: LoginComponent;
 let fixture: ComponentFixture<LoginComponent>;


 beforeEach(async () => {
  await TestBed.configureTestingModule({
   declarations: [ LoginComponent ]
  })
  .compileComponents();
 });


 beforeEach(() => {
  fixture = TestBed.createComponent(LoginComponent);
  component = fixture.componentInstance;
  fixture.detectChanges();
```

```
  });


  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

Login.component.ts

```
import { Component, OnInit } from '@angular/core';
import { NgForm } from '@angular/forms';
import { Router } from '@angular/router';
import { AuthService } from 'src/app/services/auth.service';
import { ToastrService } from 'ngx-toastr';
@Component({
  selector: 'app-login',
  templateUrl: './login.component.html',
  styleUrls: ['./login.component.css']
})
export class LoginComponent implements OnInit {


  public msg: any = [];
  public avail: boolean;
  public isAdmin: boolean;
  public isBlocked: boolean;
  // var checkArray;
  arr: any = [];
  constructor(private router: Router, private toastr: ToastrService,private authService:
```

```
AuthService) { }


 ngOnInit(): void {
  this.isAdmin = false;
  this.isBlocked=false;
  if(this.authService.msg=="successfully changed password!!")
  {
    this.toastr.success('successfully changed password!!', '', {
      timeOut: 2000,
      closeButton: true
    });
  }
  if(this.authService.msg=="successfully registered a user!")
  {
    this.toastr.success('successfully registered a user!', '', {
      timeOut: 2000,
      closeButton: true
    });
  }
  if(this.authService.msg=="successfully password-reset done!!")
  {
    this.toastr.success('successfully password-reset done!!', '', {
      timeOut: 2000,
      closeButton: true
    });


  }
```

```
  this.readUser();
}
readUser() {
  this.authService.readUser().subscribe(
    data => {
      this.arr = data['msg'];
    },
    error => {
      console.log(error);
    }
  )
}



onSubmit(f: NgForm) {

  if (!f.valid) {
    this.msg = "Invalid Email or password";
    this.avail = true;
    // console.log(this.msg);
    return;
  }

  for (var val of this.arr) {
    var a = val['email'];
    var b = f.controls.email.value;
```

```
  var c = val['role'];
  var d = "admin";
  var e = val['blocked'];
  // console.log("blocked ?" + e);
  if (a == b && c == d) {
    this.isAdmin = true;
  }
  if (a == b && e) {
    this.isBlocked = true;
  }
}
// console.log("blocked> below = " + this.isBlocked);
// console.log(this.isAdmin);
if (this.isBlocked) {
  this.msg = "You are blocked by Admin wait until admin unblock you!!!";
  this.avail = true;
  // console.log(this.msg);
  return;
}

// console.log(JSON.stringify(this.loginForm.value));
this.authService.login(JSON.stringify(f.value))
  .subscribe(
    data => {
      console.log(data);

      if(data['error'])
```

```
{
  this.msg = data['error']
  this.avail=true;
  return;
}

if(this.isAdmin)
{
  console.log("in admin");
  // if admin
  localStorage.setItem('token', data['token']);
  localStorage.setItem('N1@#I2@#M3@#D4@#A6', 'yU@$SVBs6Hh5sGggbAbf50');
  this.router.navigate(['/admin']);
}
else
{
  // if customer
  console.log("in customer");
  localStorage.setItem('token', data['token']);
  localStorage.setItem('userid', f.controls.email.value);
  localStorage.setItem('N1@#I2@#M3@#D4@#A6', 'nU@$SVBs6Hh5sGggbAbf50');
  this.router.navigate(['/userhome']);
 }

},
error => {
```

```
        console.error(error);
        this.msg = error;
       }
    )
 }
}
```

Register.component.css

```
/* font */
.font-poppins {
 font-family: "Poppins", "Arial", "Helvetica Neue", sans-serif;
}

html {
 /* -webkit-box-sizing: border-box;
 -moz-box-sizing: border-box;
 box-sizing: border-box;
 width: 100%;
 height: 100%;
 background: linear-gradient(to top, #fbc2eb 0%, #a18cd1 100%);
 background: -webkit-gradient(linear, left bottom, left top, from(#fbc2eb), to(#a18cd1));
 background: -webkit-linear-gradient(bottom, #fbc2eb 0%, #a18cd1 100%);
 background: -moz-linear-gradient(bottom, #fbc2eb 0%, #a18cd1 100%);
 background: -o-linear-gradient(bottom, #fbc2eb 0%, #a18cd1 100%);
```

```
 background-attachment: fixed;
 background-repeat: no-repeat; */
 width:100%;
  height: 100vh;
  overflow-x: hidden;
  background-image:url("../../../assets/pic/71.jpg");
  background-repeat: no-repeat;
  background-size: 100% 100%;
  position: relative;
  background-attachment: fixed;

}

*, *:before, *:after {
 -webkit-box-sizing: inherit;
 -moz-box-sizing: inherit;
 box-sizing: inherit;
}

body{
 margin: 0;
 padding: 0;
}

/* fieldset {
 min-width: 0;
```

```
  border: 0;
}
 */
button {
  outline: none;
  background: none;
  border: none;
}

/* ========================================================================
   #PAGE WRAPPER
   ======================================================================== */
.page-wrapper {
  min-height: 100vh;
}

body {
  font-family: "Poppins", "Arial", "Helvetica Neue", sans-serif;
  font-weight: 400;
  font-size: 14px;
}




/* ========================================================================
```

```
  #SPACING

  ========================================================================== */




.p-t-180 {

 padding-top: 180px;

}

.p-t-10 {

 padding-top: 10px;

}

.p-b-100 {

 padding-bottom: 100px;

}


/* ==========================================================================

  #WRAPPER

  ========================================================================== */

.wrapper {

 margin: 0 auto;

}




.wrapper--w780 {

 max-width: 780px;

}
```

```
/* ========================================================================
   #BUTTON
   ======================================================================== */
.btn {
  display: inline-block;
  line-height: 40px;
  padding: 0 33px;
  font-family: Poppins, Arial, "Helvetica Neue", sans-serif;
  cursor: pointer;
  color: #fff;
  -webkit-transition: all 0.4s ease;
  -o-transition: all 0.4s ease;
  -moz-transition: all 0.4s ease;
  transition: all 0.4s ease;
  font-size: 18px;
}

.btn--radius {
  -webkit-border-radius: 3px;
  -moz-border-radius: 3px;
  border-radius: 3px;
}

.btn--pill {
  -webkit-border-radius: 20px;
  -moz-border-radius: 20px;
```

```
  border-radius: 20px;
}

.btn--green {
  background: #57b846;
}

.btn--green:hover {
  background: #318b21;
}

/* ========================================================================
   #FORM
   ======================================================================== */
input {
  outline: none;
  margin: 0;
  border: none;
  -webkit-box-shadow: none;
  -moz-box-shadow: none;
  box-shadow: none;
  width: 100%;
  background: transparent;
  font-size: 14px;
  font-family: inherit;
}
```

```
.input-group {
  position: relative;
  margin-bottom: 33px;
  background: transparent;
  border-bottom: 1px solid rgba(255, 255, 255, 0.2);
  display: flex;
}

.fa{
  padding-right: 10px;
  margin-bottom: 0;
  padding-top: 10px;
}




.input--style-3 {
  padding: 5px 0;
  font-size: 16px;
  color: #ccc;
  background: transparent;
}

.input--style-3::-webkit-input-placeholder {
  /* WebKit, Blink, Edge */
  color: #ccc;
}
```

```
.input--style-3:-moz-placeholder {
  /* Mozilla Firefox 4 to 18 */
  color: #ccc;
  opacity: 1;
}

.input--style-3::-moz-placeholder {
  /* Mozilla Firefox 19+ */
  color: #ccc;
  opacity: 1;
}

.input--style-3:-ms-input-placeholder {
  /* Internet Explorer 10-11 */
  color: #ccc;
}

.input--style-3:-ms-input-placeholder {
  /* Microsoft Edge */
  color: #ccc;
}

/* ==========================================================================
   #TITLE
   ========================================================================== */
.title {
```

```
  font-size: 24px;
  color: #fff;
  font-weight: 400;
  margin-bottom: 36px;
}

/* ========================================================================
   #CARD
   ======================================================================== */
.card {
  overflow: hidden;
  -webkit-border-radius: 3px;
  -moz-border-radius: 3px;
  border-radius: 3px;
  background: #fff;
  /* animation: expand 1s ease forwards; */
}

.iforgot{
  animation: forgot 1s ease forwards;
}

.ichangepassword{
animation: changepassword 1s ease forwards;
}

.ilogin{
```

```
animation: login 1s ease forwards;

}


.iregister{

animation: register 1s ease forwards;

}


@keyframes login{


0%  { transform: translate(0%, -100%);}

100%{ transform: translate(0, 0);}


}

@keyframes register{




  0%  { transform: translate(0%,-100%);}

  100%{ transform: translate(0, 0);}


}




@keyframes forgot{


0%  { transform: translate(-100%, -100%);}
```

```
100%{ transform: translate(0, 0);}


}


@keyframes changepassword{


0%  { transform: translate(-100%, -100%);}

100%{ transform: translate(0, 0);}


}









.card-3 {

  opacity: 0.89;

  background: #000;

  -webkit-border-radius: 10px;

  -moz-border-radius: 10px;

  border-radius: 10px;

  -webkit-box-shadow: 0px 8px 20px 0px rgba(0, 0, 0, 0.15);

  -moz-box-shadow: 0px 8px 20px 0px rgba(0, 0, 0, 0.15);

  box-shadow: 0px 8px 20px 0px rgba(0, 0, 0, 0.15);

  width: 100%;

  display: table;
```

```
}
.card-3 .card-body {
  padding: 57px 65px;
  padding-bottom: 65px;
  display: table-cell;
}

/* .container {

  animation: expand 1s ease forwards;



} */



.niche{
padding-top: 10px;
padding-left: 10px;
font-family: "Poppins", "Arial", "Helvetica Neue", sans-serif;
}
a{
text-decoration: none;
color: steelblue;
padding-left: 10px;
}

#message {
```

```
  font-family: "Poppins", "Arial", "Helvetica Neue", sans-serif;
  text-align: center;
  align-items: center;
  display:none;
  background:transparent;
  color: #000;
  position: relative;
  padding: 20px;
  margin-top: 10px;

}




#message p {
  padding: 10px 35px;
  font-size: 18px;
}

/* Add a green text color and a checkmark when the requirements are right */
.valid {
  color:#57b846;
}

.valid:before {
  position: relative;
  left: -35px;
```

```
  content: " ✓";
}

/* Add a red text color and an "x" icon when the requirements are wrong */
.invalid {
  color: rgb(238, 36, 36);
}

.invalid:before {
  position: relative;
  left: -35px;
  content: " ✗";
}
```

