



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

HEMALATHAA K (Register No:727821TUIT036)

KABILA B S (RegisterNo:727821TUIT045)

MAHESH BOOPATHI (RegisterNo: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.

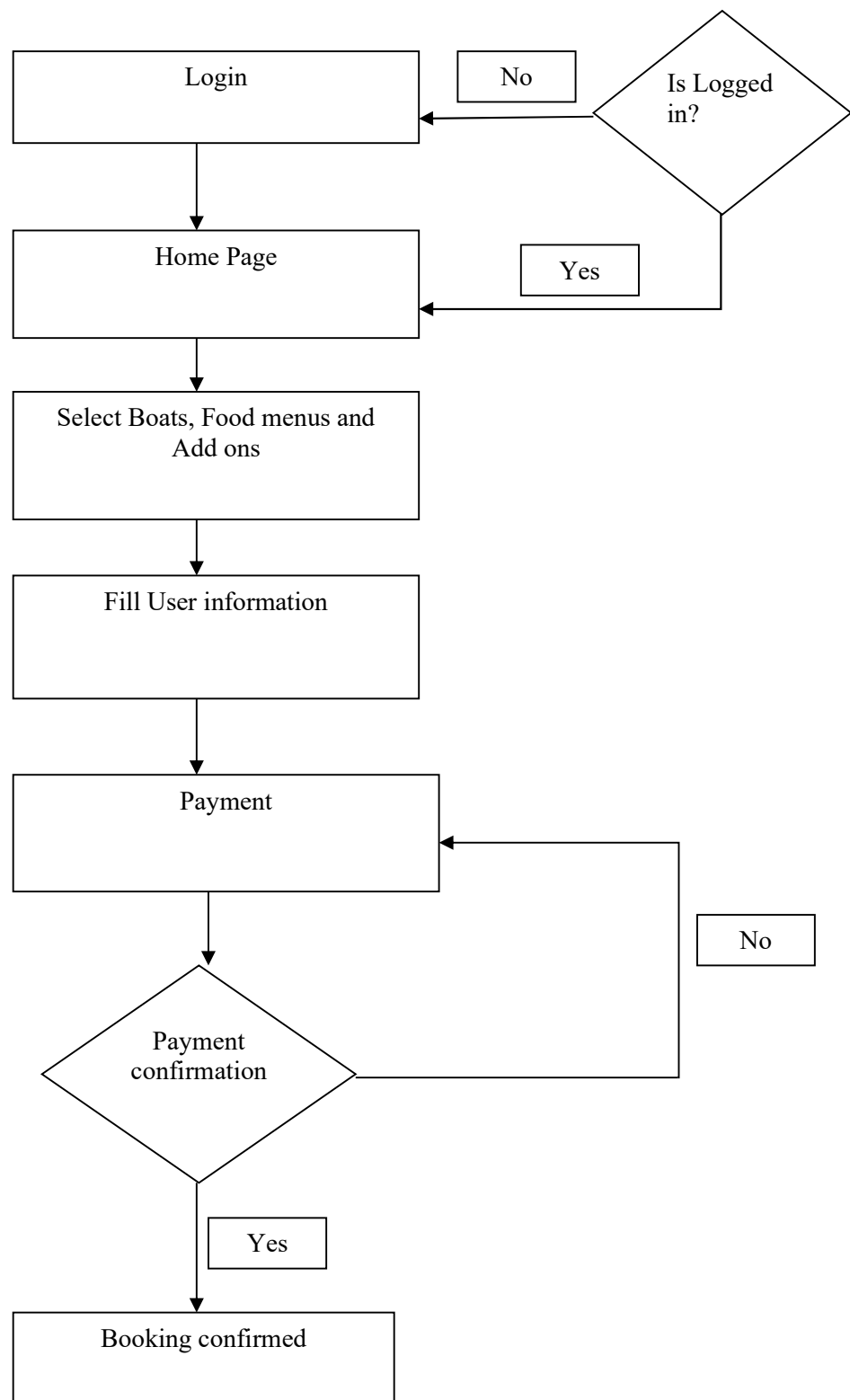
Flexibility is paramount in the realm of voyage planning. Despite careful planning, unforeseen circumstances may arise, necessitating swift thinking and adaptability to guarantee smooth sailing. By remaining composed under pressure and making quick decisions, you can overcome any challenges that may 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:

