SRI KRISHNA COLLEGE OF TECHNOLOGY
An Autonomous Institution | Accredited by NAAC with 'A' Grade
Affiliated to Anna University | Approved by AICTE
KOVAIPUDUR, COIMBATORE 641042

HOUSE BOAT VOYAGE

ADVANCED APPLICATION DEVELOPMENT

A PROJECT REPORT

Submitted by

KABILA B S (RegisterNo:727821TUIT045)
HEMALATHAA K (Register No:727821TUIT036)
MAHESH BOOPATHI A P (Register No:727821TUIT055)

in partial fulfilment for the award of the degree

Of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

APRIL - 2024

HOUSE BOAT VOYAGE

INTRODUCTION:

Planning a House Boat Voyage is an art that demands meticulous organization and creativity to craft an unforgettable experience. As a House Boat Voyage Organizer, the primary objective is to realize the vision of the individuals embarking on this special journey, whether it's a romantic getaway or a group adventure.The voyage commences with understanding the travelers' preferences, from selecting the perfect houseboat and route to arranging gourmet meals and onboard entertainment. Clear communication is vital throughout the planning process, ensuring that every detail aligns with the travelers' desires, transforming their dream voyage into a reality.

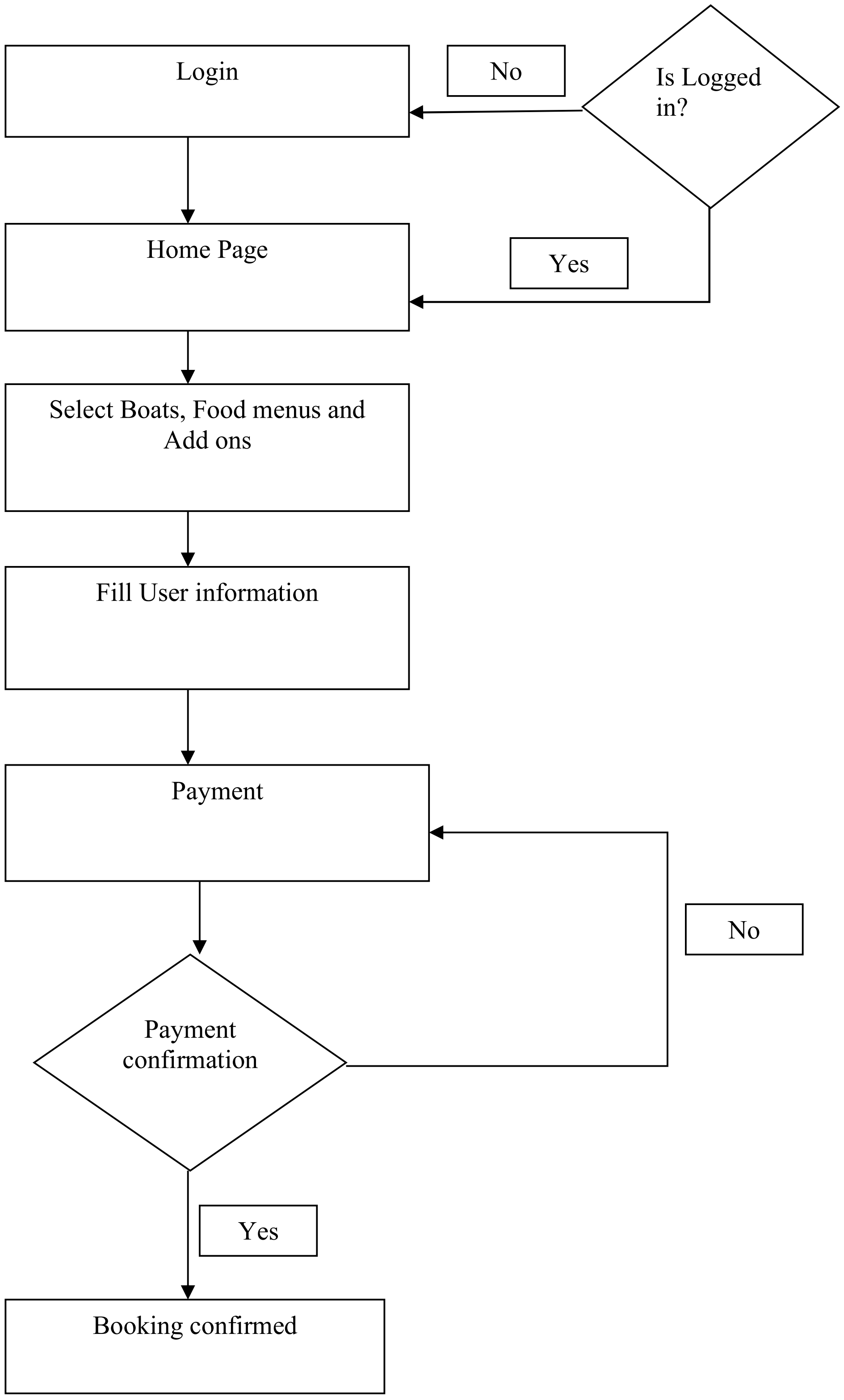
Flexibility is paramount in the realm of voyage planning. Despite careful planning, unforeseen circumstances may arise, necessitating swift thinking and adaptability to guarantee smooth sailing. By remaining composed under pressure and making quick decisions, you can overcome any challenges that may arise, ensuring that the houseboat voyage is an unparalleled success.Ultimately, being a House Boat Voyage Organizer transcends mere logistics—it's about curating extraordinary experiences that will be cherished for a lifetime.