Register.component.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <link

href="https://fonts.googleapis.com/css?family=Poppins:100,100i,200,200i,300,300i,400,400i,500,500i,600,600i,700,700i,800,800i,900,900i"
    rel="stylesheet">

  <!-- fafa -->
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">

</head>
```

```
<body>


 <div class="container">
  <div class="page-wrapper bg-gra-01 p-t-180 p-b-100 font-poppins">
   <div class="wrapper wrapper--w780">
    <div class="card card-3 iregister">
     <div class="card-body">
      <h2 class="title">Registration Info</h2>
      <div  style="color: red;" *ngIf="avail" >
      <h3>  {{msg}}</h3>
      </div>
      <!-- <div style="color: red;">
       value = {{f.value | json}}
       valid = {{f.valid}}
      </div> -->
      <form #f="ngForm" (ngSubmit)="onSubmit(f)">
       <div class="input-group">
        <input class="input--style-3" type="text" placeholder="Name" name="name"
autocomplete="off" ngModel required #name="ngModel">
       </div>
       <!-- <div style="color: red;">
        name value = {{name.value}}
        valid = {{name.valid}}
        dirty = {{name.dirty}}
        touched = {{name.touched}}
       </div> -->
```

```
        <div class="input-group">
          <input class="input--style-3" type="email" placeholder="Email" name="email"
autocomplete="off" ngModel email
            required #email="ngModel" pattern="[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,}$">
        </div>
        <div class="input-group">
          <input class="input--style-3" type="text" placeholder="Phone" name="contact"
            pattern="^(\+91[\-\s]?)?[0]?(91)?[789]\d{9}$" autocomplete="off" ngModel required
#contact="ngModel">
        </div>
        <div class="input-group">
          <input class="input--style-3" type="password" placeholder="Enter password" name="p1"
id="p1"
            autocomplete="off" pattern="(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{8,}"
            title="Plaese enter valid and strong password" ngModel required #p1="ngModel">
        </div>
        <!-- <div id="message">
          <p id="letter" class="invalid">A lowercase letter</p>
          <p id="capital" class="invalid">A capital (uppercase) letter</p>
          <p id="number" class="invalid">A number</p>
          <p id="length" class="invalid">Minimum 8 characters</p>
        </div> -->

        <div class="input-group">
          <input class="input--style-3" type="password" placeholder="Re-enter password"
name="p2" id="p2"
            pattern="(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{8,}" title="Plaese enter valid and strong
password"
```

```
            autocomplete="off" ngModel required #p2="ngModel">

          </div>

          <div class="p-t-10">

            <!-- <center> -->

              <button class="btn btn--pill btn--green" [disabled]="f.invalid"
type="submit">Register</button>

              <button class="btn btn--pill btn--green" style="margin-left: 10px;"
type="reset">Reset</button>

            <!-- </center> -->

          </div>









        </form>



        <div class="niche">

          <!-- <center> -->

            <a [routerLink]="['/']"><i class="fa fa-arrow-circle-left" aria-hidden="true"
style="color:white;">

               Goback</i></a>

            <a [routerLink]="['/login']"  id="fp">Login?</a>

          <!-- </center> -->

        </div>

       </div>

      </div>

     </div>

   </div>

 </div>
```

```
</body>


</html>
```

Register.component.spec.ts

```
import { ComponentFixture, TestBed } from '@angular/core/testing';


import { RegisterComponent } from './register.component';


describe('RegisterComponent', () => {
  let component: RegisterComponent;
  let fixture: ComponentFixture<RegisterComponent>;


  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ RegisterComponent ]
    })
    .compileComponents();
  });


  beforeEach(() => {
    fixture = TestBed.createComponent(RegisterComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });


  it('should create', () => {
```

```
    expect(component).toBeTruthy();
  });
});
```

Register.component.ts

```
import { Component, OnInit } from '@angular/core';
import { NgForm, FormGroup, FormControl, Validators } from '@angular/forms';
import { Router } from '@angular/router';
import { AuthService } from 'src/app/services/auth.service';


@Component({
  selector: 'app-register',
  templateUrl: './register.component.html',
  styleUrls: ['./register.component.css']
})
export class RegisterComponent implements OnInit {


  msg: any = [];
  avail: boolean;
  // var checkArray;
  arr: any[];
  constructor(private router: Router, private authService: AuthService) { }


  ngOnInit(): void {
    this.readUser()
  }
  readUser() {
    this.authService.readUser().subscribe(
```

```
    data => {
      this.arr = data['msg'];
    },
    error => {
      console.log(error);
    }
  )
}
onSubmit(f: NgForm) {

  for (var val of this.arr) {
    var a = val['email'];
    var b = f.controls.email.value;
    if (a == b) {
      this.msg = "User already exist with this user name (email)!!";
      this.avail = true;
      return;
    }
  }

  if (f.controls.p1.value != f.controls.p2.value) {
    this.msg = "Password   doesn't match";
    this.avail = true;
    return;
  }

  if (!f.valid) {
```

```
    this.msg = "Invalid Form Fields";
    this.avail = true;
    return;
   }



   this.authService.register(JSON.stringify(f.value))
    .subscribe(
      data => {
       this.authService.msg = "successfully registered a user!";
       this.router.navigate(['/login']);
      },
      error => { console.error(error); this.msg = error; }
    )
  }
}
```

Reset.component.css

