FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Assignment of bachelor's thesis

Title:	Web portal for EFB
Student:	Abdullah Abdullah
Supervisor:	Ing. Zdeněk Rybola, Ph.D.
Study program:	Informatics
Branch / specialization:	Web and Software Engineering
Department:	Department of Software Engineering
Validity:	until the end of summer semester 2022/2023

Instructions

Create a web portal application for the Egyptian Food Bank to the agendas of their employees. The application should mainly support the following processes:
- user management,
- management of vacations,
- management of HR letter requests,
- management of IT requests.

The thesis realization should consist of:
- analysis of related processes and requirements for the app,
- analysis of existing solutions,
- design of the solution,
- realization of the application,
- testing and documentation.

Electronically approved by Ing. Michal Valenta, Ph.D. on 16 September 2021 in Prague.

FACULTY
OF INFORMATION
TECHNOLOGY
CTU IN PRAGUE

Bachelor's thesis

Web Portal for EFB

Abdullah Abdullah

Department of Software Engineering
Supervisor: Ing. Zdeněk Rybola, Ph.D.

December 28, 2021

Acknowledgements

I would like to thank and acknowledge my supervisor Ing. Zdeněk Rybola for all his support, guidance and valuable feedback throughout my thesis process.

I would also like to thank my parents and brothers for providing me with the opportunity to study abroad in a renowned university and for their constant support and motivation throughout my studies.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as a school work under the provisions of Article 60 (1) of the Act.

In Prague on December 28, 2021 . . . . . .

Czech Technical University in Prague
Faculty of Information Technology

Citation of this thesis

Abdullah, Abdullah. *Web Portal for EFB*. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology, 2021.

Abstrakt

Cílem bakalářské práce je vytvořit aplikaci webového portálu pro Egyptskou banku potravin (EFB), která bude sloužit jako platforma pro spravování dat a komunikaci mezi zaměstnanci v rámci společnosti. To zahrnuje analýzu současných technologií a konfigurací používaných EFB a implementaci nové portálové webové aplikace, která je vhodná pro spuštění v jejich nastavení a prostředí. Projekt zahrnuje vytvoření procesů, databáze, webové aplikace, jednotkových testů, uživatelské dokumentace a finálního vydání, které EFB má použít.

Klíčová slova Egyptská potravinová banka, webová aplikace portálu, správa dat, správa svátků, správa dopisů HR, správa reklamací IT

Abstract

This bachelor thesis aims to create a Portal Web Application for the Egyptian Food Bank (EFB) to be used as a platform for data management and communication between the employees within the company. This includes analyzing the current technologies and configurations used by the EFB and implementing a new Portal Web App from scratch that is suitable to run within their setup and environment. The project includes the creation of processes, a

database, a web app, unit tests, user documentation, and a final release to be used by the EFB.

Keywords Egyptian Food Bank, Portal Web App, Data Management, Holiday Management, HR Letter Management, IT Complaints Management

Contents

List of Figures

Introduction

In today's world, companies and organizations are required to provide relevant documents and information to their partners, employees, and/or customers all over the world in order to meet the standard business requirements. Data is a crucial commodity that must be analyzed, organized, stored, and preserved in order to keep track of all the necessary information when needed, or else the relevant data gets lost or outdated. Organizations often have requirements to build specific portals for unique purposes. Whether it's to drive or enhance internal communication, manage data and tasks, or to collaborate with specific groups of people, the use cases are endless. For that reason, the EFB Portal was introduced, with the main goal of creating and delivering a simple user-based and content-based management web application with a user-friendly interface.

Portals can be used as a means of improving information and data management and communication, accessing and sharing information, and keeping track of the overall data, which can be used later on for creating statistical assessments and reports analytically. They can be personalized and configured almost freely and in any possible way, in order to fit and comply with the organization's internal hierarchical structure. A user-based and content-based portal allows different types of users, such as managers, employees, and departments to access different types of content, as well as setting different types of roles and permissions for each user. In an effort to maximize productivity and efficiency when executing certain tasks, each user is allowed to access only the relevant information and features pertaining to them only.

In brief terms, a Portal Web App serves as a single gateway through which to gain access to all the information, data, and processes used by an organization. One significant and critical aspect of the portal, is its ability to recognize and know which user is using it currently, as the portal can also be considered as a user's personal assistant. The portal is designed to present only the relevant

information and tools that each user needs, without the clutter of information and tools he/she doesn't use. The portal may look different, depending on the user's department and department. Another significant aspect of the web portal is having the ability to personalize and configure some settings within the portal, in order for the portal to be used by different organizations, with different structures, rules, and processes.

The EFB Portal is a client-oriented web application that personalizes the portal's tools and information to the specific properties and needs of the logged user, using information from organization's database. The introduction of the EFB Portal within the Egyptian Food Bank has numerous benefits. These benefits include, but are not limited to:

- moving to paperless document and data management and being more environmentally friendly by saving trees, cutting down pollution, and reducing transport,
- increasing employee productivity, by making processes and tasks less tedious to manage and complete,
- saving time and money by decreasing the amount of time needed to complete a task, hence decreasing the cost of working on that task,
- avoiding different data and information formats by having a single format for all the requests and complaints created and managed by the employees within the organization,
- creating a centralized database for storing all the necessary data and information, enabling faster and easier data lookup,
- faster communication between the employees completing the tasks, as well as keeping the users up to date with the latest information about their holiday quota, new holiday types, new HR letter types, etc.
- enabling users to keep track of their personal information, by verifying the data in their user accounts,
- enabling 24/7 access to the portal, enabling employees to create and manage their requests from anywhere and on any device,
- automating simple non-complex tasks and calculations to process and keep the data organized and up to date.

Structure

This thesis includes the following structure:

- Analysis: This section discusses the analysis of the web app, requirements, domain of the application, and analysis of the existing solutions.
  - Application Requirements: In this section, the application requirements are defined and discussed for the implementation.
  - Domain Model: In this section, the application's domain model is defined and discussed for the implementation.
  - Existing Solutions: In this section, the application is analyzed and compared to similar applications that exist on the market already.
- Design: This section describes the technologies used in the development process, architecture of the application, database model, class and sequential models.
  - Technologies: In this section, the technologies used for the application's implementation are discussed.
  - Architecture: In this section, the application's architecture and structure is discussed.
  - Database: In this section, the application's database model is discussed.
  - Class Model: In this section, the application's class models are presented and discussed.
  - Sequence Model: In this section, the application's sequence models are presented and discussed
- Implementation: This section contains the discussion of the tools used during the development process, implementation details and tests.
  - Tools: In this section, the tools used for the application's implementation are defined and discussed.
  - Implementation Details: In this section, it includes nontrivial implementations tackled during the application's implementation process.
  - Testing: In this section, the tests used in he application's implementation are discussed.
- Conclusion: This section contains the summary of the thesis.

Chapter 1

Analysis

1.1 Application Requirements

In this section, the application requirements are defined and discussed. These include a condition or capability needed by a user to achieve an objective or solve a problem within the application. Requirements analysis focuses on the tasks that determine the needs or conditions to meet for the application

The goal of the project is to create a web portal application for the EFB's employees to help them manage their data and communication between departments.

The main features of the application include:

- User Login and Management
- Management of Employee Holidays with Approval Requests
- Management of Human Resources Letter Requests
- Management of IT Complaints within the Company
- Email Notifications

The system will keep record of all data stored and modified data. A user can either be an employee or a manager, which grants them different permissions. Permissions and Roles also depend on the department in which the user belongs to. Users can edit only basic personal information such as their email address and password.

Employees can create holiday requests, HR letter requests, and IT complaints. In terms of holiday requests, the system automatically checks the availability of holiday days for each employee based on their holiday count. Any employee

from HR can approve/decline the holiday request.

Employee can create a request for different types of letters from the HR office. The request can be reviewed by any HR employee, who can approve it preliminarily or decline it. If approved preliminarily, any HR manager can view the request and approve it indicating that the requested letter was electronically signed and uploaded to the user shared directory or decline it.

Employees can create IT complaints based on the type of IT problem. Complaints can then be reviewed by all IT employees. Complaints are automatically assigned to a specific IT employee in such a manner to balance the work load between all IT employees. IT Managers can manually assign specific IT Employees for some complaints.

The web app will have an email account which would notify each user about data updates such as New Requests, Request Status Updates, and any important information. The notifications shall also be sent to the user accounts via email.

1.1.1 Non-Functional Requirements

In this section, the non-functional requirements are defined and discussed. Non-functional requirements specify criteria that can be used to judge the operation of a system in particular. Conditions, rather than specific behaviors. Non-functional requirements define how a system is supposed to be.

- NFR1: Web Application
  - The web app has the GUI for the users to interact with it. Users can add/modify/delete data based on their roles. They can also ask for approval from other users for certain tasks.
- NFR2: Backup
  - The web app will backup data on the system server as log files and keep record of any data updates.
- NFR3: Web Application Availability
  - The web app should work most of the time for data management. It should be compatible with most, if not all web browsers.
- NFR4: Localization
  - The web application will be localised in English, maybe Arabic later on after the base localization is complete and fully functional.

1.1.2 Functional Requirements

In this section, the functional requirements are defined and discussed. Functional requirements are those which are related to the technical functionality of the system. Functional requirements state how the users will interact with the application. It is something the application must do and can be testable. Functional requirements define what a system is supposed to do.

