

Lab 2:

Requirements improvement:

Functional Requirements:

Feature 1:

Priority: High

As a user, I would like to have access to account management functionality, which includes creating, editing, viewing, and deleting an account that I only have access to.

I would like to be able to **create a new user account**:

by providing the following information: Username, Password, First name and Last Name. Upon successful creation, the system shall generate a unique user ID and store the account information in the system's database.

As a user I would like also to be able to **edit my information** and I would like to have the ability to **view my account information** as well.

I also would like to be able to **delete my account**:

When a user account is deleted, the system shall:

- Prompt for confirmation before proceeding with the deletion.
- Remove the user account from the system's database. If the deleted user is currently logged in, the system shall log them out and prevent further access.

Exception Handling:

- If a non-authorized user attempts to access the account management feature, the system shall deny access and display an appropriate error message.
- When creating or editing a user account, the system shall validate the input data to ensure it adheres to predefined criteria. In case of invalid input, an error message should be displayed, and the user should be prompted to correct the data.
- When a user account is deleted, the system shall prompt the administrator for confirmation to prevent accidental deletions.

Feature 2:

Priority: High

As a user, I would like to have access to a ticket reservation page that allows me to search for tickets using two methods:

General Ticket Search,

which includes the ability to scroll through the page and view available tickets without specifying a particular time slot or destination. This provides a general overview of available options

and also a Specified Ticket Search,

which means that I want the option to enter specific search criteria, including a time slot and destination, to narrow down my ticket search and find tickets that match my preferences.

Exception Handling:

- If a non-authorized user attempts to access the ticket reservation system, the system shall deny access and display an appropriate error message.

Feature 3:

Priority: High

As a user, I would like to have the capability to pay for my reservation in a flexible manner. Which means that several payment options are available:

- **Standard Payment:** I should be able to make a standard payment for my reservation with no promotions applied.
- **Payment Based on Hours:** I want the option to pay for my reservation based on the number of hours it covers. The system should calculate the total cost according to the selected time frame.
- **Payment Based on Age:** I would like the ability to make payments based on my age, with age-based discounts or surcharges applied to the reservation cost.

- **Pay via Gift Voucher:** I want the ability to apply a gift voucher or gift card to cover part or all of the reservation cost. The system should validate the gift voucher and deduct the appropriate amount from the total payment.

Exception Handling:

- The system shall validate the payment information to ensure accuracy and prevent errors. In case of invalid payment details or insufficient funds, the system shall display an error message and prompt the user to correct the payment information.
- When applying a gift voucher, the system should verify the voucher's validity. If the gift voucher is expired or incorrect, the system should inform the user and prevent its use for payment.

Feature 4:

Priority: High

As a user, I would like to have the ability to manage reservations effectively. Which meant that I want to:

Make a reservation:

I want the capability to create a reservation for an event, activity, or service. When making a reservation, I should be able to provide details such as date, time, location, and any additional preferences or requirements.

Also I would like to **add One or More Tickets:**

I should be able to add one or more tickets to my reservation, specifying the type and quantity of tickets needed. This allows me to include multiple attendees or participants in a single reservation.

I would like to be able to **Cancel a Reservation** as well:

I want the option to cancel a reservation if my plans change or if I no longer wish to attend the event or use the service. When canceling a reservation, the system should provide a confirmation prompt to ensure the cancellation is intentional.

Exception Handling:

- The system should check for conflicts with existing reservations to avoid double-booking. If a conflict is detected, the system should notify the user and suggest alternative options.
- When canceling a reservation, the system should prompt the user for confirmation to prevent accidental cancellations. If the user confirms the cancellation, the system should release the reserved tickets and update the reservation status accordingly.

Feature 5:

Priority: Medium

As a user, I would like to have access to my trip history within the application. It means that I would like to have the ability to view both upcoming and past trips, providing me with a comprehensive history of my interactions with the application.

