Full-Stack Car Rental Project

C4 Code Diagram emerges as a powerful visualization tool in the field of software engineering. It offers an effective tool for understanding, designing, and communicating the code-level intricacies of software architecture. Within complex software systems, it is vital to understand the organization of the code, the relationships between components, and the way these elements are put together. This is where the C4 Code Diagram stands out; It offers a clear perspective focusing on the modular structure of the code base, the dependencies between components and the way these elements are brought together. When the C4 code diagram of the car rental site is examined, the diagrams below are reached.

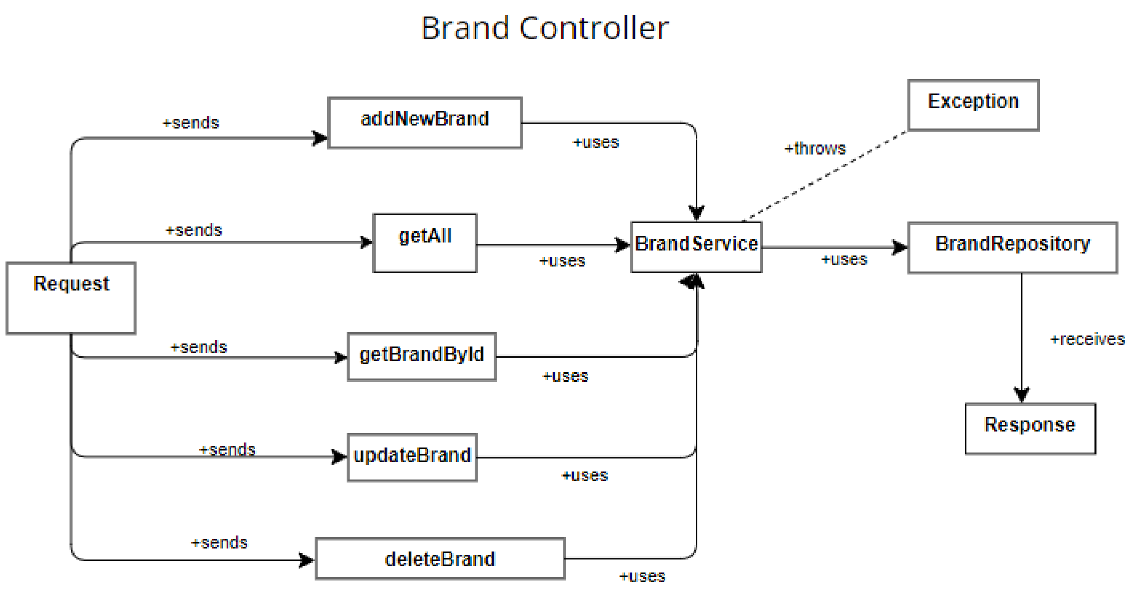


Figure 1: Brand Controller Code Diagram

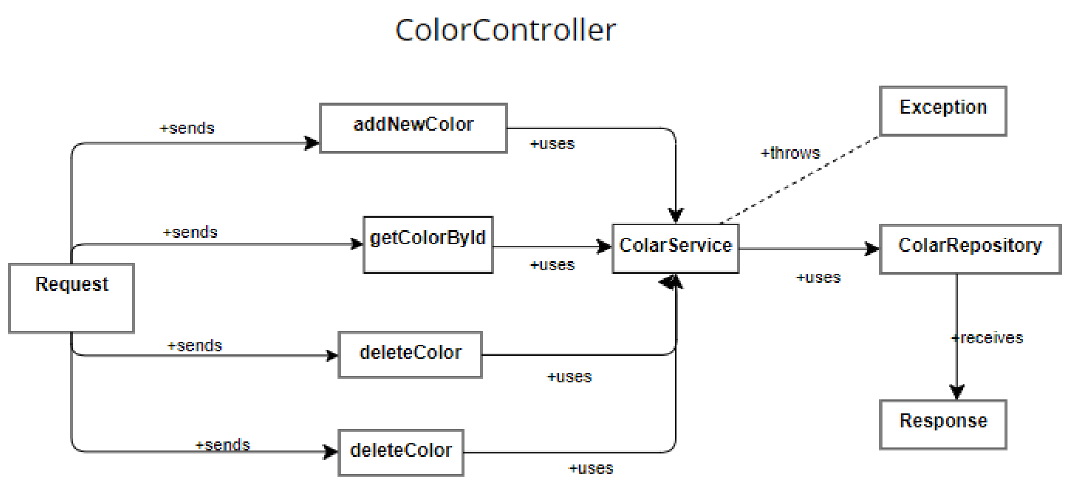


Figure 2: Color Controller Code Diagram

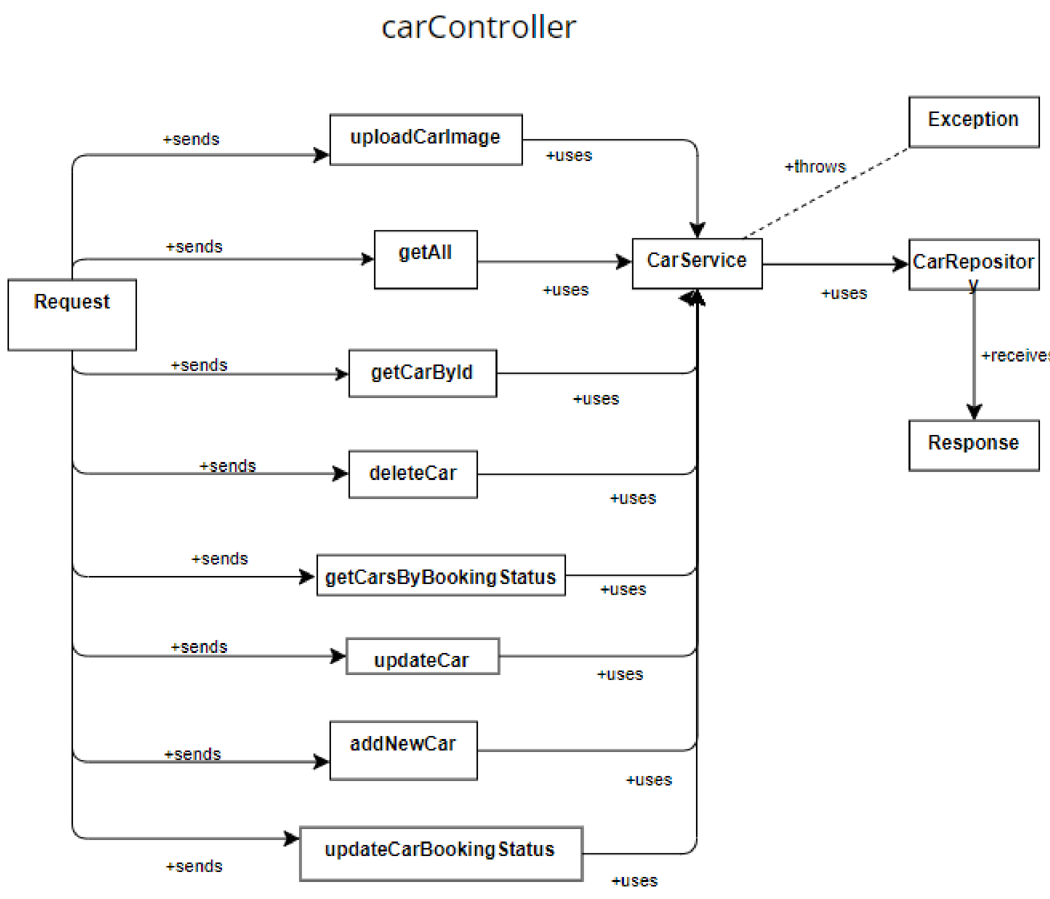


Figure 3: Car Controller Code Diagram

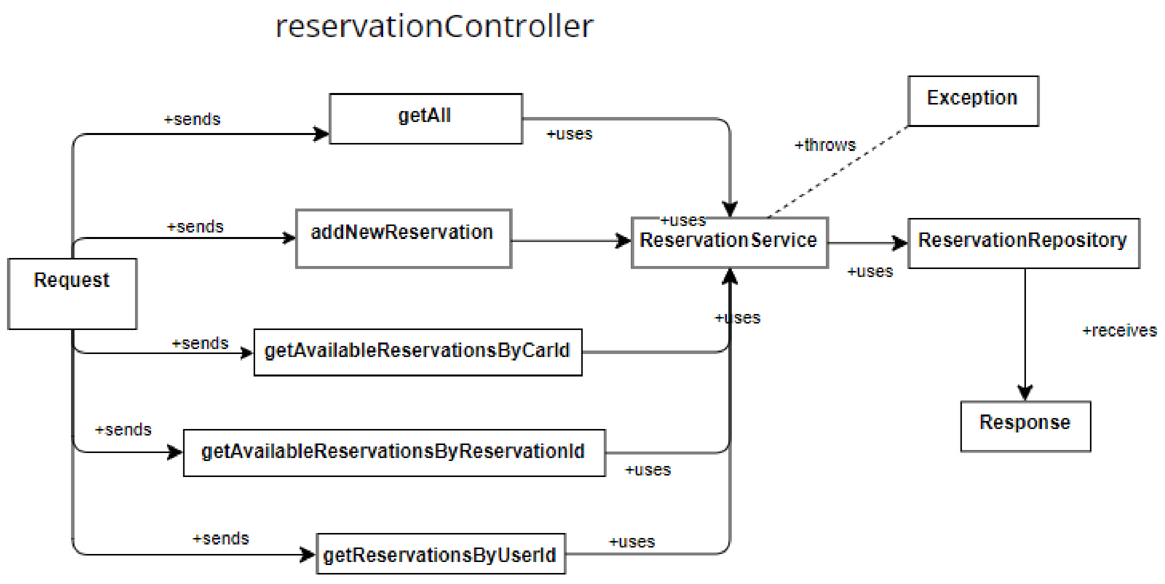


Figure 4: Reservation Controller Code Diagram

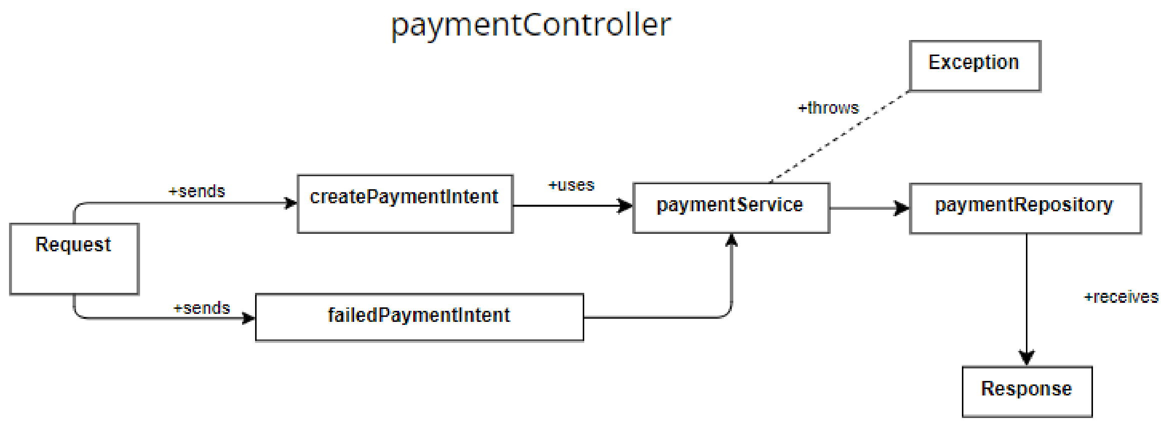


Figure 5: Payment Controller Code Diagram

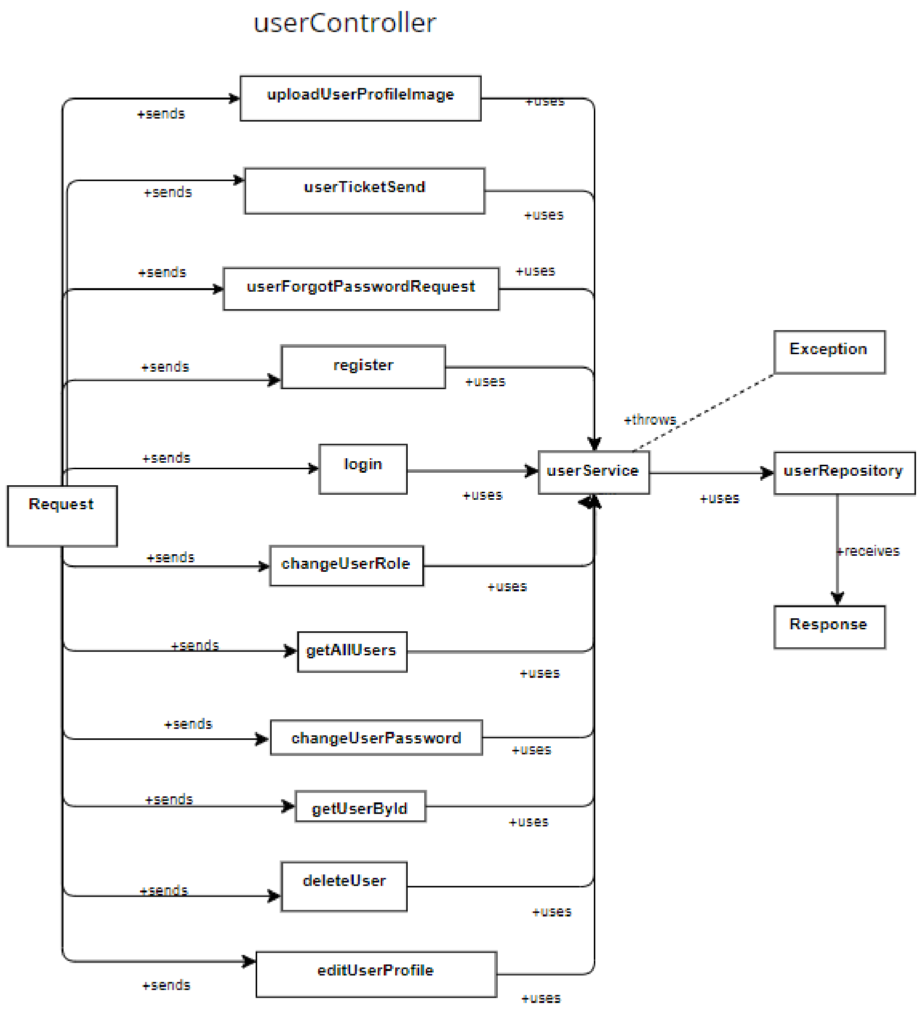


Figure 6: User Controller Code Diagram

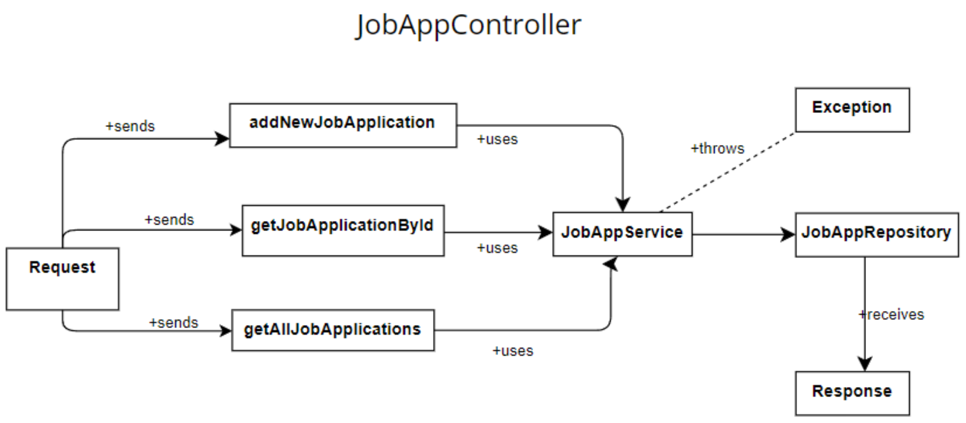


Figure 7: Job App Controller Code Diagram

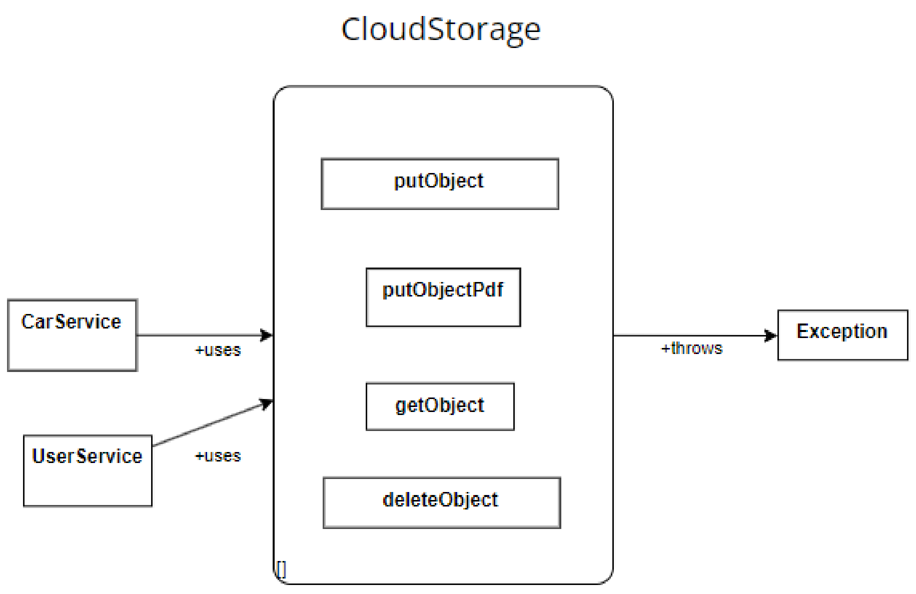


Figure 8: Cloud Storage Code Diagram

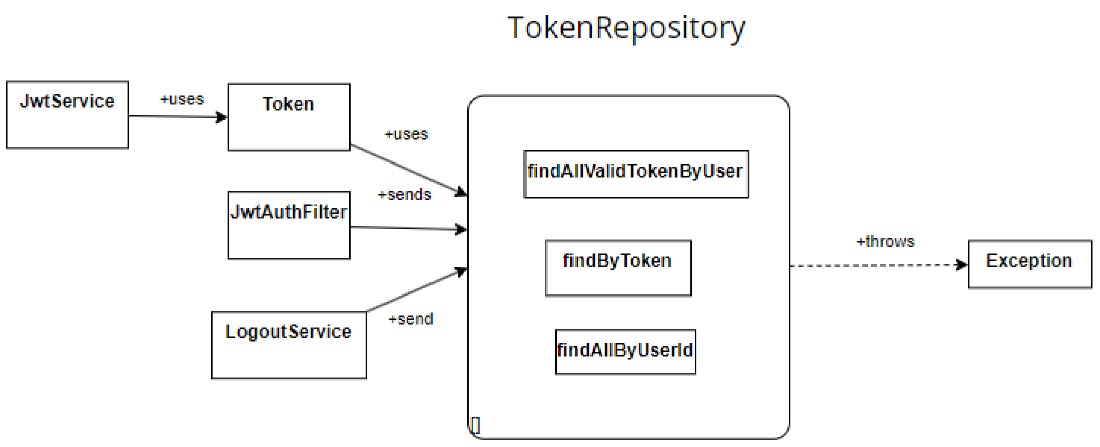


Figure 9: Token Repository Code Diagram

In the runtime view of the car rental project, interactions between users and administrators are facilitated by the HTTP protocol. Users primarily engage with the system through a web interface, allowing them to browse car listings, view detailed information, make reservations, and manage their profiles. Each user interaction triggers a series of HTTP requests to the Java Spring Boot back-end, which, behind the scenes, processes these requests. The back-end communicates with a Linux-based database, utilizing the Ubuntu distribution, to fetch or store data and applies business logic to validate user actions. The processed information is then sent back to the client as the desired output.

Administrators, on the other hand, access the system through a specialized dashboard, initiating interactions that involve privileged operations such as adding new cars, modifying details, managing reservations, and overseeing system operations. These administrator interactions also require authentication and authorization checks to ensure secure operations. The specialized dashboard communicates with the same Java Spring Boot back-end, and like user interactions, triggers a series of HTTP requests and responses.

Key events within the system, whether initiated by users or administrators, include actions like "View Car Details," "Edit Profile Picture," "Forgot Password," "Login," and "Register." Each of these events corresponds to specific HTTP requests that carry out the desired functionalities. Throughout these interactions, the HTTP protocol governs the secure and efficient transmission of data between the client and the server, ensuring a seamless experience for both users and administrators in the car rental system.