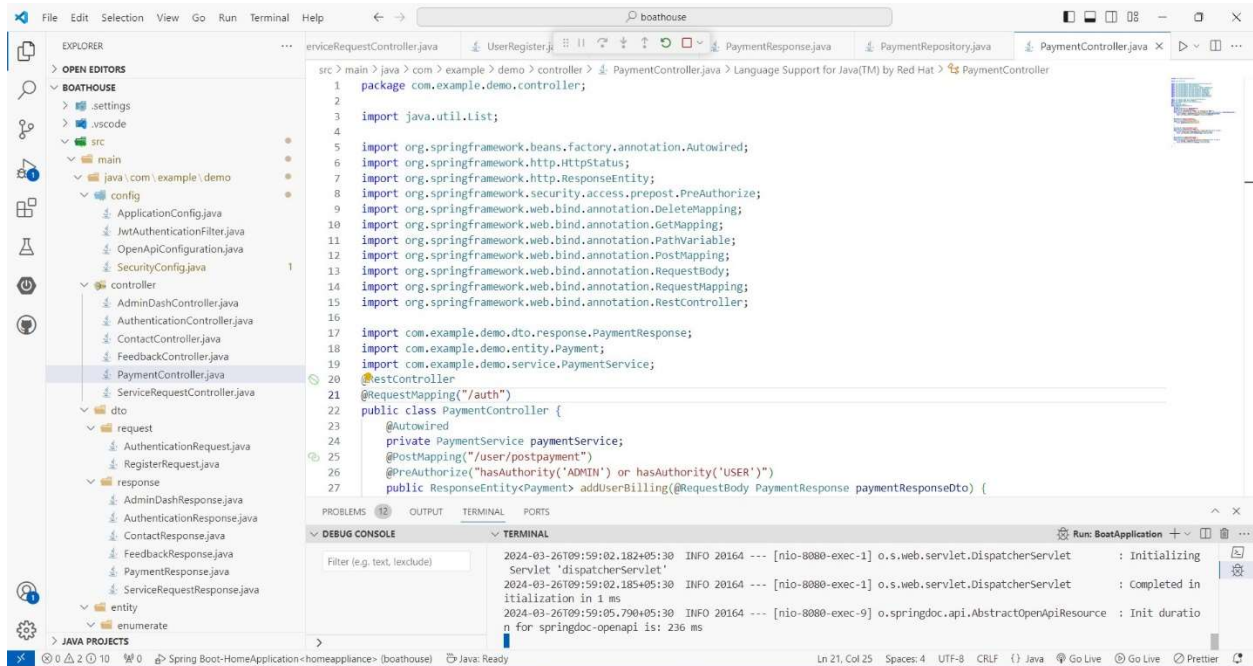
The objective of a house boat voyage is multifaceted, encompassing elements of exploration, relaxation, and connection. This unique experience provides an opportunity for travelers to come together, relishing in the serene surroundings of nature while connecting with each other and the journey itself. 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:



