Design Document

Individual Track Assignment

25/03/2022 Eindhoven
Version: 3.1

Esther Wolfs: 3329984

Tutor:
Tim Kurvers
Márcio Paixão Dantas

Version history

Version	Date	Author(s)	Changes	State
0.1	25/03/2022	Esther Wolfs	Start design document and research questions	Started
0.2	30/03/2022	Esther Wolfs	Make changes after receiving feedback	Started
0.3	05/04/2022	Esther Wolfs	Update research questions	Finished
1.0	16/05/2022	Esther Wolfs	Answer research questions	Started
1.1	16/05/2022	Esther Wolfs	Move everything research related from this document to specific applied research document	Finished
2.0	02/06/2022	Esther Wolfs	Add CICD Diagram with explanation	Started
2.1	13/06/2022	Esther Wolfs	Update C4 Diagram	Finished
3.0	13/06/2022	Esther Wolfs	Add security report structure	Started
3.1	14/06/2022	Esther Wolfs	Fill in paragraphs analysing the security risks.	Finished

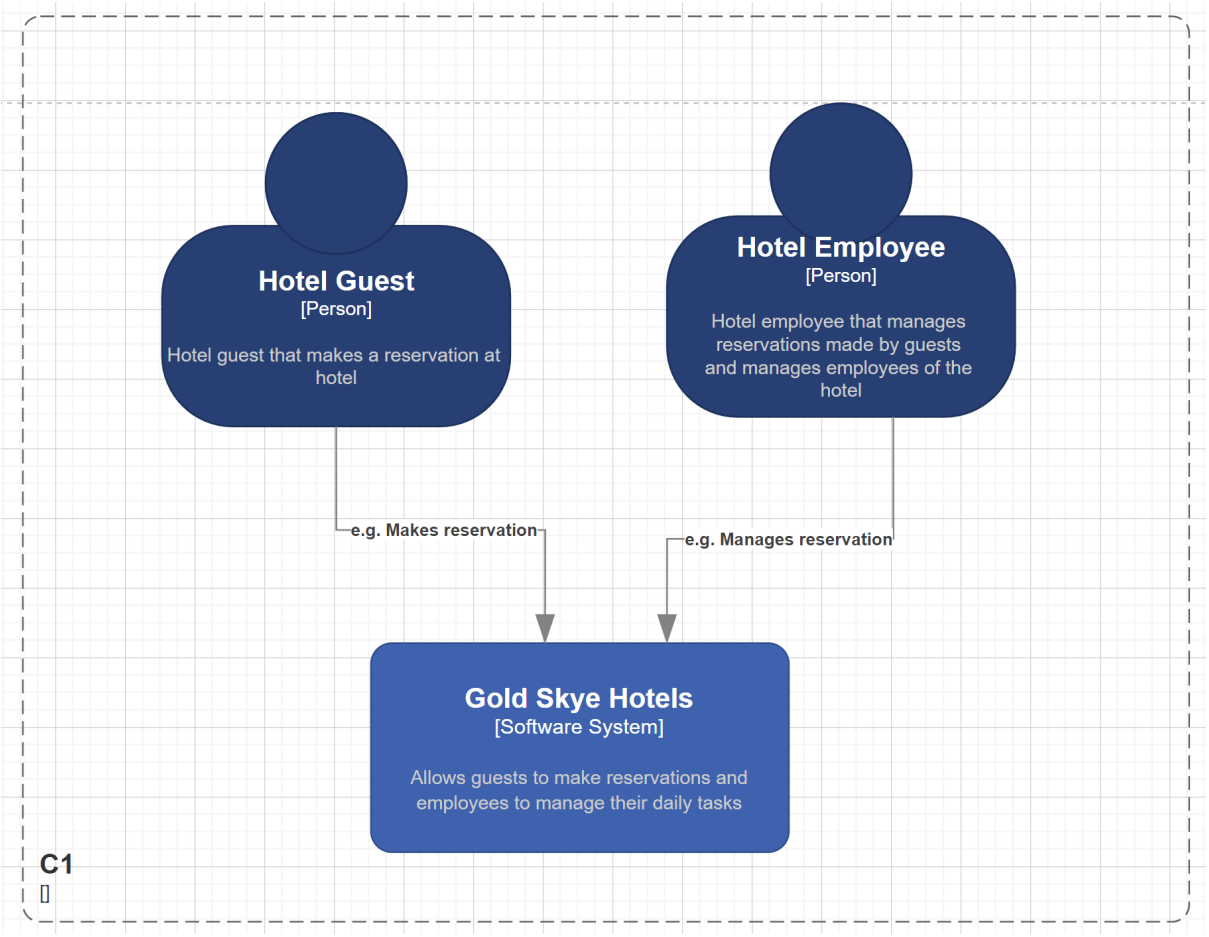
1. Introduction

In this document I will explain the decisions I have made for my project based on the research I have done. I will explain the architecture of my application.