- FR1: Users Management
  - The system will keep record of users along with roles and departments and update the data as per the requests.
  - For each user, a first name, surname, date of birth, address, role, department, username, and email address must be specified.
  - Types of roles include manager and employee, while departments include departments within the EFB.
  - Users can be added, modified, and deleted only by the IT department employees.
- FR2: Holiday Request Management
  - The system will keep record of holiday quota along with holiday requests for each employee and update the data automatically.
  - For each holiday request, a date must be specified along with the type of holiday and reason
  - Types of holidays include Paid, Unpaid, and Casual. Each of them shall be treated differently. Paid holidays deduct from the holiday quota, while unpaid holidays do not.
  - The requests are reviewed and managed by the HR department employees. They can review the request and approve or decline it after consideration.
  - In case the holiday is approved, the system deducts it from the employee's holiday quota once the date has passed.
  - In case the holiday is declined, the HR employee declines the request, the employee's holiday quota doesn't get affected in and they can discard it.
  - Employees can view past holiday requests and their upcoming holiday requests. They can also delete any `Pending` Holiday request, marking it `Cancelled`.
  - Notifications about new requests, and status updates will be available via email.
- FR3: HR Letter Request Management

  - The system will keep record of HR Letter requests for each employee and update the data automatically.
  - For each HR Letter request, a type of required document must be specified along with a reason for why the employee needs that document.
  - Types of HR letters include Confirmation Letter, Attendance Sheet, Course Sponsorship, Contract, etc.
  - The requests are reviewed and managed by the HR department employees. They can review the request and preliminarily approve or decline it after consideration.
  - After preliminary approval, the request is reviewed by any HR manager, where they can either decline it, or approve it indicating that the the requested document was electronically signed and uploaded to the user's shared directory.
  - Employees can view past HR letter requests and their upcoming HR letter requests. They can also delete any `Pending` HR Letter request, marking it `Cancelled`.
  - Notifications about new requests, and status updates will be available via email.

- FR4: IT Complaint Management
  - The system will keep record of IT complaints for all employees and update the data automatically.
  - For each IT complaint, a description of the problem needs to be specified along with the type of problem.
  - Types of IT Complaints include hardware, software, and other
  - The complaints are assigned to different IT employees based on work load and availability. IT Managers can manually assign IT employees to some complaints.
  - The requests are reviewed and managed by the IT department employees. They can review the request.
  - IT employees fix the problem and update the status of the IT complaint to `Fixed` after testing.
  - Employees can view past IT complaints and their new IT complaints. They can also delete any `Pending` IT complaint, marking it `Cancelled`.
  - Notifications about new IT complaints, and status updates will be available via email.

1.2 Domain Model

This section describes the domain of the portal, defining the properties and relations of the important objects of the domain. See Figure 1.1.

Each user works at one Department and can be either an Employee or a Manager there. Each Department and Job Position allows the User to have different permissions and roles, hence accessing and viewing different data. Each Department has list of Users and has a type, indicating if its HR, IT, or OTHER.

Holiday Requests are created by any User, and can be evaluated by any User who works at the HR Department (HR Employees). If the request is approved the holiday request creator's holiday quota gets updated based on the work day count. Each Holiday Request can have only one type of Holiday. Each Holiday, has a type indicating which holiday quota it belongs to.

HR Letter Requests are created by any User and can be preliminary evaluated by any HR Employee, and if its preliminary approved, it gets evaluated by any HR Manager who can approve it indicating that the requested letter was electronically signed and uploaded to the user's shared directory. Both HR Employees and Manager can decline the request. Each HR Letter Request can have only one type of HR Letter.

IT Complains are created by any user and handled by any IT User. They have different types of complaints to be submitted. Any IT User can view the complaints. Each IT Complaint is assigned to one IT User, in order to balance the work load. Each IT Complaint can have only one type of complaint.

1.3 Existing Solutions

In this section, existing solutions on the market are discussed and compared to the EFB Portal application. This chapter includes defining the alternatives, analyzing them, realizing the pros and cons of the existing solutions, and reaching a logical conclusion.

There are numerous portal web app providers on the market. The best ones include Unily [1], Samepage [2], Bitrix24 [3], and Speakap [4]. These services offer companies the ability to create their own portal web apps through the use of design templates and features provided by the service. Most of them include user-friendly designs and many features ranging from CRMs and Project Management to Human Resources and Marketing, while also providing the ability to integrate widely used apps such as Microsoft, Google, Zendesk, Slack, and

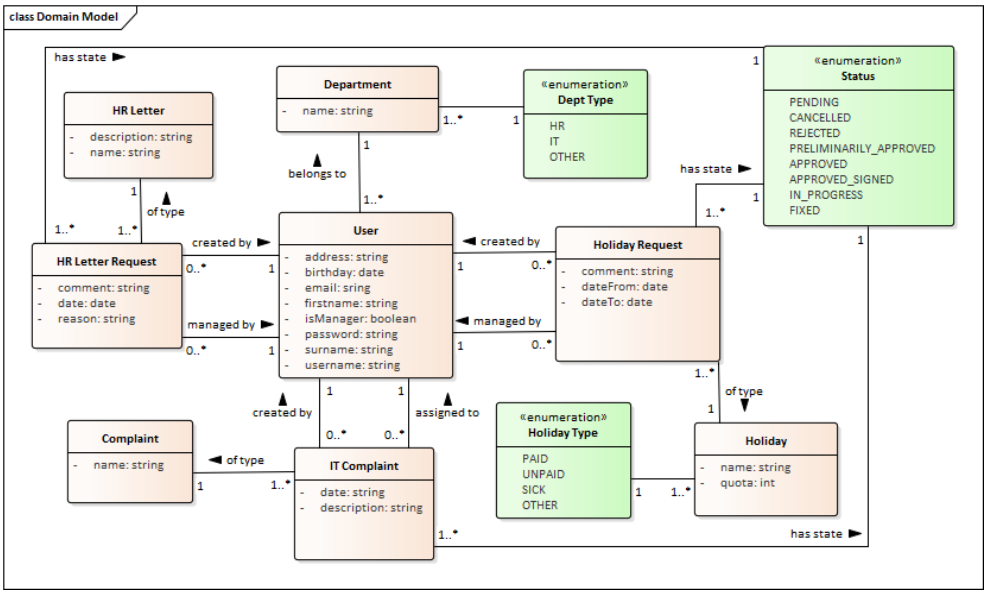

Figure 1.1: Domain Model

much more.

These services provide a wide range of features which are useful for any company, however one of their main problem is their pricing plans. Some of them offer Free packages, such as Bitrix24 and Samepage. However, those packages either have very limited features or have a maximum amount of users; i.e. 5-10 users. For a large non-profit organization like EFB, the free packages are not suitable. While the Premium or Unlimited Packages are more suitable, their complexity and usability becomes much harder due to all the features available. They are also very expensive to maintain and keep running for many years. One of the main reasons, the EFB wanted their own portal app was to reduce the complexity and scatter-ness of data everywhere due to the fact that some of the employees have a tough time using technological services and apps.

The EFB Portal offers a simple and elegant user-friendly design, that is easily understood by all employees. It shall provide only all the necessary features required by the EFB and will be very easy to maintain and understand. The UI design takes into consideration the employees' limitations and will be as minimalistic as possible, while providing all necessary data in a readable-format and all actions using simple step-by-step forms. The EFB Portal will be modifiable and maintainable by the EFP's IT employees, as User-Manuals are provided and offer easy steps to follow for any updates and modifications within the system.

Chapter 2

Design

2.1 Technologies

In this section, the process and technologies used during the development are discussed along with stating the reasons for why each technology was used.

2.1.1 Programming Language

The programming language used to develop the EFB Portal was chosen as Java.[5] This choice was made by taking into consideration several reasons noted below.

- Object-Oriented
  Java allows you to create and form standard programs and reusable code, which helps bind together the data and the functions that operate on them so that no other part of the code can access this data except that function along with implementing real-world entities like inheritance, hiding, polymorphism, etc.

- Secure
  Java has no explicit pointer. Apart from this, it has a security manager that defines the access of classes.

- Extensibility
  Java is simply extensible for web app development, helping ease the process of project scaling. It helps with the long-term projects, as the code can be easily modified over and over, without causing bugs.

- Frameworks
  There are various frameworks available for Java web development which make the development process faster and easier. Moreover, every popular framework has a huge amount of all needed documentation, support and community.

- Platform Independent
  Java code runs on any machine that doesn't need any special software to be installed, but the JVM needs to be present on the machine. This makes the code easily portable.

- Garbage Collection
  Java can easily handle garbage data, if the pile is filled, then the garbage is collected and destroyed. Which again makes this language appropriate, for instance, in terms of application scaling process.

- Experience
  I have more than 2 years of experience using Java, making it the most reasonable and obvious choice as I am much more capable of designing and programming the project in this language than any other language. It is essential that the developer is experienced with the language of development, or it will be much harder and will take a longer time to familiarize yourself with another programming language.

2.1.2 Frameworks and Libraries

- EclipseLink (JPA 2.1)
  The EclipseLink JPA provides developers with a standards based Object-Relational persistence solution with additional support for many advanced features. EclipseLink JPA provides advanced support for leading relational databases and Java containers.The software provides an extensible framework that allows Java developers to interact with various data services, including databases, web services, Object XML mapping, and Enterprise Information Systems. [6]

- Spring
  The Spring Framework is an application framework and inversion of control container for the Java platform. The framework's core features can be used by any Java application. The Spring Framework provides a comprehensive programming and configuration model for modern Java-based enterprise applications - on any kind of deployment platform.[7]
  Spring makes building web applications fast and hassle-free. By removing much of the boilerplate code and configuration associated with web development, you get a modern web programming model that streamlines the development of server-side HTML applications, REST APIs, and bidirectional, event-based systems.

  – Spring Boot
    Spring Boot is an open source Java-based framework used to create a micro Service. Spring Boot provides a good platform for Java developers to develop a stand-alone and production-grade spring

application that you can just run. The main goal of Spring Boot Framework is to reduce Development, Unit Test and Integration Test time and to ease the development of Production ready web applications very easily compared to existing Spring Framework, which really takes more time.