The project is designed with several core quality attributes:

User-Friendly Interface: The system boasts a user-friendly UI, ensuring ease of navigation and intuitive interactions. Visual cues and clear instructions guide users through various functionalities, enhancing their overall experience.

Performance: Emphasis has been placed on ensuring the system's responsiveness and speed. Techniques such as load balancing, implemented through AWS Elastic Beanstalk and EC2 instances, ensure that traffic is evenly distributed, optimizing system performance even during peak usage.

Robustness: The system incorporates robust error-handling mechanisms. By anticipating potential issues, the system provides a seamless user experience, even in the face of occasional errors or disruptions.

Security: A paramount concern, the system prioritizes data protection and secure access. Mechanisms such as Role-Based Access Control and JSON Web Tokens (JWT) are employed to safeguard system resources and user data. Furthermore, secure payment gateways like Stripe bolster transactional security.

Reliability: The system's reliability is ensured through practices like data durability and high availability. AWS RDS offers automatic backups and replication across multiple availability zones, guaranteeing data integrity and system uptime.

Scalability: Built to accommodate growth, the system leverages auto-scaling capabilities through AWS Elastic Beanstalk and EC2. This ensures that the system can handle increased loads without compromising performance.

The project architecture comprises several pivotal components:

JWT (JSON Web Token): This technology has played a crucial role in enhancing the security of the Back-End systems. It facilitates the storage of sensitive information in encrypted strings, ensuring the confidentiality of data. As security is a critical aspect of the system, JWT has likely undergone continuous updates and improvements to adapt to evolving security threats and encryption standards.

Spring Boot: As a key technology used in the development of the Back-End logic and RESTful web services, Spring Boot has been pivotal in ensuring robust and efficient functionality behind the scenes. Given its importance in the system's performance and maintainability, it is likely that Spring Boot has been subject to ongoing enhancements and optimizations to meet the evolving demands of the car rental platform.

React.js: This JavaScript library has been instrumental in building interactive and responsive user interfaces, enhancing the overall user experience. Given the dynamic nature of front-end technologies, React.js has likely undergone iterative updates and improvements to keep pace with evolving user interface design trends and user interaction patterns.