OBJECTIVE:

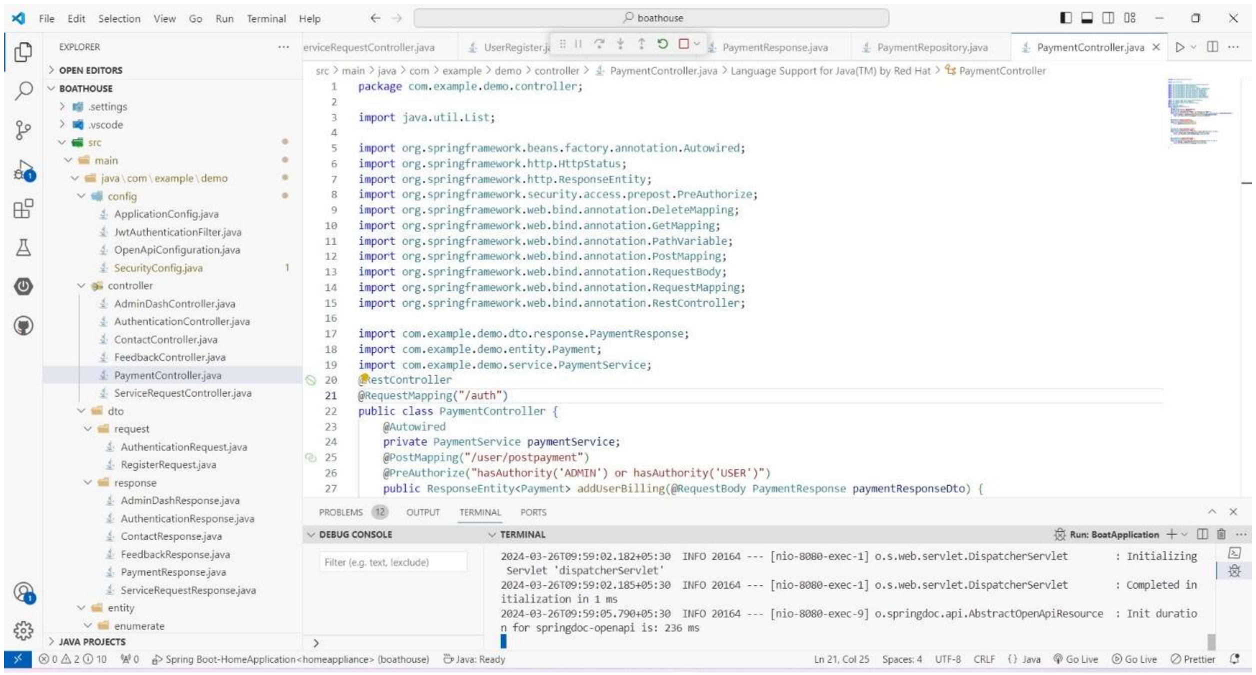
The objective of a house boat voyage is multifaceted, encompassing elements of exploration, relaxation, and connection. This unique experience provides an opportunity for travelers to come together, relishing in the serene surroundings of nature while connecting with each other and the journey itself. It's a time to appreciate not just the voyage but also the shared moments, conversations, and connections that enrich the travelers' lives.

Beyond mere sightseeing, a house boat voyage seeks to foster deeper connections among passengers, creating memorable experiences and strengthening bonds. It serves as a reminder of the beauty of exploration and the importance of human connection in experiencing life's wonders. The voyage encourages travelers to embrace the present moment, indulge in the tranquility of their surroundings, and forge lasting memories that will be cherished for years to come.

USER FLOW CHART:

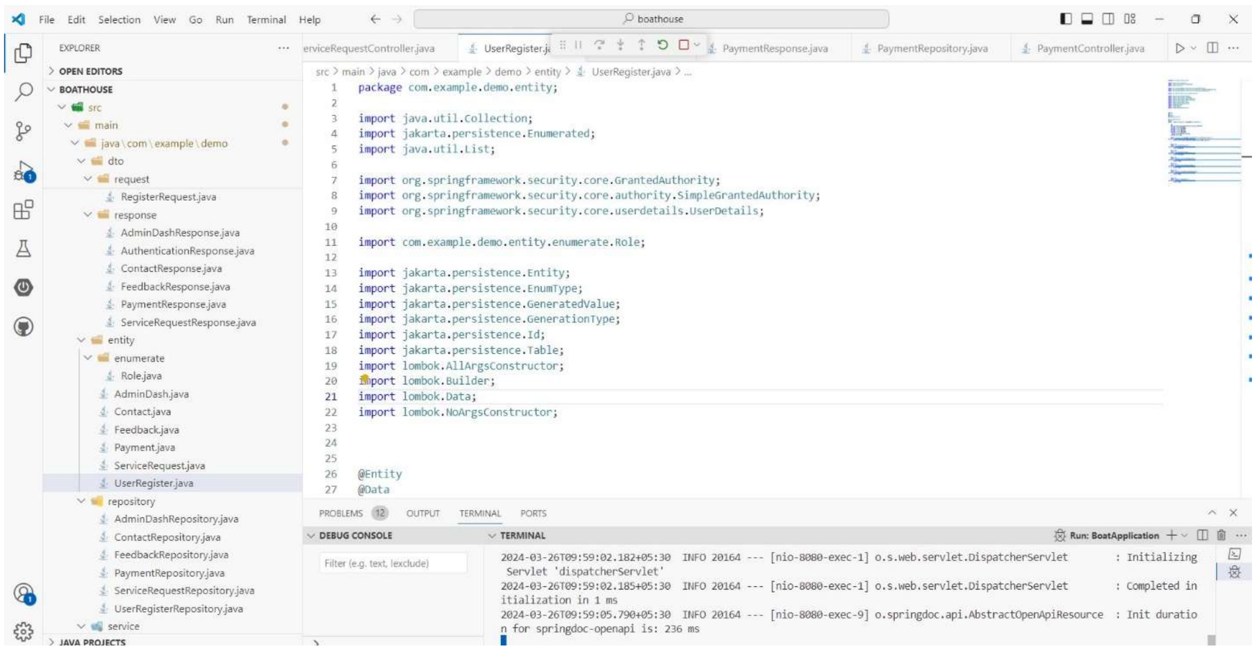


BACKEND:


File Edit Selection View Go Run Terminal Help
boathouse
EXPLORER
OPEN EDITORS
BOATHOUSE
.settings
.vscode
src
main
java \ com \ example \ demo
config
ApplicationConfig.java
JwtAuthenticationFilter.java
OpenApiConfiguration.java
SecurityConfig.java
controller
AdminDashController.java
AuthenticationController.java
ContactController.java
FeedbackController.java
PaymentController.java
ServiceRequestController.java
dto
request
AuthenticationRequest.java
RegisterRequest.java
response
AdminDashResponse.java
AuthenticationResponse.java
ContactResponse.java
FeedbackResponse.java
PaymentResponse.java
ServiceRequestResponse.java
entity
enumerate
JAVA PROJECTS
erviceRequestController.java
UserRegister.java
PaymentResponse.java
PaymentRepository.java
PaymentController.java
src > main > java > com > example > demo > controller > PaymentController.java > Language Support for Java(TM) by Red Hat > PaymentController
PROBLEMS 12 OUTPUT TERMINAL PORTS
DEBUG CONSOLE
TERMINAL
Run: BoatApplication
Filter (e.g. text, !exclude)
Spring Boot-HomeApplication<homeappliance> (boathouse)
Java: Ready
Ln 21, Col 25 Spaces: 4 UTF-8 CRLF {} Java Go Live Go Live Prettier

```java
package com.example.demo.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.security.access.prepost.PreAuthorize;
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.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.example.demo.dto.response.PaymentResponse;
import com.example.demo.entity.Payment;
import com.example.demo.service.PaymentService;
@RestController
@RequestMapping("/auth")
public class PaymentController {
    @Autowired
    private PaymentService paymentService;
    @PostMapping("/user/postpayment")
    @PreAuthorize("hasAuthority('ADMIN') or hasAuthority('USER')")
    public ResponseEntity<Payment> addUserBilling(@RequestBody PaymentResponse paymentResponseDto) {
```

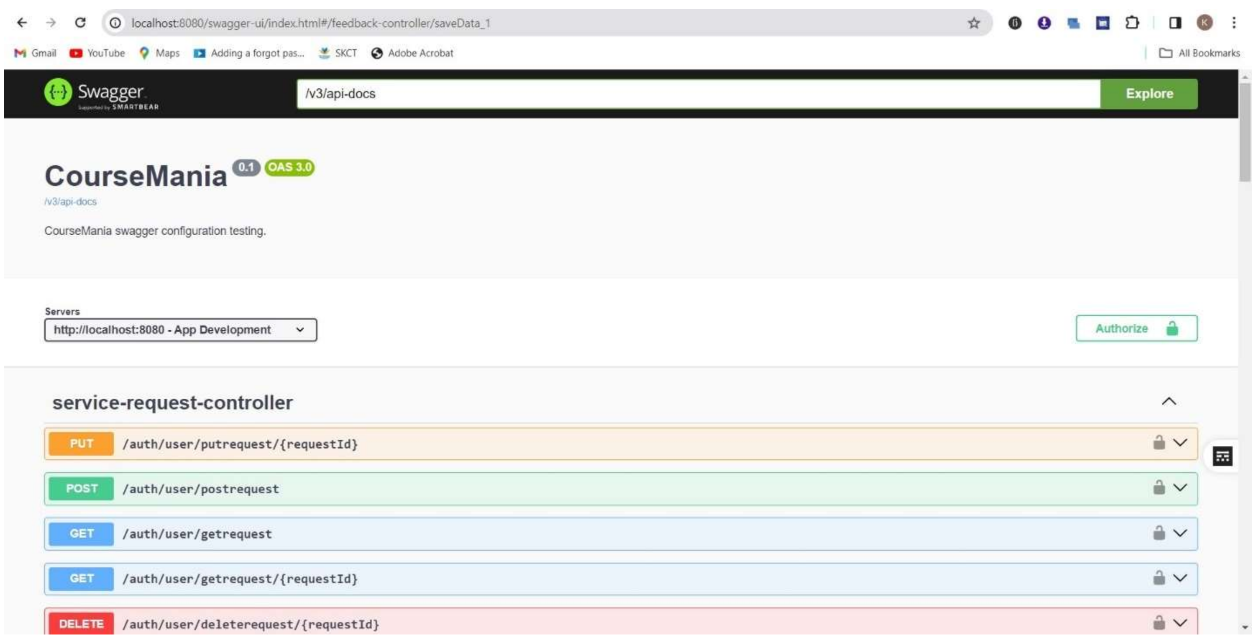
```
2024-03-26T09:59:02.182+05:30  INFO 20164 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing
 Servlet 'dispatcherServlet'
2024-03-26T09:59:02.185+05:30  INFO 20164 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Completed in
itialization in 1 ms
2024-03-26T09:59:05.790+05:30  INFO 20164 --- [nio-8080-exec-9] o.springdoc.api.AbstractOpenApiResource  : Init duratio
n for springdoc-openapi is: 236 ms
```



```java
package com.example.demo.entity;

import java.util.Collection;
import jakarta.persistence.Enumerated;
import java.util.List;

import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;

import com.example.demo.entity.enumerate.Role;

import jakarta.persistence.Entity;
import jakarta.persistence.EnumType;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.Table;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;



@Entity
@Data
```

```
2024-03-26T09:59:02.182+05:30  INFO 20164 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing
 Servlet 'dispatcherServlet'
2024-03-26T09:59:02.185+05:30  INFO 20164 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Completed in
itialization in 1 ms
2024-03-26T09:59:05.790+05:30  INFO 20164 --- [nio-8080-exec-9] o.springdoc.api.AbstractOpenApiResource  : Init duratio
n for springdoc-openapi is: 236 ms
```