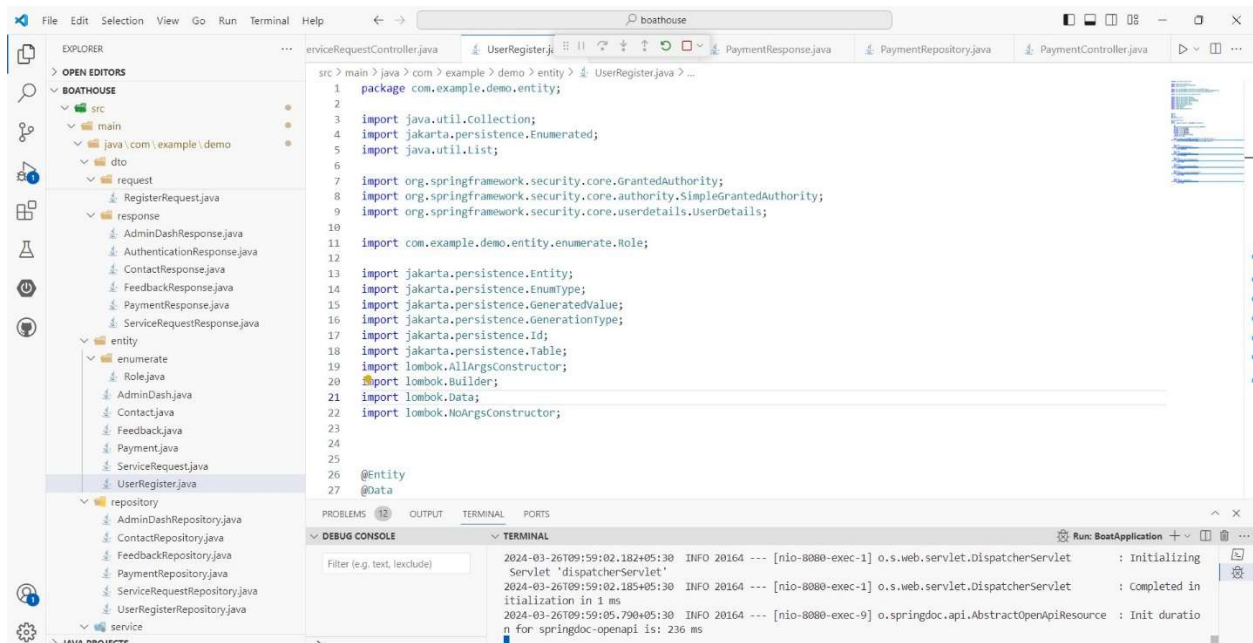
BACKEND:



The screenshot shows the VS Code editor with the `PaymentController.java` file open. The Explorer panel on the left shows the project structure, including the `controller` package. The main editor displays the following code:

```
1 package com.example.demo.controller;
2
3 import java.util.List;
4
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.http.HttpStatus;
7 import org.springframework.http.ResponseEntity;
8 import org.springframework.security.access.prepost.PreAuthorize;
9 import org.springframework.web.bind.annotation.DeleteMapping;
10 import org.springframework.web.bind.annotation.GetMapping;
11 import org.springframework.web.bind.annotation.PathVariable;
12 import org.springframework.web.bind.annotation.PostMapping;
13 import org.springframework.web.bind.annotation.RequestBody;
14 import org.springframework.web.bind.annotation.RequestMapping;
15 import org.springframework.web.bind.annotation.RestController;
16
17 import com.example.demo.dto.response.PaymentResponse;
18 import com.example.demo.entity.Payment;
19 import com.example.demo.service.PaymentService;
20
21 @RestController
22 @RequestMapping("/auth")
23 public class PaymentController {
24     @Autowired
25     private PaymentService paymentService;
26     @PostMapping("/user/postpayment")
27     @PreAuthorize("hasAuthority('ADMIN') or hasAuthority('USER')")
28     public ResponseEntity<Payment> addUserBilling(@RequestBody PaymentResponse paymentResponseDto) {
```

The bottom panel shows the terminal output, indicating the application is running successfully.