For the future roadmap, the following components are crucial for the continued evolution and enhancement of the car rental website project:

Continuous Improvement and Innovation: The project leader's focus on introducing new features to boost user engagement, streamline the booking process, and implement enhanced security measures indicates the need for ongoing innovation and feature development. This suggests that future enhancements will likely focus on introducing new functionalities and refining existing features to align with user needs and industry trends.

Technological Upgrades: The project leader's commitment to keeping up with evolving technologies and ensuring the systems remain technologically up-to-date and adaptable highlights the importance of future technological upgrades. This may involve integrating emerging technologies, such as contactless services and sustainability initiatives, to enhance the platform's capabilities and relevance in the car rental industry.

User Interface Updates: The priority given to regular updates to the user interface, incorporating user feedback to ensure an intuitive and visually appealing experience, indicates a focus on enhancing the user interface design and user experience. Future roadmap initiatives may involve iterative improvements to the user interface, driven by user feedback and design best practices.

In summary, the hotspot components from the past have likely undergone significant updates to enhance security, performance, and user experience, while the future roadmap emphasizes continuous improvement, technological upgrades, and user interface updates to drive the evolution of the car rental website project.

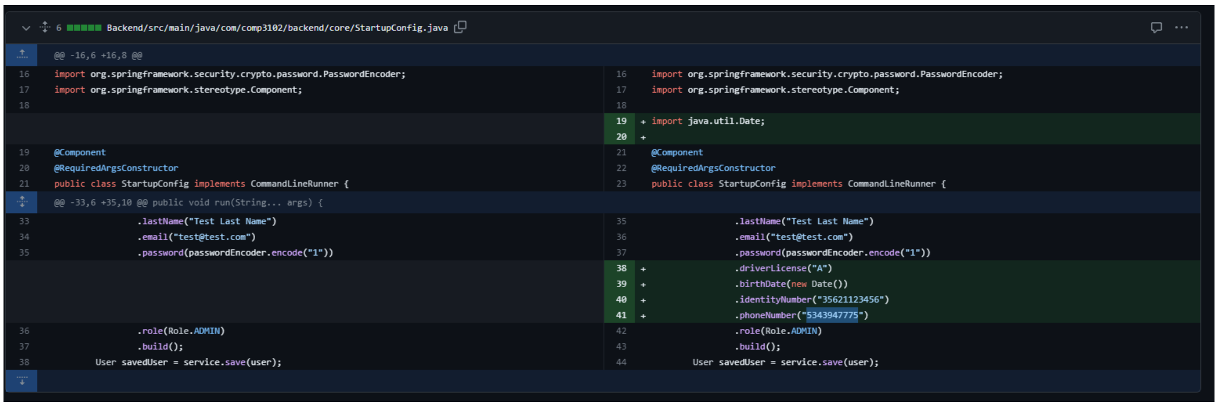
As a result of examining the car rental site, it was observed that no classes were static and were not created depending on each other when creating the back end. While creating the project, it was determined that different methods were used to transfer information between user pages and therefore many requests were made. When the user tokens were examined, it was seen that user tokens were used to create each page and these tokens were used to fill in the private information on the page. However, it was determined that the changes made with the token affected the page codes and caused great difficulties during the change.

No violation of modularity was detected. However, unhealthy inheritance situation was observed. Since bound variables were defined in a separate class and the methods that needed to be defined were specified as interfaces, the parent-child relationship could not be established as expected. Hierarchy was defined only by defining mandatory methods and variables, starting from the parent and moving towards the child, depending only on the child. Constant variables were located in separate classes instead of being defined as hard code in each class. This allowed code changes to not affect different classes.

Circular dependencies were identified in package dependencies. The car package was dependent on the user and reservation packages; It was also dependent on the reservation package, user and car packages. The user package was also dependent on the reservation package. The dependencies between these packages formed interdependent cyclic structures.

Additionally, service classes were communicating with classes in many different packages due to the need to communicate with the database. This situation brought with it many addictions. For example, the user service was dependent on classes such as UserRepository, passwordEncoder, jwtService, authenticationManager, tokenRepository, user2UserResponseDtoConverter, emailSenderService, reservationService, s3Service.

When the project was started, during the setting of dependencies, user information was entered as hard coded at the first stage, as the classes had many connections to each other due to the complexity of the project. Since it is thought that the user class must comply with quality standards, it contains placeholder information for the classes they are connected to, which are required in the pull request in figure 10 but have not been added yet.



```
6 Backend/src/main/java/com/comp3102/backend/core/StartupConfig.java

@@ -16,6 +16,8 @@
```

```java
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Component;

@Component
@RequiredArgsConstructor
public class StartupConfig implements CommandLineRunner {
```

```
@@ -33,6 +35,10 @@ public void run(String... args) {
```

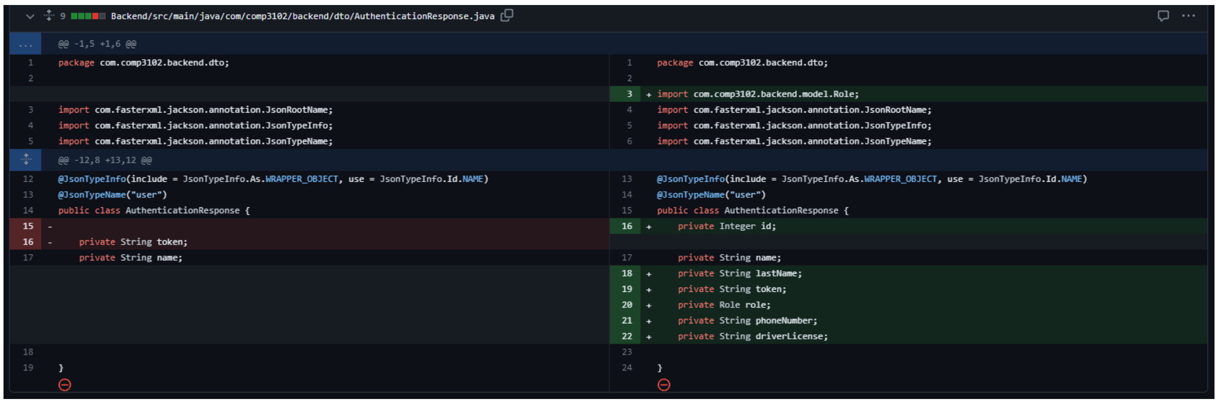
```java
                .lastName("Test Last Name")
                .email("test@test.com")
                .password(passwordEncoder.encode("1"))
                .role(Role.ADMIN)
                .build();
        User savedUser = service.save(user);
```

```java
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Component;

+ import java.util.Date;
+
@Component
@RequiredArgsConstructor
public class StartupConfig implements CommandLineRunner {
```

```java
                .lastName("Test Last Name")
                .email("test@test.com")
                .password(passwordEncoder.encode("1"))
+               .driverLicense("A")
+               .birthDate(new Date())
+               .identityNumber("35621123456")
+               .phoneNumber("5343947775")
                .role(Role.ADMIN)
                .build();
        User savedUser = service.save(user);
```

Figure 10

In Figure 11, the attributes required to reach quality standards have been added dynamically.



```
9 Backend/src/main/java/com/comp3102/backend/dto/AuthenticationResponse.java

@@ -1,5 +1,6 @@
```

```java
package com.comp3102.backend.dto;

import com.fasterxml.jackson.annotation.JsonRootName;
import com.fasterxml.jackson.annotation.JsonTypeInfo;
import com.fasterxml.jackson.annotation.JsonTypeName;
```

```
@@ -12,8 +13,12 @@
```

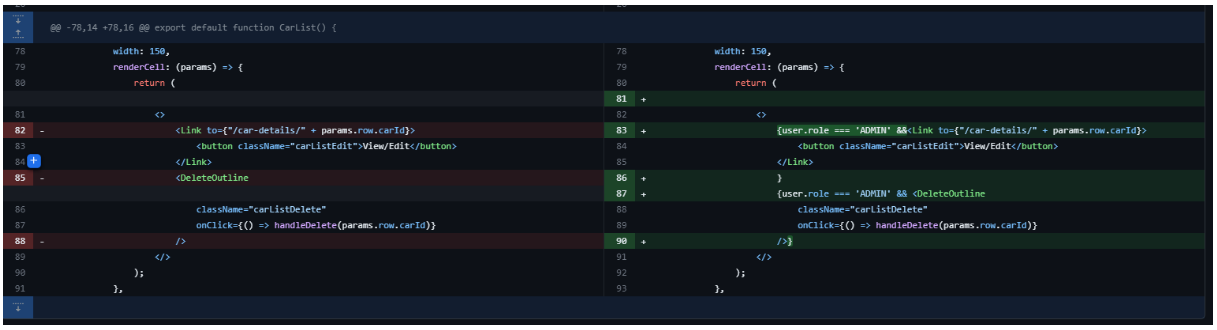
```java
@JsonTypeInfo(include = JsonTypeInfo.As.WRAPPER_OBJECT, use = JsonTypeInfo.Id.NAME)
@JsonTypeName("user")
public class AuthenticationResponse {
-
-    private String token;
     private String name;

}
```

```java
package com.comp3102.backend.dto;

+ import com.comp3102.backend.model.Role;
import com.fasterxml.jackson.annotation.JsonRootName;
import com.fasterxml.jackson.annotation.JsonTypeInfo;
import com.fasterxml.jackson.annotation.JsonTypeName;
```

```java
@JsonTypeInfo(include = JsonTypeInfo.As.WRAPPER_OBJECT, use = JsonTypeInfo.Id.NAME)
@JsonTypeName("user")
public class AuthenticationResponse {
+    private Integer id;
     private String name;
+    private String lastName;
+    private String token;
+    private Role role;
+    private String phoneNumber;
+    private String driverLicense;

}
```

Figure 11

By adding a role system in Figure 12, each user is prevented from making changes to the system.



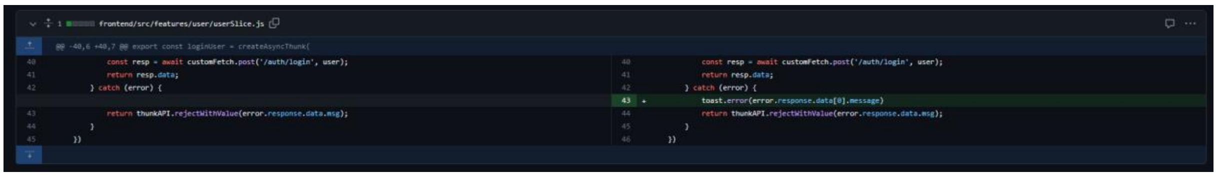
```
@@ -78,14 +78,16 @@ export default function CarList() {
```

```
            width: 150,
            renderCell: (params) => {
                return (
                    <>
-                       <Link to={"/car-details/" + params.row.carId}>
                            <button className="carListEdit">View/Edit</button>
                        </Link>
-                       <DeleteOutline
                            className="carListDelete"
                            onClick={() => handleDelete(params.row.carId)}
-                       />
                    </>
                );
            },
```

```
            width: 150,
            renderCell: (params) => {
                return (
+
                    <>
+                       {user.role === 'ADMIN' &&<Link to={"/car-details/" + params.row.carId}>
                            <button className="carListEdit">View/Edit</button>
                        </Link>
+                       }
+                       {user.role === 'ADMIN' && <DeleteOutline
                            className="carListDelete"
                            onClick={() => handleDelete(params.row.carId)}
+                       />}
                    </>
                );
            },
```

Figure 12

A pull request has been placed in Figure 13 so that users can see the mistakes they made during the login screen and a user-friendly error message will be returned.