SWAGGER UI:


localhost:8080/swagger-ui/index.html#/feedback-controller/saveData_1

/v3/api-docs

CourseMania 0.1 OAS 3.0

/v3/api-docs

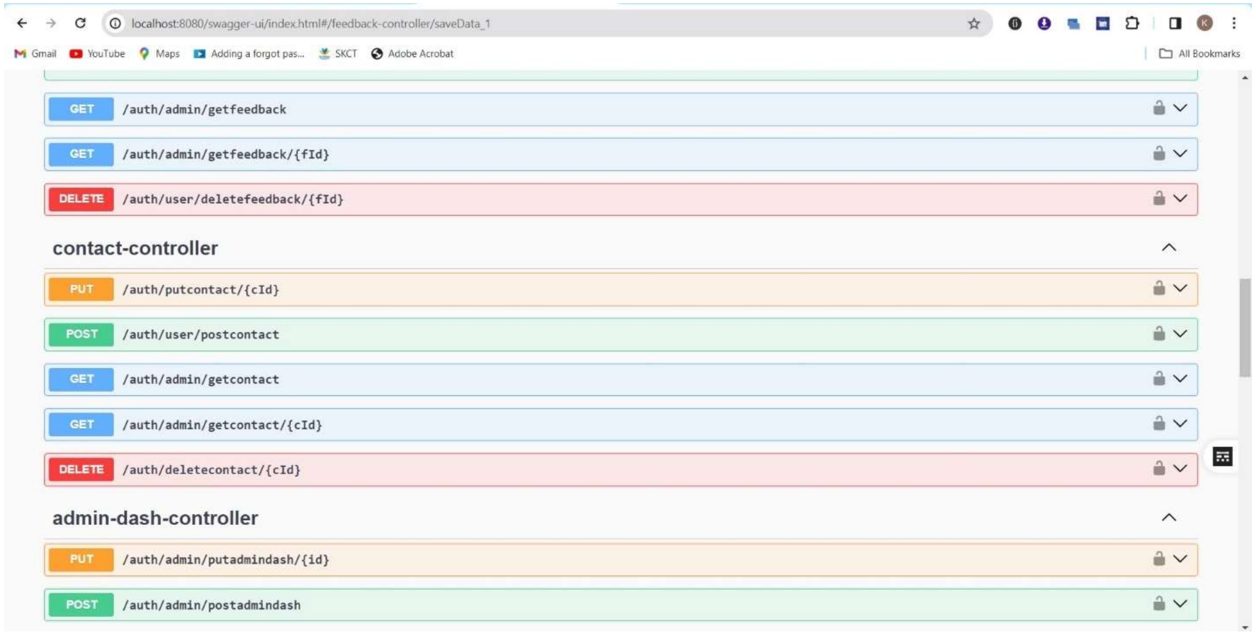
CourseMania swagger configuration testing.

Servers

http://localhost:8080 - App Development

service-request-controller

```
PUT     /auth/user/putrequest/{requestId}
POST    /auth/user/postrequest
GET     /auth/user/getrequest
GET     /auth/user/getrequest/{requestId}
DELETE  /auth/user/deleterequest/{requestId}
```


localhost:8080/swagger-ui/index.html#/feedback-controller/saveData_1

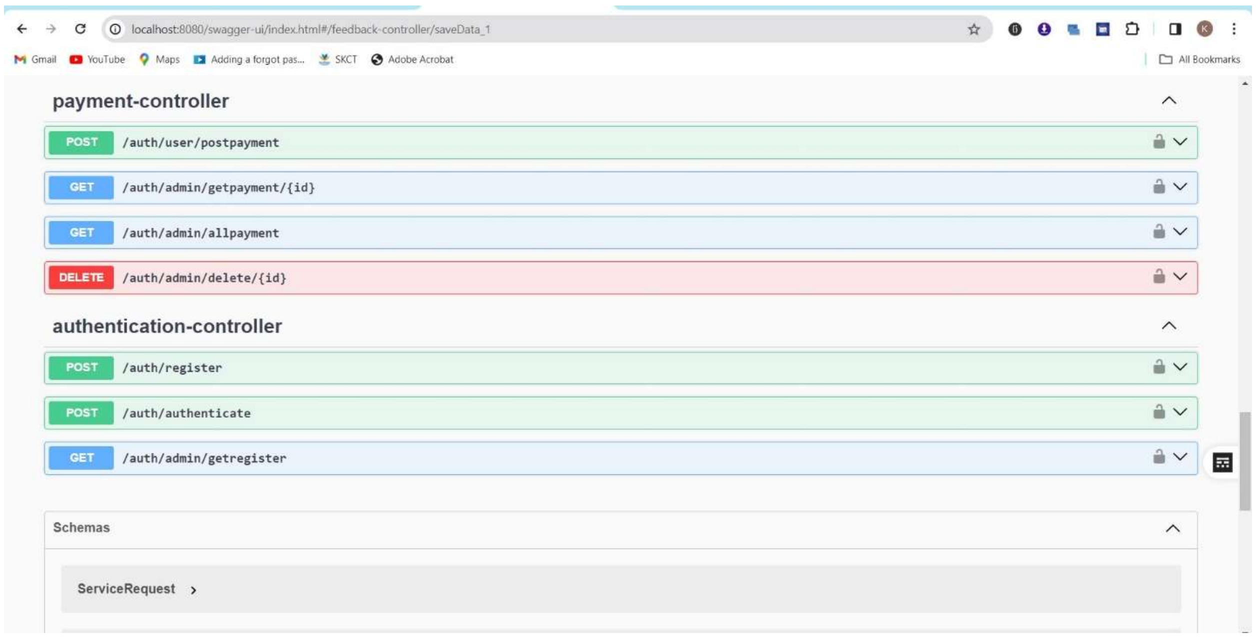
```
GET     /auth/admin/getfeedback
GET     /auth/admin/getfeedback/{fId}
DELETE  /auth/user/deletefeedback/{fId}
```

