

COMP1610 – Programming Enterprise Components Report

Your name	Yauhen Bichel	Your Student ID	001185491
-----------	---------------	-----------------	-----------

Other members in the Group:	Ivan Hulyev	Their IDs	001181952
	Yahya Chahine		001178039
	Nurkaiyr Yedige		001196862

1. BRIEF STATEMENT OF FEATURES YOU HAVE IMPLEMENTED

(THIS SECTION SHOULD BE THE SAME FOR ALL MEMBERS OF THE GROUP)

Feature	Status	Your Comments
Functionality A	Fully completed <input checked="" type="checkbox"/> Partially completed <input type="checkbox"/> Having bugs/Not working <input type="checkbox"/> Not implemented <input type="checkbox"/>	Designed and implemented
Functionality B	Fully completed <input checked="" type="checkbox"/> Partially completed <input type="checkbox"/> Having bugs/Not working <input type="checkbox"/> Not implemented <input type="checkbox"/>	Designed and implemented
Functionality C	Fully completed <input checked="" type="checkbox"/> Partially completed <input type="checkbox"/> Having bugs/Not working <input type="checkbox"/> Not implemented <input type="checkbox"/>	Designed and implemented
Functionality D	Fully completed <input type="checkbox"/> Partially completed <input checked="" type="checkbox"/> Having bugs/Not working <input type="checkbox"/> Not implemented <input type="checkbox"/>	Server side: Constraints are implemented and applied with validation of request Client side: not implemented
Functionality E	Fully completed <input checked="" type="checkbox"/> Partially completed <input type="checkbox"/> Having bugs/Not working <input type="checkbox"/> Not implemented <input type="checkbox"/>	RESTful service is designed and implemented Desktop app is implemented
Functionality F	Fully completed <input checked="" type="checkbox"/> Partially completed <input type="checkbox"/> Having bugs/Not working <input type="checkbox"/> Not implemented <input type="checkbox"/>	Message system and mail service are implemented. Created gmail mailbox as holiday booking system mailbox

Functionality G	Fully completed <input type="checkbox"/> Partially completed <input checked="" type="checkbox"/> Having bugs/Not working <input type="checkbox"/> Not implemented <input type="checkbox"/>	Prioritization is designed and implemented.
------------------------	---	---

Link to recorded video (if you record your application before submitting the report)

The recorded video will demonstrate the implemented product as group. It is roughly 10 minutes. The video you upload on Panopto should follow this naming convention – “Surname1-Surname2- etc.” You should include the surnames of all group members of all group members in the video name, so we can easily identify the group.

<https://gre.cloud.panopto.eu/Panopto/Pages/Viewer.aspx?id=00e89ecb-553f-4bcf-b795-ae83007e73b6>

2. PART A (COMPLETED AS A GROUP) (10%)

- a. **Design Documentation:** This section should contain the design documentation that you created as a group, which should contain an Entity Relationship Diagram outlining the database structure and an architecture diagram which shows the overall setup of the system.
- b. **Screenshots:** Screen shots demonstrating each of the features that you have implemented. Give captions or annotations to explain which features are being demonstrated.
- c. **Evaluation (approx. 600 words):** An evaluation of the evolution of your application. You should discuss any problems you had during implementation. You should be critical (both positive and negative) of your implementation. Be prepared to suggest alternatives. Discuss how your final implementation could be improved

Design Documentation

The Holiday System application contains the following functionality

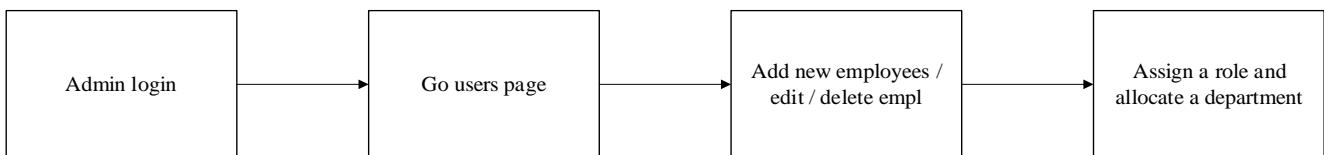


Figure 1 Functionality A: admin adds new employee



Figure 2 Functionality B: employee submit a holiday request

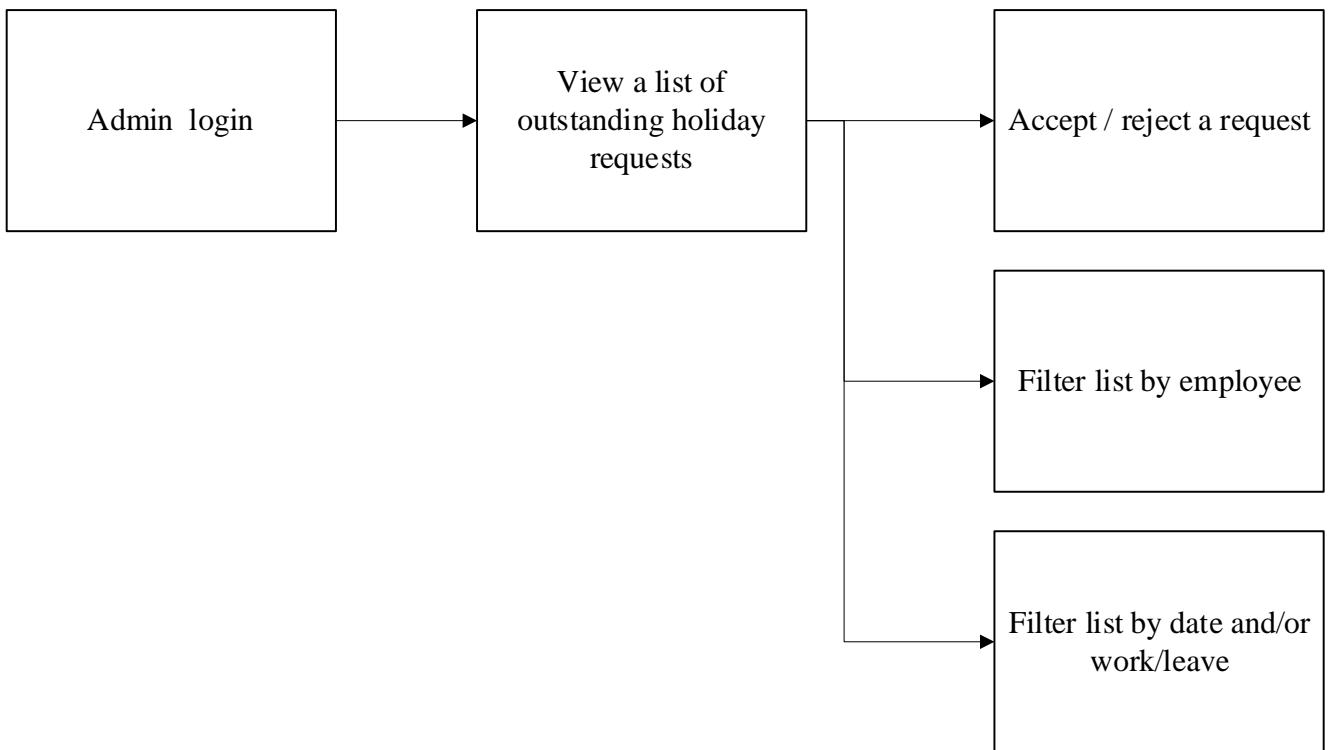


Figure 3 Functionality C: admin views requests and accept/reject them

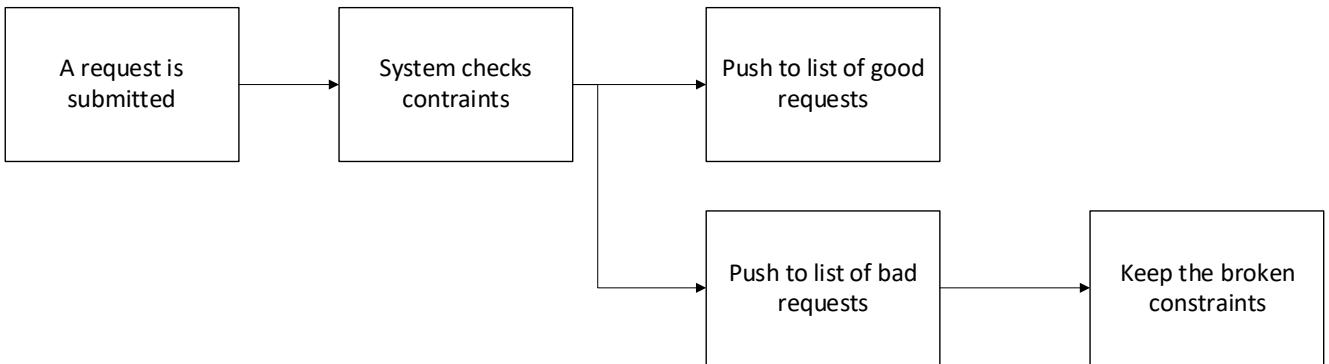


Figure 4 Functionality D: a system validates a submitted request

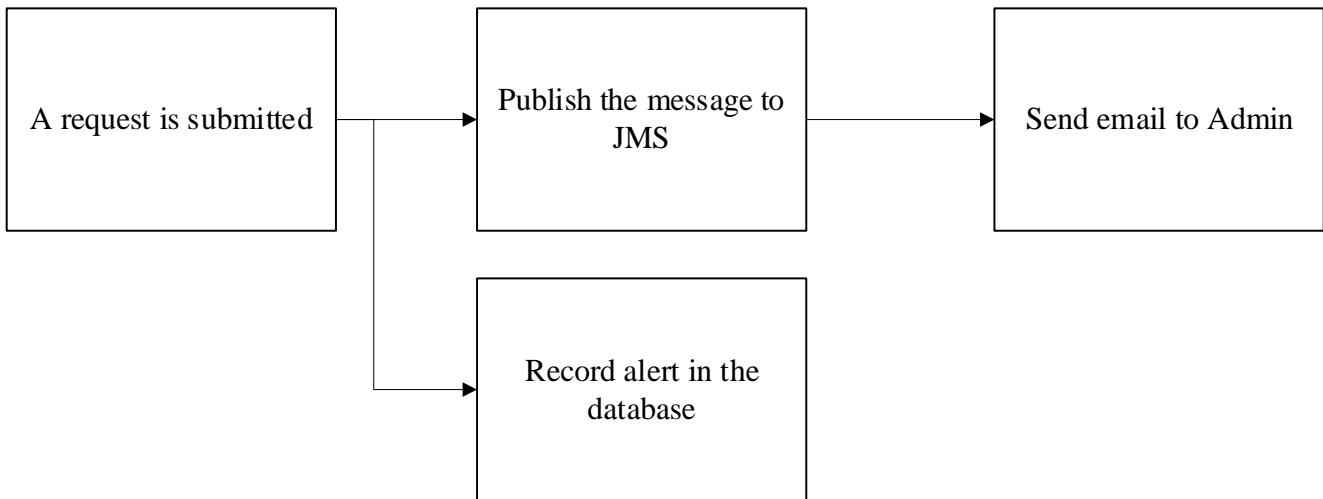


Figure 5 Functionality F: notify admin via email using JMS and database

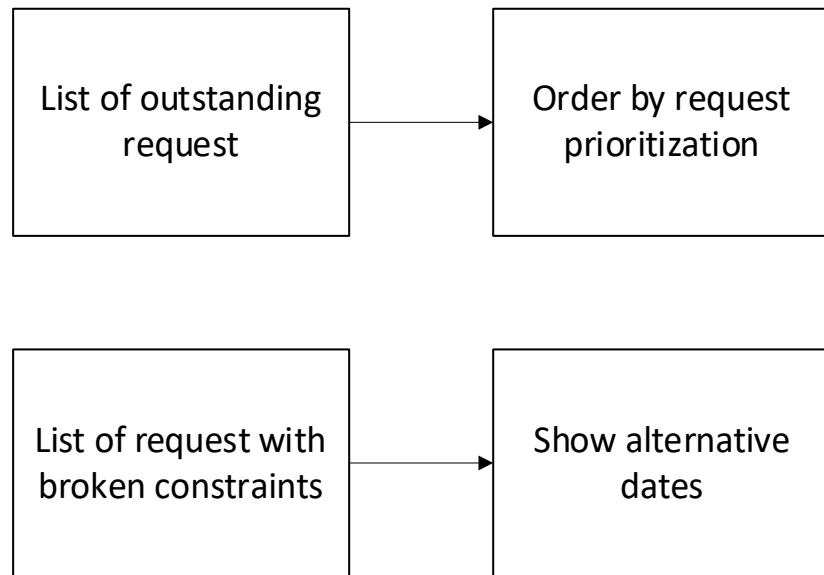


Figure 6 Functionality G: prioritization of requests and alternative dates

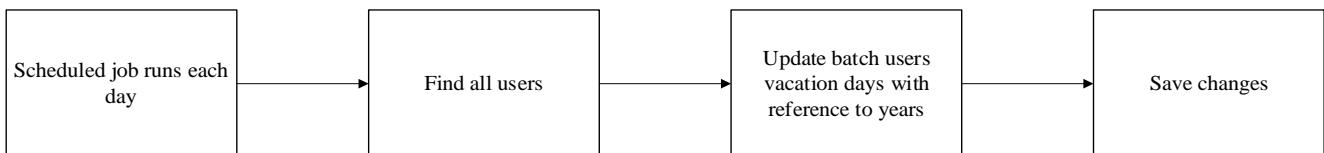


Figure 7 Scheduled job to update batch of vacation days for employees

High level network model of holiday system is demonstrated in Figure 8.