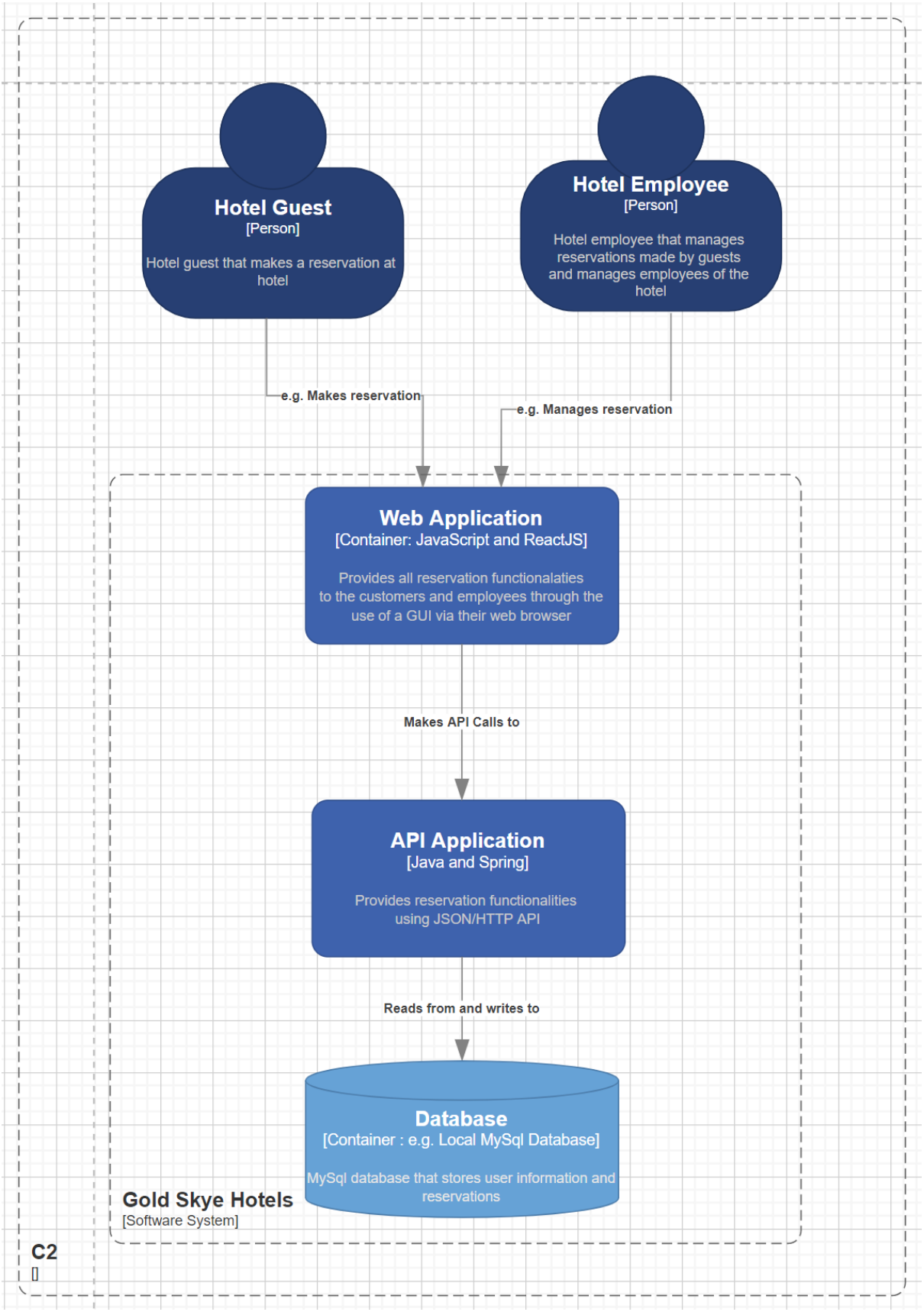
2. Architecture

2.1 C4 Diagram

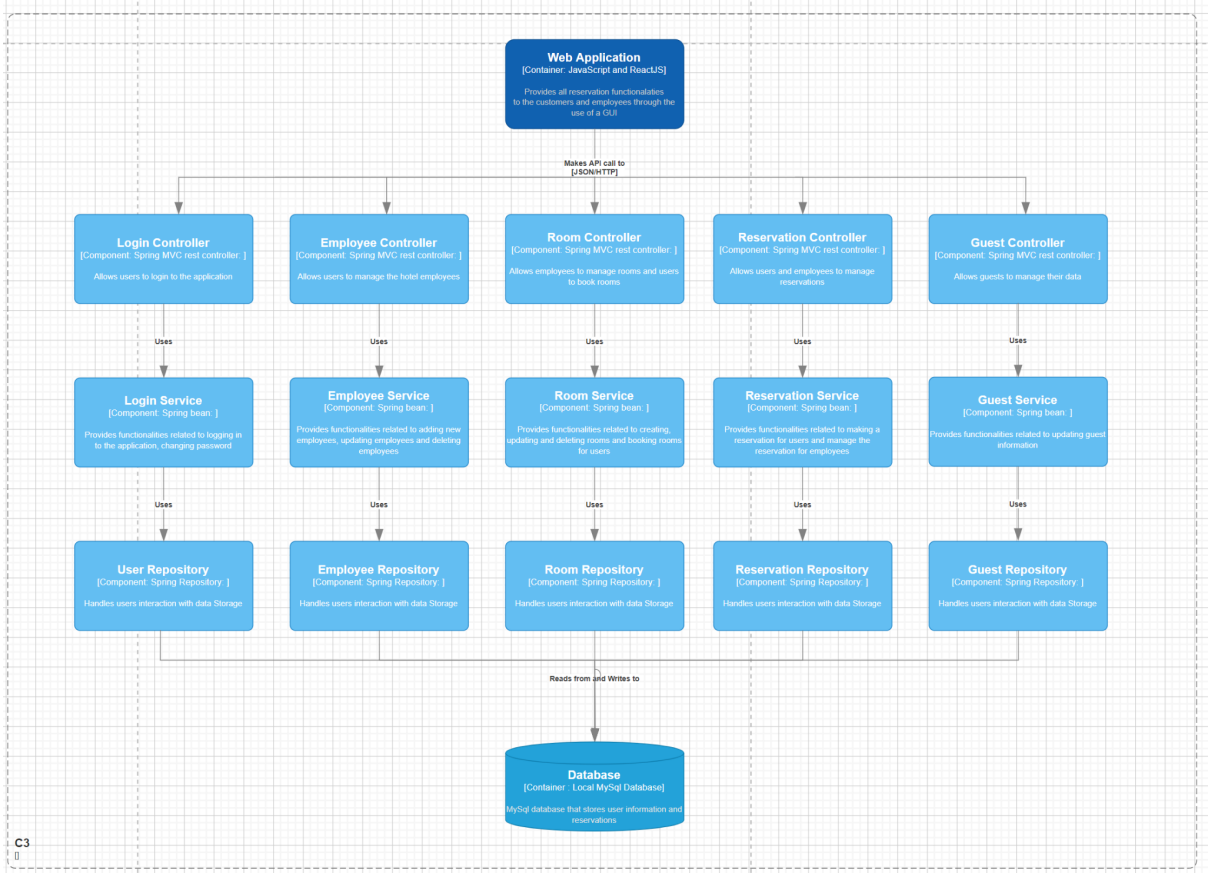
C1: There are two types of users for my application, the guests of the hotel who can manage their bookings and the employees who work at the hotel, who can manage their daily work tasks.