contact-controller

```
PUT     /auth/putcontact/{cId}
POST    /auth/user/postcontact
GET     /auth/admin/getcontact
GET     /auth/admin/getcontact/{cId}
DELETE  /auth/deletecontact/{cId}
```

admin-dash-controller

```
PUT     /auth/admin/putadmindash/{id}
POST    /auth/admin/postadmindash
```


localhost:8080/swagger-ui/index.html#/feedback-controller/saveData_1

payment-controller

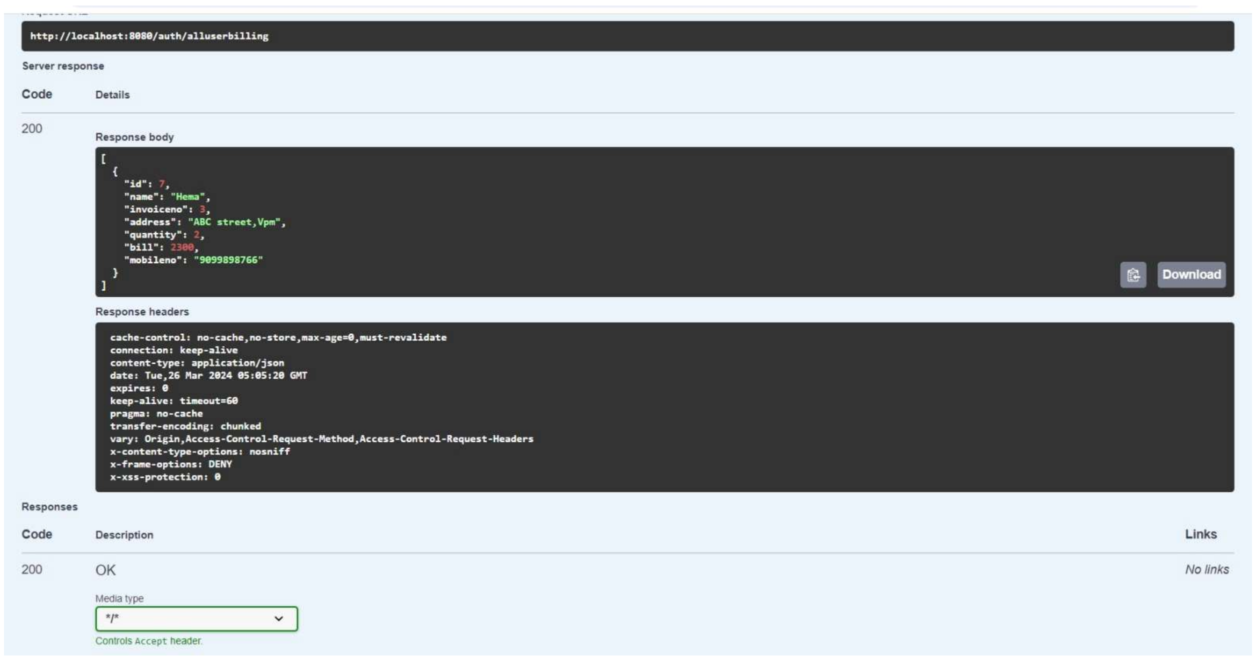
POST	/auth/user/postpayment
GET	/auth/admin/getpayment/{id}
GET	/auth/admin/allpayment
DELETE	/auth/admin/delete/{id}

authentication-controller

POST	/auth/register
POST	/auth/authenticate
GET	/auth/admin/getregister

Schemas

ServiceRequest


```
http://localhost:8080/auth/alluserbilling
```

Server response

Code | Details

200

Response body

```
[
  {
    "id": 7,
    "name": "Hema",
    "invoiceno": 3,
    "address": "ABC street,Vpm",
    "quantity": 2,
    "bill": 2300,
    "mobileno": "9099898766"
  }
]
```

Response headers

```
 cache-control: no-cache,no-store,max-age=0,must-revalidate
 connection: keep-alive
 content-type: application/json
 date: Tue,26 Mar 2024 05:05:20 GMT
 expires: 0
 keep-alive: timeout=60
 pragma: no-cache
 transfer-encoding: chunked
 vary: Origin,Access-Control-Request-Method,Access-Control-Request-Headers
 x-content-type-options: nosniff
 x-frame-options: DENY
 x-xss-protection: 0
```

Responses

Code	Description	Links
200	OK	No links

Media type: */*

Controls Accept header.

SECURITY CONFIGURATION:

1)Security.config

```
package com.example.demo.config;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.authentication.AuthenticationProvider;
import
org.springframework.security.config.annotation.method.configuration.EnableMethodS
ecurity;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecurit
y;
import
org.springframework.security.config.annotation.web.configurers.AbstractHttpConfig
urer;
import org.springframework.security.config.http.SessionCreationPolicy;
import org.springframework.security.web.SecurityFilterChain;
import
org.springframework.security.web.authentication.UsernamePasswordAuthenticationFil
ter;

import lombok.RequiredArgsConstructor;

@Configuration
@EnableWebSecurity
@EnableMethodSecurity
@RequiredArgsConstructor
public class SecurityConfig {
    @Autowired
    private final   JwtAuthenticationFilter jwtAuthFilter ;
    private final AuthenticationProvider authenticationProvider ;

    @SuppressWarnings("removal")
    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws
Exception {
        return http.csrf(AbstractHttpConfigurer::disable)
                .authorizeHttpRequests(requests -> requests
                        .requestMatchers("/auth/**","/auth/user/**", "/v3/api-
docs/**",
                                "/swagger-ui.html", "/swagger-ui/**")
                        .permitAll())
                .authorizeHttpRequests(requests ->
requests.requestMatchers("/auth/admin/**")
                        .authenticated())
                .sessionManagement(management -> management
                        .sessionCreationPolicy(SessionCreationPolicy.STATELESS))
                .authenticationProvider(authenticationProvider)
                .addFilterBefore(jwtAuthFilter,
UsernamePasswordAuthenticationFilter.class)
                .build();
    }



}
```