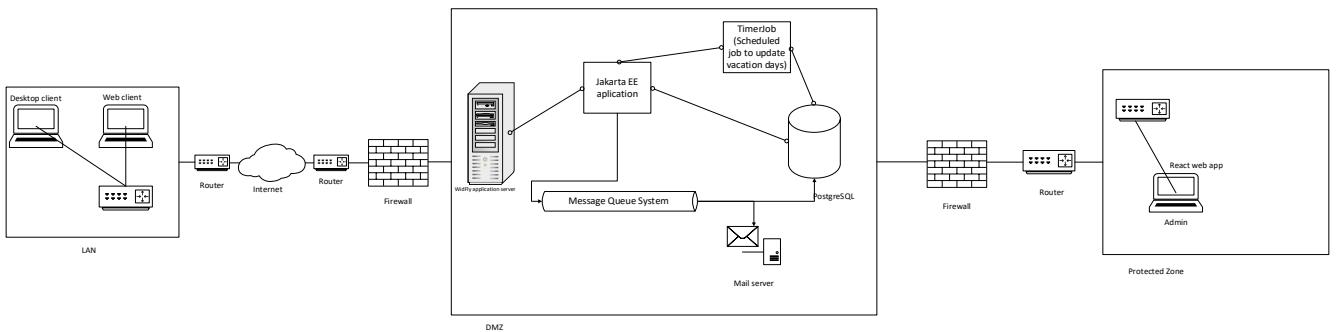


Figure 8 Simple Network Model

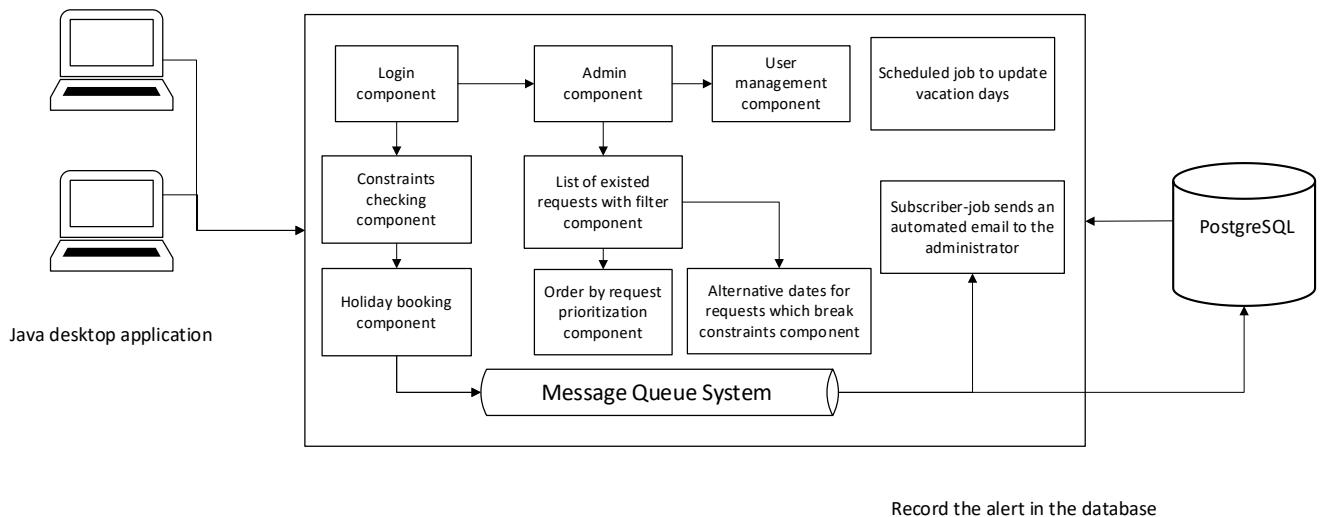


Figure 9 System Design

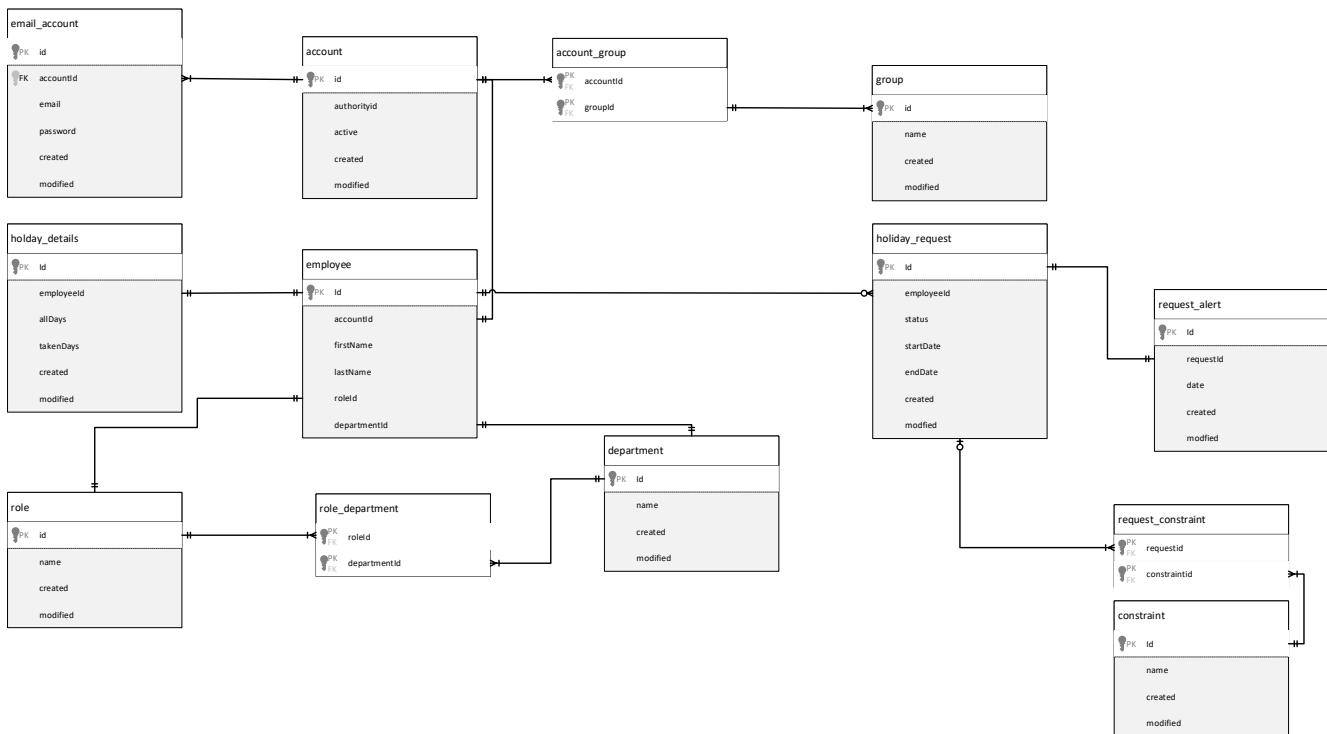


Figure 10 Entity Relationship Diagram

Screenshots

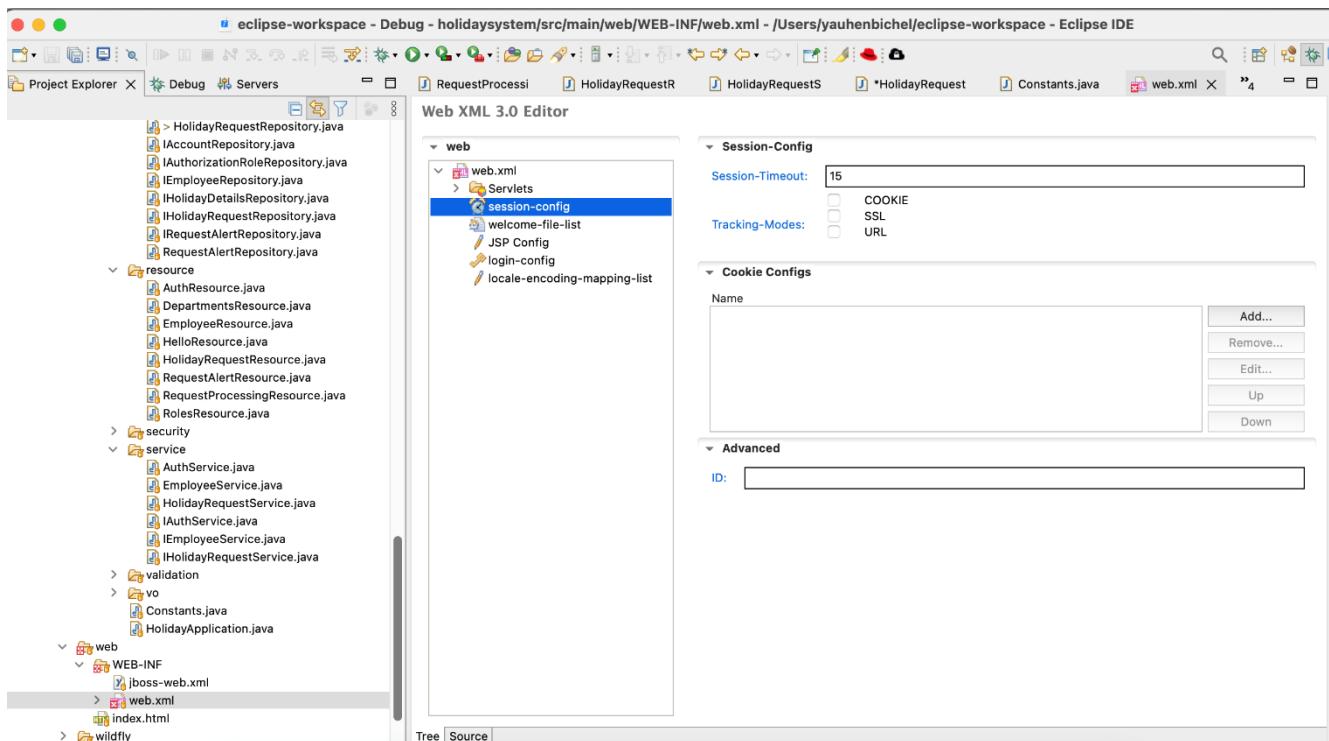


Figure 11 Session timeout

```

126     </thread-pool>
127   </subsystem>
128   <subsystem xmlns="urn:jboss:domain:bean-validation:1.0"/>
129   <subsystem xmlns="urn:jboss:domain:core-management:1.0"/>
130   <subsystem xmlns="urn:jboss:domain:datasources:6.0">
131     <datasources>
132       <datasource jndi-name="java:jboss/datasources/ExampleDS" pool-name="ExampleDS" enabled="true" use-java-context="true" statistic-
133         <connection-url>jdbc:h2:mem:test;DB_CLOSE_DELAY=-1;DB_CLOSE_ON_EXIT=FALSE</connection-url>
134         <driver>h2</driver>
135         <security>
136           <user-name>sa</user-name>
137           <password>sa</password>
138         </security>
139       </datasource>
140       <datasource jndi-name="java:/PostgresDS" pool-name="PostgresDS">
141         <connection-url>jdbc:postgresql://ec2-52-214-125-106.eu-west-1.compute.amazonaws.com:5432/dfejis77nibjdr</connection-url>
142         <driver-class>org.postgresql.Driver</driver-class>
143         <driver>postgresql-42.3.3.jar</driver>
144         <security>
145           <user-name>qexdvxvgdqtdgh</user-name>
146           <password>bf7f8df51b60c17d1fcc333c3f7b63e57df7300022deb67d60eb6cee2bd4e5dd</password>
147         </security>
148         <validation>
149           <valid-connection-checker class-name="org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLValidConnectionChecker">
150             <validate-on-match>true</validate-on-match>
151             <exception-sorter class-name="org.jboss.jca.adapters.jdbc.extensions.postgres.PostgreSQLExceptionSorter"/>
152           </validation>
153         </datasource>
154       <drivers>
155         <driver name="h2" module="com.h2database.h2">
156           <xadatasource-class>org.h2.idb.IdbDataSource</xadatasource-class>

```

Figure 12 Database settings

KEY	VALUE	DESCRIPTION	Bulk Edit
Key	Value	Description	

```

1 [
2   {
3     "id": "e51b3404-810f-49ba-a725-2f0a1e7d1180",
4     "accountId": "2e1293be-a549-4464-aa18-09a714a1ef1c",
5     "email": "tom.litvin@belarus.com",
6     "firstName": "Tom",
7     "lastName": "Litvin",
8     "role": "HEAD",
9     "department": "ENGINEERING",
10    "years": 0,
11    "totalDays": 30,
12    "takenDays": 0,
13    "holidayStatus": "ON_DUTY"
14  },
15 ]

```

Figure 13 Holiday System API Request/Response collection

My Workspace

APIs

Comments

Servers

Monitors

Logs

History

Actions

Import

Get All Holiday Request

GET Hello

GET Get Pages of Holiday Requests

GET Get All Employees

POST Get All Employees By Date

POST Get On Duty Employees By D...

POST Get On Holiday Employees B...

GET Get All Holiday Requests

GET Get Pages of Holiday Requests

GET Get Prioritized Holiday Requ...

GET Get Alternative Dates

GET Get All Holiday Requests By S...