```
/* font */
.font-poppins {
  font-family: "Poppins", "Arial", "Helvetica Neue", sans-serif;
}



html {
  /* -webkit-box-sizing: border-box;
  -moz-box-sizing: border-box;
```

```
  box-sizing: border-box;
  width: 100%;
  height: 100%;
  background: linear-gradient(to top, #fbc2eb 0%, #a18cd1 100%);
  background: -webkit-gradient(linear, left bottom, left top, from(#fbc2eb), to(#a18cd1));
  background: -webkit-linear-gradient(bottom, #fbc2eb 0%, #a18cd1 100%);
  background: -moz-linear-gradient(bottom, #fbc2eb 0%, #a18cd1 100%);
  background: -o-linear-gradient(bottom, #fbc2eb 0%, #a18cd1 100%);




  background-attachment: fixed;
  background-repeat: no-repeat; */
  width:100%;
    height: 100vh;
    overflow-x: hidden;
    background-image:url("../../../assets/pic/71.jpg");
    background-repeat: no-repeat;
    background-size: 100% 100%;
    position: relative;
    background-attachment: fixed;

}


*, *:before, *:after {
  -webkit-box-sizing: inherit;
  -moz-box-sizing: inherit;
```

```
  box-sizing: inherit;
}

body{
  margin: 0;
  padding: 0;
}

/* fieldset {
  min-width: 0;

  border: 0;
}
 */
button {
  outline: none;
  background: none;
  border: none;
}

/* ========================================================================
   #PAGE WRAPPER
   ======================================================================== */
.page-wrapper {
  min-height: 100vh;
}
```

```
body {
  font-family: "Poppins", "Arial", "Helvetica Neue", sans-serif;
  font-weight: 400;
  font-size: 14px;
}




/* ========================================================================
   #SPACING
   ======================================================================== */



.p-t-180 {
  padding-top: 180px;
}
.p-t-10 {
  padding-top: 10px;
}
.p-b-100 {
  padding-bottom: 100px;
}


/* ========================================================================
   #WRAPPER
```

```
  ========================================================================== */

.wrapper {

 margin: 0 auto;

}



.wrapper--w780 {

 max-width: 780px;

}



/* ==========================================================================

  #BUTTON

  ========================================================================== */

.btn {

 display: inline-block;

 line-height: 40px;

 padding: 0 33px;

 font-family: Poppins, Arial, "Helvetica Neue", sans-serif;

 cursor: pointer;

 color: #fff;

 -webkit-transition: all 0.4s ease;

 -o-transition: all 0.4s ease;

 -moz-transition: all 0.4s ease;

 transition: all 0.4s ease;

 font-size: 18px;

}
```

```
.btn--radius {
  -webkit-border-radius: 3px;
  -moz-border-radius: 3px;
  border-radius: 3px;
}

.btn--pill {
  -webkit-border-radius: 20px;
  -moz-border-radius: 20px;
  border-radius: 20px;
}

.btn--green {
  background: #57b846;
}

.btn--green:hover {
  background: #318b21;
}

/* ========================================================================
   #FORM
   ======================================================================== */
input {
  outline: none;
  margin: 0;
```

```
  border: none;
  -webkit-box-shadow: none;
  -moz-box-shadow: none;
  box-shadow: none;
  width: 100%;
  background: transparent;
  font-size: 14px;
  font-family: inherit;
}

.input-group {
  position: relative;
  margin-bottom: 33px;
  background: transparent;
  border-bottom: 1px solid rgba(255, 255, 255, 0.2);
  display: flex;
}

.fa{
  padding-right: 10px;
  margin-bottom: 0;
  padding-top: 10px;
}




.input--style-3 {
```

```
  padding: 5px 0;
  font-size: 16px;
  color: #ccc;
  background: transparent;
}

.input--style-3::-webkit-input-placeholder {
  /* WebKit, Blink, Edge */
  color: #ccc;
}

.input--style-3:-moz-placeholder {
  /* Mozilla Firefox 4 to 18 */
  color: #ccc;
  opacity: 1;
}

.input--style-3::-moz-placeholder {
  /* Mozilla Firefox 19+ */
  color: #ccc;
  opacity: 1;
}

.input--style-3:-ms-input-placeholder {
  /* Internet Explorer 10-11 */
  color: #ccc;
}
```

```
.input--style-3:-ms-input-placeholder {
  /* Microsoft Edge */
  color: #ccc;
}

/* ============================================================================
   #TITLE
   ============================================================================ */
.title {
  font-size: 24px;
  color: #fff;
  font-weight: 400;
  margin-bottom: 36px;
}

/* ============================================================================
   #CARD
   ============================================================================ */
.card {
  overflow: hidden;
  -webkit-border-radius: 3px;
  -moz-border-radius: 3px;
  border-radius: 3px;
  background: #fff;
  /* animation: expand 1s ease forwards; */
}
```

```
.iforgot{
  animation: forgot 1s ease forwards;
}

.ichangepassword{
animation: changepassword 1s ease forwards;
}

.ilogin{
animation: login 1s ease forwards;
}

.iregister{
animation: register 1s ease forwards;
}

@keyframes login{

0%  { transform: translate(0%, -100%);}
100%{ transform: translate(0, 0);}

}
@keyframes register{
```

```
 0%  { transform: translate(0%,-100%);}

 100%{ transform: translate(0, 0);}


}




@keyframes forgot{


0%  { transform: translate(-100%, -100%);}

100%{ transform: translate(0, 0);}


}


@keyframes changepassword{


0%  { transform: translate(-100%, -100%);}

100%{ transform: translate(0, 0);}


}









.card-3 {
```

```
  opacity: 0.89;
  background: #000;
  -webkit-border-radius: 10px;
  -moz-border-radius: 10px;
  border-radius: 10px;
  -webkit-box-shadow: 0px 8px 20px 0px rgba(0, 0, 0, 0.15);
  -moz-box-shadow: 0px 8px 20px 0px rgba(0, 0, 0, 0.15);
  box-shadow: 0px 8px 20px 0px rgba(0, 0, 0, 0.15);
  width: 100%;
  display: table;
}
.card-3 .card-body {
  padding: 57px 65px;
  padding-bottom: 65px;
  display: table-cell;
}

/* .container {

  animation: expand 1s ease forwards;



} */



.niche{
padding-top: 10px;
```

```
padding-left: 10px;
font-family: "Poppins", "Arial", "Helvetica Neue", sans-serif;
}
a{
text-decoration: none;
color: steelblue;
padding-left: 10px;
}

#message {
  font-family: "Poppins", "Arial", "Helvetica Neue", sans-serif;
  text-align: center;
  align-items: center;
  display:none;
  background:transparent;
  color: #000;
  position: relative;
  padding: 20px;
  margin-top: 10px;

}

#message p {
  padding: 10px 35px;
  font-size: 18px;
```

```
}

/* Add a green text color and a checkmark when the requirements are right */
.valid {
  color:#57b846;
}

.valid:before {
  position: relative;
  left: -35px;
  content: " ✓";
}

/* Add a red text color and an "x" icon when the requirements are wrong */
.invalid {
  color: rgb(238, 36, 36);
}