  - Spring Security
    Spring Security is a powerful and highly customizable authentication and access-control framework. It is the de-facto standard for securing Spring-based applications. Spring Security is a framework that focuses on providing both authentication and authorization to Java applications. Like all Spring projects, the real power of Spring Security is found in how easily it can be extended to meet custom requirements
  - Spring Mail
    Spring framework provides many useful interfaces and classes for sending and receiving mails. The org.springframework.mail package is the root package that provides mail support in Spring framework.

- Thymeleaf
  Thymeleaf is a Java XML/XHTML/HTML5 template engine that can work both in web and non-web environments. It is better suited for serving XHTML/HTML5 at the view layer of MVC-based web applications, but it can process any XML file even in offline environments. It provides full Spring Framework integration.[8]

2.1.3 Database

MySQL is the world's most popular open source relational database management system that uses Structured Query Language. It is commonly used in web applications due to its speed, flexibility and reliability. MySQL employs SQL, or Structured Query Language, for accessing and processing data contained in databases.[9] It was chosen due to its scalability, flexibility, high performance, and high availability. As well as its ease of management and overall comprehensive application development, meaning that it provides support for every application development need.

2.1.4 Web Server

The Apache Tomcat software is an open source implementation of the Java Servlet, JavaServer Pages, Java Expression Language and Java WebSocket technologies. It is designed to execute Java Servlets and render web pages that use Java Server Page (JSP) coding.[10] It was used for the EFB Portal application mainly because its a free open-source software. Moreover, it is

highly flexible and lightweight, meaning that you can run it in virtually any fashion you choose, and it'll still work as intended as well as providing relatively quick load and redeploy times. Depending on the configuration, Tomcat can be used as an extra layer of security when installed behind a firewall.

2.1.5 PaaS

Platform as a service (PaaS) is a complete development and deployment environment in the cloud, with resources that enable you to deliver everything from simple cloud-based apps to sophisticated, cloud-enabled enterprise applications. You purchase the resources you need from a cloud service provider on a pay-as-you-go basis and access them over a secure Internet connection. PaaS includes infrastructure—servers, storage, and networking—but also middleware, development tools, business intelligence (BI) services, database management systems, and more. PaaS is designed to support the complete web application lifecycle: building, testing, deploying, managing, and updating.

The main advantages of using PaaS include, but are not limited to, cutting coding time, adding development capabilities without adding staff, developing for multiple platforms more easily, efficiently managing the application lifecycle, and using sophisticated tools using a pay-as-you-go model. This technology was used for the deployment of the test versions and final prototype version of the EFB Portal application.

2.1.6 DBaaS

Database as a service (DBaaS) (also known as managed database service) is a cloud computing service that lets users access and use a cloud database system without purchasing and setting up their own hardware, installing their own database software, or managing the database themselves. The cloud provider takes care of everything from periodic upgrades to backups to ensuring that the database system remains available and secure 24/7.

The main advantages of using DBaaS include, but are not limited to, cost saving, application scalability (up or down), rapid development and configuration, and data security. The EFB Portal is connected to a PaaS, while also being connected to a cloud DB as a form of DBaaS.

2.2 Architecture

The EFB Portal system consists of a central database and the Java Web Application. The application and the database can run on different servers or share the same, depending on the client's needs. Also, the web application and database can be deployed on a PaaS or live in the local server of the company.

Possible deployment model demonstrated in Figure 2.1. The application ar-

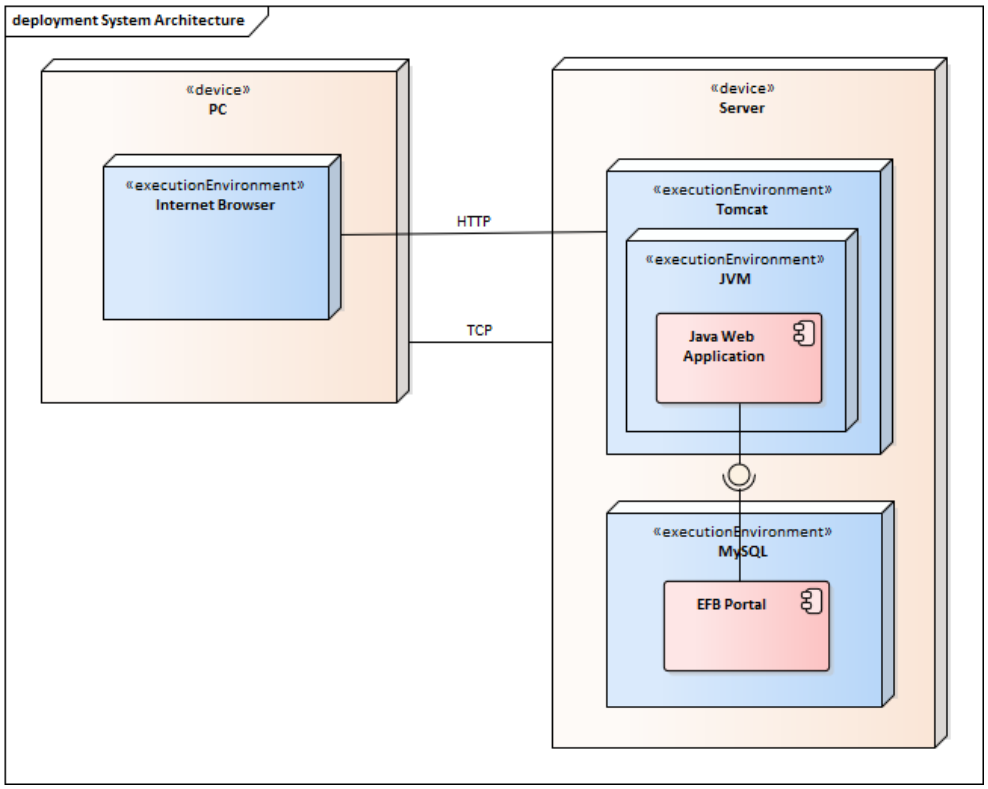

Figure 2.1: Deployment Model

chitecture is based on the 3-layers MVC architecture, as shown in Figure 2.2. A 3-layer architecture is a client-server architecture in which the functional process logic, data access, computer data storage and user interface are developed and maintained as independent modules on separate platforms. There are various reasons for why this architecture was chosen. Firstly, it provides the ability to update the technology stack of one layer, without impacting other areas of the application. It also allows for different development teams to each work on their own areas of expertise. The architecture also eases the process of scaling an application. By separating the layers, it makes it possible to deploy to a variety of databases instead of being locked into one particular technology. This all helps with providing an ease of maintenance of the code, managing presentation code and business logic separately, while adding more independence on the underlying servers and services and code reliability.

The three layers are defined as follows:

- Presentation Layer - The layer responsible for presentation of data to the user, by sending content to browsers in the form of HTML/JS/CSS, as well as processing and validating user input.

- Application Layer- The layer responsible using an application server and processing the business logic for the application.

- Data Layer - The layer responsible for the database management system that provides access to application data.

The layers are isolated using five interfaces:

- Presentation Layer:
  - Views - Resources directory that includes HTML, CSS, and JS files for each web page.
  - RESTController - Interface between the Presentation and Business Layers. Responsible for sending the correct data and rendering the presentation layer.

- Business Layer:
  - ControllerInterface - Interface between the Presentation and Business layers. It consists of several concrete Controller interfaces for processing individual parts of the business logic.

- Data Layer:
  - DAOs - Public Interface that extends the CRUDRepository of each Entity, and allows for managing the entity's data.
  - Entities - Entity class files essentially for object wrapping for each associated database table .

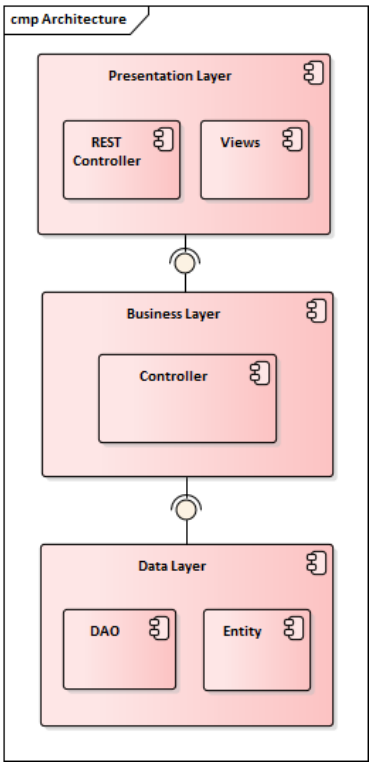

Figure 2.2: Architecture

2.3 Database

In this section will be shown and discussed the database as shown in Figure 2.3 The database consists of 9 entities, each with their own attributes and responsible for storing data used within the web app.