GET Get All Request Alerts

GET Find Employee By Id

GET Find Holiday Request By Id

GET Validate Holiday Request

POST Add Employee

POST Add Holiday Request

PUT Update Holiday Request

Search: Find and Replace

Console

localhost

Save

Send

Params: offset=1, limit=2

Body:

```

2   {
3     "id": "674dbcb7-9fd1-48d4-b976-713375306db0",
4     "employeeId": "e51b3404-810f-49ba-a725-2f0a1e7d1180",
5     "status": "BROKEN",
6     "requestedDays": null,
7     "startDate": "2022-06-01T00:00",
8     "endDate": "2022-06-15T00:00",
9     "years": 0,
10    "totalDays": 30,
11    "takenDays": 0
12  },
13  {
14    "id": "fbaa23b4-442c-4b99-a127-2c5199414fff",
15    "employeeId": "e51b3404-810f-49ba-a725-2f0a1e7d1180",
  
```

200 OK 59 ms 809 B Save Response

Figure 14 Pagination: query params: offset and limit sorted by created

Enter a part of object name here

MoleCare

COMP-1610-Holidays - ec2-52-214-125-106.eu

Databases

dfejis77nibjdr

Schemas

public

Tables

holiday_request

holiday_details

request_alert_queue

Views

Materialized Views

Indexes

Functions

Sequences

Data types

Aggregate functions

Event Triggers

Extensions

Storage

System Info

Roles

Administer

System Info

DBeaver Sample Database (SQLite)

postgres - localhost:5432

PostgreSQL - backend - localhost:5433

Script

Auto

COMP-1610-Holidays Script-6

```

inner join holiday_details hd
on hd.employeeid = emp.id
inner join holiday_request hr
on hr.employeeid = emp.id
WHERE hr.startdate <= '2022-10-02 00:00:00.000' AND hr.enddate >= '2022-10-02 00:00:00.000';

select row_number() over() as id, hr.*
from holiday_request hr
ORDER BY hr.created
limit 5 offset 4

select hr.*
from holiday_request hr
ORDER BY hr.created
limit 2 offset 1
  
```

Grid

	ID	EmployeeID	Status	Created
1	674dbcb7-9fd1-48d4-b976-713375306db0	e51b3404-810f-49ba-a725-2f0a1e	BROKEN	2022-04-03 19:53:09.973
2	fbaa23b4-442c-4b99-a127-2c5199414fff	e51b3404-810f-49ba-a725-2f0a1e	APPROVED	2022-04-06 14:18:50.963

2 row(s) fetched - 17ms (1ms fetch), on Apr 23, 16:51:04

Figure 15 Pagination: database testing returns the same records

```

69  /**
70  * Gets all holiday requests
71  * limit 100
72  * @return
73  */
74  @GET
75  @Path("all")
76  http://localhost:8080/holiday-request/all
77
78  public Response getHolidayRequests() {
79
80      List<HolidayRequestModel> models = holidayRequestService.getHolidayRequests();
81
82      List<HolidayResponse> holidayRequestResponses = new ArrayList<>();
83      for(HolidayRequestModel model: models) {
84          HolidayResponse holidayResponse = holidayRequestMapper.toResponse(model);
85          holidayRequestResponses.add(holidayResponse);
86      }
87
88      return Response.ok(holidayRequestResponses)
89          .header("Access-Control-Allow-Origin", "*")
90          .build();
91
92  /**
93  * Gets pages of holiday requests sorted by created by default
94  * @param offset skip number of requests
95  * @param limit size of the page
96  * @return list of HolidayResponse
97  */
98  @GET
99  @Path("query")
100 http://localhost:8080/holiday-request/query
101
102  public Response getHolidayRequestPages(@QueryParam("offset") int offset,
103                                         @QueryParam("limit") int limit) {
104
105      List<HolidayRequestModel> models = holidayRequestService.getHolidayRequests(offset, limit);
106
107      List<HolidayResponse> holidayRequestResponses = new ArrayList<>();
108      for(HolidayRequestModel model: models) {
109          HolidayResponse holidayResponse = holidayRequestMapper.toResponse(model);
110          holidayRequestResponses.add(holidayResponse);
111      }
112
113  }

```

Figure 16 Paging implementing in REST API interface

Package com.holidaysystem.service									
OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP									
<h3>Interface Summary</h3> <table border="1"> <thead> <tr> <th>Interface</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>IAuthService</td> <td>Interface for Auth service provides user register and login</td> </tr> <tr> <td>IEmployeeService</td> <td>Interface for Employee service provides employee details</td> </tr> <tr> <td>IHolidayRequestService</td> <td>Interface for HolidayRequest service provides user register and login</td> </tr> </tbody> </table>		Interface	Description	IAuthService	Interface for Auth service provides user register and login	IEmployeeService	Interface for Employee service provides employee details	IHolidayRequestService	Interface for HolidayRequest service provides user register and login
Interface	Description								
IAuthService	Interface for Auth service provides user register and login								
IEmployeeService	Interface for Employee service provides employee details								
IHolidayRequestService	Interface for HolidayRequest service provides user register and login								
<h3>Class Summary</h3> <table border="1"> <thead> <tr> <th>Class</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>AuthService</td> <td>Auth service provides user register and login</td> </tr> <tr> <td>EmployeeService</td> <td>Employee service provides employee details</td> </tr> <tr> <td>HolidayRequestService</td> <td>HolidayRequest service provides user register and login</td> </tr> </tbody> </table>		Class	Description	AuthService	Auth service provides user register and login	EmployeeService	Employee service provides employee details	HolidayRequestService	HolidayRequest service provides user register and login
Class	Description								
AuthService	Auth service provides user register and login								
EmployeeService	Employee service provides employee details								
HolidayRequestService	HolidayRequest service provides user register and login								

Figure 17 Javadoc Generated Documentation

The screenshot shows the Eclipse IDE interface with the following details:

- Project Explorer:** Shows the project structure for "holidaysystem [holidaysystem temp]". It includes JAX-WS Web Services, JPA Content, src/main/java, JRE System Library [JavaSE-11], Maven Dependencies, Deployed Resources, doc, and src/main/java/com/holidaysystem. Under src/main/java/com/holidaysystem, there are common, comparator, config, constraints, entity, enumeration, mail, mapper, message, model, repository, resource, security, service, validation, vo, Constants.java, and HolidayApplication.java.
- Code Editor:** Displays the content of HolidayApplication.java. The code defines a HolidayApplication class that extends Application. It includes methods for getting singletons and classes, and a constructor that logs the application's entry point.
- Console:** Shows the output of Javadoc Generation, listing various files being generated such as allpackages-index.html, deprecated-list.html, and index.html.

Figure 18 RESTful service project structure

The screenshot shows the Postman application interface with the following details:

- Header Bar:** Includes Home, Workspaces, API Network, Reports, Explore, Search Postman, Invite, Upgrade, and a dropdown menu.
- Left Sidebar:** My Workspace section with collections for APIs, Environments, Mock Servers, Monitors, Flows, and History. The "Holiday System" collection is expanded, showing various API endpoints like Hello, Get All Employees, and Add Holiday Request.
- Request Panel:** A POST request to "http://localhost:8080/holidaysystem/api/holiday-request". The Body tab is selected, showing a JSON payload:


```

1 {
2     "employeeId": "e51b3404-810f-49ba-a725-2f0a1e7d1180",
3     "status": "PENDING",
4     "startDate": "2022-12-01 00:00:00",
5     "endDate": "2022-12-15 00:00:00"
6 }
```
- Response Preview:** An illustration of a spaceman launching a rocket.
- Bottom Navigation:** Find and Replace, Console, Cookies, Capture requests, Bootcamp, Runner, Trash, and Help icons.

Figure 19 Add holiday request API request

```

public void publish(UUID id, HolidayRequest request) {
    String jsonEntity = holidayRequestMapper.toJson(id, request);
    try {
        Context jndiContext = new InitialContext();
        ConnectionFactory factory = (ConnectionFactory) jndiContext.lookup(Constants.JNDI_CONNECTION_FACTORY);
        Queue calculationQueue = (Queue) jndiContext.lookup(Constants.JMS_MESSAGE_QUEUE);
        Connection connect = factory.createConnection();
        Session session = connect.createSession(false, Session.AUTO_ACKNOWLEDGE);
        MessageProducer sender = session.createProducer(calculationQueue);
        MapMessage message = session.createMapMessage();
        message.setString(Constants.QUEUE_KEY_MESSAGE, jsonEntity);
        logger.debug("jsonEntity: " + jsonEntity);
        sender.send(message);
        connect.close();
    } catch (NamingException e) {
        logger.error("id : " + id);
    } catch (JMSException e) {
        logger.error("employeeId : " + employeeId);
        logger.error("startDate : " + startDate);
    } catch (Exception e) {
        logger.error(e.getMessage());
    }
}

```

Figure 20 Put message to message queue

```

public void onMessage(Message message) {
    logger.info("Message received from message queue");
    try {
        MapMessage requestMsg = (MapMessage) message;
        String jsonRequest = requestMsg.getString(Constants.QUEUE_KEY_MESSAGE);
        logger.info(String.format("The jsonRequest: %s", jsonRequest));
        HolidayRequestMessage holidayRequestMessage = holidayRequestMapper.toMessage(jsonRequest);
        RequestAlertEntity entity = requestAlertMapper.toEntity(UUID.randomUUID(), holidayRequestMessage);
        requestAlertRepository.save(entity);
        mailService.send();
    } catch (JMSException e) {
        logger.error(e.getMessage(), e);
    } catch (Exception e) {
    }
}

```

Figure 21 Receiving message from the message queue

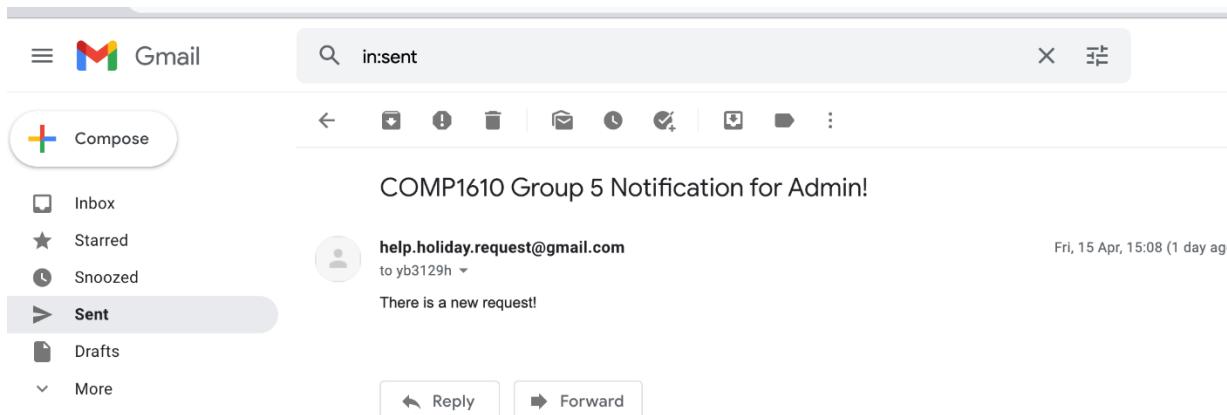


Figure 22 Holiday system app mail box to send a message to Admin. Sending by the application

The screenshot shows a dark-themed email client interface. On the left, there's a sidebar titled 'Deleted Items' with a list for 'Yesterday' containing an item from 'help.holiday.request...' with the subject 'COMP1610 Gro...'. Below this is a section for 'This Week' with an item from 'Greenwich Students'. At the bottom of the sidebar are buttons for 'Add to Favorites', 'Categorize', 'Mark as Private', 'Email', 'Meeting', 'Chat', and '...'. The main pane displays an email from 'help.holiday.request@gmail.com <help.h...>' to 'Yauhen Bichel' with the subject 'COMP1610 Group 5 Notification for Admin!'. The message body says 'There is a new request!'. The timestamp is 'Yesterday at 3:08 PM'. Below the email is the recipient's contact information: 'Yauhen Bichel • yb3129h@gre.ac.uk'. At the bottom of the contact card are buttons for 'Send email' and 'Start chat'. The contact card also includes tabs for 'Overview' (which is selected) and 'Contact'.