.invalid:before {
  position: relative;
  left: -35px;
  content: " ✗";
}
```

Reset.component.html

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

  <link

href="https://fonts.googleapis.com/css?family=Poppins:100,100i,200,200i,300,300i,400,400i,500,
500i,600,600i,700,700i,800,800i,900,900i"

    rel="stylesheet">


</head>


<body>




  <div class="container">

    <div class="page-wrapper p-t-180 p-b-100 font-poppins">

      <div class="wrapper wrapper--w780">

        <div class="card card-3 iforgot">

          <div class="card-body">

            <h2 class="title">Forgot Password</h2>

            <div  style="color: red;" *ngIf="avail" >

             <h3>  {{msg}}</h3>

             </div>

            <form #f="ngForm" (ngSubmit)="onSubmit(f)">

              <div class="input-group">


                  <input class="input--style-3" pattern="[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,}$"
type="email" placeholder="Enter Your Email" name="email" ngModel required
```

```
#email="ngModel"
                        email
                        autocomplete="off">
                    </div>

                    <div class="p-t-10">
                    <!-- <center> -->
                     <button class="btn btn--pill btn--green" [disabled]="f.invalid"
type="submit">submit</button>
                    <!-- </center> -->

                    </div>
                  </form>
                  <div class="niche">
                  <!-- <center> -->
                    <a [routerLink]="['/login']"><i class="fa fa-arrow-circle-left" aria-hidden="true"
                          style="color:white;"> Goback</i></a>
                        <!-- </center> -->
                  </div>
                </div>

              </div>

          </div>
      </div>

    </div>
```

```
</body>


</html>
```

Reset.component.spec.ts

```
import { ComponentFixture, TestBed } from '@angular/core/testing';


import { ResetComponent } from './reset.component';


describe('ResetComponent', () => {
  let component: ResetComponent;
  let fixture: ComponentFixture<ResetComponent>;


  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ ResetComponent ]
    })
    .compileComponents();
  });


  beforeEach(() => {
    fixture = TestBed.createComponent(ResetComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });


  it('should create', () => {
```

```
    expect(component).toBeTruthy();
  });
});
```

Reset.component.ts

```
import { Component, OnInit } from '@angular/core';
import { NgForm } from '@angular/forms';
import { Router } from '@angular/router';
import { AuthService } from 'src/app/services/auth.service';


@Component({
  selector: 'app-reset',
  templateUrl: './reset.component.html',
  styleUrls: ['./reset.component.css']
})
export class ResetComponent implements OnInit {
  msg : any =[];
  avail:boolean;
  arr:any[];
  checkmail:boolean;
  constructor(private router:Router, private authService:AuthService) { }


  ngOnInit(): void {
    this.checkmail=false;
    this.readUser()
  }
  readUser()
  {
```

```
  this.authService.readUser().subscribe(
    data=>{
      this.arr = data['msg'];
    },
    error=>{
      console.log(error);
    }
  )
}

onSubmit(f:NgForm)
{
  for (var val of this.arr)
  {
    var a = val['email'];
    var b = f.controls.email.value;
    if(a==b)
    {
      this.checkmail=true;
    }
  }
  if(this.checkmail==false)
  {
    this.msg = "User does not exist with this mail!!";
    this.avail=true;
    return;
  }
```

```
  this.authService.reset(JSON.stringify(f.value))
  .subscribe(
    data=> {console.log(data); this.router.navigate(['/reset-password']);},
    error=>{console.error(error);this.msg = error;}
  )
 }

}
```

Resetpasswod.component.css

```
/* font */
.font-poppins {
 font-family: "Poppins", "Arial", "Helvetica Neue", sans-serif;
}



html {
 /* -webkit-box-sizing: border-box;
 -moz-box-sizing: border-box;
 box-sizing: border-box;
 width: 100%;
 height: 100%;
 background: linear-gradient(to top, #fbc2eb 0%, #a18cd1 100%);
 background: -webkit-gradient(linear, left bottom, left top, from(#fbc2eb), to(#a18cd1));
 background: -webkit-linear-gradient(bottom, #fbc2eb 0%, #a18cd1 100%);
```

```
  background: -moz-linear-gradient(bottom, #fbc2eb 0%, #a18cd1 100%);
  background: -o-linear-gradient(bottom, #fbc2eb 0%, #a18cd1 100%);




  background-attachment: fixed;
  background-repeat: no-repeat; */
  width:100%;
    height: 100vh;
    overflow-x: hidden;
    background-image:url("../../../assets/pic/71.jpg");
    background-repeat: no-repeat;
    background-size: 100% 100%;
    position: relative;
    background-attachment: fixed;


}


*, *:before, *:after {
  -webkit-box-sizing: inherit;
  -moz-box-sizing: inherit;
  box-sizing: inherit;
}


body{
  margin: 0;
  padding: 0;
```

```
}


/* fieldset {
  min-width: 0;

  border: 0;
}
 */
button {
  outline: none;
  background: none;
  border: none;
}

/* ========================================================================
   #PAGE WRAPPER
   ======================================================================== */
.page-wrapper {
  min-height: 100vh;
}

body {
  font-family: "Poppins", "Arial", "Helvetica Neue", sans-serif;
  font-weight: 400;
  font-size: 14px;
}
```

```
/* ================================================================================
  #SPACING
  ================================================================================ */



.p-t-180 {
  padding-top: 180px;
}
.p-t-10 {
  padding-top: 10px;
}
.p-b-100 {
  padding-bottom: 100px;
}


/* ================================================================================
  #WRAPPER
  ================================================================================ */
.wrapper {
  margin: 0 auto;
}
```

```
.wrapper--w780 {
  max-width: 780px;
}



/* ========================================================================
   #BUTTON
   ======================================================================== */
.btn {
  display: inline-block;
  line-height: 40px;
  padding: 0 33px;
  font-family: Poppins, Arial, "Helvetica Neue", sans-serif;
  cursor: pointer;
  color: #fff;
  -webkit-transition: all 0.4s ease;
  -o-transition: all 0.4s ease;
  -moz-transition: all 0.4s ease;
  transition: all 0.4s ease;
  font-size: 18px;
}


.btn--radius {
  -webkit-border-radius: 3px;
  -moz-border-radius: 3px;
  border-radius: 3px;
}
```

```
.btn--pill {
  -webkit-border-radius: 20px;
  -moz-border-radius: 20px;
  border-radius: 20px;
}

.btn--green {
  background: #57b846;
}

.btn--green:hover {
  background: #318b21;
}

/* ========================================================================
   #FORM
   ======================================================================== */
input {
  outline: none;
  margin: 0;
  border: none;
  -webkit-box-shadow: none;
  -moz-box-shadow: none;
  box-shadow: none;
  width: 100%;
  background: transparent;
```

```
  font-size: 14px;
  font-family: inherit;
}

.input-group {
  position: relative;
  margin-bottom: 33px;
  background: transparent;
  border-bottom: 1px solid rgba(255, 255, 255, 0.2);
  display: flex;
}

.fa{
  padding-right: 10px;
  margin-bottom: 0;
  padding-top: 10px;
}



.input--style-3 {
  padding: 5px 0;
  font-size: 16px;
  color: #ccc;
  background: transparent;
}
```

```
.input--style-3::-webkit-input-placeholder {
  /* WebKit, Blink, Edge */
  color: #ccc;
}

.input--style-3:-moz-placeholder {
  /* Mozilla Firefox 4 to 18 */
  color: #ccc;
  opacity: 1;
}

.input--style-3::-moz-placeholder {
  /* Mozilla Firefox 19+ */
  color: #ccc;
  opacity: 1;
}

.input--style-3:-ms-input-placeholder {
  /* Internet Explorer 10-11 */
  color: #ccc;
}

.input--style-3:-ms-input-placeholder {
  /* Microsoft Edge */
  color: #ccc;
}
```

```
/* ==========================================================================

  #TITLE

  ========================================================================== */

.title {

 font-size: 24px;

 color: #fff;

 font-weight: 400;

 margin-bottom: 36px;

}


/* ==========================================================================

  #CARD

  ========================================================================== */

.card {

 overflow: hidden;

 -webkit-border-radius: 3px;

 -moz-border-radius: 3px;

 border-radius: 3px;

 background: #fff;

 /* animation: expand 1s ease forwards; */

}


.iforgot{

 animation: forgot 1s ease forwards;

}


.ichangepassword{
```

```
animation: changepassword 1s ease forwards;

}


.ilogin{

animation: login 1s ease forwards;

}


.iregister{

animation: register 1s ease forwards;

}


@keyframes login{


0%  { transform: translate(0%, -100%);}

100%{ transform: translate(0, 0);}


}

@keyframes register{




  0%  { transform: translate(0%,-100%);}

  100%{ transform: translate(0, 0);}


}
```

```
@keyframes forgot{

0%  { transform: translate(-100%, -100%);}

100%{ transform: translate(0, 0);}

}

@keyframes changepassword{

0%  { transform: translate(-100%, -100%);}

100%{ transform: translate(0, 0);}

}









.card-3 {
  opacity: 0.89;
  background: #000;
  -webkit-border-radius: 10px;
  -moz-border-radius: 10px;
  border-radius: 10px;
  -webkit-box-shadow: 0px 8px 20px 0px rgba(0, 0, 0, 0.15);
```

```
  -moz-box-shadow: 0px 8px 20px 0px rgba(0, 0, 0, 0.15);
  box-shadow: 0px 8px 20px 0px rgba(0, 0, 0, 0.15);
  width: 100%;
  display: table;
}
.card-3 .card-body {
  padding: 57px 65px;
  padding-bottom: 65px;
  display: table-cell;
}

/* .container {

  animation: expand 1s ease forwards;



} */



.niche{
padding-top: 10px;
padding-left: 10px;
font-family: "Poppins", "Arial", "Helvetica Neue", sans-serif;
}
a{
text-decoration: none;
color: steelblue;
```

```
padding-left: 10px;
}

#message {
  font-family: "Poppins", "Arial", "Helvetica Neue", sans-serif;
  text-align: center;
  align-items: center;
  display:none;
  background:transparent;
  color: #000;
  position: relative;
  padding: 20px;
  margin-top: 10px;

}



#message p {
  padding: 10px 35px;
  font-size: 18px;
}

/* Add a green text color and a checkmark when the requirements are right */
.valid {
  color:#57b846;
}
```

```
.valid:before {
  position: relative;
  left: -35px;
  content: " ✓ ";
}

/* Add a red text color and an "x" icon when the requirements are wrong */
.invalid {
  color: rgb(238, 36, 36);
}

.invalid:before {
  position: relative;
  left: -35px;
  content: " ✗ ";
}
```