User is an entity that represents the registered users on the system. Only IT Managers are allowed to create new users. It includes the following attributes:

- `firstname` - Given First Name
- `surname` - Given Surname
- `birthdate` - Date of Birth
- `address` - Street Address
- `department` - Department in which User works at
- `email` - Email Address

- `username` - Login Username
- `password` - Login Password
- `paid quota` - Paid Holiday Quota
- `unpaid quota` - Un-Paid Holiday Quota
- `sick quota` - Sick Holiday Quota
- `isManager` - User's Role (true = manager, false = employee)

Department is an entity that represents the departments within the Egyptian Food Bank. Each department has a unique name and type, represented by string enumeration values. It includes the following attributes:

- `name` - Name of Department
- `surname` - Type of Department (ENUM)

HR Letter Request is an entity that represents the HR Letter requests that are on the system. Any user can create a request, and only HR employees can approve/reject it. It includes the following attributes:

- `date` - Date of Request
- `type` - Type of HR Letter Document
- `reason` - Purpose of Document Request
- `comments` - Additional Comments (Optional)
- `status` - Status of the Request (ENUM)
- `creator` - User who Requested
- `manager` - User who Managed (Approved/Rejected)

HR Letter is an entity that represents the HR Letter types that are on the system. Only HR User's and IT Managers can modify this data. It includes the following attributes:

- `name` - Type of Document
- `description` - Brief Description of Document

Holiday Request is an entity that represents the Holiday requests that are on the system. Any user can create a request, and only HR employees can approve/reject it. It includes the following attributes:

- `dateFrom` - Start Date of Holiday

- `dateTo` - End Date of Holiday
- `type` - Type of Holiday
- `daysCount` - Number of Working Days within the requested Period
- `comments` - Additional Comments (Optional)
- `status` - Status of the Request (ENUM)
- `creator` - User who Requested
- `manager` - User who Managed (Approved/Rejected)

Holiday is an entity that represents the Holiday types that are on the system. Only HR User's and IT Managers can modify this data. The type attribute represents enumeration values that correspond to which type belongs to which quota. It includes the following attributes:

- `name` - Title of Holiday
- `description` - Brief Description of Holiday
- `quota` - Limit of Days for Holiday Type
- `type` - Type of Holiday (ENUM)

State Holiday is an entity that represents the State Holidays that are on the system. Only HR User's and IT Managers can modify this data. It includes the following attributes:

- `name` - Name of State Holiday
- `dateFrom` - Start Date for Holiday
- `dateTo` - End Date for Holiday

IT Complaint is an entity that represents the IT complaints that are on the system. Any user can create a complaint, and only IT User's can modify it. It includes the following attributes:

- `date` - Date of Complaint
- `type` - Type of IT Complaint
- `description` - Brief Description of Complaint
- `status` - Status of the Complaint (ENUM)
- `creator` - User who Complained

- `assignee` - User who is Assigned

Complaint is an entity that represents the Complaint types that are on the system. Only IT User's can modify this data. It includes the following attributes:

- `name` - Type of Complaint
- `itComplaints` - History of IT Complaints

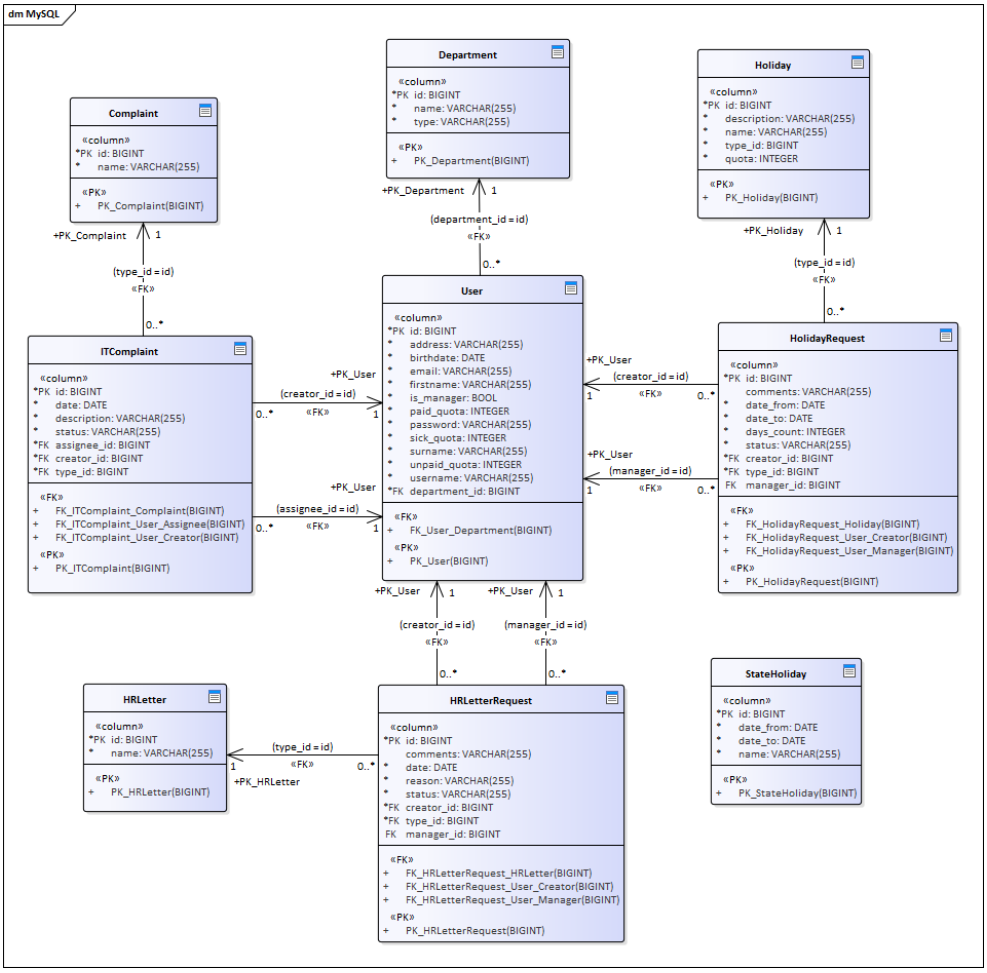

Figure 2.3: Database Model

2.4 Class Model

In this section will be shown examples of classes using class models.

2.4.1 Overview

This subsection covers the overview of the implemented sources to help with understanding the project structure, packages, and classes. The project's code structure could be seen in Figure 3.1, and is described in more detail in Section 3.2.2. The business logic is located within the `controllers` package, while the presentation layer is located within the `views` package. The REST-Controllers within the views package allow the system to recognize any user request sent while the application is running, such as `GET` and `POST` mappings, which are linked to methods. These methods allow the system to send the necessary data to the the correct business logic controller method, which validates the data and sends back the result. The result is then checked and the method returns the name of the web-page located in the `/resources/views` directory, so that the user is presented with the correct web-page and result.

2.4.2 HR Letter Controller Class

The HR Letter Controller Class acts as one of the main functionality classes in the project. The class implements the HR Letter Interface containing all the functionality related the HR Letter processes. The class is used to verify and communicate data between the presentation layer and the data layer. The HR Letter Controller Class Diagram can be seen in Figure 2.4

2.4.3 Holiday Controller Class

The Holiday Controller Class acts as one of the main functionality classes in the project. The class implements the Holiday Interface containing all the functionality related the Holiday processes. The class is used to verify and communicate data between the presentation layer and the data layer. The Holiday Controller Class Diagram can be seen in Figure 2.5

2.4.4 IT Complaint Controller Class

The IT Complaint Controller Class acts as one of the main functionality classes in the project. The class implements the IT Complaint Interface containing all the functionality related the IT Complaint processes. The class is used to verify and communicate data between the presentation layer and the data layer. The IT Complaint Controller Class Diagram can be seen in Figure 2.6.