Upcoming Trips:

- I want to be able to view a list of my upcoming trips. This should include details such as the trip date, time, destination, and any relevant information about the trip. Upcoming trips should be presented in a clear and organised manner, allowing me to plan accordingly.

Viewing Old Trips:

- I also want to have access to my past trips, allowing me to review my trip history. This should include details about past trips, such as the date, time, destination, and any other relevant trip-related information. Past trips should be easily accessible for reference and review.

Exception Handling:

- If a non-authorized user attempts to access the ticket reservation system, the system shall deny access and display an appropriate error message.

Feature 6:

Priority: High

As an administrator, I need to be able to set up and manage access roles within the system, allowing for different levels of access and permissions for users. The system should support the following access roles:

- No roles:
 - Every user will have access to its own account regarding its role{CLIENT or ADMIN}.
- **'CLIENT'** Role:
 - Clients with this role should have the ability: to view and purchase tickets.
- **'ADMINISTRATOR'** Role:
 - People with this role will have access to the system information.
- **'ADMIN'** Role:
 - People with this role should have the following permissions:
 - Add new tickets to the system.
 - Edit ticket data, including details and pricing.
 - Modify operating hours for events or services.
 - Delete tickets from the system.

Exception Handling:

- If a non-authorized user attempts to access the ticket reservation system, the system shall deny access and display an appropriate error message.
- If an user with role different from 'CLIENT' tries to access the edit page of the 'ADMIN's, the system shall deny access and display an appropriate error message.
- If a 'CLIENT' or 'ADMIN' tries to enter an 'ADMINISTRATOR's' page, the system shall deny access and display an appropriate error message.

Feature 7:

Priority: High

As an administrator, I require the system to support various payment methods and currencies to facilitate transactions. The payment support feature should include:

Credit Card Support:

- Accept VISA credit card payments for reservations and purchases.

Debit Card Support:

- Accept MASTERCARD debit card payments for reservations and purchases.

Multiple Currencies:

- The system should support three different currencies for transactions: EUR, USD, and BGN, allowing users to make payments in their preferred currency.

Voucher Selling:

- 'ADMIN's should have the capability to add and redeem vouchers or gift cards for use in the system. It includes the ability to verify and deduct the voucher value during transactions.

Exception Handling:

- The system should validate payment details to ensure accuracy and prevent errors. In case of invalid payment information or insufficient funds, the system should display an error message and prompt the administrator to rectify the payment.
- When selling or redeeming vouchers, the system should verify the validity of vouchers. If a voucher is expired or incorrect, the system should inform the administrator and prevent its use for transactions.

Feature 8:

Priority: Medium

As an administrator, I need the system to support multiple languages to cater to a diverse user base and be able to add new languages based on client requests. The language support feature should include:

Supported Languages:

- Initially, the system should support three languages: English (EN), German (DE), and Bulgarian (BG).

Add Another Language:

- The system should provide the flexibility for administrators to add additional languages upon client request. This should include language translation for all user-facing interfaces and content.

Exception Handling:

- If a non-authorized user attempts to access the ticket reservation system, the system shall deny access and display an appropriate error message.

Non-Functional Requirements:

9. Security

User Authentication:

- The system should implement a robust user authentication mechanism, requiring users to provide valid credentials (e.g., username and password) to access the system. Strong password policies and multi-factor authentication should be supported to enhance security.

Role Verification:

- After successful authentication, the system must verify the user's assigned role. Role-based access control (RBAC) should be employed to ensure that users can only access features and data aligned with their authorised roles (e.g., CLIENT, ADMIN or ADMINISTRATOR).

10: Performance

Response Time:

- The system should ensure that responses to user requests do not exceed a maximum response time of 2 seconds. This responsiveness is crucial for providing a smooth and efficient user experience.

11: Reliability

Continuous Availability:

- The application should be available 24 hours a day, 7 days a week, without scheduled downtime, to meet the needs of users across different time zones and ensure uninterrupted service.

12: Localisation

Time Zones Support:

- The development team should support various time zones, including Germany, Bulgaria, the UK, and France, to accommodate users worldwide.