Figure 23 Admin receives the notification about new holiday request

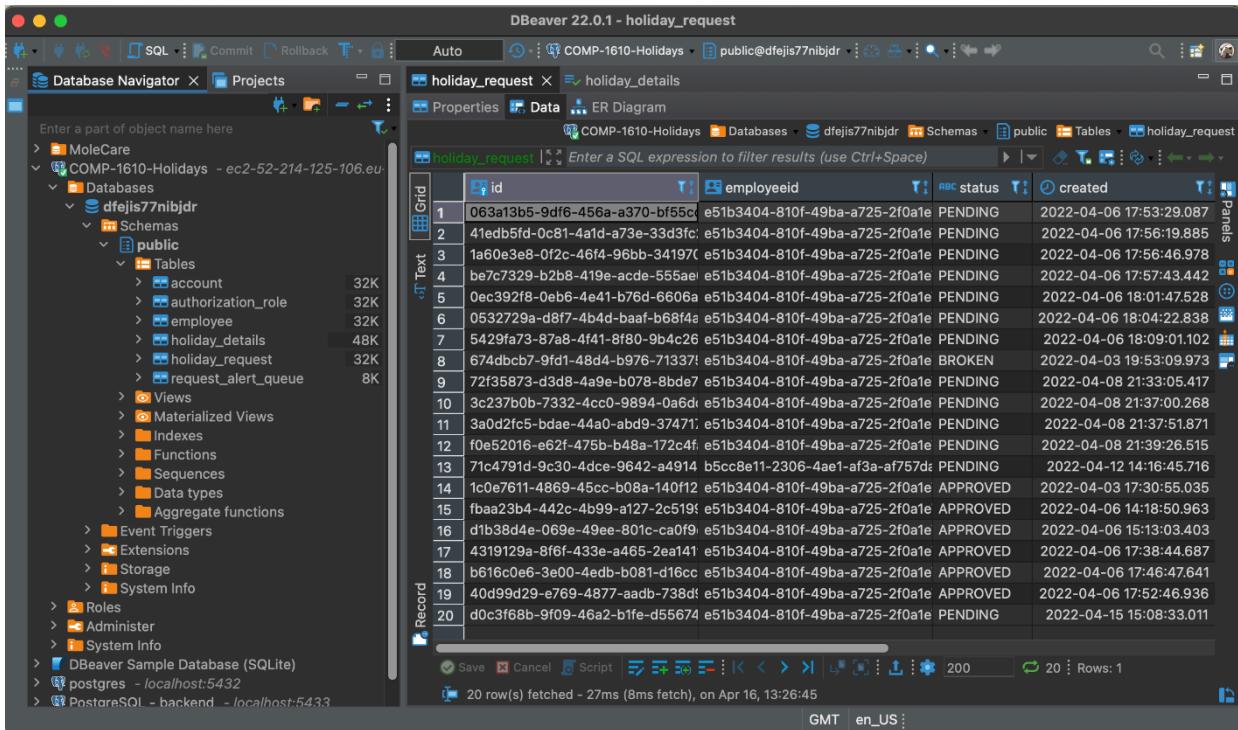


Figure 24 Prototype database, which should be implemented by design in debt tasks