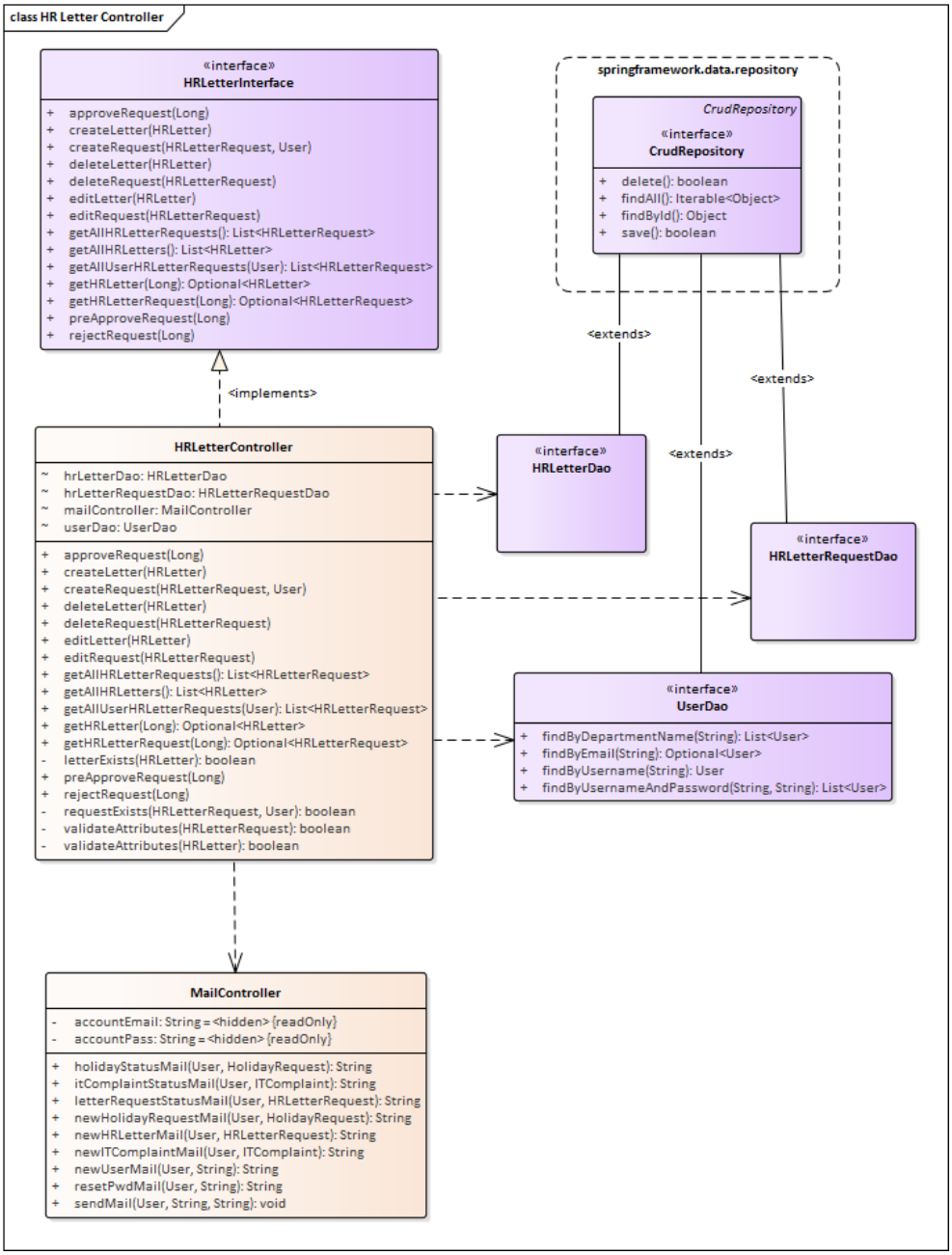

Figure 2.4: HR Letter Controller Class

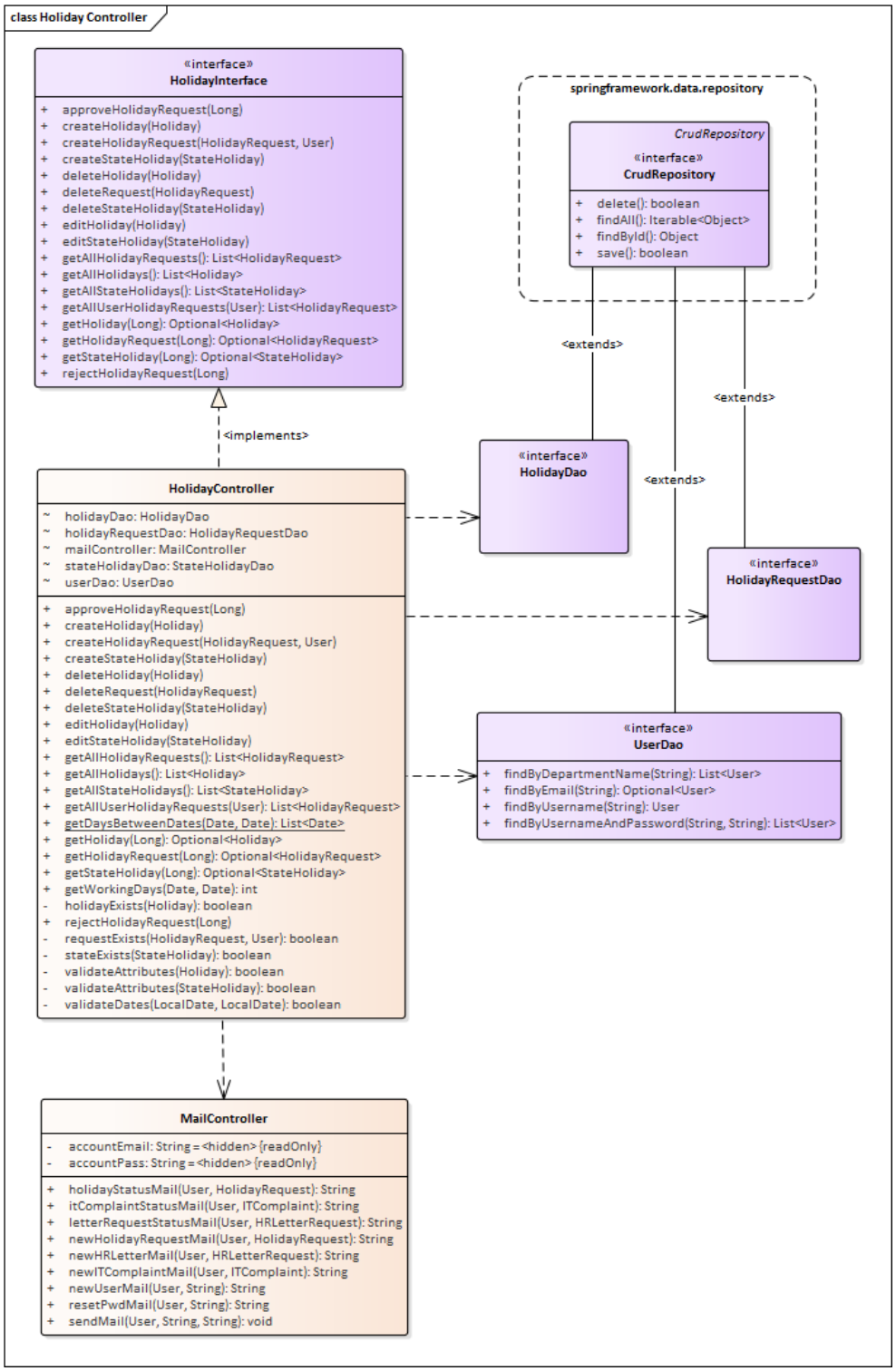

Figure 2.5: Holiday Controller Class

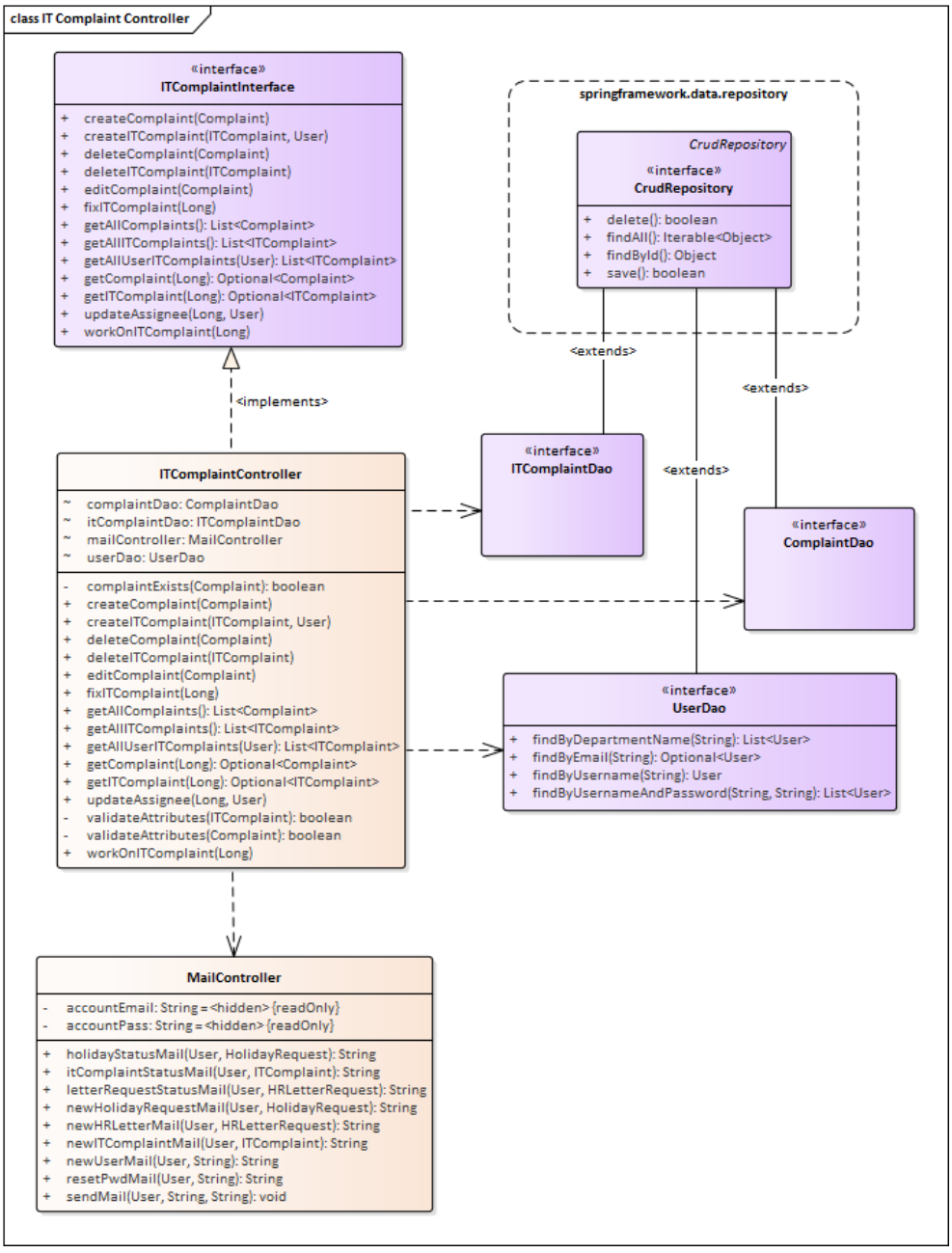

Figure 2.6: IT Complaint Controller Class

2.5 Sequence Model

This section contains the description of two use case scenarios, and a sequence diagram for one of the sampled use case scenarios to provide a visualization. The Create HR Letter Request and Manage Holiday request scenarios were chosen to be explained.

2.5.1 Create HR Letter Request

The Create HR Letter Request Sequence Model is shown in Figure 2.7. This sequence can only be triggered after the user has successfully logged into the web app and accessed their Personal HR Letter Requests web-page. User clicks on 'Create Request' button. System loads and displays the Create HR Letter Form web-page. User fills in all the required fields including Letter Type, Reason, and any Additional Comments if necessary. User clicks on the 'Create' button. System checks if all required fields are filled and continues with processing the data. System confirms validity of entered data first, and then adds the provided HR Letter Request to the database. System notifies the User by sending an email with information about the newly created request. System then redirects the User back to their Personal HR Letter Requests web-page, where the User can view all their personal requests. Alternatively, the system can show an error while processing the data and display it to the User in the form of an alert or the User can also cancel the HR Letter Request creation by clicking on the 'Cancel' button, where the System redirects the User back to their Personal HR Letter Requests web-page.

2.5.2 Manage Holiday Request

The Manage Holiday Request sequence can only be triggered after the user has successfully logged into the web app and accessed the Holiday Requests Management from within the Control Panel. Only Users from within the HR Department and IT Managers can access that web-page. System displays list of all Holiday Requests within the database. User will select one of the requests to manage and click on the 'View' Button found next to the selected request. System loads and displays the Edit Holiday Request Form web-page. User checks and verifies data and then clicks on the 'Approve' button. The system calculates the amount of working days within the requested holiday period; taking into consideration weekends and state holidays, and subtracts it from the Creator's holiday quota. System then updates the status of the request to 'Approved' and notifies the Creator via Email about the status update of their request. Alternatively, the User can reject the request by clicking on the 'Reject' button. In that case, the system updates the status the request to 'Rejected' and notifies the Creator via Email about the status update of their request.