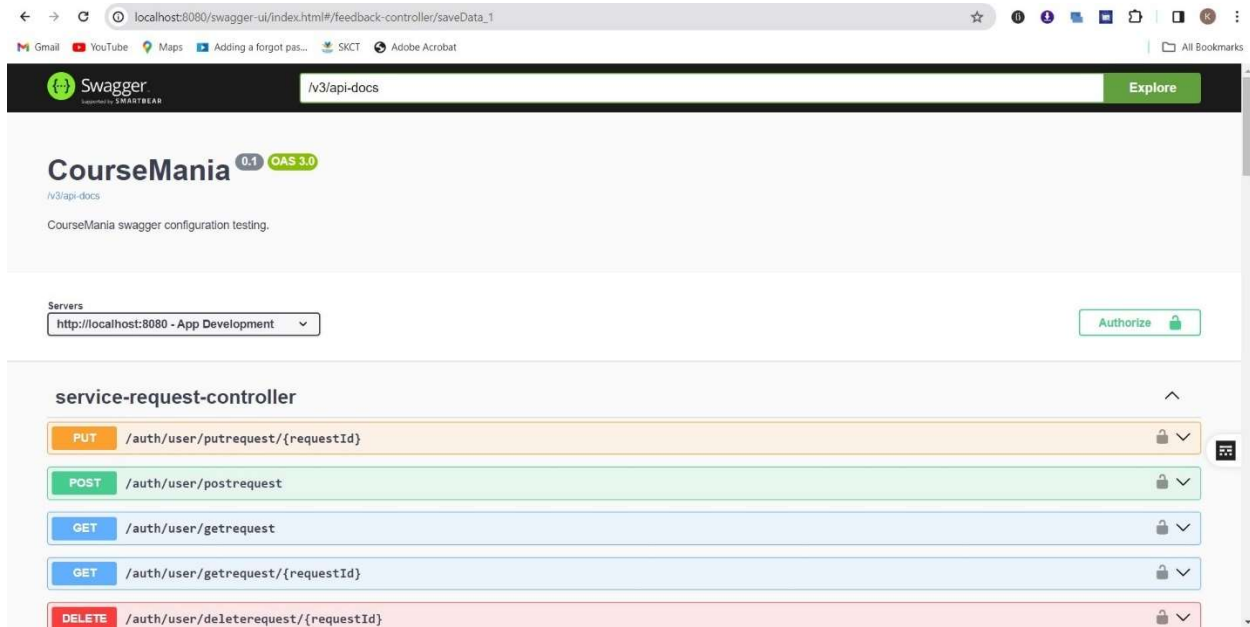


The screenshot shows the VS Code editor with the `UserRegister.java` file open. The Explorer panel on the left shows the project structure, including the `entity` package. The main editor displays the following code:

```
1 package com.example.demo.entity;
2
3 import java.util.Collection;
4 import jakarta.persistence.EnumType;
5 import java.util.List;
6
7 import org.springframework.security.core.GrantedAuthority;
8 import org.springframework.security.core.authority.SimpleGrantedAuthority;
9 import org.springframework.security.core.userdetails.UserDetails;
10
11 import com.example.demo.entity.enumerate.Role;
12
13 import jakarta.persistence.Entity;
14 import jakarta.persistence.EnumType;
15 import jakarta.persistence.GeneratedValue;
16 import jakarta.persistence.GenerationType;
17 import jakarta.persistence.Id;
18 import jakarta.persistence.Table;
19 import lombok.AllArgsConstructor;
20 import lombok.Builder;
21 import lombok.Data;
22 import lombok.NoArgsConstructor;
23
24 @Entity
25 @Data
26 @Builder
27 @NoArgsConstructor
```

The bottom panel shows the terminal output, indicating the application is running successfully.

SWAGGER UI:

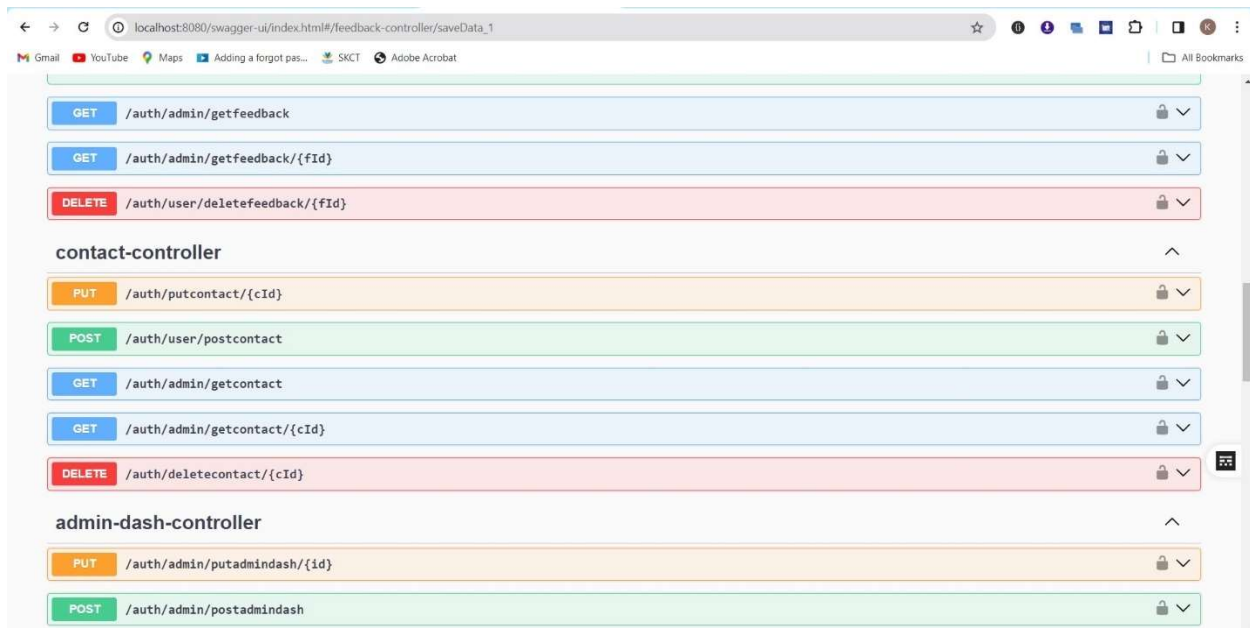


A screenshot of the Swagger UI for the CourseMania API. The browser address bar shows the URL `localhost:8080/swagger-ui/index.html#/feedback-controller/saveData_1`. The Swagger logo is in the top left, and a search bar contains `/v3/api-docs` with an **Explore** button. The main header displays **CourseMania** with version `0.1` and `OAS 3.0`. Below the header, it says `/v3/api-docs` and `CourseMania swagger configuration testing.`