2)OpenApiConfiguration:

```
package com.example.demo.config;

import java.util.Arrays;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

import io.swagger.v3.oas.models.Components;
import io.swagger.v3.oas.models.OpenAPI;
import io.swagger.v3.oas.models.info.Info;
import io.swagger.v3.oas.models.servers.Server;
import io.swagger.v3.oas.models.security.SecurityRequirement;
import io.swagger.v3.oas.models.security.SecurityScheme;

@Configuration
public class OpenApiConfiguration {
@Autowired
      @Bean
        public OpenAPI defineOpenApi() {
                Server server = new Server();
                server.setUrl("http://localhost:8080");
                server.setDescription("App Development");

                Info info = new Info()
```

```
                                .title("CourseMania")
                                .version("0.1")
                                .description("CourseMania swagger configuration
testing.");

                // Define JWT security scheme
                SecurityScheme securityScheme = new SecurityScheme()
                                .type(SecurityScheme.Type.HTTP)
                                .scheme("bearer")
                                .bearerFormat("JWT");

                // Add the JWT security scheme to the components
                Components components = new
Components().addSecuritySchemes("bearerAuth", securityScheme);

                // Add security requirement with JWT authentication
                SecurityRequirement securityRequirement = new
SecurityRequirement().addList("bearerAuth");

                // Create the OpenAPI object with components and security
requirements
                return new OpenAPI()
                                .info(info)
                                .servers(Arrays.asList(server))
                                .components(components)
                                .addSecurityItem(securityRequirement);
      }

}
```

3)JwtAuthenticationFilter:

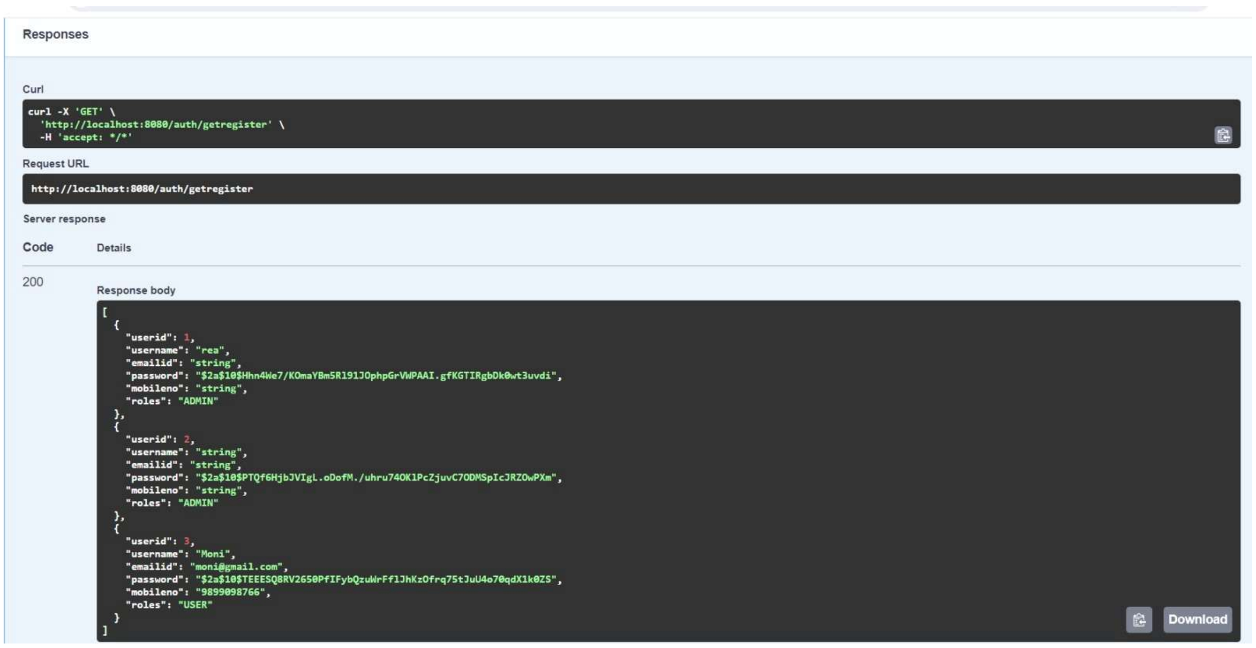
```
package com.example.demo.config;

import java.io.IOException;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.lang.NonNull;
import
org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
```

```
import
org.springframework.security.web.authentication.WebAuthenticationDetailsSource;
import org.springframework.stereotype.Component;
import org.springframework.web.filter.OncePerRequestFilter;

import com.example.demo.service.JwtService;

import jakarta.servlet.FilterChain;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import lombok.RequiredArgsConstructor;

@Component
@RequiredArgsConstructor
public class JwtAuthenticationFilter extends OncePerRequestFilter{
    @Autowired
    private final JwtService jwtservice ;
    private final UserDetailsService userDetailsService;

    @Override
    protected void doFilterInternal(
             @NonNull HttpServletRequest request,
             @NonNull HttpServletResponse response,
             @NonNull FilterChain filterChain)
            throws ServletException, IOException {
         if (request.getServletPath().contains("/api/v1/auth")) {
              filterChain.doFilter(request, response);
              return;}
         final String authHeader = request.getHeader("Authorization");
            final String token;
            final String username;
            if (authHeader == null || !authHeader.startsWith("Bearer ")) {
                filterChain.doFilter(request, response);
                return;
            }
        token = authHeader.substring(7);
        username = jwtservice.extractUsername(token);
         if (username != null &&
SecurityContextHolder.getContext().getAuthentication() == null) {
                UserDetails userDetails =
this.userDetailsService.loadUserByUsername(username);
```

```
                if (jwtservice.isTokenValid(token, userDetails)) {
                    UsernamePasswordAuthenticationToken
usernamePasswordAuthenticationToken = new UsernamePasswordAuthenticationToken(
                            userDetails, null, userDetails.getAuthorities());
                    usernamePasswordAuthenticationToken
                            .setDetails(new
WebAuthenticationDetailsSource().buildDetails(request));

SecurityContextHolder.getContext().setAuthentication(usernamePasswordAuthenticati
onToken);
                }
                filterChain.doFilter(request, response);
        }
    }

}
```