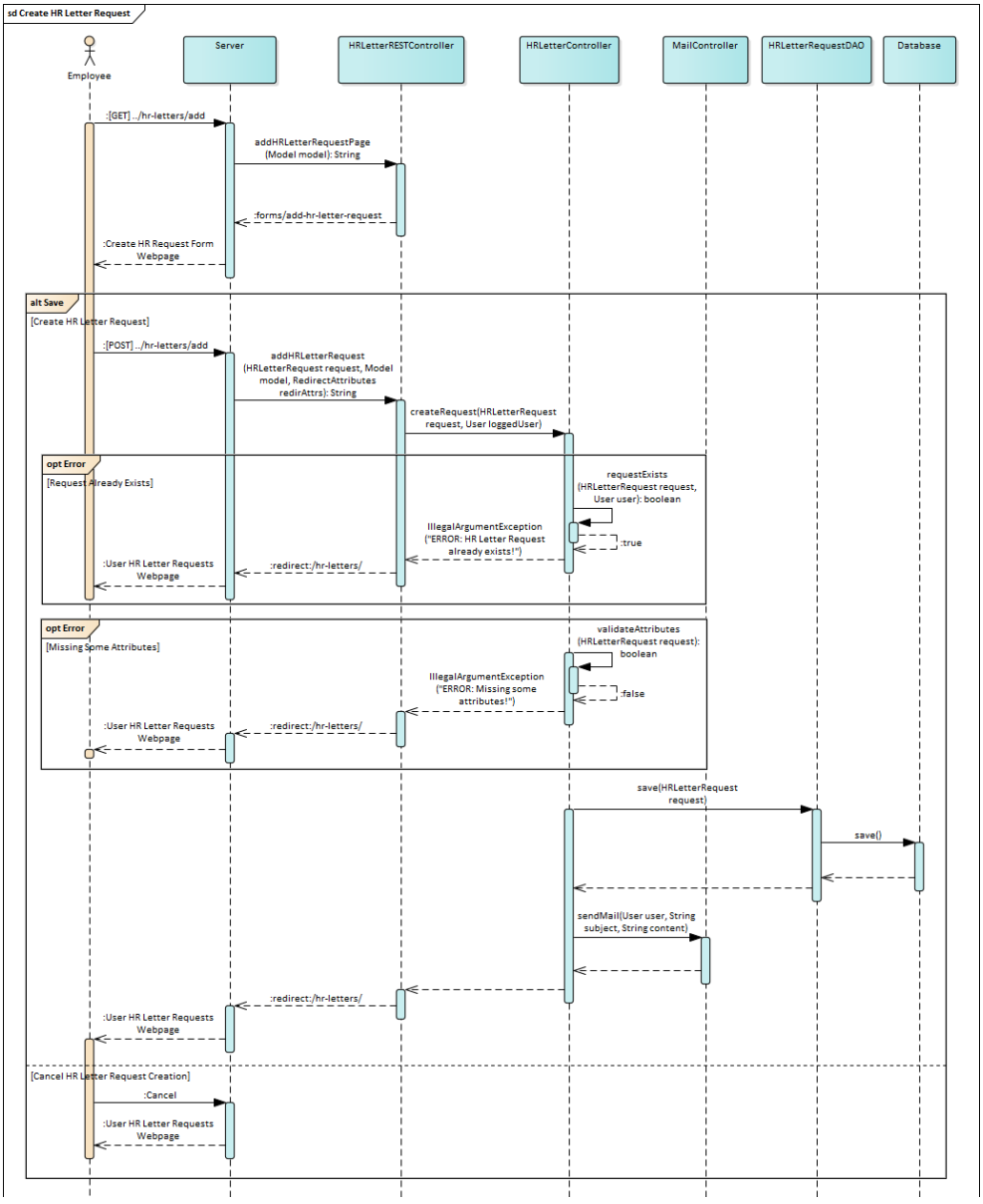

Figure 2.7: Create HR Letter Request Sequence Model

Chapter 3

Implementation

In this chapter the tools choice, implementation details and testing of the EFB Portal application are discussed.

3.1 Tools

This section contains the list of tools used during the implementation process.

- JDK 1.8
  The JDK is a development environment for building applications, and components using the Java programming language. It includes tools useful for developing and testing programs written in the Java programming language and running on the Java platform. [11]

- Maven
  Maven is a powerful project management tool that is based on POM (project object model). It is used for project build, dependency and documentation.[12]

- JUnit
  Junit is widely used testing framework along with Java Programming Language. It is used as automation framework for unit testing.[13]

- Github VCS
  Github Version Control System is used to keep track of the changes during the software development process.[14]

- Github CI/CD
  Github Continuous Integration/Continuous Delivery. CI packages and tests software builds and alerts if the recent changes failed any unit tests. CD is the automation that delivers changes to infrastructure. Is used for building and testing every commit to GitLab and also for deployment to test environment.[14]

- Heroku
  Heroku provides a platform as a service (PaaS).[15] The benefits and the reason of using PaaS were discussed in the Technologies Section 2.1.2.

- ClearDB
  ClearDB provides a database as a service (DBaaS).[16] The benefits and the reason of using DBaaS were discussed in the Technologies Section 2.1.2.

- XAMPP
  XAMPP is a tool that provides the Apache web server, MySQL, phpMyAdmin and control over this software.[17] This tool ease the development process.

- phpMyAdmin
  phpMyAdmin is used as a database administration tool.[18]

- Sequel Pro
  Sequel Pro is a full-featured administration tool for database.[19] This tool is used for connection and manipulation with ClearDB.

- Apache Netbeans 11.1
  NetBeans is an integrated development environment (IDE) for Java. NetBeans allows applications to be developed from a set of modular software components called modules. NetBeans runs on Windows, macOS, Linux and Solaris.[20] This was used as a main IDE.

3.2 Implementation Details

In this section will be discussed several key points faced during the implementation process.

3.2.1 Organization's Special Requirements

One of the major things to be noted, is the Egyptian Food Bank's special requirements as per their organization. The implementation of the business processes and use-cases must follow and abide by the organization's requirements. For that reason a few points must be noted. Firstly, it was requested that user's should not be able to edit their personal information, as the data must be verified based on official documents. For that reason, only IT employees can add users and edit their personal information. Users are only allowed to update their email address and password. In regards to the Holiday Requests, users can only submit holiday requests three days in advance, in order to give the HR employees time to process the request.

Moreover, half-day holidays are not allowed within the organization, for that reason only the dates must be specified, without taking into consideration the time. The EFB also wanted to continue using their local shared Employee directories on their network to upload the electronically signed HR Letters. For that reason the system allows only HR managers to approve the HR Letter request marking it as 'Signed and Uploaded', while HR employees can preliminary approve the request marking it 'Preliminary Approved'. Finally, the IT Complaints were implemented in to distribute the complaints among the all users of the IT department in order to balance the work load. However, IT managers are allowed to manually update the assigned user to another user, as long as the IT complaint is in the 'Pending' state.

3.2.2 Code Structure and Architecture

The project is structured as shown in Figure 3.1. This structure was created to follow the rules of the 3-layer architecture. The data layer is represented by the `data` package includes two sub-packages `models` and `DAO`, which contain the database entity classes and DAO Interfaces respectively. The business layer is represented by the `controllers` package, which contains the logic controllers and their implemented interfaces. The presentation layer is represented by the `views` package, which contains the REST controllers, while the HTML web-pages are located within the `views` directory found inside the `resources` directory. Moreover, the main package includes the Main Spring Boot application class, and the `config` package includes configuration class files that are primarily used for configuring the authentication and security aspect of the application.

It is important to note that the project includes different packages each responsible for handling specific functions for each layer of the architecture. The REST Controllers were used to handle the communication between the presentation layer and business logic layer, while the Logic Controllers were used to handle the communication between the business logic and data layer, as well as returning the response to the REST Controller to be presented to the User. While the separation of the REST Controllers and Logic Controllers is not necessary, it was done to conform to the rules of robust design. The separation of these two packages helps when changing or updating the code, as the developer is able to change one file without affecting the other files. Moreover, it makes it easier for new developers to use new technologies if needed, and update the necessary file without needing to update the technology used for the other files. The data layer is also divided into two packages, one that includes the database entity classes, while the other includes the DAO classes, responsible for implementing CRUD functions for each entity.

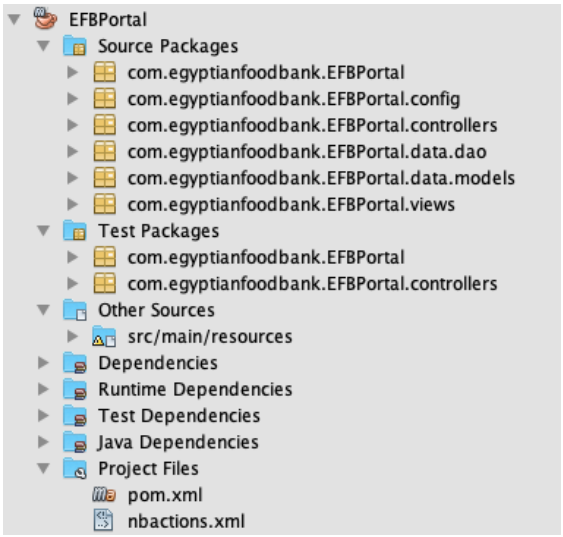

Figure 3.1: Project Code Structure

3.2.3 Maven

Maven was introduced to the application in order to help ease the process of the build script. By defining the dependencies required in the `pom.xml` (Project Object Model) file, maven automates the process of downloading the dependency jar files and importing them to the project. Moreover, by defining the parameters needed for building the application package within the pom file, and executing the build command, maven creates the JAR artifact of the application in the target directory of the project. Maven requires internet to download the jars, which is a one time process that happens during the build phase of the application. The code snippet below shows part of the project's `pom.xml` file which includes some of the application package build parameters and some of the dependencies used along with the Spring Boot framework for the project.