Resetpasswod.componenthtml

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
  <link

href="https://fonts.googleapis.com/css?family=Poppins:100,100i,200,200i,300,300i,400,400i,500,
```

```
500i,600,600i,700,700i,800,800i,900,900i"

   rel="stylesheet">


</head>


<body>



 <div class="container">

  <div class="page-wrapper p-t-180 p-b-100 font-poppins">

    <div class="wrapper wrapper--w780">

      <div class="card card-3 iforgot">

        <div class="card-body">

          <h2 class="title">Change Password</h2>

          <form>

            <div class="input-group">


              <input class="input--style-3" type="text"

                 name="email" required  autocomplete="off" disabled

                value="we sended you otp click below link for change password">

            </div>




          </form>

          <div class="niche">

            <!-- <center> -->

            <a [routerLink]="['/reset-password-done']">change-password</a>
```

```
          <a [routerLink]="['/login']"><i class="fa fa-arrow-circle-left" aria-hidden="true"
style="color:white;margin-left: 10px;">

              Goback</i></a>

           <!-- </center> -->

          </div>

        </div>


      </div>


    </div>

   </div>


  </div>


</body>


</html>
```

Resetpasswod.component.spec.ts

```
import { ComponentFixture, TestBed } from '@angular/core/testing';


import { ResetPasswordComponent } from './reset-password.component';


describe('ResetPasswordComponent', () => {

  let component: ResetPasswordComponent;

  let fixture: ComponentFixture<ResetPasswordComponent>;


  beforeEach(async () => {
```

```
  await TestBed.configureTestingModule({
    declarations: [ ResetPasswordComponent ]
  })
  .compileComponents();
 });

 beforeEach(() => {
  fixture = TestBed.createComponent(ResetPasswordComponent);
  component = fixture.componentInstance;
  fixture.detectChanges();
 });

 it('should create', () => {
  expect(component).toBeTruthy();
 });
});
```

Resetpasswod.component.ts

```
import { Component, OnInit } from '@angular/core';
import { AuthService } from 'src/app/services/auth.service';

@Component({
 selector: 'app-reset-password',
 templateUrl: './reset-password.component.html',
 styleUrls: ['./reset-password.component.css']
})
export class ResetPasswordComponent implements OnInit {
```

```
  constructor(private authService:AuthService) { }


  ngOnInit(): void {
    this.authService.msg = "";
  }


}
```

User.cart.component.css

```
/* hello */


*{
  margin: 0;
  padding: 0;
  box-sizing: border-box;
}
body{
  /* background: #e7e7e7; */
  font-family: sans-serif;
}
.column {
  float: left;
  width: 100%;
  padding: 10px;
  height: 10%;


}
```

```
.parent{
  background: #fff;
  border-radius: 10px;
  box-shadow: 0px 10px 10px rgba(0, 0, 0,0.2) ;
  display: flex;
  max-width: 100%;
  overflow: hidden;
  width: 700px;
  margin-left: auto;
  margin-right: auto;
  position: relative;
  transition: 0.4s ease;
}

.parent:hover{
  transform: translateY(-10px);
  box-shadow: 0 0 20px 0 rgba(0,0,0,0.7);
}
.parent h6{
  margin-top:5px ;
  margin-bottom:5px ;
  opacity: 0.6;
  /* color: gray; */
  letter-spacing: 1px;
  text-transform: uppercase;
}
```

```
.parent h2{
  letter-spacing: 1px;
  margin:  10px 0;

}
.form-group{
  padding-top: 10px;
  display: flex;
}
input[type=number] {
  /* -moz-appearance: textfield; */
  width: 50px;
}
/* input::-webkit-outer-spin-button,
input::-webkit-inner-spin-button {
  -webkit-appearance: none;
  margin: 0;
}
*/

.preview{
  color: #fff;
  padding: 30px;
  width: 250px;
  position: relative;
}
```

```
.preview a{
  color: #fff;
  font-size: 12px;
  opacity: 0.6;
  margin-top: 30px;
  text-decoration: none;
}

.info{
  padding: 30px;
  position: relative;
  width: 100%;
}
.info p{
  font-size: 12px;
  font-weight: bold;
  text-transform: uppercase;
  opacity: 0.6;
}

.btnn{
  background: #e40046;
  border: 1px solid transparent;
  box-shadow: 0px 5px 10px rgba(0, 0, 0,0.6);
  color: #fff;
  font-size: 16px;
  padding: 10px 20px;
```

```
  position: absolute;
  bottom: 30px;
  right: 30px;
  letter-spacing: 1px;
  cursor: pointer;
  transition: all 0.4s ease-in-out;
  font-weight: bold;
  border-radius: 20px;
}
.it{
  padding: 10px 20px;
  position: absolute;
  top: 30px;
  right: 20px;
  letter-spacing: 1px;
}

.btnn:hover{
  background: #fff;
  color: #e40046;
  border: 1px solid #e40046;
  transform: scale(1.05);
}
.btnn:focus{
  outline: none;
}
```

```
@media(max-width:768px)
{
 .parent{
    flex-direction: column;
    width: 90%;
 }
 .preview
 {
    width: 100%;
    padding-top: 10px;
    padding-bottom: 0;
 }
 .preview h2{
    margin: 10px 0 0;
 }
 .preview a{
    margin-top: 10px;
 }
 .info h2{
    margin-top: 20px;
 }
 .info p{
    margin-bottom: 50px;
 }
```

```
 .btnn{
    padding: 10px 15px ;
    font-size: 14px;
 }
}



@media(max-width:320px)
{

 .preview
 {
    width: 100%;
    padding-top: 10px;
    padding-bottom: 0;
 }



 .info p{
    margin-bottom: 50px;
 }
 .btnn{
    padding: 10px 15px ;
    font-size: 14px;
    bottom: 37px;
 }
 #last{
```