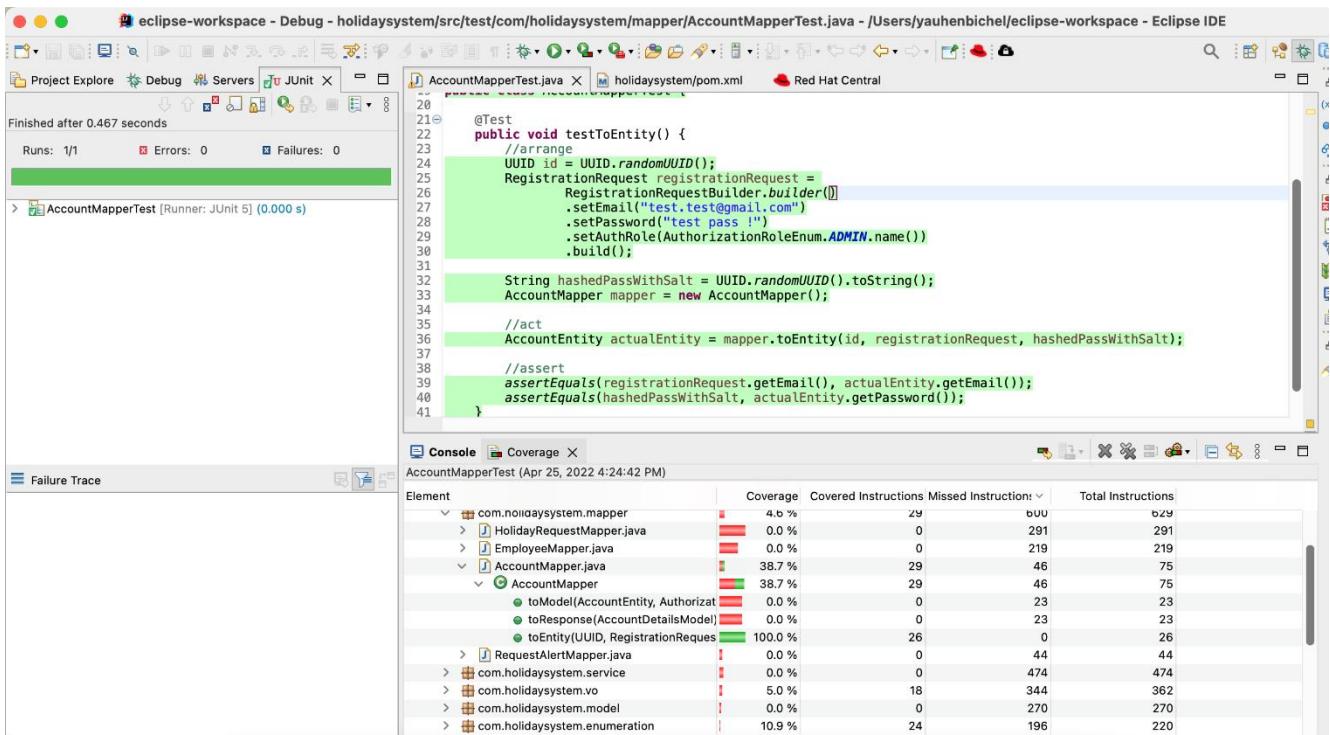


Figure 25 Unit tests and code coverage

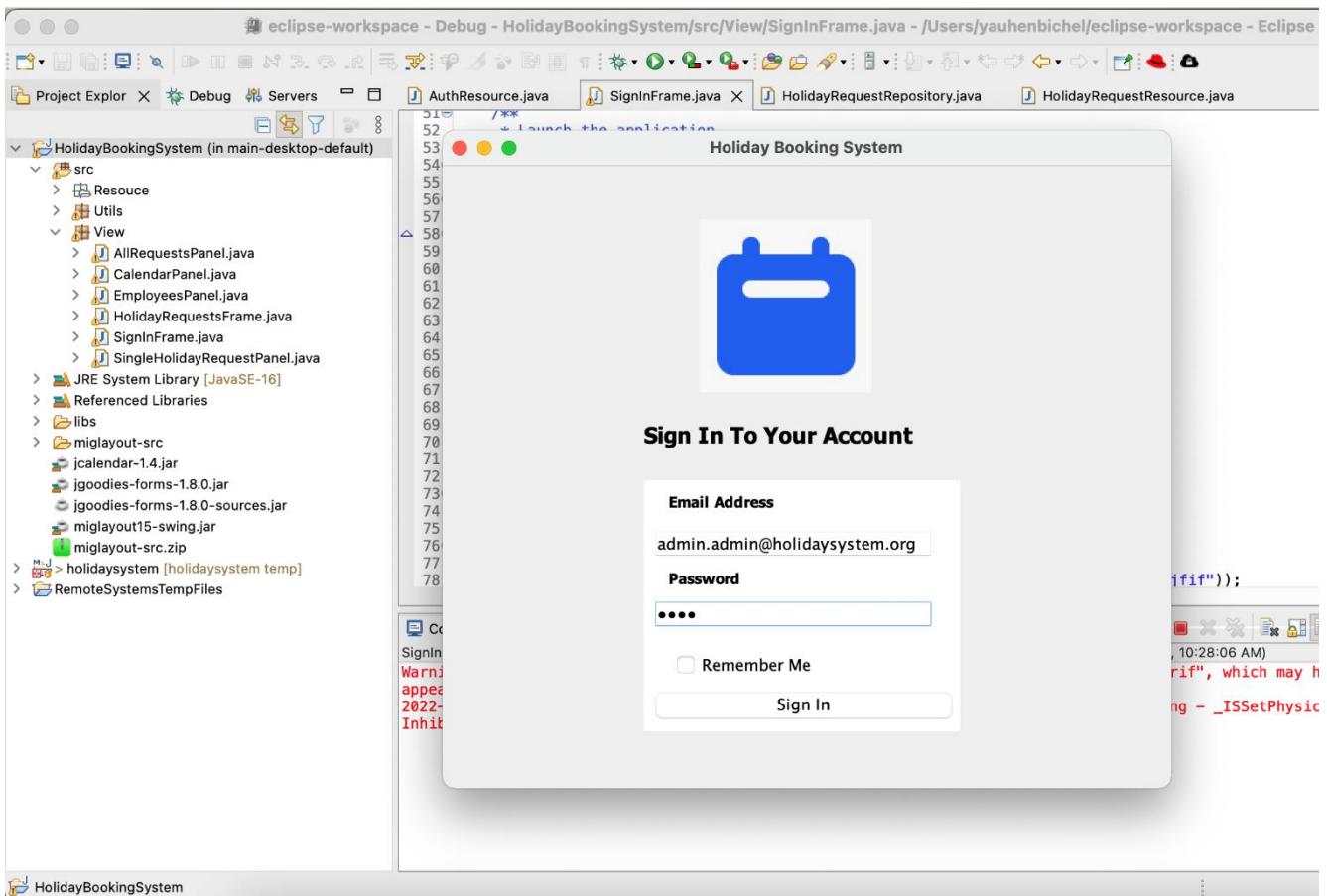


Figure 26 Desktop App Sign In form

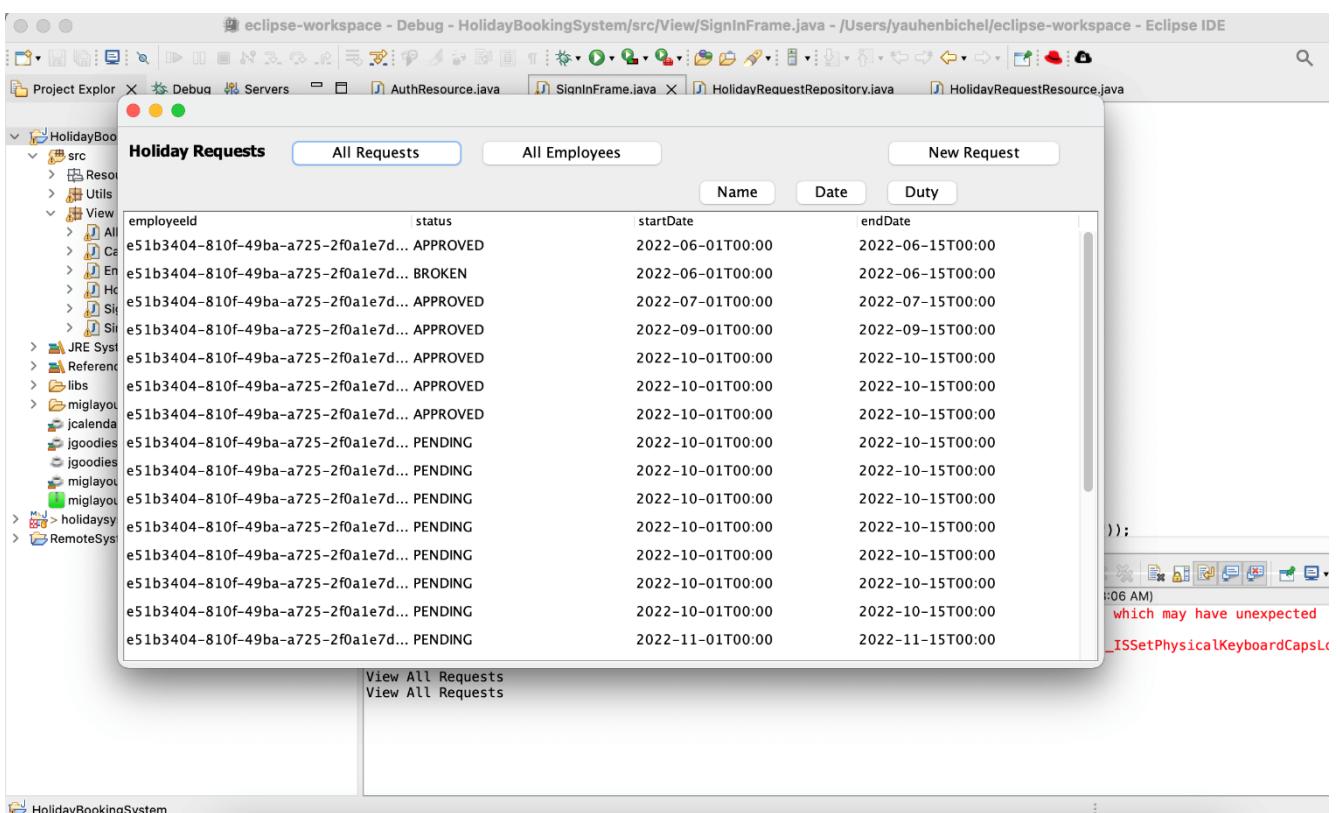


Figure 27 Desktop App Get Requests

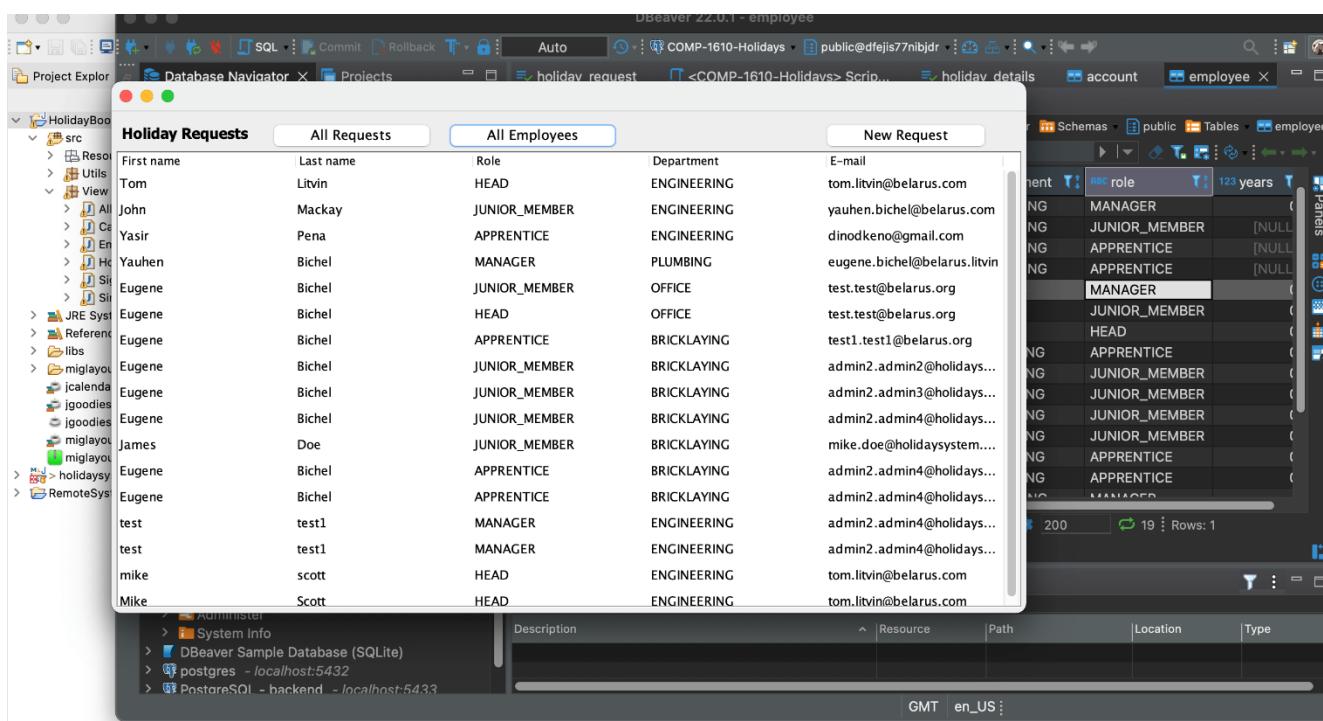


Figure 28 Desktop App: Get Employees

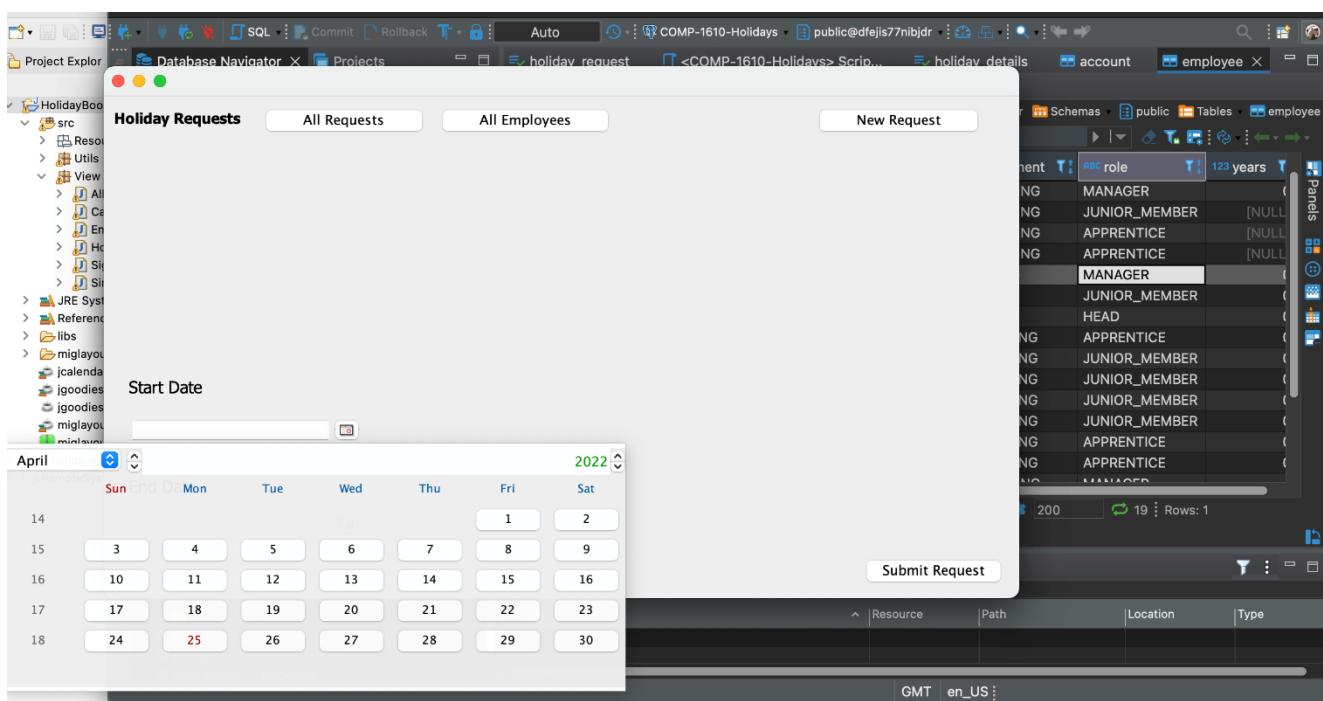


Figure 29 Desktop App: new holiday request

The screenshot shows the 'Holiday Requests' dashboard. At the top, there are three summary boxes: 'Pending Requests' (18), 'Approved Requests' (6), and 'Rejected Requests' (0). Below these are sections for 'Pending Requests' and 'Approved Requests'. The 'Pending Requests' section lists five entries, each with an employee ID, start date (01/10/2022), end date (15/10/2022), and a 'More Details' link.

EMPLOYEE	START DATE	END DATE	
e51b3404-810f-49ba-a725-2f0a1e7d1180	01/10/2022	15/10/2022	More Details
e51b3404-810f-49ba-a725-2f0a1e7d1180	01/10/2022	15/10/2022	More Details
e51b3404-810f-49ba-a725-2f0a1e7d1180	01/10/2022	15/10/2022	More Details
e51b3404-810f-49ba-a725-2f0a1e7d1180	01/10/2022	15/10/2022	More Details
e51b3404-810f-49ba-a725-2f0a1e7d1180	01/10/2022	15/10/2022	More Details

Figure 30 Web: dashboard

The screenshot shows the 'My Requests' view. At the top, there are five entries for 'Jane Cooper' listed under 'NAME'. Each entry includes the title 'Regional Paradigm Technician', role 'Admin', start date '23 Jan. 2022', end date '31 Jan. 2022', email 'jane.cooper@example.com', and a 'More Details' link. A 'New Request' button is located in the top right corner of the table header.

NAME	TITLE	ROLE	START DATE	END DATE	EMAIL	
Jane Cooper	Regional Paradigm Technician	Admin	23 Jan. 2022	31 Jan. 2022	jane.cooper@example.com	More Details
Jane Cooper	Regional Paradigm Technician	Admin	23 Jan. 2022	31 Jan. 2022	jane.cooper@example.com	More Details
Jane Cooper	Regional Paradigm Technician	Admin	23 Jan. 2022	31 Jan. 2022	jane.cooper@example.com	More Details
Jane Cooper	Regional Paradigm Technician	Admin	23 Jan. 2022	31 Jan. 2022	jane.cooper@example.com	More Details
Jane Cooper	Regional Paradigm Technician	Admin	23 Jan. 2022	31 Jan. 2022	jane.cooper@example.com	More Details

Figure 31 Employee requests view

The screenshot shows a web application titled "Holiday Requests" with a navigation bar including "Dashboard", "My Requests", "All Requests" (which is underlined), and "All Employees". A blue button for "+ New Request" and a user profile icon are also present. The main content area is titled "All Requests" and contains a table with columns: EMPLOYEE ID, START DATE, END DATE, and STATUS. The table has 9 rows, all of which have a "More Details" link. The first row is approved, while the second row is broken. The last row is pending.

EMPLOYEE ID	START DATE	END DATE	STATUS	
e51b3404-810f-49ba-a725-2f0a1e7d1180	01/06/2022	15/06/2022	APPROVED	More Details
e51b3404-810f-49ba-a725-2f0a1e7d1180	01/06/2022	15/06/2022	BROKEN	More Details
e51b3404-810f-49ba-a725-2f0a1e7d1180	01/07/2022	15/07/2022	APPROVED	More Details
e51b3404-810f-49ba-a725-2f0a1e7d1180	01/09/2022	15/09/2022	APPROVED	More Details
e51b3404-810f-49ba-a725-2f0a1e7d1180	01/10/2022	15/10/2022	APPROVED	More Details
e51b3404-810f-49ba-a725-2f0a1e7d1180	01/10/2022	15/10/2022	APPROVED	More Details
e51b3404-810f-49ba-a725-2f0a1e7d1180	01/10/2022	15/10/2022	PENDING	More Details

Figure 32 Requests without filters

This screenshot is identical to Figure 32, showing the same list of requests. The status filter "BROKEN" is selected, and only the second row from the previous table is visible, indicating that no other requests are currently broken.

EMPLOYEE ID	START DATE	END DATE	STATUS	
e51b3404-810f-49ba-a725-2f0a1e7d1180	01/06/2022	15/06/2022	BROKEN	More Details

Figure 33 Requests with status BROKEN

The screenshot shows a web browser window with the URL `localhost:3001/all-employees`. The page title is "Holiday Requests". The main content area is titled "All Employees" and displays a grid of employee profiles. Each profile card contains the employee's name, role (e.g., HEAD, MANAGER, APPRENTICE), department (e.g., ENGINEERING, PLUMBING, BRICKLAYING), and a small green circular badge. A blue button labeled "Add Employees" is located in the top right corner of the grid.

Employee	Role	Department
Tom Litvin	HEAD	ENGINEERING
John Mackay	JUNIOR_MEMBER	ENGINEERING
Yasir Pena	APPRENTICE	ENGINEERING
Yauhen Bichel	MANAGER	PLUMBING
Eugene Bichel	JUNIOR_MEMBER	OFFICE
Eugene Bichel	HEAD	OFFICE
Eugene Bichel	APPRENTICE	BRICKLAYING
Eugene Bichel	JUNIOR_MEMBER	BRICKLAYING
Eugene Bichel	JUNIOR_MEMBER	BRICKLAYING
James Doe	JUNIOR_MEMBER	BRICKLAYING
Eugene Bichel	APPRENTICE	BRICKLAYING

Figure 34 Web: all employees

The screenshot shows a web browser window with the URL `localhost:3001/profile`. The page title is "Holiday Requests". The main content area is titled "Personal Information" and contains a form with various input fields. The fields include: First name (Yauhen), Last name (Bichel), Email address (yb3129h@gre.ac.uk), Country (United States), Street address (empty), City (empty), State / Province (empty), and ZIP / Postal code (empty).

Figure 35 Personal view of employee

Evaluation

The solution for holiday booking system was designed and developed for 5 weeks. The product includes server part as RESTful service with PostgreSQL database, React web application as browser client and desktop application as one more client application for clients.

Design of server part was implemented by one student, Yauhen Bichel, the author of the report. Also, the server application configuration, REST API design, project structure design, code structure, implementing SOLID principles and GRASP patterns are designed and developed by Yauhen Bichel also. Apart from that, entity relationship diagram was designed and implemented by me. Moreover, I configured unit tests environment, start writing unit tests, run code coverage. Unfortunately, with reference to a short period of the module, I provided a demonstration how to run unit tests and use code coverage. Also, I help to implement function in web application, which uses React library, in particular, I implemented filter functionality, fixed dates format for REST service.

On the one hand, I think, we are, as the team, did the great job in a very small time period with quite big list of functionalities with supporting 2 different client applications as web application and desktop application. At the same time, I designed low coupled design of the RESTful service, designed names of resources, used DI for decreasing the dependency from implementations of interfaces, used repositories to separate the layers between domain entities and calls to database and business layer of the backend application. Moreover, I designed and realized message system, mail notification, logging system and providing human friendly messages from exception handlers. Also, interfaces methods and classes have Javadoc comments, and I generated Javadoc documentation. In addition to that, the Javadoc documentation, I created postman collection of REST requests/responses, which are added to zip file with source code of the backend application.

On the other hand, with reference to short period of time, which was given, the database schema contains simplified structure than entity relational diagram in the design. The exception handling does not cover all possible scenarios and it can be improved. It is important to mention, that I do not have enough time to write unit tests, add a few integration tests to verify communication between database and the application. The components do not cover all possible API requests, does not check all possible requests validation. Moreover, the communication with database was not analyzed using SQL query analyzer. I need to mention, that logging should be improved as well. As the result, the current solution is not stable, the users can see exception messages, server errors, database can reach inconsistent state. The application does not have retry strategy for failed connections or weak connection between application server and database server. The logs can have user critical data, which should be tested and fixed, also some components can have low covering by logs, and it should be improved as well.

Apart from backend RESTful service, the client applications are in prototype state too. They are not tested well, and they do not have good structure. They do not have any procedure to repeat requests when the connection is weak.

Also, I want to mention, that the significant part of the configuration, design and developed was done by me, Yauhen Bichel. I configured, designed and developed Jakarta EE application by myself. Other students designed and developed React application and desktop application, also they added integration to RESTful service with my help. Moreover, I helped to setup the WildFly application server and the backend application in their local machines. I created the GitHub repository, where I consistently make pull requests, and each member of the team can see the changes. Unfortunately, other members of the team do not have the time and some experience to do the same. As a result, the gap of knowledge and experience between students cannot help in development of the big systems. I have to mention, that we had weekly 1-2 hours meetings during the almost 3 months.

Taking everything into account, the team worked hard, and we implemented the functionalities from requirements. With reference to the fact, that the system is a prototype, in my opinion, the provided solution is quite good. At the same time, the application has a long list of technical debt and secure and stability improvement tasks, which should be designed and developed for starting to use the application in testing with external testers.