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>2.3.2.RELEASE</version>
</parent>
<groupId>com.egyptianfoodbank</groupId>
<artifactId>EFBPortal</artifactId>
<version>1.0-SNAPSHOT</version>
<name>EFBPortal</name>
<description>Portal Web App for the Egyptian Food Bank</description>
...
```

```
<dependencies>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-test</artifactId>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.springframework.security</groupId>
        <artifactId>spring-security-web</artifactId>
    </dependency>
    <dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-mail</artifactId>
     </dependency>
     ...
</dependencies>
...
```

3.2.4 EclipseLink/JPA

One of the main reasons why EclipseLink JPA was chosen was because it provides standards based Object-Relational persistence solution with additional support for many advanced features for leading relational databases and Java containers. EclipseLink is JPA's reference implementation and implements JPA version 2.2. The Java Persistence API (JPA) was chosen due to its rich features provided within its specification. This includes bootstrapping and basic entity mappings, as well as mapping associations. By allowing you to map associations between database tables to entity attributes, this made designing and controlling the database much simpler, as mapping associations were easily implemented in the code by adding one line of code before any entity attribute and defining a simple annotation such as @ManyToOne, @OneToMany, @ManyToMany, etc.
The code snippet below shows the `Holiday Request` entity class, which includes map associations between `Holiday Request` and `Holiday` type, as well as `Holiday Request` and `User`.

```
@Entity
public class HolidayRequest implements Serializable {

    private static final long serialVersionUID = 1L;
```

```
@Id
@GeneratedValue(strategy = GenerationType.IDENTITY)
private Long id;
@NotNull
@DateTimeFormat(pattern = "yyyy-MM-dd")
private LocalDate dateFrom;
@NotNull
@DateTimeFormat(pattern = "yyyy-MM-dd")
private LocalDate dateTo;
@NotNull
@OneToOne
private Holiday type;
@NotNull
private Integer daysCount;

private String comments;
@NotNull
@Enumerated(EnumType.STRING)
private Status status;
@NotNull
@OneToOne
private User creator;
...
```

3.2.5 DAO Interfaces

In order to provide an abstract interface for each of the database entities to be able to complete data operations, a DAO interface must be realized. Since the project is using the Spring framework, the `CrudRepository` was chosen to help ease the process of the creation of the DAO interfaces. This is a Spring Data interface for generic CRUD operations on a repository of a specific type. It provides several methods out of the box for interacting with a database. It contains methods such as `save`, `findById`, `delete`, `count` etc. Spring boot automatically detects the JPA repositories and matches them with the entity names, as well as having a simple syntax when creating additional methods. For example, the `findByUsername` method is a custom method, yet it is still understood by the application and functions as intended. This is because custom methods are allowed within the CrudRepositpry, as long as they match with the rules and key terms. Methods can start with key terms such as find, read, query, get, count, and end with the names of the attributes to be accessed. The example above should return a User object if found, by searching the users data and finding the user with the given username. The extension of the CrudRepository for the DAO Interfaces meant that we can eliminate the need and use of SQL queries.

3.2.6 Github CI/CD

Another great feature provided by Maven, is having a simple process for preparing the execution environment and separating the different stages. This made the integration of the Continuous Integration/Continuous Delivery (CI/CD) process into the project a very simple process including one file with a mere few lines of code. The code mainly includes two stages: a build stage, which is run via the 'mvn compile' command and is responsible for compiling and building the application for production, and a test stage, which is run via the 'mvn test' command and is responsible for cleaning, compiling, and running the unit tests defined within the project. The code snippet below shows the code used for the CI/CD process.

```
build:
     stage: build
     script:
          - mvn compile
test:
     stage: test
     script:
          - mvn dependency::tree
          - mvn clean compile
          - mvn install -DskipTests
          - mvn test
```

3.2.7 Security

In terms of Security, only basic measures were taken during the implementation since the Java compiler converts the Java code into byte code (.class file) and these byte codes are then run by Java Virtual Machine (JVM). The JVM can then take care of the security, when the byte codes are executed. The basic security measures include the use of private variables and methods in order to allow restrictive access and ensure they are not overridden. As well as providing a strong form of encapsulation abiding by the principles of OOP. Moreover, users are assigned roles based on their role and department, allowing restrictive access to certain content and functions.

As the application includes a login form to authenticate users before accessing the portal, the user passwords must be stored in the database, therefore encryption must be used. BCrypt was used as a password-hashing function based on the Blowfish cipher. When a new user is created, a randomly generated string is created and then encrypted using bcrypt creating the user's password. The user's login credentials are then sent to their email in plain text format. Upon logging in, the password string inputted by the user is encrypted using bcrypt and then compared to the encrypted password in the

database. If both password-hashes correspond to the same text string, then it's a match, or else it is not. All of this ensured that the system met the basic security measures needed to guarantee production use.

3.2.8 Logging

As a means of easing the troubleshooting part of the application, logging was integrated within the application. The system keeps daily logs of all requests sent and their responses. While the logs do not provide too much details, it provides enough information to understand the problem and monitor the acts that led to this problem. The log files help provide visibility into how the application is running and who is doing what on the portal. The text snippet below is taken from a `log.log` file created during the implementation process.

```
2021-Dec-05 10:20:22.817 TRACE c.e.E.v.MainRESTController -
     [VIEW] Guest visited Login Page
2021-Dec-05 10:20:35.510 TRACE c.e.E.v.MainRESTController -
     [VIEW] Abrasheed visited Dashboard Page.
2021-Dec-05 10:22:16.625 TRACE c.e.E.v.HolidayRESTController -
     [VIEW] Abrasheed visited Holiday Types CP Page.
2021-Dec-05 10:22:20.752 TRACE c.e.E.v.HolidayRESTController -
     [VIEW] Abrasheed visited Add Holiday Type CP Form.
2021-Dec-05 10:22:36.935 INFO  c.e.E.v.HolidayRESTController -
     [ADD] Holiday Type 'Other' added by 'Abrasheed' successfully!
2021-Dec-05 10:22:36.975 TRACE c.e.E.v.HolidayRESTController -
     [VIEW] Abrasheed visited Holiday Types CP Page.
2021-Dec-05 10:23:25.594 TRACE c.e.E.v.ITComplaintRESTController -
     [VIEW] Abrasheed visited IT Complaints Page.
2021-Dec-05 10:23:58.664 INFO  c.e.E.v.ITComplaintRESTController -
     [DELETE] Abrasheed deleted IT Complaint '2' successfully.
2021-Dec-05 10:24:27.765 TRACE c.e.E.v.ITComplaintRESTController -
     [VIEW] Abrasheed visited IT Complaints CP Page.
2021-Dec-05 10:24:30.504 TRACE c.e.E.v.ITComplaintRESTController -
     [VIEW] Abrasheed visited Edit IT Complaint '16' CP Form.
2021-Dec-05 10:24:34.603 INFO  c.e.E.v.ITComplaintRESTController -
     [UPDATE] IT Complaint '16' updated Assignee to 'Abrasheed' successfully.
2021-Dec-05 10:25:16.213 TRACE c.e.E.v.ITComplaintRESTController -
     [VIEW] Abrasheed visited IT Complaints CP Page.
2021-Dec-05 10:25:19.729 TRACE c.e.E.v.ITComplaintRESTController -
     [VIEW] Abrasheed visited Edit IT Complaint '16' CP Form.
2021-Dec-05 10:25:35.434 INFO  c.e.E.v.ITComplaintRESTController -
     [UPDATE] IT Complaint '16' marked as 'Fixed' by 'Abrasheed' successfully!
```

3.3 Testing

In this section the unit testing and manual user testing of the EFB Portal application are discussed.

3.3.1 Unit Tests

Unit Testing is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. In procedural programming, a unit may be an individual program, function, procedure, etc. In object-oriented programming, the smallest unit is a method, which may belong to a base/ super class, abstract class or derived/ child class. In this project, the unit testing is done with the help of JUnit as discussed in Section 3.1. Upon building the project, each unit test is run, and if any test fails, the build fails, if not, the build success. This allows us to see if there is any bugs or issues within the code, which can be caused by an update in the database, or the addition of a newly implemented feature, or any update within the code. In other words, the tests are used as an indication of how well the application works as intended, as well as a method for detecting problems within the project. Currently, there are over a hundred unit tests, which cover all the data methods within each one the controller classes. These methods include but are not limited to, the creation, modification, and deletion of Users, Departments, Holiday Requests, HR Letter Requests, IT Complaints.
The code snippet below shows a sample unit test method, for testing the `createStateHoliday()` method, of the `HolidayController` class.