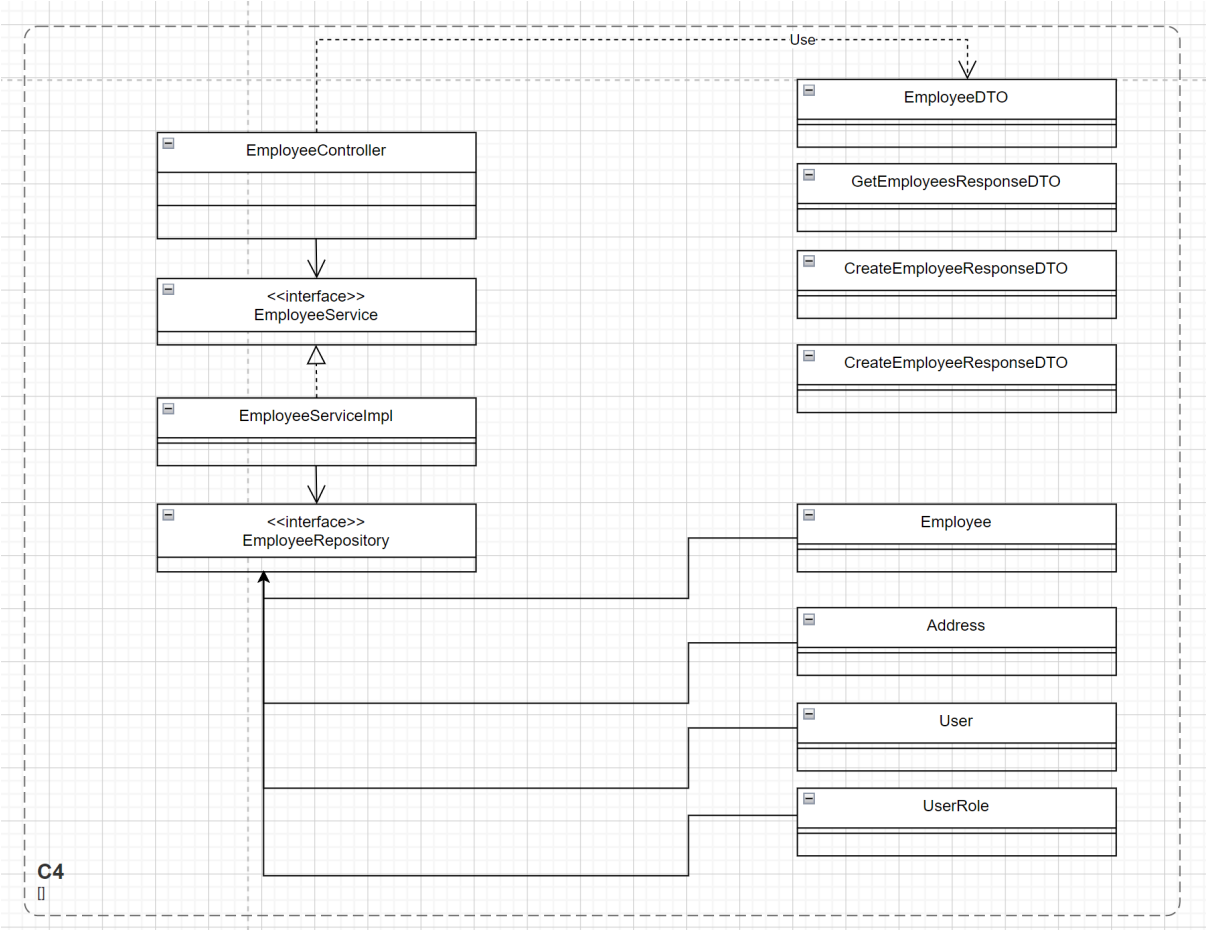
C2: The system is divided in two parts, the web application that is built using the front end JavaScript library React and the backend Java APIs.



C3: For now, the API is divided into 5 sections, a section for rooms, reservations, accounts, employees and login. These 5 sections are separated into 3 layers, the Controller, Service and Repository layer, to guarantee the three layer design.



C4: This is one part of the C4. It only shows the employee part. The DTOs that are used by the EmployeeController are the CreateEmployeeResponseDTO, GetEmployeesResponseDTO and the EmployeeDTO. It is connected to the Business Layer through the EmployeeService interface. EmployeeServiceImpl implements this interface. This is connected to the repository layer by using the EmployeeRepository. This layer uses the Employee, Address, User and UserRole entities.