3. PART B (COMPLETED INDIVIDUALLY) (20%):

1. **Research (approx. 600 words):** Jakarta EE makes it possible to create enterprise applications. Carry out some research and critically discuss what other technologies and frameworks are available and how they compare to Jakarta EE.
2. **Individual Reflection (approx. 300 words):** Please include a reflection of your role within the team and discuss lessons learnt. What you think went well and what you think could have been improved and how.

1. Research

Jakarta EE is formerly known as Java EE or J2EE. “Java Platform Enterprise Edition (Java EE) is an umbrella standard for Java's enterprise computing facilities”. (docs.oracle.com, 2022). So, as an umbrella of standards, there is a set of different specifications, which allows to keep compatibility of different libraries, tools and frameworks. For example, most of applications, in particular, enterprise applications have security functionality, transactions, logging libraries and logging analysis modules, often message queue systems for supporting asynchronous and robust behaviour, JDBC API to connect to different databases, persistence API for object relational mapping, also all enterprise applications require validation, authentication, authorization and other subsystems, which are commonly used in

applications. There are many frameworks, libraries, different platforms, which should be comparable for writing difficult enterprise applications. With reference to the fact, that most of applications can be divided by tiers and their have common components and subsystems, Jakarta EE provides a set of standards for different systems, which used for developing enterprise applications using Java. That is why, when the application is built using Jakarta EE model, using frameworks, libraries and tools, which are support a set of standards from Java EE, then the application can work in different environment without changes in source code.

During my prototype development, I used WildFly, which supports Jakarta EE standards. WildFly allows to use different subsystems, as fact, I used data source subsystem to add JDBC driver and set connection settings, messaging subsystem to add queue and topic. For example, WildFly provides RESTEasy implementation of JAX-RS standard for building REST services, and ActiveMQ is messaging system in WildFly by default.

There are many different technologies and frameworks, which can be used for developing Java applications and enterprise applications. However, one of the main requirements for most of enterprise applications is compatibility with standards, and there is Jakarta EE standards for Java enterprise applications. WildFly is one of the application servers, which is compatible with Jakarta EE standards. Also, GlassFish server is one of other examples of Jakarta EE compatible application server. (javaee.github.io/glassfish, 2022). WildFly and GlassFish manage Java EE applications. Moreover, Oracle WebLogic Server is one of popular application servers for managing and deploying Jakarta EE applications.

One of the alternatives for using applications servers as WildFly is Spring framework. Spring provides components for all required subsystems in enterprise applications. As a developer, we can find all components in Spring as in Jakarta EE applications. Spring provides security, batch component and scheduling services, persistence API and other modules and components. At the same time, Spring provides their own implementations as Spring MVC for REST services, when Jakarta EE compatible applications use JAX-RS compatible implementations as RESTEasy and Jersey. Also, when Jakarta EE applications use EJB for implementing business logic in enterprise applications, Spring framework provides Spring Beans like Service, Component, Repository. The idea of them are the same using interfaces and providing different implementation. Apart of it, Jakarta EE applications use message driven beans for message systems, when Spring suggests Spring JMS.

Taking everything into account, there are many different Jakarta EE applications servers for managing and deploying Java applications, also as an alternative for traditional Jakarta EE servers, Java enterprise applications can be developed using Spring framework. Some of the main criteria for choosing the application server and framework is documentation, professional community and Jakarta EE standards compatibility. At the same time, Spring framework provides some compatibility, for

example while Java EE uses CDI contexts and dependency injection standards, Spring provides a dependency injection container. But CDI is an option for DI in Spring.

2. Individual Reflection

Design for the Jakarta EE application was made by me. I separated functionality into logical components, I defined user flows, I created entity relationship diagram.

Also, I setup WildFly application server, configured relational database for the application. Helped to setup Eclipse IDE and WildFly application server for other members of the team.

Apart from it, I setup datasource, message system, mail, created gmail account for the application.

After that, I created README.md file with settings and added configured standalone file with jdbc postgresql driver jar file into the application folder.

I designed and developed from scratch Jakarta EE application. For all designed and implemented public API, I provided examples of REST requests and responses in postman collection using the following tool <https://www.postman.com/downloads>.

Moreover, I teach all other members of the group how to think about design, what structure of the code should be, how to write code, how to provide low coupling with high cohesion, implement unit tests.

During the collaboration with my team, I scheduled the library meetings, discussed our time management, tried to fix different issues like CORS errors.

Regarding the learnt lessons, I would like to mention that it was my first experience of developing Jakarta EE applications, and I learnt many new tools and servers, and how setup application server for enterprise applications, which supports Jakarta EE standards. With reference to my experience with Spring framework, I can compare development of Java applications using for example WildFly application server and Spring framework.

Taking everything into account, I learnt WildFly application server, Jakarta EE standards, configuration datasource, message subsystem, mail subsystem, logging, debug application in application servers. However, I found very difficult to collaborate in the team in very short time periods in two quite big projects. I think, the developed product, which includes quite big backend part with separated web application and desktop application requires at least 6 months of development. That is why, I found the module is quite difficult because of very short time period for developing the product in a team with very different level of knowledge and experience. For example, I designed and developed a full RESTful service and configured the application server by myself. And, in my opinion, it is not related with management or time management, the problem is very different level of students

in a very short time period. Also, the implemented by me prototype does not mirror my design fully. During the implantation I simplified the domain model and functionality to provide worked prototype.

I am glad to learn WildFly application server and developing enterprise applications using Jakarta EE specifications.

4. COURSEWORK CONTRIBUTION (COMPLETED AS A GROUP)

In percentage, please indicate the work contribution of each member. This should be agreed by all group members. The total of all members work must add to 100%

Team member name	Student ID	individual overall work contribution (%)	Signature	Note
Student: Yauhen Bichel	001185491	30	Yauhen Bichel	<ul style="list-style-type: none">- A full design of the system and backend part design in particular.- WildFly configuration and writing steps for setup. Help with setup for team members.- Design and develop Jakarta EE application using WildFly server and its subsystems.- Selecting database, adding tables to database, configuration Heroku for the application.- Providing API requests;- Implement filter functionality in client side;- Design and implement Pagination

				<ul style="list-style-type: none"> - Design and implement message queue system and mail service - Design and implement validation, exception handling, logging and added Javadoc comments - Generated Javadoc documentation
Student: Ivan Huliayev	001181952	20	Ivan Huliayev	
Student: Yahya Chahine	001178039	25	Yahya Chahine	
Student: Nurkaiyr Yedige	001196862		Nurkaiyr Yedige	
Total 100%				

Describe each task you performed	Student: 001185491 Yauhen Bichel	work contribution in %	Student: 001181952 Ivan Huliayev	work contribution in %	Student: 001178039 Yahya Chahine	work contribution in %	Student : 001196862 Nurkaiyr Yedige	work contribution in %	total
----------------------------------	-------------------------------------	------------------------	-------------------------------------	------------------------	-------------------------------------	------------------------	--	------------------------	--------------

Feature A:	70	10	10	10	100%
Feature B:	40	20	20	20	100%
Feature C:	60	10	20	10	100%
Feature D:	40	10	10	40	100%
Feature E:	25	25	25	25	100%
Feature F:	70	10	10	10	100%
Feature G:	40	10	30	20	100%

5. MANUAL DOCUMENT

This section explains how your prototype should be run with clear instructions

There are 3 separated program modules: RESTful service, React web application and desktop application.

RESTful service contains README.md file with the following steps:

- how to setup WildFly server;
- how to run the application in the application server with settings for standalone from the folder with the application source code;
- where database is deployed and how to connect to the PostgreSQL database for the web service.
- Mail box of the application, which is used for sending the message.

1. Firstly, please setup WildFly application server, which supports Jakarta EE standards.

- a. Download the WildFly 26.0.1.Final zip file here <https://www.wildfly.org/downloads/>
- b. Copy the unzipped folder to the place in disk, where the system contains application servers.

- c. Rename wildfly-26.0.01.Final folder to wildfly-24.0.0.Final for supporting by Eclipse
2. After that, please install Eclipse and run the application in Eclipse IDE.
- a. Download the Eclipse <https://www.eclipse.org/downloads/>
 - b. Download the Jakarta EE from the group 5 github repository
<https://github.com/COMP-1610-cw-team/backend> . Also, the zip folder can be downloaded from Moodle submission folder in Moodle COMP-1610
<https://moodlecurrent.gre.ac.uk/course/view.php?id=70195>
 - c. Open Eclipse IDE and click File -> Open Projects from File System
 - d. Open folder with the holidaysystem project
 - e. Find standalone.xml file with settings for WildFly application server for the application in src->main->wildfly folder.

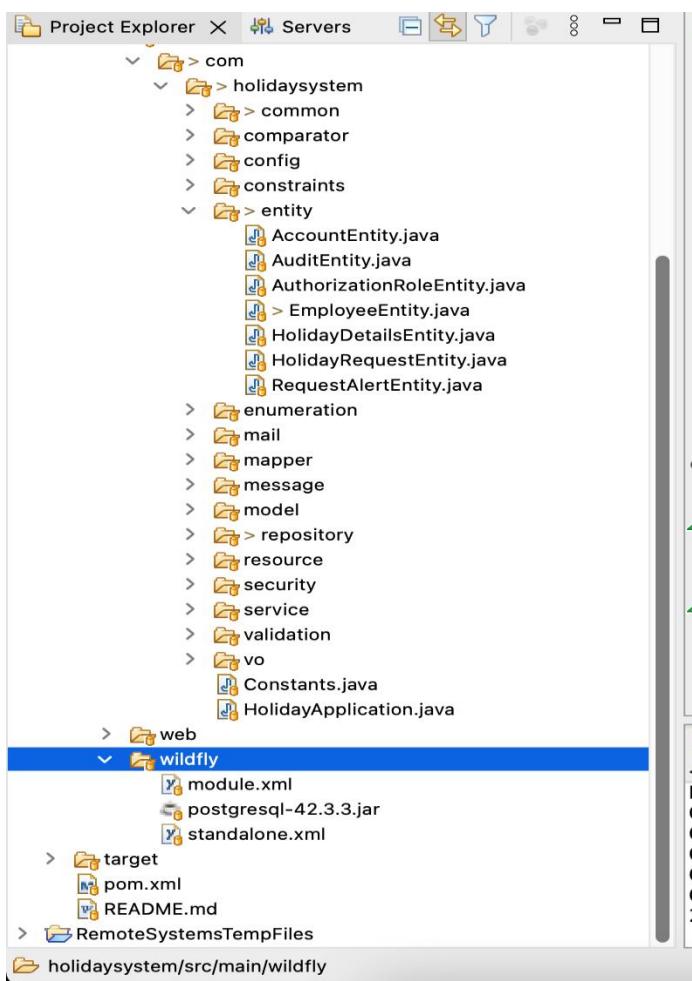
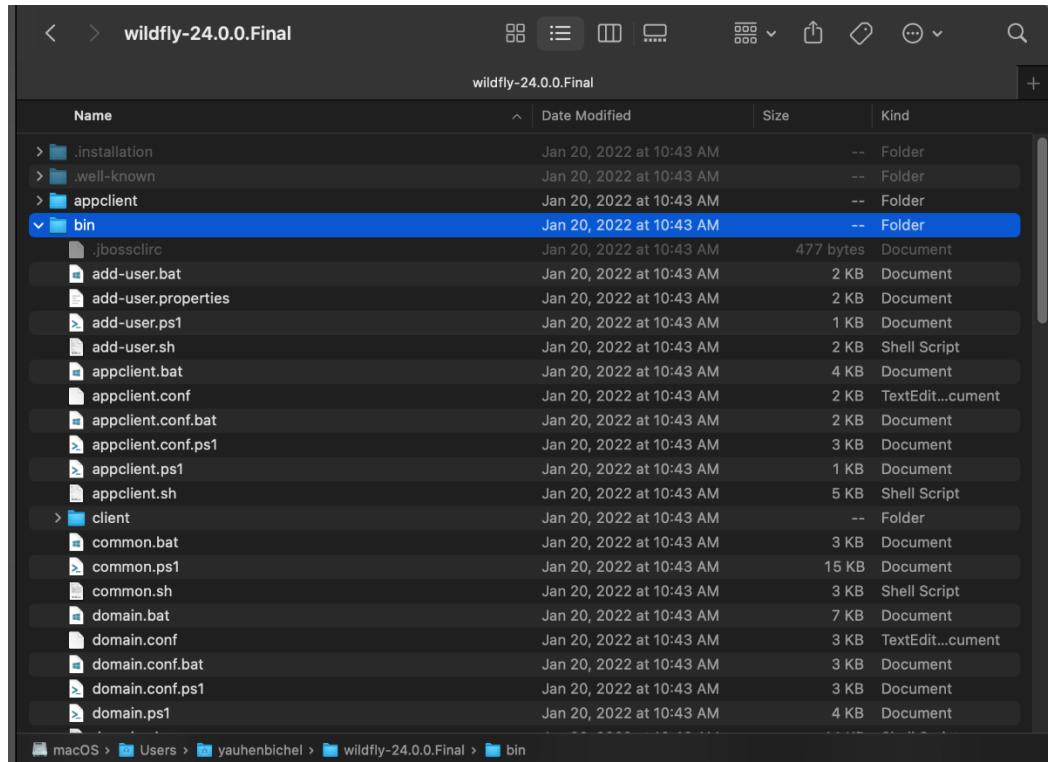


Figure 36 WildFly folder with standalone.xml file

- f. Copy standalone.xml file into wildfly-24.0.0.Final->standalone->configuration folder.