```
@Test
public void testCreateHolidayRequestDuplicate() {
    System.out.println("Create Holiday Request Test - Duplicate");

    User user = mock(User.class);
    when(user.getPaidQuota()).thenReturn(5);

    Holiday holidayType = mock(Holiday.class);
    when(holidayType.getType()).thenReturn(HolidayType.PAID);
    when(holidayType.getName()).thenReturn("Test Holiday");

    HolidayRequest holidayRequest = mock(HolidayRequest.class);
    when(holidayRequest.getType()).thenReturn(holidayType);
    when(holidayRequest.getDateFrom()).thenReturn(LocalDate.of(2025, 12, 27));
    when(holidayRequest.getDateTo()).thenReturn(LocalDate.of(2025, 12, 31));
    when(holidayRequest.getCreator()).thenReturn(user);

    when(holidayRequestDao.findAll()).thenReturn(Collections.singletonList(holidayRe
```

```
    assertThatThrownBy(() -> controller.createHolidayRequest(holidayRequest, user))
            .isInstanceOf(IllegalArgumentException.class)
            .hasMessage("ERROR: Holiday Request 'Test Holiday' exists already!");
}
```

3.3.2 User Testing

Usability testing is a technique used in user-centered interaction design to evaluate a product by testing it on users. This can be seen as an irreplaceable usability practice, since it gives direct input on how real users use the system. All scenarios, use-cases, and features of the application were thoroughly tested first by the developer before starting the User Testing phase.

The application was given to two people, along with scenarios and use-cases to test, and were asked to report their feedback. One tester was the General Manager of the EFB, where he was responsible for testing the usability and process of creating requests and checking personal data as an employee, while the other tester was the Head of the IT department in EFB, where he was responsible for testing the usability and process of managing requests and the data. In order to understand if the application is user-friendly and easy to understand and follow, the scenarios below were given to the testers to test, with minimal detailed instructions about the step-by-step process, in order to evaluate the application's ease-of-use.

- Scenario 1 - General Manager (Creation of Requests)
  - Navigating the Portal App
    User is able to Login, view Personal Requests and view the User's Profile.
  - Editing Personal User Information
    User is able to check their email address and password, and update them easily.
  - Create Holiday Request
    User is is able to check their own Holiday Requests, add new request, check the status of their request, and confirm the email notifications.
  - Create HR Letter Request
    User is is able to check their own HR Letter Requests, add new request, check the status of their request, and confirm the email notifications.

  - Create IT Complaint
    User is is able to check their own IT Complaints, add new complaint, check the status of their complaint, and confirm the email notifications.

- Scenario 2 - IT Manager (Management of Data and Requests)
  - Navigating the Portal App
    User is able to Login, view Personal Requests and view the User's Profile.
  - Manage Departments and Users
    User is able to add, edit, and delete different departments and add, edit, and delete users from the system.
  - Manage Holiday Types and State Holidays
    User is able to add, edit, and delete different Holiday types, assign different quota amounts for each holiday type, and add, edit, and delete State Holidays from the system.
  - Manage HR Letter Types
    User is able to add, edit, and delete different HR Letter types.
  - Manage Complaint Types
    User is able to add, edit, and delete different IT Complaint types.
  - Manage Holiday Request
    User is is able to check any and all Holiday Requests, view and edit any request, and approve or decline them.
  - Manage HR Letter Request
    User is is able to check any and all HR Letter Requests, view and edit any request, and approve or decline them.
  - Manage IT Complaint
    User is is able to check any and all IT Complaints, view and edit any complaint, update the assignee responsible for the complaint, and update their status.

After the completion of the above test case scenarios, the users gave their feedback. The feedback given was mostly positive, where the tester's felt like the application's design was very nice, simple, and easy to understand, which was a key requirement. The application's main features are easy to understand and manage. However, there was a request to add alerts for indicating the exact reasons why a request, complaint, or type was not added or updated in the database. For example, if the date is not valid, the application used to return `ERROR:Request could not be created!`, instead it was requested by the client after testing to update these error alert messages with exact error descriptions. The application was then updated as per the request and currently displays descriptive alert error messages, for example `ERROR:Request`

`could not be created due to invalid Start Date!`. Overall, the application was tested thoroughly and performed well, however it is recommended that the application be released to a group of employees first as a beta version, before the official release.

Conclusion

In conclusion, the EFB Portal was created as a means of improving information and data management and communication within the Egyptian Food Bank. In an effort to upgrade to the latest technologies and move to paperless data management, the portal web application was introduced. The application was designed and developed to help with the management of three main processes, HR Letter Requests Management, Holiday Requests Management, and IT Complaints Management. The portal allows its users to create requests and report complaints and deliver them to the appropriate department, where the department users manage those requests and complaints. The system keeps tracks of all the data, and updates the status and user data appropriately.

This thesis and project includes the analysis, design, and implementation of the EFB Portal web application. The definition of the application requirements and domain model, along with the study and research of existing solutions helped with the analysis of the project. As part of the design process, the study and choice of which technologies to use was determined, along with the creation of the system architecture, the database, the business class models for the main processes and the sequence models all helped express the design of the system to be implemented. The choice of tools, implementation process and details, and testing process were discussed in the implementation chapter of the thesis. The combination of all of that led to the development and implementation of the final version release of EFB Portal application.

The application was given to certain members of the Egyptian Food Bank to test and report their feedback. The feedback was overall positive, as the application met all the requirements of the organization for the HR Letter Management, Holiday Request Management, and IT Complaints Management, as well as fully fulfilling the need for a simple and user-friendly design that can be easily understood and used by all employees. The latest beta version release of the web application can be found on the following

address: https://efb-portal.herokuapp.com/, while the source code, documentation, models, and instructions can be found in the enclosed CD.

The system will be delivered to the Egyptian Food Bank in the form of a Flash Drive or CD, that will include the application, database scripts, user manual, and installation guide. The application is a jar artifact, that gets deployed to a maven repository, in which it can be run. The database scripts include two MySQL scripts, one used for the creation of the database, and the other for inserting dummy testing data. The creation script will generate all the entities with their respective attributes, and then insert a user with admin privileges, to be able to access the portal. The user manual is intended to give assistance to users using the application and maintaining the system. It is provided in the form of a PDF and can be found on the portal, specifically the Help web page. The installation guide is a PDF that includes step-by-step instructions for the installation and setup of the system. These files and documents should help the organization with the initial setup and management of the system.

Since the Egyptian Food Bank is a renowned non-profit organization, the system will be provided for free to them. However, future updates will include the ability to add advertisements on the portal, as a means of generating ad money to support the development of the system and any future updates. For non-profit organizations, the system will be provided for free, but the ads will be managed by the developer in order to generate some income. On the other hand, companies and organizations are able to buy the system, and will be given the option to include the advertisement manager, as a means of generating ad income for the organization.

Nevertheless, there is always room for improvement and the addition of new features and processes. Since the application was mainly designed for the Egyptian Food Bank, the source code is not 100% dynamic, meaning that application cannot be given to a different organization without having to modify main parts of the code due to the fact that different organizations have different methods of processing tasks and data. While it is very hard to create a system that will fulfill all the requirements of every organization, the system can still be modified to be a bit more dynamic to allow for greater control and flexibility. Even though the UI received many positive comments, I believe there still exists minor updates and features that could improve the user experience, such as sorting tables and adding graphs for statistical data. While the application seems simple, it is intended to be that way, in order for the EFB employees understand it and adapt to the new system easily. After monitoring the use of the portal, additional updates and features could be released. It is planned in the future to add the salaries for users on the system, and allow the users to create holiday requests even after the consuming all their quota,

while deducting from their salaries. Additionally, the localization of the portal application is planned for the future in order to support both the English and Arabic language.

Bibliography

[1] Unily. Unite your enterprise. Available from: https://www.unily.com/

[2] Samepage. Work gets better on the same page. Available from: https://www.samepage.io/

[3] Bitrix24. Your company. United. Available from: https://www.bitrix24.com/

[4] Speakap. Reinvent Communication with Your Non-Desk Employees. Available from: https://www.speakap.com/

[5] Oracle. Java. Available from: https.//www.java.com/

[6] TheEclipseFoundation. EclipseLink. Available from: https.//www.eclipse.org/

[7] Spring.io. Spring Framework. The right stack for the right job. Available from: https.//spring.io/

[8] TheThymeleafTeam. Thymeleaf. Available from: https.//www.thymeleaf.org/

[9] MySQL. An open-source relational database management system. Available from: https.//www.mysql.com/

[10] Apache. Apache Tomcat. Available from: http.//tomcat.apache.org/

[11] Oracle. Java Development Kit (JDK). Available from: https.//www.oracle.com/java/

[12] Maven. Welcome to Apache Maven. Available from: http.//maven.apache.org/

[13] JUnit. A simple framework to write repeatable tests. Available from: https.//junit.org/

[14] Github. The world's leading software development platform. Available from: https://github.com/

[15] Heroku. Cloud platform as a service. Available from: https://www.heroku.com/

[16] ClearDB. Geo distributed data services platform. Available from: https://w2.cleardb.net/

[17] Apachefriends.org. XAMPP Installers and downloads for Apache. Available from: https://www.apachefriends.org/

[18] phpMyAdmin. Bringing MySQL to the web. Available from: https://www.phpmyadmin.net/

[19] SequelPro. A fast, easy-to-use Mac database management application. Available from: https://www.sequelpro.com/

[20] Netbeans. Development Environment, Tooling Platform and Application Framework. Available from: https://netbeans.apache.org/

[21] Fakhroutdinov, K. (UML) The Unified Modeling Language. Available from: https://www.uml-diagrams.org/

[22] Mockito. Reinvent Communication with Your Non-Desk Employees. Available from: https://site.mockito.org/

Appendix A

Acronyms

API Application Programming Interface

CRM Customer Relationship Management

CSS Cascading Style Sheets

DBaaS Database-as-a-Service

EFB Egyptian Food Bank

GUI Graphical User Interface

HTML HyperText Markup Language

JPA Java Persistence API

JS JavaScript

JVM Java Virtual Machine

MVC Model View Controller

OOP Object-Oriented Programming

ORM Object-Relational Mapping

PaaS Platform-as-a-Service

POM Product Object Model

UI User Interface

XML Extensible Markup Language

Appendix B

Contents of enclosed CD

- `readme.txt` ..... the file with CD contents description
- `apidocs` ..... the directory with the source code documentation
- `jar` ..... the directory with the artifact
- `manuals` ..... the directory with installation guide and user manual
- `scripts` ..... database SQL scripts
- `models` ..... enterprise architect models
- `src` ..... the directory of source codes
  - `wbdcm` ..... implementation sources
  - `thesis` ..... the directory of LaTeX source code of the thesis
- `text` ..... the thesis text directory
  - `thesis.pdf` ..... the thesis text in PDF format