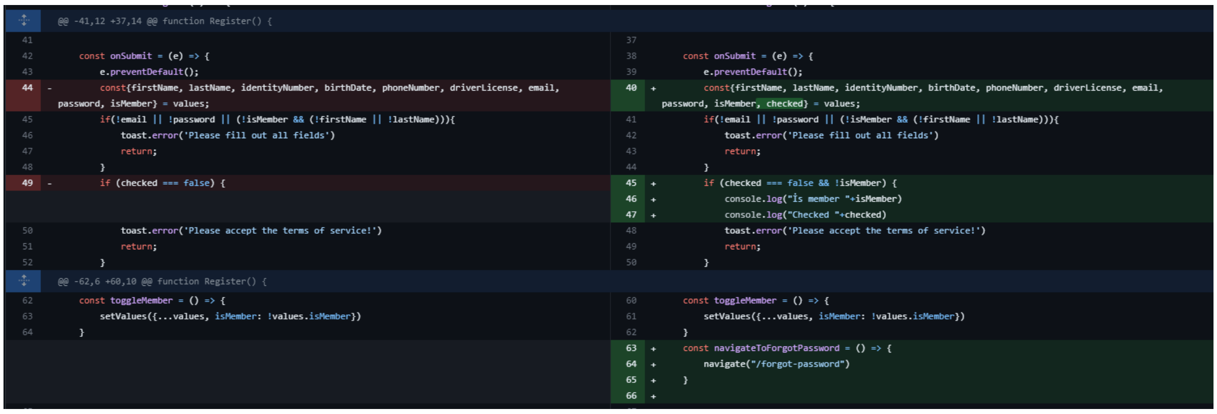
```
1 frontend/src/features/user/userSlice.js
@@ -40,6 +40,7 @@ export const loginUser = createAsyncThunk(
```

```
            const resp = await customFetch.post('/auth/login', user);
            return resp.data;
        } catch (error) {
            return thunkAPI.rejectWithValue(error.response.data.msg);
        }
    })
```

```
            const resp = await customFetch.post('/auth/login', user);
            return resp.data;
        } catch (error) {
+           toast.error(error.response.data[0].message)
            return thunkAPI.rejectWithValue(error.response.data.msg);
        }
    })
```

Figure 13

In Figure 14, the logging structure has been added to the project, thus creating a recording system that can be tracked and analyzed by recording important events and errors in the system.



```
@@ -41,12 +37,14 @@ function Register() {
```

```

    const onSubmit = (e) => {
        e.preventDefault();
-       const{firstName, lastName, identityNumber, birthDate, phoneNumber, driverLicense, email,
password, isMember} = values;
        if(!email || !password || (!isMember && (!firstName || !lastName))){
            toast.error('Please fill out all fields')
            return;
        }
-       if (checked === false) {
            toast.error('Please accept the terms of service!')
            return;
        }
```

```

    const onSubmit = (e) => {
        e.preventDefault();
+       const{firstName, lastName, identityNumber, birthDate, phoneNumber, driverLicense, email,
password, isMember, checked} = values;
        if(!email || !password || (!isMember && (!firstName || !lastName))){
            toast.error('Please fill out all fields')
            return;
        }
+       if (checked === false && !isMember) {
+           console.log("İs member "+isMember)
+           console.log("Checked "+checked)
            toast.error('Please accept the terms of service!')
            return;
        }
```

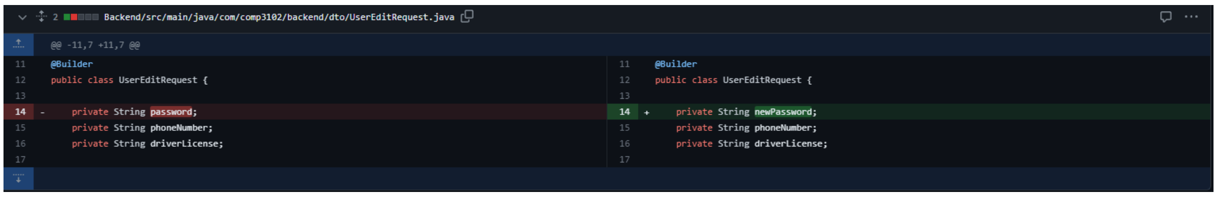
```
@@ -62,6 +60,10 @@ function Register() {
```

```
    const toggleMember = () => {
        setValues({...values, isMember: !values.isMember})
    }
```

```
    const toggleMember = () => {
        setValues({...values, isMember: !values.isMember})
    }
+   const navigateToForgotPassword = () => {
+       navigate("/forgot-password")
+   }
+
```

Figure 14

In Figure 15, the nomenclature in the code has been updated in the parts changed by the user in order to avoid confusion for the parts containing information changed by the user.



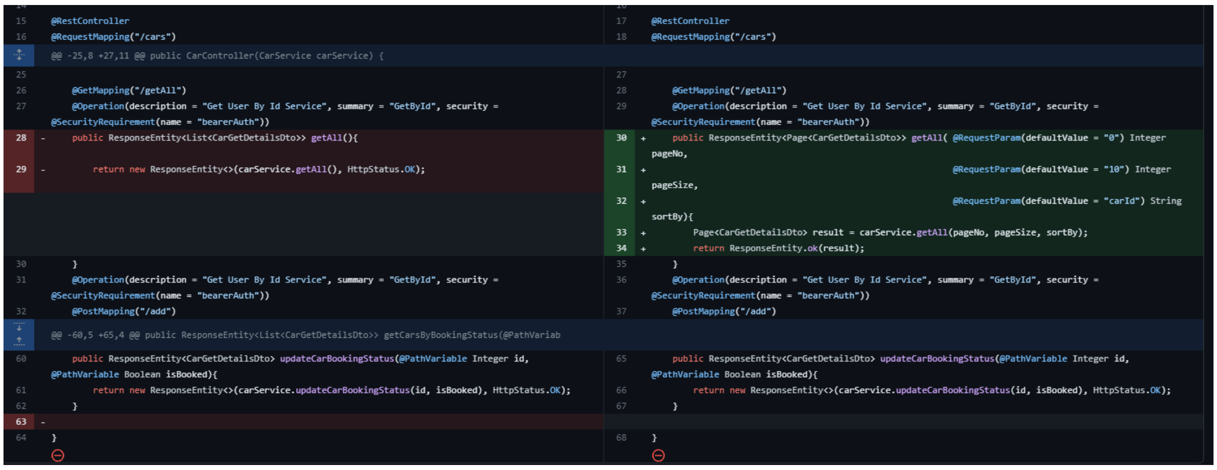
```
2 Backend/src/main/java/com/comp3102/backend/dto/UserEditRequest.java
@@ -11,7 +11,7 @@
```

```
@Builder
public class UserEditRequest {

-   private String password;
    private String phoneNumber;
    private String driverLicense;

```

```
@Builder
public class UserEditRequest {

+   private String newPassword;
    private String phoneNumber;
    private String driverLicense;

```

Figure 15

In Figure 16, the specific endpoint processes an HTTP GET request used to retrieve the details of the cars. This feature has been added to offer users the ability to obtain detailed information of cars.

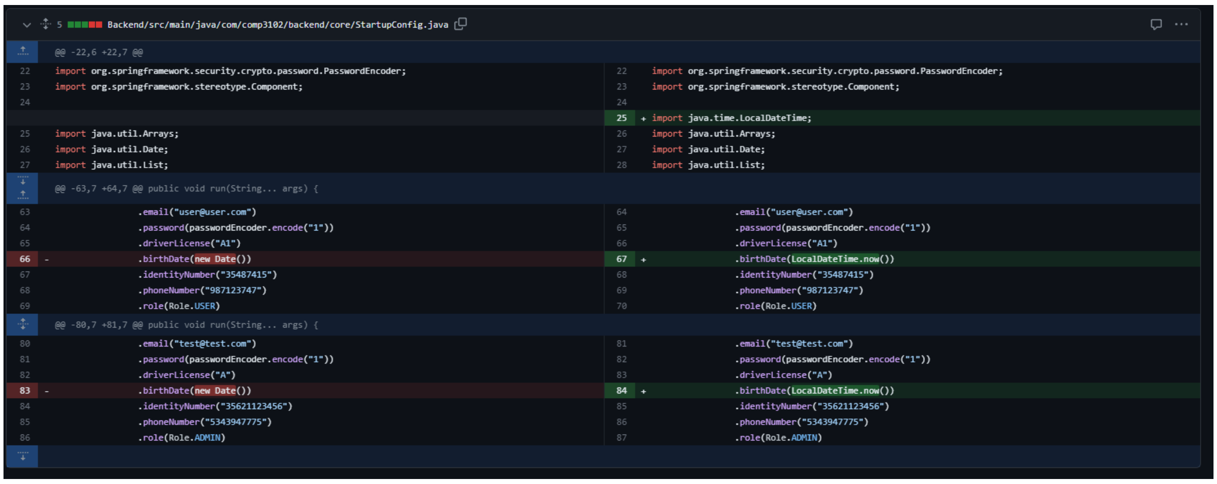


```
@RestController
@RequestMapping("/cars")
@@ -25,8 +27,11 @@ public CarController(CarService carService) {

    @GetMapping("/getAll")
    @Operation(description = "Get User By Id Service", summary = "GetById", security =
@SecurityRequirement(name = "bearerAuth"))
-    public ResponseEntity<List<CarGetDetailsDto>> getAll(){

-        return new ResponseEntity<>(carService.getAll(), HttpStatus.OK);

    }
    @Operation(description = "Get User By Id Service", summary = "GetById", security =
@SecurityRequirement(name = "bearerAuth"))
    @PostMapping("/add")
@@ -60,5 +65,4 @@ public ResponseEntity<List<CarGetDetailsDto>> getCarsByBookingStatus(@PathVariab
    public ResponseEntity<CarGetDetailsDto> updateCarBookingStatus(@PathVariable Integer id,
@PathVariable Boolean isBooked){
        return new ResponseEntity<>(carService.updateCarBookingStatus(id, isBooked), HttpStatus.OK);
    }
-
}
```

```
@RestController
@RequestMapping("/cars")

    @GetMapping("/getAll")
    @Operation(description = "Get User By Id Service", summary = "GetById", security =
@SecurityRequirement(name = "bearerAuth"))
+    public ResponseEntity<Page<CarGetDetailsDto>> getAll( @RequestParam(defaultValue = "0") Integer
pageNo,
+                                                          @RequestParam(defaultValue = "10") Integer
pageSize,
+                                                          @RequestParam(defaultValue = "carId") String
sortBy){
+        Page<CarGetDetailsDto> result = carService.getAll(pageNo, pageSize, sortBy);
+        return ResponseEntity.ok(result);
    }
    @Operation(description = "Get User By Id Service", summary = "GetById", security =
@SecurityRequirement(name = "bearerAuth"))
    @PostMapping("/add")

    public ResponseEntity<CarGetDetailsDto> updateCarBookingStatus(@PathVariable Integer id,
@PathVariable Boolean isBooked){
        return new ResponseEntity<>(carService.updateCarBookingStatus(id, isBooked), HttpStatus.OK);
    }

}
```