Name	Date Modified	Size	Kind
> bin	Jan 20, 2022 at 10:43 AM	--	Folder
copyright.txt	Jan 20, 2022 at 10:43 AM	19 KB	Plain Text
> docs	Jan 20, 2022 at 10:43 AM	--	Folder
> domain	Jan 20, 2022 at 10:43 AM	--	Folder
jboss-modules.jar	Jan 20, 2022 at 10:43 AM	456 KB	Java JAR file
LICENSE.txt	Jan 20, 2022 at 10:43 AM	27 KB	Plain Text
> modules	Apr 6, 2022 at 12:33 PM	--	Folder
README.txt	Jan 20, 2022 at 10:43 AM	2 KB	Plain Text
> standalone	Apr 8, 2022 at 9:28 PM	--	Folder
configuration	Yesterday at 11:48 PM	--	Folder
application-roles.properties	Jan 20, 2022 at 10:43 AM	711 bytes	Document
application-users.properties	Jan 20, 2022 at 10:43 AM	935 bytes	Document
logging.properties	Today at 4:02 PM	2 KB	Document
mgmt-groups.properties	Apr 6, 2022 at 11:56 AM	681 bytes	Document
mgmt-users.properties	Apr 6, 2022 at 11:56 AM	1 KB	Document
standalone copy.xml	Apr 5, 2022 at 7:25 PM	31 KB	XML Document
> standalone_xml_history	Today at 4:02 PM	--	Folder
standalone-full-ha.xml	Jan 20, 2022 at 10:43 AM	37 KB	XML Document
standalone-full.xml	Apr 5, 2022 at 7:50 PM	33 KB	XML Document
standalone-ha.xml	Jan 20, 2022 at 10:43 AM	33 KB	XML Document
standalone-load-balancer.xml	Jan 20, 2022 at 10:43 AM	13 KB	XML Document
standalone-microprofile-ha.xml	Jan 20, 2022 at 10:43 AM	26 KB	XML Document
standalone-microprofile.xml	Jan 20, 2022 at 10:43 AM	23 KB	XML Document
standalone.xml	Yesterday at 11:48 PM	34 KB	XML Document

- g. Setup WildFly in Eclipse
 - i. Open Eclipse IDE
 - ii. Click Window -> Show View -> Other -> Servers
 - iii. Setup server by choosing WildFly 24+ and selecting folder the installed application server.
 - iv. Add the application in the application server.
- h. When the application server is running, open the management console
<http://127.0.0.1:9990/console/index.html>
- i. WildFly will ask to login. Then add a user using add-user.sh file in wildfly-24.0.0.Final -> bin folder. (comp1610y / password1)



- j. Restart the application server and open the management console. After login, please check that datasource is set (postgresql-42.3.3.jar), message contains topic and jms
- k. Please find postgresql-42.3.3.jar jdbc driver in the application folder in src->main->wildfly folder.
- l. Install all maven dependencies using command: >mvn install
- m. Check the application is running by sending GET request
<http://127.0.0.1:9990/console/index.html>
3. The application database is PostgreSQL, which is hosted in Heroku cloud provider. The database has the following settings:
 - a. Host: ec2-52-214-125-106.eu-west-1.compute.amazonaws.com
 - b. Database: dfejis77nibjdr
 - c. User: qexdvxvgdqtdgh
 - d. Port: 5432
 - e. Password:
bf7f8df51b60c17d1fcc333c3f7b63e57df7300022deb67d60eb6cee2bd4e5dd
 - f. jdbc:postgresql://ec2-52-214-125-106.eu-west-1.compute.amazonaws.com:5432/dfejis77nibjdr
 - g. Heroku CLI: heroku pg:psql postgresql-infinite-73558 --app holiday-system

The screenshot shows the HAL Management Console interface. The left sidebar has sections like Subsystems, Interfaces, Socket Bindings, Paths, and System Properties. The main area is titled 'Configuration' and shows a tree view under 'Subsystem (34)'. The 'Datasources & Drivers' section is expanded, showing 'Datasources' and 'JDBC Drivers'. Under 'Datasources', 'PostgresDS' is selected, indicated by a blue border. A green success message at the top right says 'Successfully tested connection for datasource PostgresDS.' Below it, another message says 'The datasource PostgresDS is enabled. Disable'. On the right, there's a panel for 'Main Attributes' with fields for JNDI Name (java:/PostgresDS), Driver Name (postgresql-42.3.3.jar), Connection URL (jdbc:postgresql://ec2-52-214-125-106.eu...), Enabled (true), and Statistics Enabled (false).

4. The mail box was created by me for the application in gmail with the following username and password: help.holiday.request@gmail.com / 1Vacation2!Request%

The screenshot shows a Gmail inbox. The left sidebar has 'Compose', 'Inbox', 'Starred', 'Snoozed', and 'Sent' (which is highlighted). The main area shows an incoming email from 'help.holiday.request@gmail.com' with the subject 'COMP1610 Group 5 Notification for Admin!'. The email was sent on 'Fri, 15 Apr, 15:08 (1 day ago)'. The message body says 'There is a new request!'.

Figure 37 help.holiday.request@gmail.com mailbox

The screenshot shows the HAL Management Console interface for configuring a JMS Queue. The left sidebar has a dark theme with icons for Core Queue, JMS Queue (selected), JMS Topic, Security Setting, Address Setting, and Divert. The main content area has a light background. The title is "JMS Queue" and the sub-title is "Defines a Jakarta Messaging queue." A table lists three items: DLQ, ExpiryQueue, and HolidayRequestQueue, with HolidayRequestQueue selected. Below the table are sections for "Selector" (Entries: java:jms/HolidayRequestQueue) and "Durable" (true). Navigation links at the top include "Back", "Configuration", "Subsystem", "Messaging", "Category", "Server", "Settings", and "Destinations".

Figure 38 JMS Queue configuration

The screenshot shows the HAL Management Console interface for configuring a JMS Topic. The left sidebar has a dark theme with icons for Core Queue, JMS Queue, JMS Topic (selected), Security Setting, Address Setting, and Divert. The main content area has a light background. The title is "JMS Topic" and the sub-title is "Defines a Jakarta Messaging topic." A table lists one item: HolidayRequestTopic, which is selected. Below the table is a section for "Entries" (java:jms/HolidayRequestTopic). Navigation links at the top include "Back", "Configuration", "Subsystem", "Messaging", "Category", "Server", "Settings", and "Destinations".

Figure 39 JMS Topic configuration

wildfly-24.0.0.Final				
Name	Date Modified	Size	Kind	
> standalone_xml_history	Today at 4:02 PM	--	Folder	
standalone-full-ha.xml	Jan 20, 2022 at 10:43 AM	37 KB	XML Document	
standalone-full.xml	Apr 5, 2022 at 7:50 PM	33 KB	XML Document	
standalone-ha.xml	Jan 20, 2022 at 10:43 AM	33 KB	XML Document	
standalone-load-balancer.xml	Jan 20, 2022 at 10:43 AM	13 KB	XML Document	
standalone-microprofile-ha.xml	Jan 20, 2022 at 10:43 AM	26 KB	XML Document	
standalone-microprofile.xml	Jan 20, 2022 at 10:43 AM	23 KB	XML Document	
standalone.xml	Yesterday at 11:48 PM	34 KB	XML Document	
> data	Apr 6, 2022 at 11:54 AM	--	Folder	
> deployments	Today at 4:02 PM	--	Folder	
> lib	Jan 20, 2022 at 10:43 AM	--	Folder	
> log	Today at 12:00 AM	--	Folder	
audit.log	Apr 8, 2022 at 9:27 PM	Zero bytes	Log File	
holidaySystem.log	Today at 4:03 PM	989 KB	Log File	
holidaySystem.log.2022-04-11	Apr 11, 2022 at 11:53 PM	30.9 MB	Document	
holidaySystem.log.2022-04-12	Apr 12, 2022 at 11:49 PM	24.8 MB	Document	
holidaySystem.log.2022-04-13	Apr 13, 2022 at 11:48 PM	48.1 MB	Document	
holidaySystem.log.2022-04-14	Apr 14, 2022 at 10:58 PM	20 MB	Document	
holidaySystem.log.2022-04-15	Yesterday at 11:59 PM	9.3 MB	Document	
server.log	Apr 11, 2022 at 4:36 PM	12 KB	Log File	
server.log.2022-04-08	Apr 8, 2022 at 11:57 PM	359 KB	Document	
server.log.2022-04-09	Apr 9, 2022 at 1:14 PM	159 KB	Document	
server.log.2022-04-10	Apr 10, 2022 at 8:24 PM	28 KB	Document	
> tmp	Today at 4:02 PM	--	Folder	
> welcome-content	Jan 20, 2022 at 10:43 AM	--	Folder	

macOS > Users > yauhenbichel > wildfly-24.0.0.Final

Figure 40 Logs are stored in log folder of WildFly

5. How to run unit tests

Right mouse click and then run Junit tests

```

20
21  @Test
22  public void testToEntity() {
23      //Arrange
24      UUID id = UUID.randomUUID();
25      RegistrationRequest registrationRequest =
26          RegistrationRequestBuilder.builder()
27              .setEmail("test.test@gmail.com")
28              .setPassword("test pass !")
29              .setAuthRole(AuthorizationRoleEnum.ADMIN.name())
30              .build();
31
32      String hashedPassWithSalt = UUID.randomUUID().toString();
33      AccountMapper mapper = new AccountMapper();
34
35      //act
36      AccountEntity actualEntity = mapper.toEntity(id, registrationRequest, hashedPassWithSalt);
37
38      //assert
39      assertEquals(registrationRequest.getEmail(), actualEntity.getEmail());
40      assertEquals(hashedPassWithSalt, actualEntity.getPassword());
41  }

```

Element	Coverage	Covered Instructions	Missed Instruction	Total Instructions
com.holidaySystem.mapper	4.0 %	29	600	629
HolidayRequestMapper.java	0.0 %	0	291	291
EmployeeMapper.java	0.0 %	0	219	219
AccountMapper.java	38.7 %	29	46	75
AccountMapper	38.7 %	29	46	75
toModel(AccountEntity, AuthorizationRoleEnum)	0.0 %	0	23	23
toResponse(AccountDetailsModel)	0.0 %	0	23	23
toEntity(UUID, RegistrationRequest)	100.0 %	26	0	26
RequestAlertMapper.java	0.0 %	0	44	44
com.holidaySystem.service	0.0 %	0	474	474
com.holidaySystem.vo	5.0 %	18	344	362
com.holidaySystem.model	0.0 %	0	270	270
com.holidaySystem.enumeration	10.9 %	24	196	220