```
    bottom: 20px;
  }

}



img.img-responsive {
  margin-left: auto;
  margin-right: auto;
  width: 125px;
  height: 125px;
}



#itemprice{

  padding-top: 10px;
  text-align: center;
  color: rgb(68, 52, 52);
  font-weight: 300;
  letter-spacing: 1px;

}

#lastcol{
  margin-bottom: 50px;
  box-shadow: none;
```

```
}


#lastcol p {
  color: black;
  font-size: medium;
}
```

User.cart.component.html

```
<!DOCTYPE html>
<html>


<head>
  <meta charset='utf-8'>
  <meta http-equiv='X-UA-Compatible' content='IE=edge'>
  <meta name='viewport' content='width=device-width, initial-scale=1'>
  <!-- <link rel='stylesheet' type='text/css' media='screen' href='uhome.css'> -->
  <!-- nav font -->
  <link href="https://fonts.googleapis.com/css?family=Audiowide&display=swap"
rel="stylesheet">









  <!-- Bootstrap CSS -->
  <link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/css/bootstrap.min.css"
    crossorigin="anonymous">
```

```
 <!-- fafa -->
 <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.7.0/css/font-awesome.min.css">



</head>

<body>
 <app-navbar></app-navbar>
 <!-- </div> -->

 <div class="container" id="container1">
  <div class="row">

   <div class="column" *ngFor="let recipe of recipes">
    <div class="parent">
     <div class="preview">
      <!-- <center> -->
      <img src="../../../assets/recipe/{{recipe.dishimage}}" alt="" class="img-responsive">
      <!-- </center> -->
     </div>

     <div class="info">

      <table>
       <tr>
```

```
 <td colspan="2">
  <h6>{{recipe.dishname}}</h6>
 </td>
</tr>
<tr>
 <td>
  <h6>Size:-</h6>
 </td>
 <td>
  <h6>{{recipe.quantity}}</h6>
 </td>
</tr>
<tr>
 <td>
  <h6>Price:-</h6>
 </td>

 <td>
  <h6><span class="product-price">{{recipe.price}}</span>RS</h6>
 </td>

</tr>



<tr>
 <td>
  <h6>Quantity:-</h6>
```

```
            </td>


            <td>

              <h6>{{recipe.qty}}</h6>

            </td>


           </tr>


          </table>


          <!-- <div class="product-quantity">

            <input type="number" id="qty_input4" class="ab form-control form-control-sm"  min="1"

              title="no of recipe"   (change)="update(recipe)">

          </div> -->

          <button class="btnn" (click)="deletefromcart(recipe)"> <i class="fa fa-trash" aria-
hidden="true"></i></button>

        </div>


      </div>

    </div>


      <!----------------------------------------------total section----------------------------------------------------
------  -->


    <div class="parent" style="background-color:rgb(235, 243, 251);margin-top: 15px;margin-
bottom: 150px;" id="lastcol">


      <div class="info">
```

```
        <p style="margin-top: 5px;margin-bottom: 5px;">total = <b><span
class="tot">{{total}}</span>Rs</b></p>

        <p style="margin-top: 5px;margin-bottom: 5px;">delivery charge = <b><span
class="dc">50</span>Rs</b></p>

        <p style="margin-top: 5px;margin-bottom: 5px;">grand total =<b><span
class="gt">{{total+50}}</span>Rs</b></p>

        <button class="btn btn-success" id="last" style="margin-top: 5px;margin-bottom: 5px;"
(click)="checkout()">Check out</button>

      </div>

     </div>




   </div>


  </div>




  <!-- hello -->

</body>


</html>
```

User.cart.component.spec.ts

```
import { ComponentFixture, TestBed } from '@angular/core/testing';


import { CartComponent } from './cart.component';


describe('CartComponent', () => {
```

```
  let component: CartComponent;
  let fixture: ComponentFixture<CartComponent>;


  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ CartComponent ]
    })
    .compileComponents();
  });


  beforeEach(() => {
    fixture = TestBed.createComponent(CartComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });


  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

User.cart.component.ts

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { AuthService } from 'src/app/services/auth.service';
import { HttpErrorResponse } from '@angular/common/http';
import { CartService } from 'src/app/services/cart.service';
@Component({
```

```
  selector: 'app-cart',
  templateUrl: './cart.component.html',
  styleUrls: ['./cart.component.css']
})
export class CartComponent implements OnInit {
  arr: any[];
  recipes: any[];
  total:any;
  emptychechk:boolean;
  paymentHandler:any = null;

  constructor(private router: Router, private authService: AuthService,private
cartService:CartService) { }

  ngOnInit(): void {
    this.invokeStripe();
    this.check()
    // this.getItem()
    this.empty()
  }

  check() {
    this.authService.check().subscribe(
      data => {
        console.log(data);
      },
      (error) => {
```

```
    if (error instanceof HttpErrorResponse) {

       this.router.navigate(['/login'])

    }
    console.log(error);
   }
  )
}



empty()
{
  this.cartService.EmptyCheck().subscribe(
    data => {
     // console.log(data);
     if(data['msg']=="yes empty cart")
     {
       // console.log("empty cart");
       this.router.navigate(['/empty-cart'])
       return;
     }
     else
     {
       this.getItem();
     }
```

```
    },
    (error) => {
      console.log(error);
    }
  )
}

getItem()
{
  this.authService.getCartItem().subscribe(
    data => {

      this.arr = data[0];

      this.total=this.arr['total'];

      this.recipes=this.arr['recipe']

      if(this.recipes.length==0)
      {
        this.router.navigate(['/empty-cart'])
        return;
      }

    },
    error => {
      console.log(error);
```

```
    }
  )
}



deletefromcart(recipe)
{

  this.cartService.deleteRecipe(recipe).subscribe(
    data => {
     // console.log(data);
     if(data['msg']=="recipe deleted from the cart")
     {
       // console.log("hello");
       this.authService.msg="recipe deleted from the cart";
       this.authService.avail=true;
       this.router.navigate(['/userhome'])
     }
    },
    error => {

     console.log(error);
    }
  )
}
checkout()
{
```

```
  this.initializePayment();


 }



 invokeStripe() {
  if(!window.document.getElementById('stripe-script')) {
   const script = window.document.createElement("script");
   script.id = "stripe-script";
   script.type = "text/javascript";
   script.src = "https://checkout.stripe.com/checkout.js";
   script.onload = () => {
    this.paymentHandler = (<any>window).StripeCheckout.configure({
      key:
'pk_test_51K5CjwSIYDqfPBTO6rpnautOVkXzl2zImxJCvCY26sdy7tLGl9D2KoBkPCrBWoTlhUvJCMPV
N5NPOed7AmNn6xH600FuDAlLxL',
      locale: 'auto',
      token: function (stripeToken: any) {
       console.log(stripeToken)
       alert('Payment has been successfull!');
      }


    });
   }
   window.document.body.appendChild(script);
  }
```

```
}

initializePayment() {

  const paymentHandler = (<any>window).StripeCheckout.configure({
    key: 'pk_test_sLUqHXtqXOkwSdPosC8ZikQ800snMatYMb',
    locale: 'auto',
    token: function (stripeToken: any) {
      console.log({stripeToken})
      this.router.navigate(['/cart'])
      alert('Paid amount sucessfully!');
      this.loadSuccessPage();
    }
  });

  paymentHandler.open({
    name: 'Paying amount',
    description: 'Paying payment for the order',
    amount: this.total * 100
  });

}

loadSuccessPage(){
  this.cartService.deletecart().subscribe(
    data => {
```

```
      if(data['msg']=="order placed")
      {

        this.authService.msg="sucessfully order placed!!!!";
        this.authService.avail=true;
        this.router.navigate(['/userhome'])
      }
    },
    error => {
      console.log(error);
    }
  )
 }