Figure 16

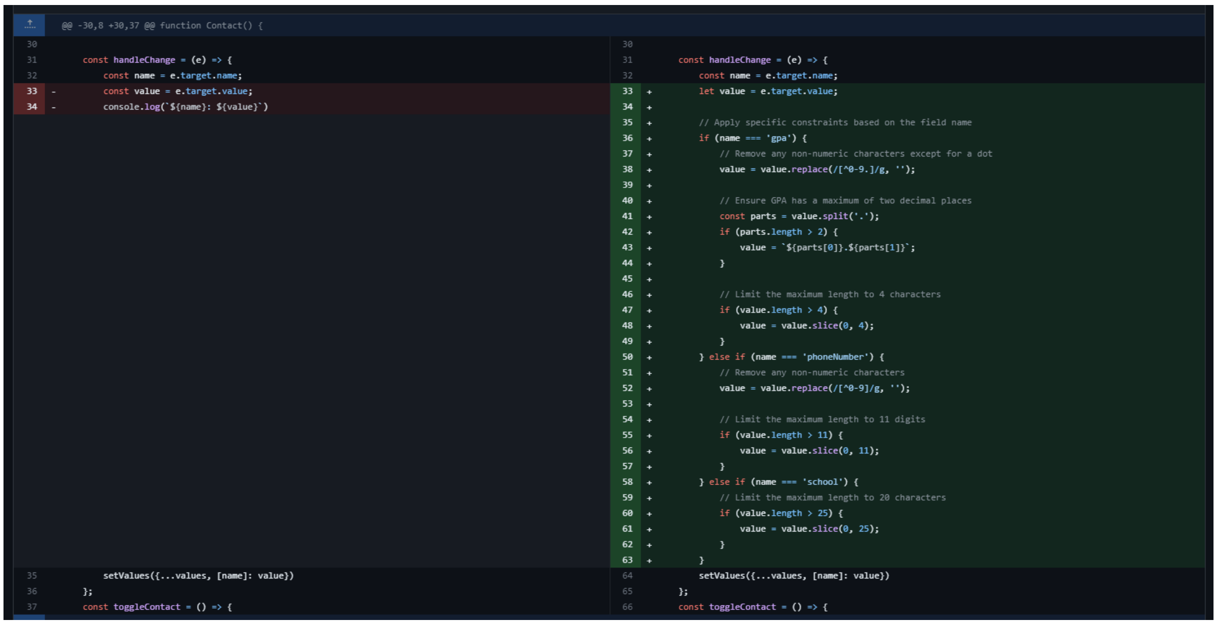
In Figure 17, a switch has been made from datetime to local time in order to increase the code quality.



```
5 Backend/src/main/java/com/comp3102/backend/core/StartupConfig.java
@@ -22,6 +22,7 @@
 import org.springframework.security.crypto.password.PasswordEncoder;
 import org.springframework.stereotype.Component;

+import java.time.LocalDateTime;
 import java.util.Arrays;
 import java.util.Date;
 import java.util.List;
@@ -63,7 +64,7 @@ public void run(String... args) {
                .email("user@user.com")
                .password(passwordEncoder.encode("1"))
                .driverLicense("A1")
-               .birthDate(new Date())
+               .birthDate(LocalDateTime.now())
                .identityNumber("35487415")
                .phoneNumber("987123747")
                .role(Role.USER)
@@ -80,7 +81,7 @@ public void run(String... args) {
                .email("test@test.com")
                .password(passwordEncoder.encode("1"))
                .driverLicense("A")
-               .birthDate(new Date())
+               .birthDate(LocalDateTime.now())
                .identityNumber("35621123456")
                .phoneNumber("5343947775")
                .role(Role.ADMIN)
```

Figure 17

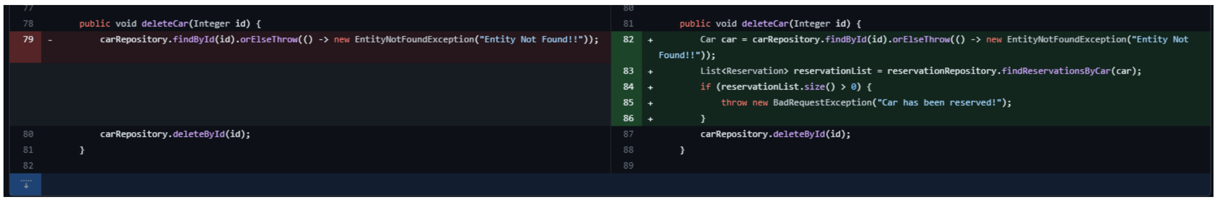
In Figure 18, a control system has been created to prevent errors that may occur in the system, taking into account the errors that the user may make.



```
@@ -30,8 +30,37 @@ function Contact() {

    const handleChange = (e) => {
        const name = e.target.name;
-       const value = e.target.value;
-       console.log(`${name}: ${value}`)
+       let value = e.target.value;
+
+       // Apply specific constraints based on the field name
+       if (name === 'gpa') {
+           // Remove any non-numeric characters except for a dot
+           value = value.replace(/[^0-9.]/g, '');
+
+           // Ensure GPA has a maximum of two decimal places
+           const parts = value.split('.');
+           if (parts.length > 2) {
+               value = `${parts[0]}.${parts[1]}`;
+           }
+
+           // Limit the maximum length to 4 characters
+           if (value.length > 4) {
+               value = value.slice(0, 4);
+           }
+       } else if (name === 'phoneNumber') {
+           // Remove any non-numeric characters
+           value = value.replace(/[^0-9]/g, '');
+
+           // Limit the maximum length to 11 digits
+           if (value.length > 11) {
+               value = value.slice(0, 11);
+           }
+       } else if (name === 'school') {
+           // Limit the maximum length to 20 characters
+           if (value.length > 25) {
+               value = value.slice(0, 25);
+           }
+       }
        setValues({...values, [name]: value})
    };
    const toggleContact = () => {
```

Figure 18

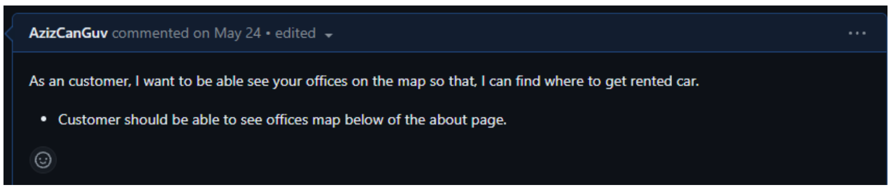
In Figure 19, the feature of checking whether the deleted car has been reserved by any user has been added. In this way, users' rented cars are prevented from being accidentally deleted.



```
    public void deleteCar(Integer id) {
-       carRepository.findById(id).orElseThrow(() -> new EntityNotFoundException("Entity Not Found!!"));
+       Car car = carRepository.findById(id).orElseThrow(() -> new EntityNotFoundException("Entity Not Found!!"));
+       List<Reservation> reservationList = reservationRepository.findReservationsByCar(car);
+       if (reservationList.size() > 0) {
+           throw new BadRequestException("Car has been reserved!");
+       }
        carRepository.deleteById(id);
    }
```

Figure 19

Issues are tracking units used to keep track of specific issues, tasks or problems in a software project or collaboration platform. Issues can be created for a variety of reasons, including bug reporting, new feature requests, task assignment, discussion, documentation request, and performance improvement. These units are used to identify problems arising in the project, facilitate communication between developers, make recommendations, and monitor the development of the project. The figures below show some of the issues created in the car rental project.

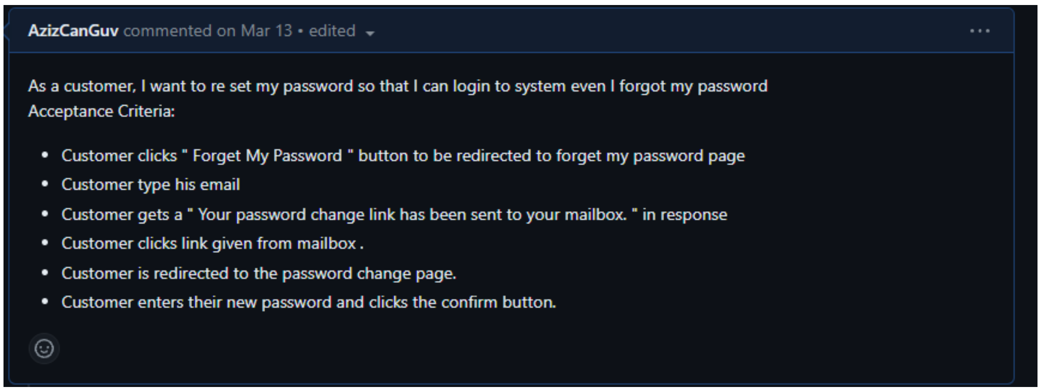


AzizCanGuv commented on May 24 • edited

As an customer, I want to be able see your offices on the map so that, I can find where to get rented car.

- Customer should be able to see offices map below of the about page.

Figure 20

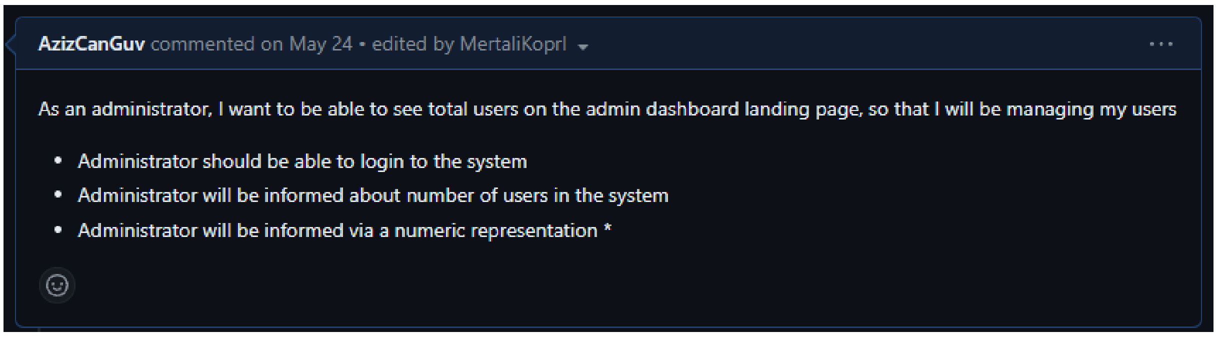


AzizCanGuv commented on Mar 13 • edited

As a customer, I want to re set my password so that I can login to system even I forgot my password
Acceptance Criteria:

- Customer clicks " Forget My Password " button to be redirected to forget my password page
- Customer type his email
- Customer gets a " Your password change link has been sent to your mailbox. " in response
- Customer clicks link given from mailbox .
- Customer is redirected to the password change page.
- Customer enters their new password and clicks the confirm button.