4)ApplicationConfig:

```
package com.example.demo.config;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.AuthenticationProvider;
import org.springframework.security.authentication.dao.DaoAuthenticationProvider;
import
org.springframework.security.config.annotation.authentication.configuration.Authe
nticationConfiguration;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;

import com.example.demo.repository.UserRegisterRepository;

import lombok.RequiredArgsConstructor;

@Configuration
@RequiredArgsConstructor
public class ApplicationConfig {
    @Autowired
    private final UserRegisterRepository userRepository;
    @Bean
    public UserDetailsService userDetailService()
    {
        return username -> userRepository.findByUsername(username)
                .orElseThrow(() -> new UsernameNotFoundException("User not 
found!"));
    }

    @Bean
    public AuthenticationProvider authenticationProvider() {
        DaoAuthenticationProvider authProvider = new DaoAuthenticationProvider();
        authProvider.setUserDetailsService(userDetailService());
        authProvider.setPasswordEncoder(passwordEncoder());
        return authProvider;
    }

      @Bean
        public AuthenticationManager 
authenticationManager(AuthenticationConfiguration authenticationConfiguration)
                throws Exception {
            return authenticationConfiguration.getAuthenticationManager();
        }
    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }
}
```


Responses

Curl

```
curl -X 'GET' \
  'http://localhost:8080/auth/getregister' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:8080/auth/getregister
```

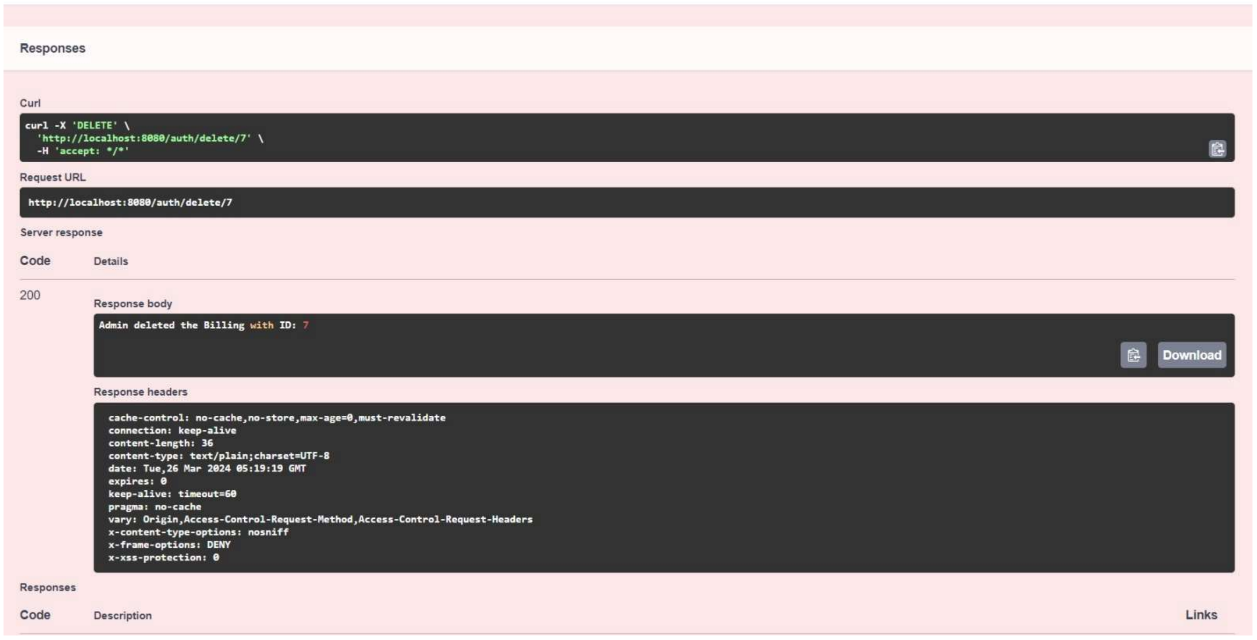
Server response

Code Details

200

Response body

```
[
  {
    "userid": 1,
    "username": "rea",
    "emailid": "string",
    "password": "$2a$10$Hhn4We7/KOmaYBm5R191JOphpGrVWPAAI.gfKGTIRgbDk0wt3uvdi",
    "mobileno": "string",
    "roles": "ADMIN"
  },
  {
    "userid": 2,
    "username": "string",
    "emailid": "string",
    "password": "$2a$10$PTQf6HjbJVIgL.oDofM./uhru74OK1PcZjuvC7ODMSpIcJRZOwPXm",
    "mobileno": "string",
    "roles": "ADMIN"
  },
  {
    "userid": 3,
    "username": "Moni",
    "emailid": "moni@gmail.com",
    "password": "$2a$10$TEEESQ8RV2650PfIFybQzuWrFflJhKzOfrq75tJuU4o70qdX1k0ZS",
    "mobileno": "9899098766",
    "roles": "USER"
  }
]
```