The **Servers** section shows a dropdown menu with `http://localhost:8080 - App Development` and an **Authorize** button.

The **service-request-controller** section is expanded, showing a list of endpoints:

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



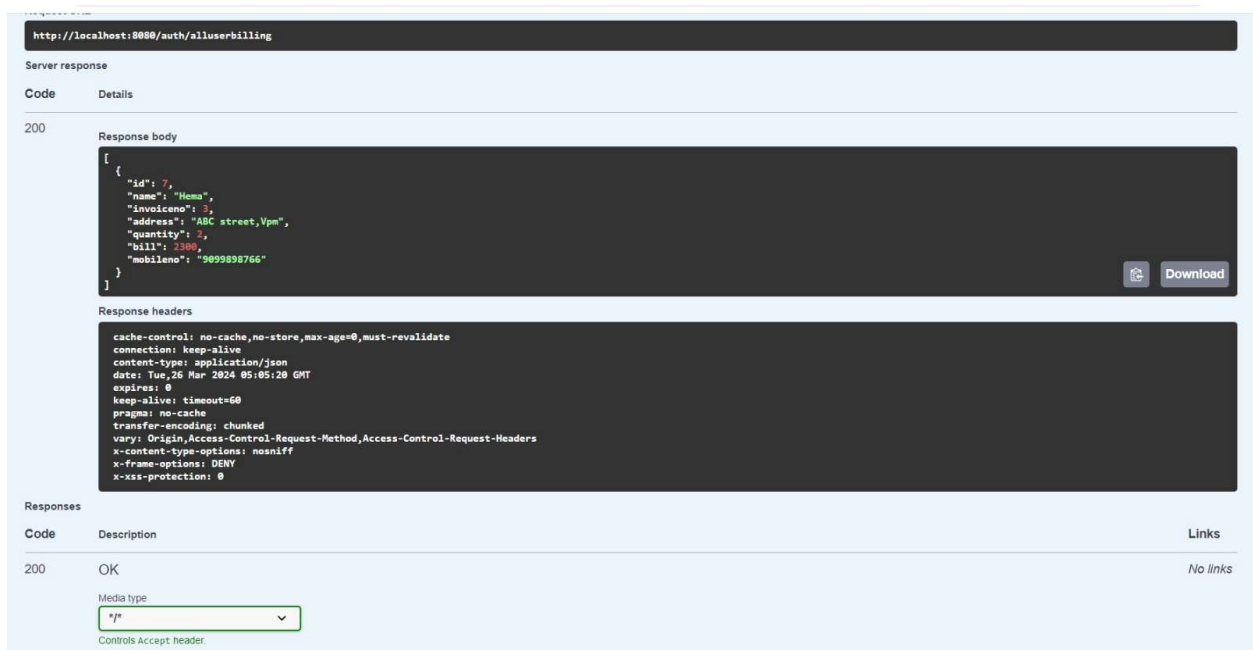
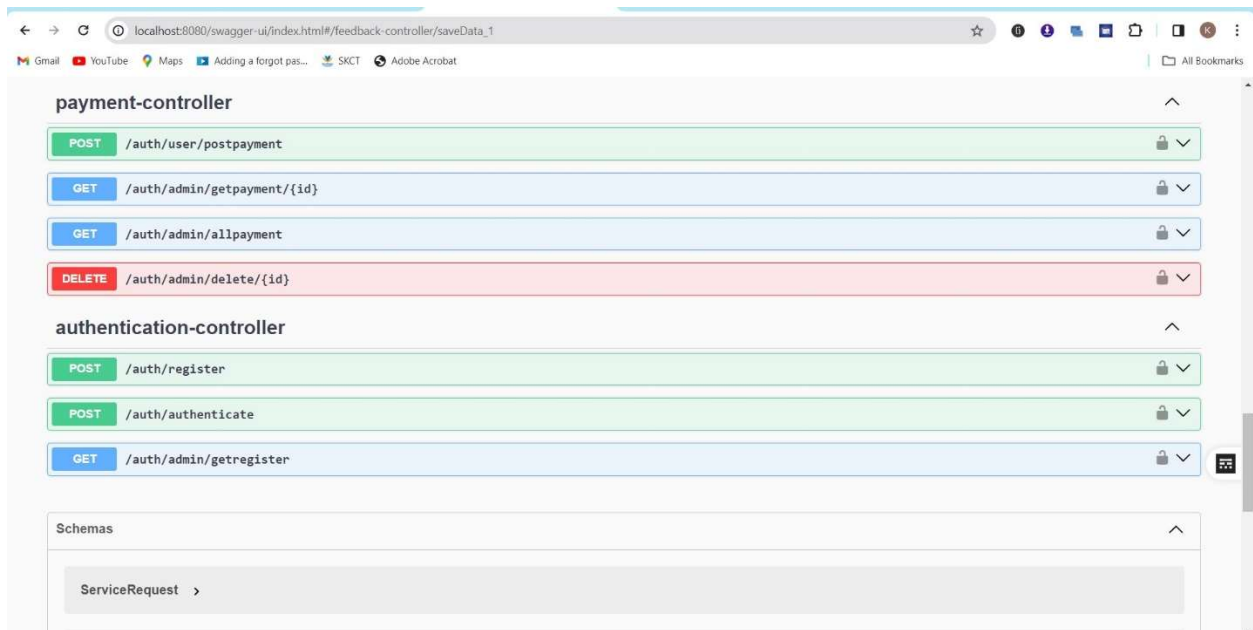
A continuation of the Swagger UI screenshot, showing the **contact-controller** and **admin-dash-controller** sections.