Figure 21

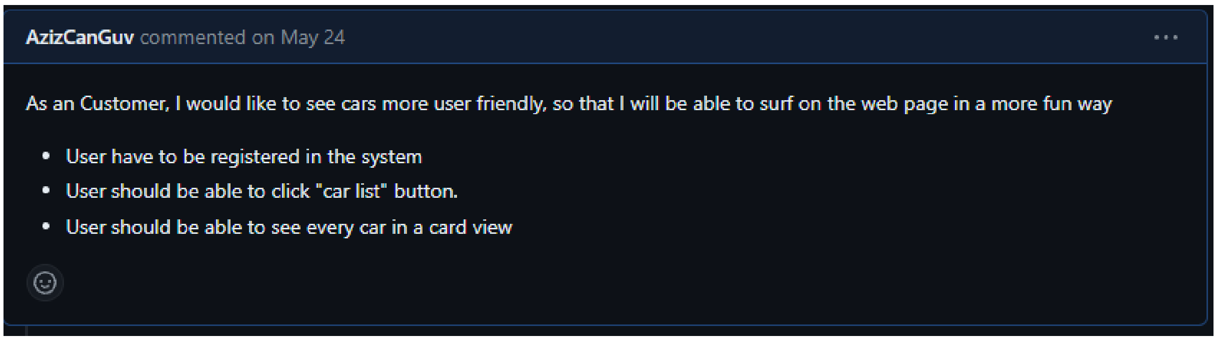


AzizCanGuv commented on May 24 • edited by MertaliKoprl

As an administrator, I want to be able to see total users on the admin dashboard landing page, so that I will be managing my users

- Administrator should be able to login to the system
- Administrator will be informed about number of users in the system
- Administrator will be informed via a numeric representation *

Figure 22

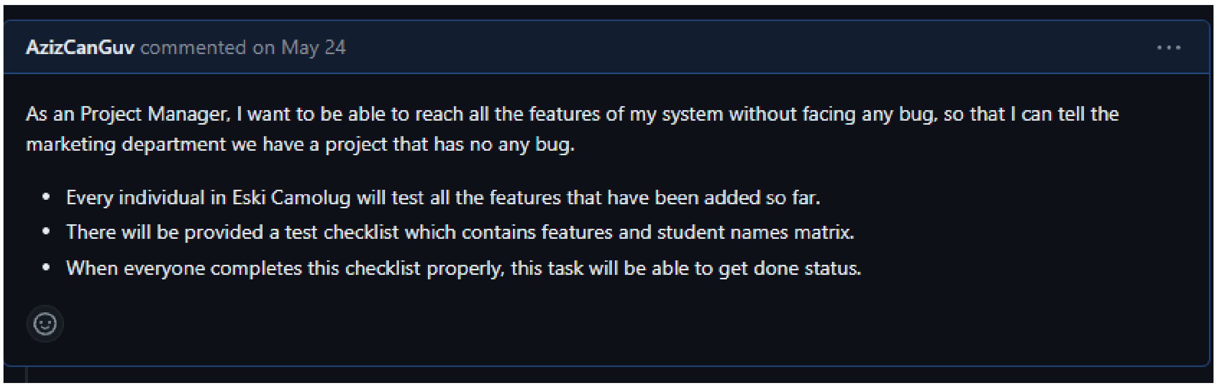


AzizCanGuv commented on May 24

As an Customer, I would like to see cars more user friendly, so that I will be able to surf on the web page in a more fun way

- User have to be registered in the system
- User should be able to click "car list" button.
- User should be able to see every car in a card view

Figure 23

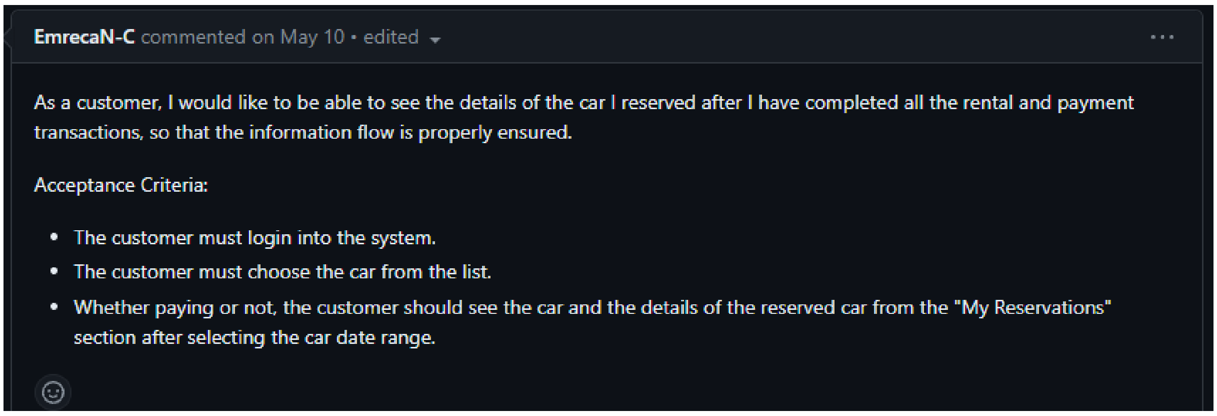


AzizCanGuv commented on May 24

As an Project Manager, I want to be able to reach all the features of my system without facing any bug, so that I can tell the marketing department we have a project that has no any bug.

- Every individual in Eski Camolug will test all the features that have been added so far.
- There will be provided a test checklist which contains features and student names matrix.
- When everyone completes this checklist properly, this task will be able to get done status.

Figure 24



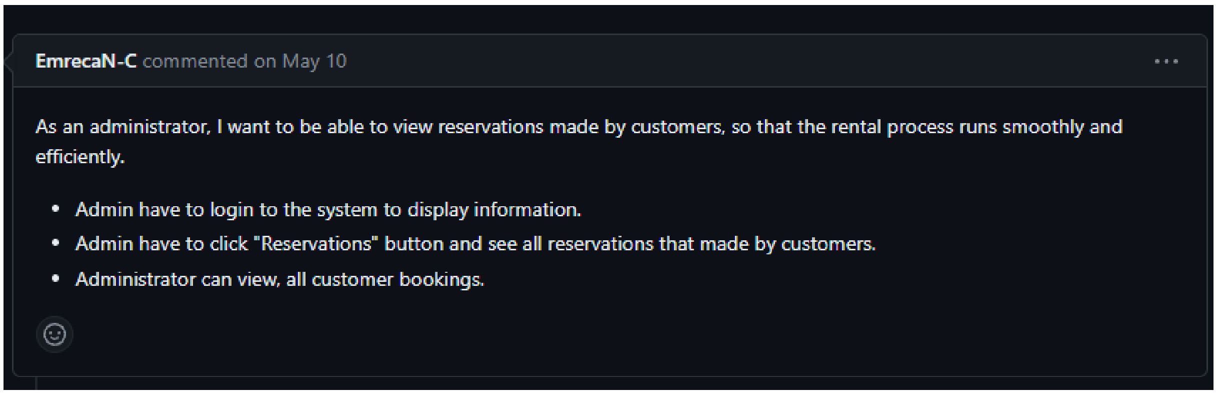
EmrecaN-C commented on May 10 • edited

As a customer, I would like to be able to see the details of the car I reserved after I have completed all the rental and payment transactions, so that the information flow is properly ensured.

Acceptance Criteria:

- The customer must login into the system.
- The customer must choose the car from the list.
- Whether paying or not, the customer should see the car and the details of the reserved car from the "My Reservations" section after selecting the car date range.

Figure 25

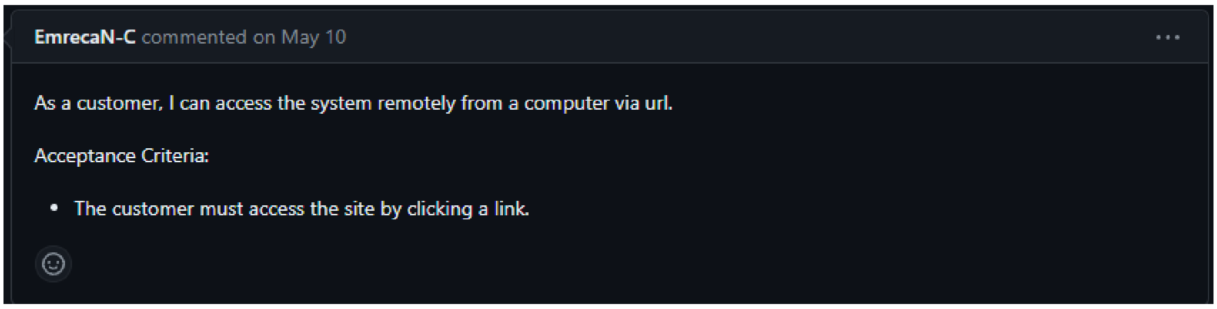


EmrecaN-C commented on May 10

As an administrator, I want to be able to view reservations made by customers, so that the rental process runs smoothly and efficiently.

- Admin have to login to the system to display information.
- Admin have to click "Reservations" button and see all reservations that made by customers.
- Administrator can view, all customer bookings.

Figure 26



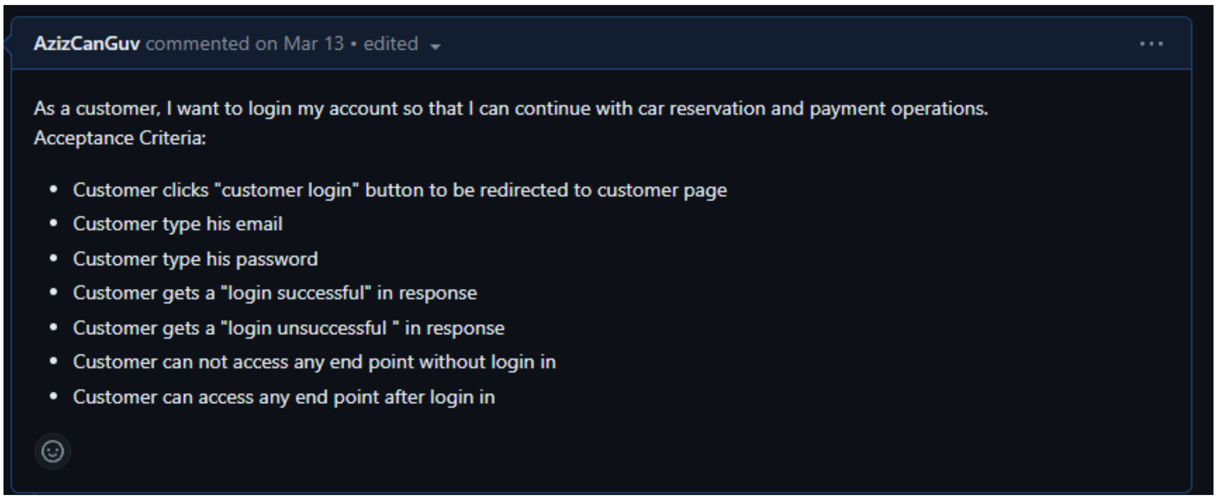
EmrecaN-C commented on May 10

As a customer, I can access the system remotely from a computer via url.

Acceptance Criteria:

- The customer must access the site by clicking a link.

Figure 27

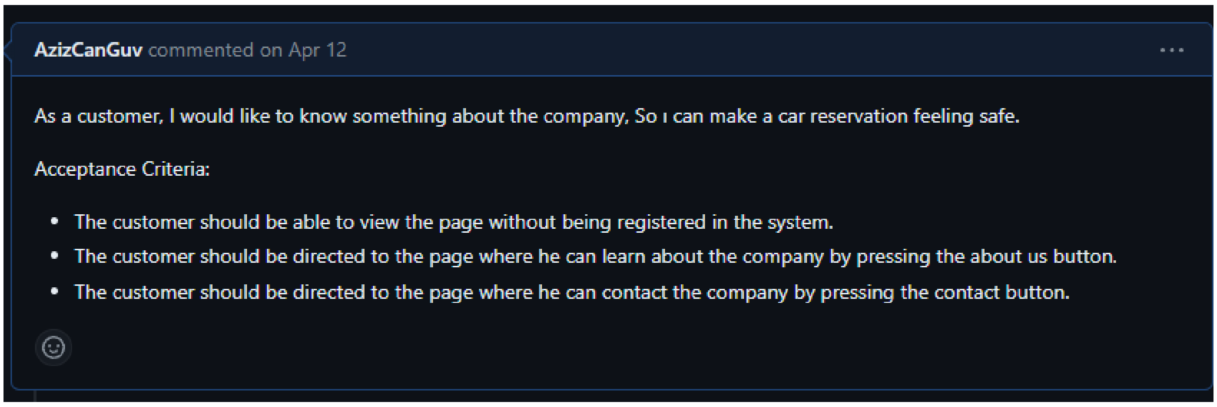


AzizCanGuv commented on Mar 13 • edited

As a customer, I want to login my account so that I can continue with car reservation and payment operations.
Acceptance Criteria:

- Customer clicks "customer login" button to be redirected to customer page
- Customer type his email
- Customer type his password
- Customer gets a "login successful" in response
- Customer gets a "login unsuccessful " in response
- Customer can not access any end point without login in
- Customer can access any end point after login in

Figure 28



AzizCanGuv commented on Apr 12

As a customer, I would like to know something about the company, So ı can make a car reservation feeling safe.