Responses

Curl

```
curl -X 'DELETE' \
  'http://localhost:8080/auth/delete/7' \
  -H 'accept: */*'
```

Request URL

```
http://localhost:8080/auth/delete/7
```

Server response

Code Details

200

Response body

```
Admin deleted the Billing with ID: 7
```

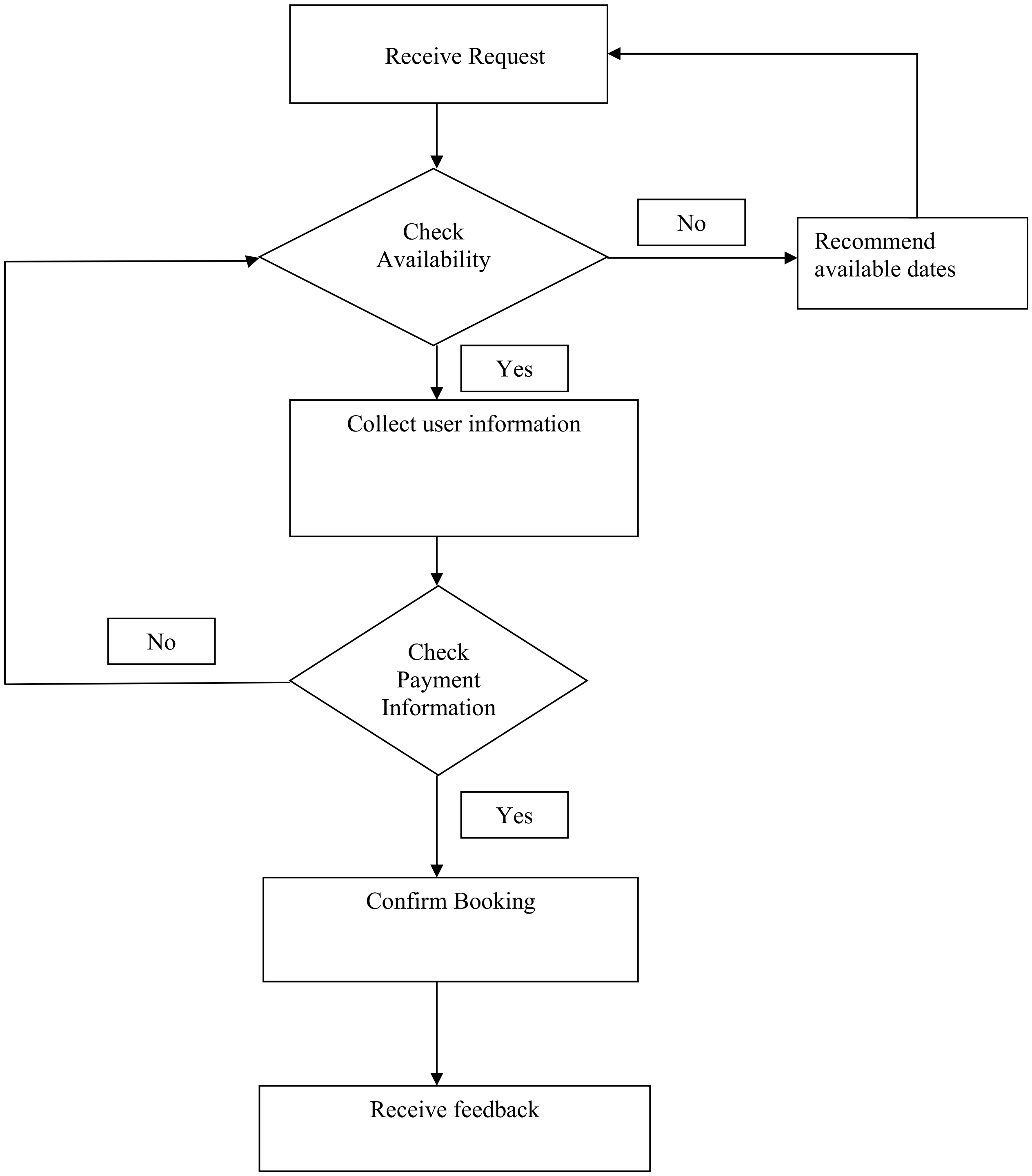
Response headers

```
cache-control: no-cache,no-store,max-age=0,must-revalidate
connection: keep-alive
content-length: 36
content-type: text/plain;charset=UTF-8
date: Tue,26 Mar 2024 05:19:19 GMT
expires: 0
keep-alive: timeout=60
pragma: no-cache
vary: Origin,Access-Control-Request-Method,Access-Control-Request-Headers
x-content-type-options: nosniff
x-frame-options: DENY
x-xss-protection: 0
```

Responses

Code Description Links

ADMIN CHART:


Receive Request
Check Availability
No
Recommend available dates
Yes
Collect user information
No
Check Payment Information
Yes
Confirm Booking
Receive feedback

CONCLUSION:

In conclusion, the development of an online platform dedicated to planning and organizing house boat voyages offers a transformative solution to the challenges faced by travelers seeking unique and memorable experiences. By harnessing technology to simplify the voyage planning process, this platform empowers users to customize their journeys according to their preferences and desires.

Through seamless coordination of boat rentals, route planning, onboard amenities, and guest management, this platform promises to revolutionize the way house boat voyages are organized. It ensures that every journey is meticulously planned and executed, guaranteeing an unforgettable experience for all passengers.

With this innovative tool at their disposal, travelers can look forward to effortless and enjoyable voyage planning experiences, marking the dawn of a new era in nautical exploration. Overall, an online platform for house boat voyages not only streamlines logistics but also enhances the overall journey experience, promising memorable adventures and cherished memories for years to come.