The **contact-controller** section is expanded, showing a list of endpoints:

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

The **admin-dash-controller** section is expanded, showing a list of endpoints:

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



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.Authen
ticationConfiguration;
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 userDetailsService()
{
    return username -> userRepository.findByUsername(username)
        .orElseThrow(() -> new UsernameNotFoundException("User not
found!"));
}

@Bean
public AuthenticationProvider authenticationProvider() {
    DaoAuthenticationProvider authProvider = new DaoAuthenticationProvider();
    authProvider.setUserDetailsService(userDetailsService());
    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$10$Hn4Me7/KQmaYBm5R191J0phGvVmPAAI.gFKGIRgbDk0wt3uvdi",
    "mobilen": "string",
    "roles": "ADMIN"
  },
  {
    "userid": 2,
    "username": "string",
    "emailid": "string",
    "password": "$2a$10$PTQf6HjbJVigL.oDofM..uhru74QK1PcZjuvC700MSPlcJR2OuPXm",
    "mobilen": "string",
    "roles": "ADMIN"
  },
  {
    "userid": 3,
    "username": "Moni",
    "emailid": "moni@gmail.com",
    "password": "$2a$10$TEESQ8RV2650PFIFybQzuMrFf1zhKxOfrq75t3uU4o70qdXl8Z5",
    "mobilen": "9899098766",
    "roles": "USER"
  }
]
```

[Download](#)

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

[Download](#)

Response headers

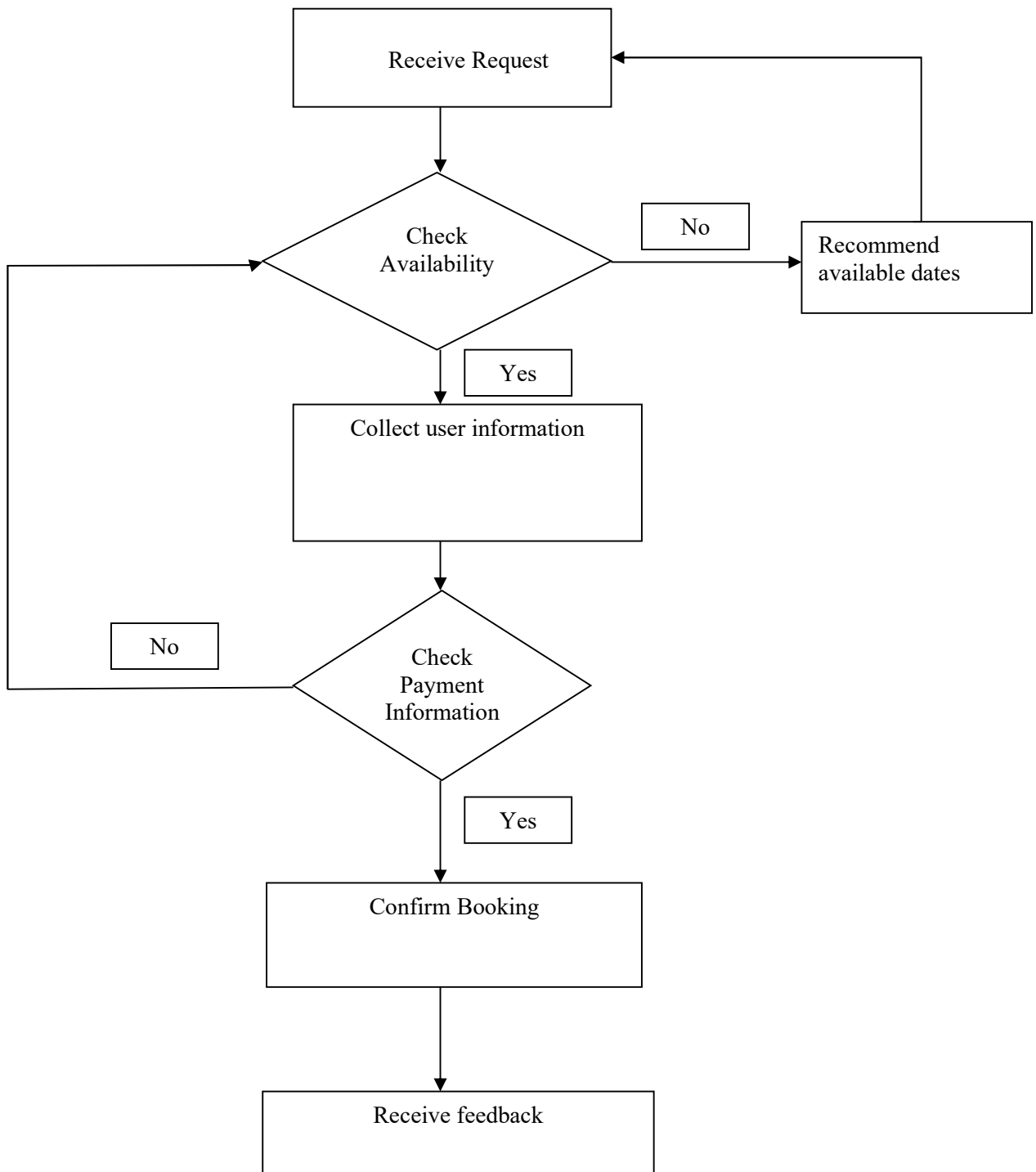
```
cache-control: no-cache,no-store,max-age=0,must-revalidate
connection: keep-alive
content-length: 36
content-type: text/plain;charset=UTF-8
date: Tue, 26 Mar 2024 05:19:19 GMT
expires: 0
keep-alive: timeout=60
pragma: no-cache
vary: Origin,Access-Control-Request-Method,Access-Control-Request-Headers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

Responses

Code	Description
------	-------------

[Links](#)

ADMIN CHART:



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.