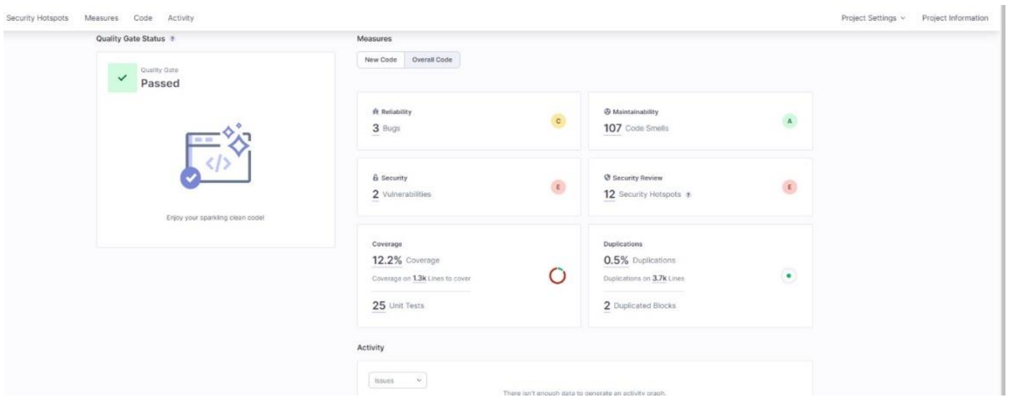
Acceptance Criteria:

- The customer should be able to view the page without being registered in the system.
- The customer should be directed to the page where he can learn about the company by pressing the about us button.
- The customer should be directed to the page where he can contact the company by pressing the contact button.

Figure 29

Statatic Code Analysis

The project has passed the static analysis test. There are 3 reliability errors, 2 security errors, 12 security revision errors and 107 maintainability errors in the project. It is seen that the majority of the 107 errors were made on the front-end side. In addition, 0.5% duplication has occurred.

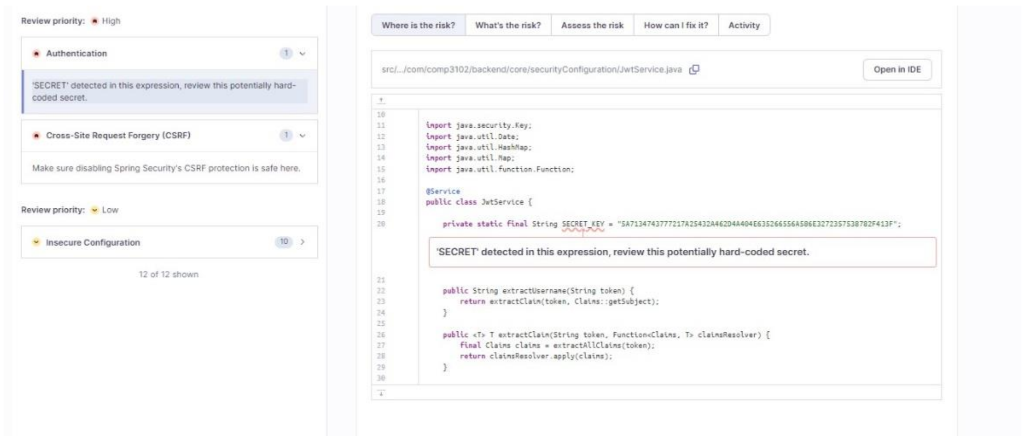


Quality Gate
Passed

Reliability: 3 Bugs (C)
Maintainability: 107 Code Smells (A)
Security: 2 Vulnerabilities (E)
Security Review: 12 Security Hotspots (E)
Coverage: 12.2% Coverage — Coverage on 1.3k Lines to cover — 25 Unit Tests
Duplications: 0.5% Duplications — Duplications on 3.7k Lines — 2 Duplicated Blocks

There isn't enough data to generate an activity graph.

To discuss the security errors in detail, as seen in the picture below, the Secret key is not hidden and it is clearly written. This is not a suitable situation and is a problem that will cause security problems.



Review priority: High

Authentication 1

'SECRET' detected in this expression, review this potentially hard-coded secret.

Cross-Site Request Forgery (CSRF) 1

Make sure disabling Spring Security's CSRF protection is safe here.

Review priority: Low

Insecure Configuration 10

12 of 12 shown

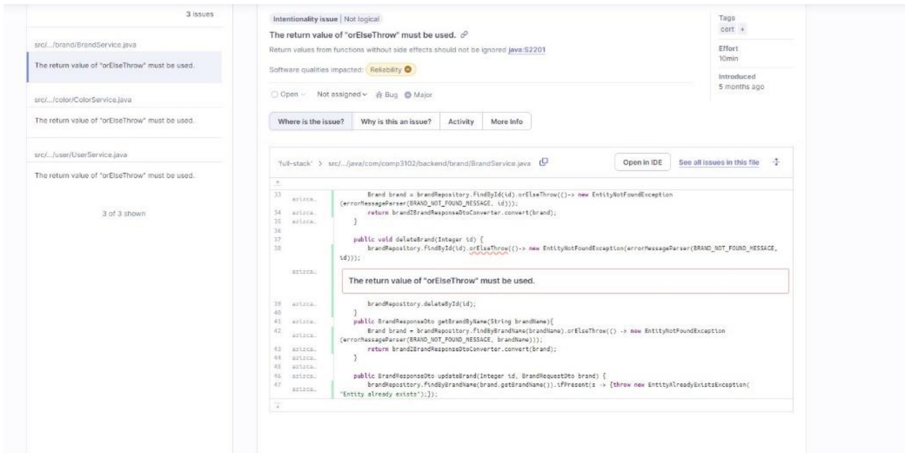
src/.../com/comp3102/backend/core/securityConfiguration/JwtService.java

```java
import java.security.Key;
import java.util.Date;
import java.util.HashMap;
import java.util.Map;
import java.util.function.Function;

@Service
public class JwtService {

    private static final String SECRET_KEY = "5A7134743777217A25432A462D4A404E635266556A586E3272357538782F413F";
```

'SECRET' detected in this expression, review this potentially hard-coded secret.

```java

    public String extractUsername(String token) {
        return extractClaim(token, Claims::getSubject);
    }

    public <T> T extractClaim(String token, Function<Claims, T> claimsResolver) {
        final Claims claims = extractAllClaims(token);
        return claimsResolver.apply(claims);
    }

```

Sonarqube has shown 3 orElseThrow() errors as reliability errors. By ignoring the potential absence of a return value, there is a risk of encountering errors or unexpected outcomes further along in the code. This assumption of a constant presence of a value may lead to issues in how the program operates.



3 issues

src/.../brand/BrandService.java
The return value of "orElseThrow" must be used.

src/.../color/ColorService.java
The return value of "orElseThrow" must be used.

src/.../user/UserService.java
The return value of "orElseThrow" must be used.

3 of 3 shown

Intentionality issue | Not logical

The return value of "orElseThrow" must be used.

Return values from functions without side effects should not be ignored java:S2201

Software qualities impacted: Reliability

Open Not assigned Bug Major

Tags: cert
Effort: 10min
Introduced: 5 months ago

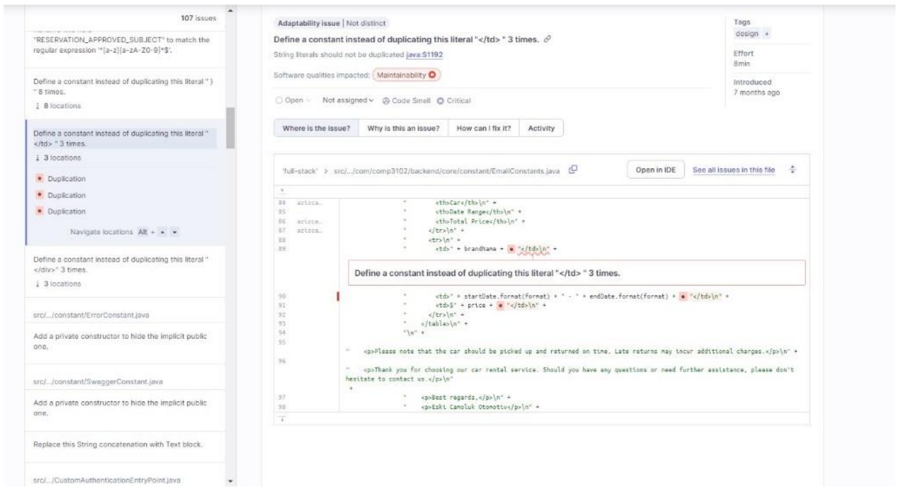
'full-stack' > src/.../java/com/comp3102/backend/brand/BrandService.java

```java
        Brand brand = brandRepository.findById(id).orElseThrow(()-> new EntityNotFoundException
(errorMessageParser(BRAND_NOT_FOUND_MESSAGE, id)));
        return brand2BrandResponseDtoConverter.convert(brand);
    }

    public void deleteBrand(Integer id) {
        brandRepository.findById(id).orElseThrow(()-> new EntityNotFoundException(errorMessageParser(BRAND_NOT_FOUND_MESSAGE,
id)));
```

The return value of "orElseThrow" must be used.

```java
        brandRepository.deleteById(id);
    }
    public BrandResponseDto getBrandByName(String brandName){
        Brand brand = brandRepository.findByBrandName(brandName).orElseThrow(() -> new EntityNotFoundException
(errorMessageParser(BRAND_NOT_FOUND_MESSAGE, brandName)));
        return brand2BrandResponseDtoConverter.convert(brand);
    }

    public BrandResponseDto updateBrand(Integer id, BrandRequestDto brand) {
        brandRepository.findByBrandName(brand.getBrandName()).ifPresent(s -> {throw new EntityAlreadyExistsException(
"Entity already exists");});
```

107 The majority of maintainability errors are in the front-end part, an example is shown in the image below. It is important to reduce these errors so that the code can run more stable and error-free.



107 issues

"RESERVATION_APPROVED_SUBJECT" to match the regular expression '^[a-z][a-zA-Z0-9]*$'.

Define a constant instead of duplicating this literal ") " 8 times.
8 locations

Define a constant instead of duplicating this literal " </td> " 3 times.
3 locations
Duplication
Duplication
Duplication
Navigate locations Alt + ▲ ▼

Define a constant instead of duplicating this literal " </div>" 3 times.
3 locations

src/.../constant/ErrorConstant.java
Add a private constructor to hide the implicit public one.

src/.../constant/SwaggerConstant.java
Add a private constructor to hide the implicit public one.

Replace this String concatenation with Text block.

src/.../CustomAuthenticationEntryPoint.java

Adaptability issue | Not distinct

Define a constant instead of duplicating this literal "</td> " 3 times.