}
```

Changepassword.component.css

```
/* font */
.font-poppins {
  font-family: "Poppins", "Arial", "Helvetica Neue", sans-serif;
}



html {
  /* -webkit-box-sizing: border-box;
  -moz-box-sizing: border-box;
```

```
  box-sizing: border-box;
  width: 100%;
  height: 100%;
  background: linear-gradient(to top, #fbc2eb 0%, #a18cd1 100%);
  background: -webkit-gradient(linear, left bottom, left top, from(#fbc2eb), to(#a18cd1));
  background: -webkit-linear-gradient(bottom, #fbc2eb 0%, #a18cd1 100%);
  background: -moz-linear-gradient(bottom, #fbc2eb 0%, #a18cd1 100%);
  background: -o-linear-gradient(bottom, #fbc2eb 0%, #a18cd1 100%);




  background-attachment: fixed;
  background-repeat: no-repeat; */
  width:100%;
    height: 100vh;
    overflow-x: hidden;
    background-image:url("../../../assets/pic/71.jpg");
    background-repeat: no-repeat;
    background-size: 100% 100%;
    position: relative;
    background-attachment: fixed;

}


*, *:before, *:after {
  -webkit-box-sizing: inherit;
  -moz-box-sizing: inherit;
```

```
  box-sizing: inherit;
}

body{
  margin: 0;
  padding: 0;
}

/* fieldset {
  min-width: 0;

  border: 0;
}
 */
button {
  outline: none;
  background: none;
  border: none;
}

/* ========================================================================
   #PAGE WRAPPER
   ======================================================================== */
.page-wrapper {
  min-height: 100vh;
}
```

```
body {
  font-family: "Poppins", "Arial", "Helvetica Neue", sans-serif;
  font-weight: 400;
  font-size: 14px;
}




/* ========================================================================
   #SPACING
   ======================================================================== */



.p-t-180 {
  padding-top: 180px;
}
.p-t-10 {
  padding-top: 10px;
}
.p-b-100 {
  padding-bottom: 100px;
}


/* ========================================================================
   #WRAPPER
```

```
  ========================================================================== */

.wrapper {

 margin: 0 auto;

}



.wrapper--w780 {

 max-width: 780px;

}



/* ==========================================================================

  #BUTTON

  ========================================================================== */

.btn {

 display: inline-block;

 line-height: 40px;

 padding: 0 33px;

 font-family: Poppins, Arial, "Helvetica Neue", sans-serif;

 cursor: pointer;

 color: #fff;

 -webkit-transition: all 0.4s ease;

 -o-transition: all 0.4s ease;

 -moz-transition: all 0.4s ease;

 transition: all 0.4s ease;

 font-size: 18px;

}
```

```
.btn--radius {
  -webkit-border-radius: 3px;
  -moz-border-radius: 3px;
  border-radius: 3px;
}

.btn--pill {
  -webkit-border-radius: 20px;
  -moz-border-radius: 20px;
  border-radius: 20px;
}

.btn--green {
  background: #57b846;
}

.btn--green:hover {
  background: #318b21;
}

/* ========================================================================
   #FORM
   ======================================================================== */
input {
  outline: none;
  margin: 0;
```

```
  border: none;
  -webkit-box-shadow: none;
  -moz-box-shadow: none;
  box-shadow: none;
  width: 100%;
  background: transparent;
  font-size: 14px;
  font-family: inherit;
}

.input-group {
  position: relative;
  margin-bottom: 33px;
  background: transparent;
  border-bottom: 1px solid rgba(255, 255, 255, 0.2);
  display: flex;
}

.fa{
  padding-right: 10px;
  margin-bottom: 0;
  padding-top: 10px;
}




.input--style-3 {
```

```
  padding: 5px 0;
  font-size: 16px;
  color: #ccc;
  background: transparent;
}

.input--style-3::-webkit-input-placeholder {
  /* WebKit, Blink, Edge */
  color: #ccc;
}

.input--style-3:-moz-placeholder {
  /* Mozilla Firefox 4 to 18 */
  color: #ccc;
  opacity: 1;
}

.input--style-3::-moz-placeholder {
  /* Mozilla Firefox 19+ */
  color: #ccc;
  opacity: 1;
}

.input--style-3:-ms-input-placeholder {
  /* Internet Explorer 10-11 */
  color: #ccc;
}
```

```
.input--style-3:-ms-input-placeholder {
  /* Microsoft Edge */
  color: #ccc;
}

/* ============================================================================
   #TITLE
   ============================================================================ */
.title {
  font-size: 24px;
  color: #fff;
  font-weight: 400;
  margin-bottom: 36px;
}

/* ============================================================================
   #CARD
   ============================================================================ */
.card {
  overflow: hidden;
  -webkit-border-radius: 3px;
  -moz-border-radius: 3px;
  border-radius: 3px;
  background: #fff;
  /* animation: expand 1s ease forwards; */
}
```

```
.iforgot{
  animation: forgot 1s ease forwards;
}

.ichangepassword{
animation: changepassword 1s ease forwards;
}

.ilogin{
animation: login 1s ease forwards;
}

.iregister{
animation: register 1s ease forwards;
}

@keyframes login{

0%  { transform: translate(0%, -100%);}
100%{ transform: translate(0, 0);}

}
@keyframes register{
```

```
 0%  { transform: translate(0%,-100%);}

 100%{ transform: translate(0, 0);}


}




@keyframes forgot{


0%  { transform: translate(-100%, -100%);}

100%{ transform: translate(0, 0);}


}


@keyframes changepassword{


0%  { transform: translate(-100%, -100%);}

100%{ transform: translate(0, 0);}


}









.card-3 {
```

```
  opacity: 0.89;
  background: #000;
  -webkit-border-radius: 10px;
  -moz-border-radius: 10px;
  border-radius: 10px;
  -webkit-box-shadow: 0px 8px 20px 0px rgba(0, 0, 0, 0.15);
  -moz-box-shadow: 0px 8px 20px 0px rgba(0, 0, 0, 0.15);
  box-shadow: 0px 8px 20px 0px rgba(0, 0, 0, 0.15);
  width: 100%;
  display: table;
}
.card-3 .card-body {
  padding: 57px 65px;
  padding-bottom: 65px;
  display: table-cell;
}

/* .container {

  animation: expand 1s ease forwards;



} */



.niche{
padding-top: 10px;
```

```
padding-left: 10px;
font-family: "Poppins", "Arial", "Helvetica Neue", sans-serif;
}
a{
text-decoration: none;
color: steelblue;
padding-left: 10px;
}

#message {
  font-family: "Poppins", "Arial", "Helvetica Neue", sans-serif;
  text-align: center;
  align-items: center;
  display:none;
  background:transparent;
  color: #000;
  position: relative;
  padding: 20px;
  margin-top: 10px;

}

#message p {
  padding: 10px 35px;
  font-size: 18px;
```

```
}

/* Add a green text color and a checkmark when the requirements are right */
.valid {
  color:#57b846;
}

.valid:before {
  position: relative;
  left: -35px;
  content: " ✓";
}

/* Add a red text color and an "x" icon when the requirements are wrong */
.invalid {
  color: rgb(238, 36, 36);
}