Currency Support:

- The application should also support different currencies, such as Euro, Dollar, and Bulgarian Lev, to facilitate transactions in various regions.

13: Analytics

Response Generation:

- The system should generate responses to the user via various methods, including file generation (e.g., ticket purchase confirmation), status updates (e.g., changes to ticket times), and a combination of file and status updates (e.g., reservation modifications).

Promotion Notifications:

- The system should provide users with notifications of promotions and sales to keep them informed of special offers and discounts.

14: Performance

System Monitoring:

- The development team should employ a system monitoring tool to continuously monitor the system's performance and health.

Data Flow Control:

- The team should exercise control over the data flow within the system to optimise efficiency and resource utilisation.

Problem Alerts:

- Developers should receive alerts in the event of system issues or anomalies, enabling them to address problems promptly.

User Notifications:

- In case of system issues affecting users, notifications should be generated and delivered to inform users of the problem and suggest possible actions.

15: Recovery**Data Backup Support:**

- The system should support data backup mechanisms to safeguard user data and system integrity in the event of data loss or system failures.

16: Compatibility**Cross-Browser Compatibility:**

- The developers should ensure that the system is responsive and functions correctly when accessed from different web browsers, promoting a seamless user experience.

Responsive Design:

- The application should exhibit responsive design characteristics, ensuring that it adapts gracefully to screens with various resolutions, preventing content overlapping and ensuring readability.

17: Integration**Payment System Integration:**

- The development team should integrate external payment systems, such as PayPal for example, to facilitate secure and convenient payment processing.

Security System Integration:

- Integration of robust security systems, like Keycloak, should be implemented to enhance user data protection and access control.

18: Compliance**Industry Standards Adherence:**

- The development team should adapt the application to conform to industry standards and regulations, ensuring compliance with best practices and legal requirements.

19: Usability**User-Friendly Interface:**

- The developers should implement a user-friendly interface to enhance the ease of use and overall user experience.

Accessibility Features:

- The development team should incorporate accessibility features to accommodate individuals with disabilities, such as voice control for blind users, enhancing inclusivity and usability.

=====

Task 2:**Self-code review:****a.) Implement the calculation of the price according to dynamic pricing rules.****Branch: main**

<https://github.com/DesislavaaSimeonova/RailwayTicketingPortal/tree/main>

b.) Create your own code review checklist


















Example taken from: <https://www.java-success.com/30-java-code-review-checklist-items/>

My checklist:

1. DRY principle
2. SOLID principle
3. KISS principle
4. Not so much comments
5. Descriptive comments if needed
6. Meaningful method names

7. Meaningful variable names
8. Focused classes(doen one thing)
9. Focused functions(doen one thing 20~25 lines)
10. Reduced, better escaped, code duplication
11. Small scope for variables
12. No commented code
13. Minimised access to classes, variables, methods, packages
14. Right data types
15. Handle null pointers
16. No sensitive data is public
17. Clear documentation
18. Handle exceptions(escape publishing paths and stack trace)
19. Follow security guidelines
20. Write proper tests
21. Don't forget non-functional requirements by feature implementation

c.) Execute self-code review - 28.10.2023

1. DRY principle: 
2. SOLID principle: 
3. KISS principle: 
4. Not so much comments: 
5. Descriptive comments if needed: 
6. Meaningful method names: 
7. Meaningful variable names: 
8. Focused classes(doen one thing): 
9. Focused functions(doen one thing 20~25 lines): 
10. Reduced, better escaped, code duplication: 
11. Small scope for variables: 
12. No commented code: unused liquidate change sets should be deleted
13. Minimised access to classes, variables, methods, packages: 
14. Right data types: 
15. Handle null pointers: 
16. No sensitive data is public: 
17. Clear documentation: no documentation, for now
18. Handle exceptions(escape publishing paths and stack trace): 
19. Follow security guidelines: 
20. Write proper tests: no tests currently, only pricing calculation imply
21. Don't forget non-functional requirements by feature implementation: 