String literals should not be duplicated java:S1192

Software qualities impacted: Maintainability

Open · Not assigned · Code Smell · Critical

Tags: design
Effort: 8min
Introduced: 7 months ago

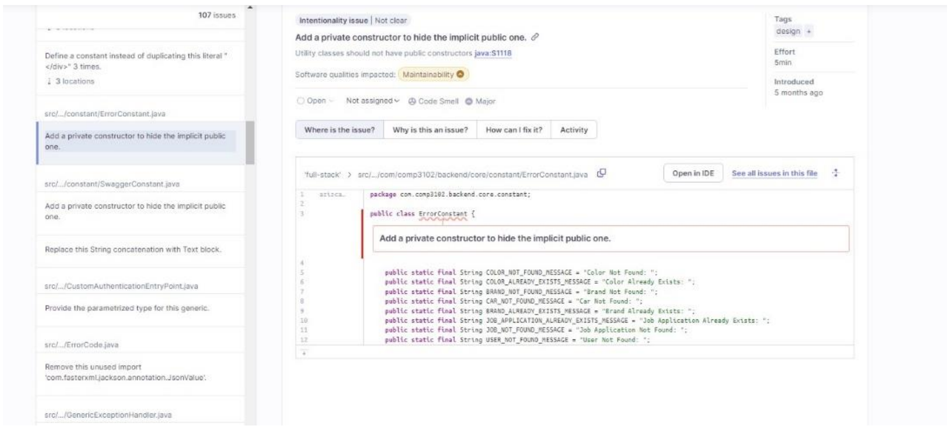
'full-stack' > src/.../com/comp3102/backend/core/constant/EmailConstants.java

```
                    "        <th>Car</th>\n" +
                    "        <th>Date Range</th>\n" +
                    "        <th>Total Price</th>\n" +
                    "      </tr>\n" +
                    "      <tr>\n" +
                    "        <td>" + brandName + "</td>\n" +
```

Define a constant instead of duplicating this literal "</td> " 3 times.

```
                    "        <td>" + startDate.format(format) + " - " + endDate.format(format) + "</td>\n" +
                    "        <td>$" + price + "</td>\n" +
                    "      </tr>\n" +
                    "    </table>\n" +
                    "\n" +
"    <p>Please note that the car should be picked up and returned on time. Late returns may incur additional charges.</p>\n" +
"    <p>Thank you for choosing our car rental service. Should you have any questions or need further assistance, please don't hesitate to contact us.</p>\n"
 +
                    "    <p>Best regards,</p>\n" +
                    "    <p>Eski Camoluk Otomotiv</p>\n" +
```

In addition, it is recommended by SonarQube that some public methods be private and is seen as a maintainability error.



107 issues

Define a constant instead of duplicating this literal " </div>" 3 times.
3 locations

src/.../constant/ErrorConstant.java
Add a private constructor to hide the implicit public one.

src/.../constant/SwaggerConstant.java
Add a private constructor to hide the implicit public one.

Replace this String concatenation with Text block.

src/.../CustomAuthenticationEntryPoint.java
Provide the parametrized type for this generic.

src/.../ErrorCode.java
Remove this unused import 'com.fasterxml.jackson.annotation.JsonValue'.

src/.../GenericExceptionHandler.java

Intentionality issue | Not clear

Add a private constructor to hide the implicit public one.

Utility classes should not have public constructors java:S1118

Software qualities impacted: Maintainability

Open · Not assigned · Code Smell · Major

Tags: design
Effort: 5min
Introduced: 5 months ago

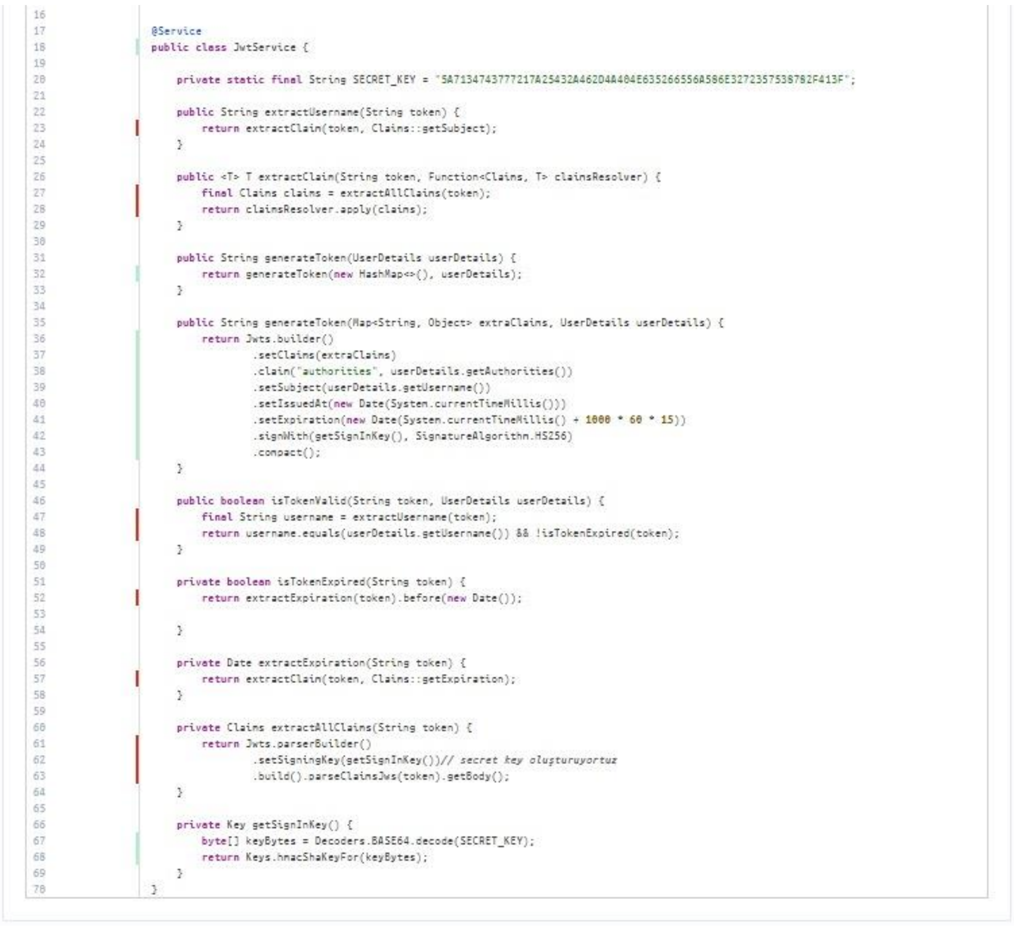
'full-stack' > src/.../com/comp3102/backend/core/constant/ErrorConstant.java

```
package com.comp3102.backend.core.constant;

public class ErrorConstant {
```

Add a private constructor to hide the implicit public one.

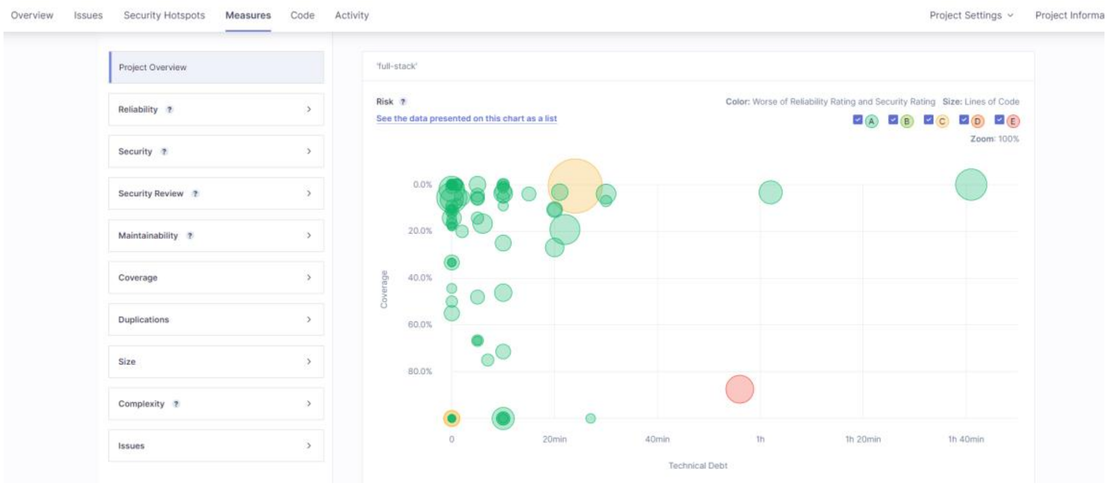
```

    public static final String COLOR_NOT_FOUND_MESSAGE = "Color Not Found: ";
    public static final String COLOR_ALREADY_EXISTS_MESSAGE = "Color Already Exists: ";
    public static final String BRAND_NOT_FOUND_MESSAGE = "Brand Not Found: ";
    public static final String CAR_NOT_FOUND_MESSAGE = "Car Not Found: ";
    public static final String BRAND_ALREADY_EXISTS_MESSAGE = "Brand Already Exists: ";
    public static final String JOB_APPLICATION_ALREADY_EXISTS_MESSAGE = "Job Application Already Exists: ";
    public static final String JOB_NOT_FOUND_MESSAGE = "Job Application Not Found: ";
    public static final String USER_NOT_FOUND_MESSAGE = "User Not Found: ";
```

In the Uncovered code section, lines of code written but not used are discussed. According to SonarQube analysis, 3.7k lines written were not used.



```java
@Service
public class JwtService {

    private static final String SECRET_KEY = "5A7134743777217A25432A462D4A404E635266556A586E3272357538782F413F";

    public String extractUsername(String token) {
        return extractClaim(token, Claims::getSubject);
    }

    public <T> T extractClaim(String token, Function<Claims, T> claimsResolver) {
        final Claims claims = extractAllClaims(token);
        return claimsResolver.apply(claims);
    }

    public String generateToken(UserDetails userDetails) {
        return generateToken(new HashMap<>(), userDetails);
    }

    public String generateToken(Map<String, Object> extraClaims, UserDetails userDetails) {
        return Jwts.builder()
                .setClaims(extraClaims)
                .claim("authorities", userDetails.getAuthorities())
                .setSubject(userDetails.getUsername())
                .setIssuedAt(new Date(System.currentTimeMillis()))
                .setExpiration(new Date(System.currentTimeMillis() + 1000 * 60 * 15))
                .signWith(getSignInKey(), SignatureAlgorithm.HS256)
                .compact();
    }

    public boolean isTokenValid(String token, UserDetails userDetails) {
        final String username = extractUsername(token);
        return username.equals(userDetails.getUsername()) && !isTokenExpired(token);
    }

    private boolean isTokenExpired(String token) {
        return extractExpiration(token).before(new Date());

    }

    private Date extractExpiration(String token) {
        return extractClaim(token, Claims::getExpiration);
    }

    private Claims extractAllClaims(String token) {
        return Jwts.parserBuilder()
                .setSigningKey(getSignInKey())// secret key oluşturuyortuz
                .build().parseClaimsJws(token).getBody();
    }

    private Key getSignInKey() {
        byte[] keyBytes = Decoders.BASE64.decode(SECRET_KEY);
        return Keys.hmacShaKeyFor(keyBytes);
    }
}
```

As seen in the picture below, although the code quality is generally good, there is a security error that needs to be fixed.


Overview
Issues
Security Hotspots
Measures
Code
Activity
Project Settings
Project Informa
Project Overview
Reliability
Security
Security Review
Maintainability
Coverage
Duplications
Size
Complexity
Issues
'full-stack'
Risk
See the data presented on this chart as a list
Color: Worse of Reliability Rating and Security Rating Size: Lines of Code
A B C D E
Zoom: 100%
Coverage
20min
40min
1h
1h 20min
1h 40min
Technical Debt