.invalid:before {
  position: relative;
  left: -35px;
  content: " ✗";
}
```

Changepassword.component.html

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>

 <link


href="https://fonts.googleapis.com/css?family=Poppins:100,100i,200,200i,300,300i,400,400i,500,
500i,600,600i,700,700i,800,800i,900,900i"

  rel="stylesheet">


 <!-- fafa -->

 <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/4.7.0/css/font-awesome.min.css">


</head>


<body>


 <div class="container">

  <div class="page-wrapper bg-gra-01 p-t-180 p-b-100 font-poppins">

   <div class="wrapper wrapper--w780">

    <div class="card card-3 iregister">

     <div class="card-body">

      <h2 class="title">Change Password </h2>

      <div style="color: red;" *ngIf="avail">

       <h3> {{msg}}</h3>

      </div>

      <form #f="ngForm" (ngSubmit)="onSubmit(f)">

       <div class="input-group">

        <input class="input--style-3" type="email" placeholder="Email" name="email"
autocomplete="off" ngModel
```

```
            email required #email="ngModel" pattern="[a-z0-9._%+-]+@[a-z0-9.-]+\.[a-z]{2,}$">

          </div>

          <div class="input-group">

            <input class="input--style-3" type="password" placeholder="Enter old password"
name="op" id="op"

              autocomplete="off" pattern="(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{8,}"

              title="Plaese enter valid and strong password" ngModel required #op="ngModel">

          </div>



          <div class="input-group">

            <input class="input--style-3" type="password" placeholder="Enter new password"
name="p1" id="p1"

              autocomplete="off" pattern="(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{8,}"

              title="Plaese enter valid and strong password" ngModel required #p1="ngModel">

          </div>



          <div class="input-group">

            <input class="input--style-3" type="password" placeholder="Re-enter new password"
name="p2" id="p2"

              pattern="(?=.*\d)(?=.*[a-z])(?=.*[A-Z]).{8,}" title="Plaese enter valid and strong
password"

              autocomplete="off" ngModel required #p2="ngModel">

          </div>

          <div class="p-t-10">

            <!-- <center> -->



            <button class="btn btn--pill btn--green" [disabled]="f.invalid"
type="submit">ChangePasswod</button>

            <button class="btn btn--pill btn--green" style="margin-left: 10px;"
```

```
type="reset">Reset</button>
            <!-- </center> -->
          </div>

        </form>
        <div class="niche" style="margin-top:10px;">
          <!-- <center> -->
          <a [routerLink]="['/login']" id="fp">Login</a>

          <a [routerLink]="['/']"><i class="fa fa-arrow-circle-left" aria-hidden="true"
             style="color:white;">Goback</i></a>
          <!-- </center> -->
        </div>
       </div>
      </div>
     </div>
   </div>
 </div>

</body>

</html>
```

Changepassword.component.spec.ts

```
import { ComponentFixture, TestBed } from '@angular/core/testing';

import { ChangepasswordComponent } from './changepassword.component';
```

```
describe('ChangepasswordComponent', () => {
  let component: ChangepasswordComponent;
  let fixture: ComponentFixture<ChangepasswordComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [ ChangepasswordComponent ]
    })
    .compileComponents();
  });

  beforeEach(() => {
    fixture = TestBed.createComponent(ChangepasswordComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

Changepassword.componen.ts

```
import { HttpErrorResponse } from '@angular/common/http';
import { Component, OnInit } from '@angular/core';
import { NgForm } from '@angular/forms';
import { Router } from '@angular/router';
import { AuthService } from 'src/app/services/auth.service';
```

```
@Component({
  selector: 'app-changepassword',
  templateUrl: './changepassword.component.html',
  styleUrls: ['./changepassword.component.css']
})
export class ChangepasswordComponent implements OnInit {
  msg: any = [];
  avail: boolean;
  arr: any[];
  isValid: boolean = false;
  constructor(private router: Router, private authService: AuthService) { }

  ngOnInit(): void {
    this.readUser()
    // first we logout current user
    localStorage.removeItem('token')
    localStorage.removeItem('userid')
    localStorage.removeItem('N1@#I2@#M3@#D4@#A6');
  }



  readUser() {
    this.authService.readUser().subscribe(
      data => {
        this.arr = data['msg'];
```

```
    },
    error => {
      console.log(error);
    }
  )
}
onSubmit(f: NgForm) {

  for (var val of this.arr) {
    var a = val['email'];
    var b = f.controls.email.value;
    if (a == b) {
      // console.log("yes same");
      this.isValid = true;
    }
  }
  if (this.isValid) {
    console.log("email exist");
  }
  else {
    // console.log("email does not exist");
    this.msg = "Email does not exist!!";
    this.avail = true;
    return;
  }
```

```
  if (f.controls.p1.value != f.controls.p2.value) {
    this.msg = "Password   doesn't match";
    this.avail = true;
    return;
  }

  this.authService.changepassword(JSON.stringify(f.value))
    .subscribe(
      data => {
        // console.log(data);
        if(data['error'])
        {
          console.log("in if err");
          console.log(data['error']);
          this.msg = data['error']
          this.avail=true;
          return;
        }
        else
        {
          this.authService.msg="successfully changed password!!";
          this.router.navigate(['/login']);
        }
      },
      error => { console.error(error); this.msg = error; }
    )
  }
```

```
}
```

Backend:-Server.js to connect ot mongodb

```
require('dotenv').config()
const express = require('express')
const mongoose = require('mongoose')
const url = 'mongodb://localhost/opzdb'
var bodyParser = require('body-parser')
var cors = require('cors')
const app = express()
var appRoutes = require("./routes/appRoutes")
var adminRoutes = require('./routes/adminRoute')

// process.env.PORT hoy to e else 3000
const PORT = process.env.PORT || 3000

// some useful library
app.use(bodyParser.urlencoded({ extended: true }))
app.use(bodyParser.json())
app.use(cors())



// data base connection
mongoose.Promise=global.Promise
mongoose.connect(url, { useNewUrlParser: true ,useUnifiedTopology: true})
const con = mongoose.connection
```

```
con.on('open', () => {
   console.log("database connected");
})



// for testing purposr
app.get('/', (req, res) => {
   res.send("Hello UtSwathi Kanduri from server")
})



// route configure
app.use('/', appRoutes)
app.use('/admin', adminRoutes)



// print console message
app.listen(PORT, () => {
   console.log(`Listing onnn port ${PORT}`);
})
```

Cart.js

```
var mongoose = require('mongoose')
var cartSchema = mongoose.Schema({
   whichuser: {
      type: String,
   },
   recipe: {
      type  :Array,
      default:[]
   },
   total:{
      type:Number,
      default:0
   },
   createdAt: {type: Date, default: Date.now}
})
module.exports = mongoose.model('cart',cartSchema)
```

feedback.js

```
var mongoose = require('mongoose')
var feedbackSchema = mongoose.Schema({
   whichuser: {
      type: String,
   },
   email: {
      type: String,
   },
```

```
    name: {
        type: String,

    },
    msg: {
        type: String,
    },
    createdAt: {type: Date, default: Date.now}
})
module.exports = mongoose.model('feedback',feedbackSchema)
```

order.js

```
var mongoose = require('mongoose')
var orderSchema = mongoose.Schema({
    whichuser: {
        type: String,
    },
            recipe: {
        type  :Array,
        default:[]
    },
    total:{
        type:Number,
    },
    createdAt: {type: Date, default: Date.now}
})
```

```
module.exports = mongoose.model('order',orderSchema)
```

otp.js

```
var mongoose = require('mongoose')
var otpSchema = mongoose.Schema({
   otp: {
      type: Number,
      required: true
   },
   email: {
      type: String,
      required: true
   },
   createdAt: {type: Date, default: Date.now}
})
module.exports = mongoose.model('otp',otpSchema)
```

recipe.js

```
var mongoose = require('mongoose')
var recipeSchema = mongoose.Schema({

   dishname: {
      type: String,

   },
```

```
    quantity: {
        type: String,

    },
    price: {
        type: Number,

    },
    dishimage: {
        type: String,
    },
    createdAt: {type: Date, default: Date.now}
})
module.exports = mongoose.model('recipe',recipeSchema)
```

user.js

```
var mongoose = require('mongoose')
var bcrypt = require('bcrypt');
var userSchema = mongoose.Schema({
    name: {
        type: String,
        require: true
    },
    contact: {
        type: Number,
```

```
    required: true,
  },
  email: {
    type: String,
    required: true,
    unique: true
  },
  password: {
    type: String,
    required: true,
  },
  role: {
    type: String, default: 'customer'
  },
  blocked: {
    type: Boolean, default: false
  }
}, { timestamp: true }
)

userSchema.statics.hashPassword = function hashPassword(password){
  return bcrypt.hashSync(password,10);
}

userSchema.methods.isValid = function(hashedpassword){
  return  bcrypt.compareSync(hashedpassword, this.password);
}
```

```
module.exports = mongoose.model('user',userSchema)
```