6. How to run desktop application

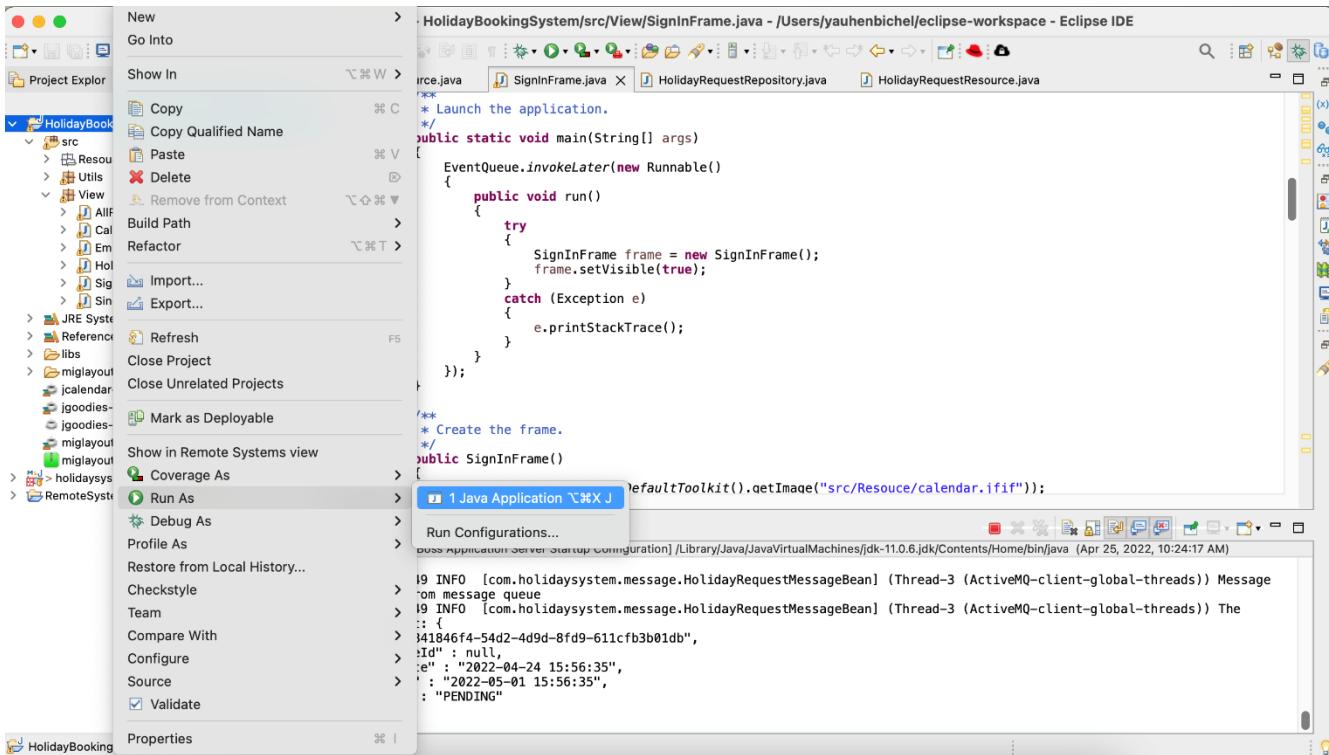


Figure 41 Right mouse click, choose Run As

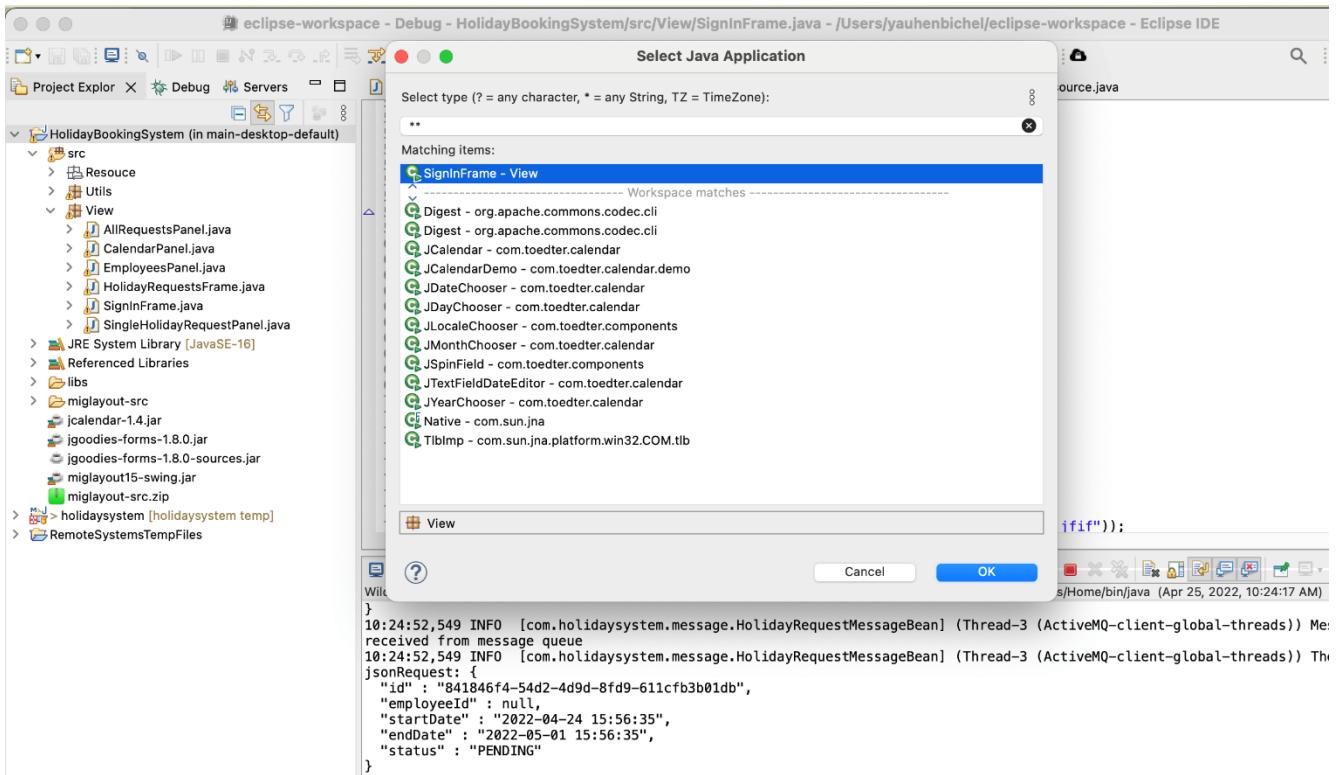
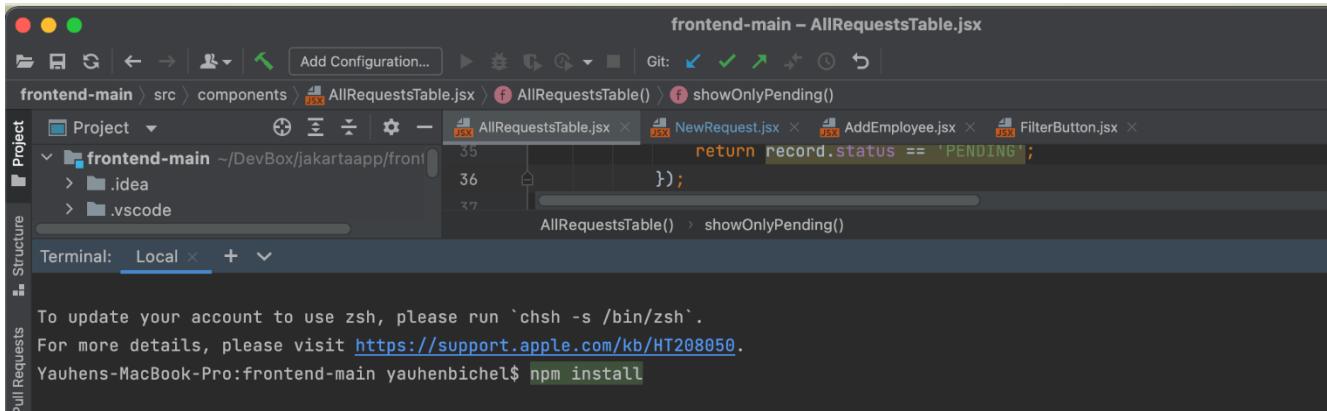


Figure 42 Choose SignIn frame

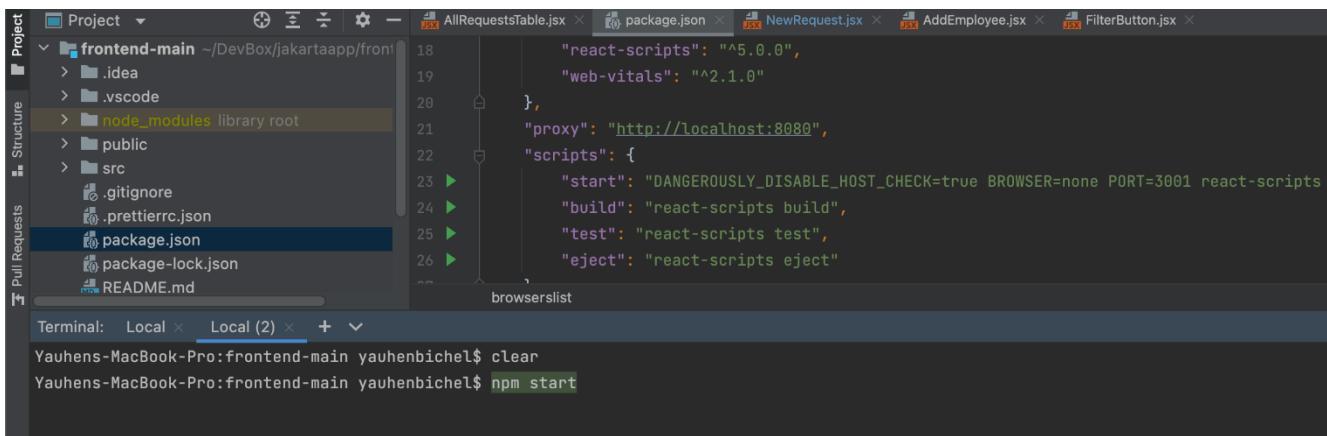
6. How to run web application



The screenshot shows the VS Code interface with the title bar "frontend-main – AllRequestsTable.jsx". The left sidebar has "Project" selected. The main area shows a code editor with several files: AllRequestsTable.jsx, NewRequest.jsx, AddEmployee.jsx, and FilterButton.jsx. The terminal at the bottom shows the command "npm install" being run, with the output:

```
To update your account to use zsh, please run `chsh -s /bin/zsh`.  
For more details, please visit https://support.apple.com/kb/HT208050.  
Yauhens-MacBook-Pro:frontend-main yauhenbichel$ npm install
```

Figure 43 Download npm package dependencies



The screenshot shows the VS Code interface with the title bar "frontend-main – AllRequestsTable.jsx × package.json × NewRequest.jsx × AddEmployee.jsx × FilterButton.jsx". The left sidebar has "Project" selected. The main area shows the "package.json" file open. The "scripts" section contains the "start" script: "start": "DANGEROUSLY_DISABLE_HOST_CHECK=true BROWSER=none PORT=3001 react-scripts start". The terminal at the bottom shows the command "npm start" being run.

Figure 44 Run the following cmd to run the web app: npm start

7. Possible issues during starting wildfy application server

If WildFly can not be started, then some ports can be taken by other applications. In this case, please find the processes Ids and kill them using the steps as provided in Figure 45.

```
~/eclipse-workspace/holidaysystem -- bash — 117x30
~/DevBox/Jakartaapp -- bash +
```

	PID	PPID	CPUTIME	SIZE	NAME
4	29	4	524288	524288	com.apple.net.utun_control
5	21	0	65536	65536	com.apple.net.ipsec_control
6	0	52	8192	2048	com.apple.netsrc
7	18	3	8192	2048	com.apple.network.statistics
8	5	0	8192	32768	com.apple.network.tcp_ccdebug
9	1	0	8192	2048	com.apple.network.advisory
a	1	1	16384	2048	com.apple.nke.sockwall
b	1	0	8192	2048	com.checkpoint.cpfw.ctl
c	1	0	8192	2048	com.checkpoint.cpfw.fwnotify
d	1	0	1048576	2048	com.checkpoint.cpfw.debug
e	0	0	8192	8192	com.apple.fileutil.kext.stateful.ctl
f	0	0	8192	2048	com.apple.fileutil.kext.stateless.ctl

```
Yauhens-MBP:holidaysystem yauhenbichel$ kill -9 8080
-bash: kill: (8080) - No such process
Yauhens-MBP:holidaysystem yauhenbichel$ kill -9 8443
-bash: kill: (8443) - No such process
[Yauhens-MBP:holidaysystem yauhenbichel$ sudo lsof -i :8080
[Password: ]]
COMMAND PID      USER FD   TYPE DEVICE SIZE/OFF NODE NAME
java    4868 yauhenbichel  521u  IPv4 0xb85b9b3f5ca707ff    0t0  TCP localhost:http-alt (LISTEN)
java    24777 yauhenbichel   45u  IPv6 0xb85b9b3f5c89c207    0t0  TCP *:http-alt (LISTEN)
[Yauhens-MBP:holidaysystem yauhenbichel$ kill -p 4868
-bash: kill: p: invalid signal specification
[Yauhens-MBP:holidaysystem yauhenbichel$ kill -p 24777
-bash: kill: p: invalid signal specification
[Yauhens-MBP:holidaysystem yauhenbichel$ kill -9 4868
[Yauhens-MBP:holidaysystem yauhenbichel$ kill -9 24777
-bash: kill: (24777) - No such process
[Yauhens-MBP:holidaysystem yauhenbichel$ sudo lsof -i :8443
Yauhens-MBP:holidaysystem yauhenbichel$ ]]
```

Figure 45 Find processes and kill them

8. REFERENCES

Docs.oracle.com, (2022), Standards and Specifications

[Online]

Available at:

https://docs.oracle.com/cd/E82085_01/140/rib_implementation_guide/Chapter%202%20-%20Standards%20and%20Specifications.htm

[Accessed 16 March 2022]

javaee.github.io/glassfish, (2022), GlassFish Documentation

[Online]

Available at: <https://javaee.github.io/glassfish/documentation>

[Accessed 12 March 2022]

9. BIBLIOGRAPHY

Martinez-Torres, Rafael, (2022), COMP 1610 Coursework Specification
[Online]

Available at: <https://moodlecurrent.gre.ac.uk/mod/resource/view.php?id=1729371>
[Accessed 10 Mar 2022]

Martinez-Torres, Rafael, (2022), Lectures, Programming Enterprise Components module materials
[Online]

Available at: <https://moodlecurrent.gre.ac.uk/course/view.php?id=70195>
[Accessed 05 January 2022]

Pressman, R., Maxim, B., (2019), Software Engineering: a practitioner's approach
[Online]

Available at: <https://ebookcentral.proquest.com/lib/gre/reader.action?docID=5989441>
[Accessed 25 Feb 2022]

Eclipse-ee4j.github.io, (2022), The Jakarta EE Tutorial
[Online]

Available at: <https://eclipse-ee4j.github.io/jakartaee-tutorial/#introduction-to-jakarta-ee>
[Accessed 14 March 2022]

Devcenter.heroku.com, (2022), Heroku Postgres
[Online]

Available at: <https://devcenter.heroku.com/articles/heroku-postgresql>
[Accessed 10 January 2022]

Baeldung.com, (2022), How to Set Up a WildFly Server
[Online]

Available at: <https://www.baeldung.com/wildfly-server-setup>
[Accessed 01 February 2022]

Wildfly.org, (2022), WildFly Documentation
[Online]

Available at: <https://docs.wildfly.org/26/#administrator-guides>

[Accessed 01 March 2022]

Eclipse-ee4j.github.io, (2022), The Jakarta EE Tutorial

[Online]

Available at: <https://eclipse-ee4j.github.io/jakartaee-tutorial/#introduction-to-jakarta-ee>

[Accessed 14 March 2022]

Jakarta.ee, (2022), Specifications

[Online]

Available at: <https://jakarta.ee/specifications/>

[Accessed 12 March 2022]

Docs.oracle.com, (2022), Oracle WebLogic Server

[Online]

Available at: <https://docs.oracle.com/en/middleware/fusion-middleware/weblogic-server/>

[Accessed 12 March 2022]