2.2 How is SOLID guaranteed

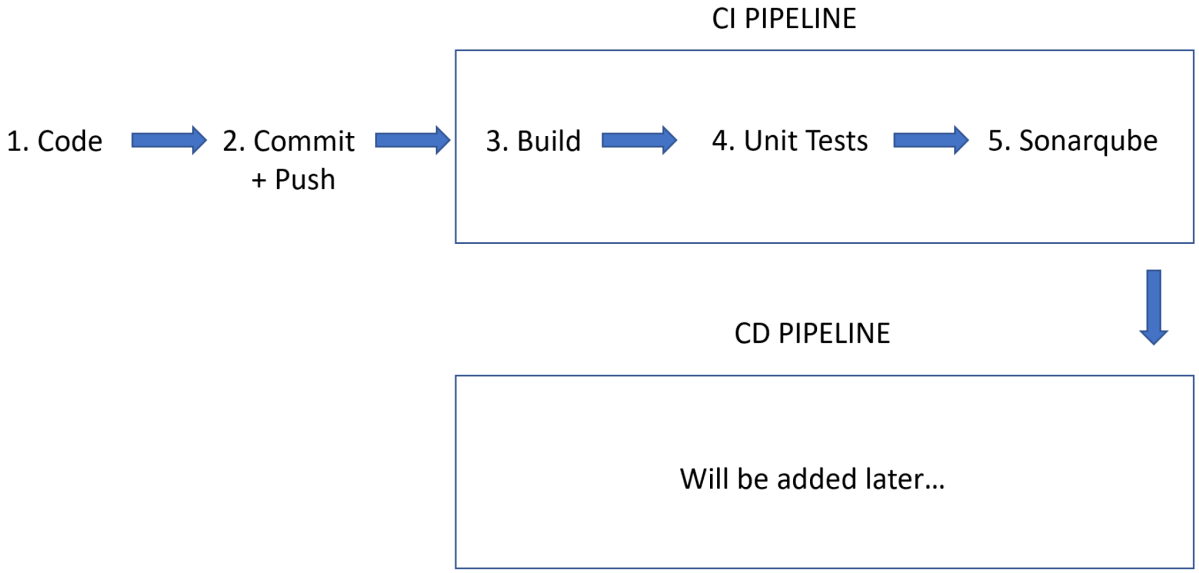
The solid principles are used in Object Oriented Design, to help create a design that is easily maintainable and extendable.

- **S**ingle Responsibility:
  Every single class created is responsible for only one task. This is possible by dividing the controllers by feature, so there are no controllers responsible for example both the employees and the reservations at the same time.
- **O**pen/Closed Principle:
  To keep the code extendable, the open/closed principle is used. Using interfaces will help with this, because this means you don't have to change the existing code if you want to add a new feature. The objects are open for extension but closed for modification. An example for this in my application is the payment method. If the company decides to add another payment method, they can add it without having to modify the current payment methods.
- **L**iskov substitution:
  The liskov substitution means that every subclass should be able to be substituted by the base class. For my application I use inheritance in the Person class. The person class is the main class and Guest and Employee are subclasses. For the different Employee types I will add more subclasses in the future, to separate the functionalities.

- **I**nterface Segregation Principle:
  I will create multiple interfaces to segregate functionalities between users, so that for example a front desk employee doesn't have access to the statistics meant for the manager.
- **D**ependency Inversion Principle:
  The layers are separated by interfaces, the classes depend on the interfaces and are given them when created. For example if you want to switch between using a fake database and a real database you can give the class another instance of the database, without getting any errors or having to modify the existing code.

3. CI/CD Diagram

This diagram shows the CI/CD Pipelines. The CD pipeline will be added later.



3.1 CI Pipeline

1. The first stage is the code stage. In this stage the new code will be created.
2. After the new code is added, the code needs to be committed and then pushed to git. This triggers the pipeline.
3. Stage 3 is the first stage of the CI Pipeline. It will build the application, if there are no errors in the build it will pass, otherwise it will fail.
4. Stage 4 triggers the unit tests, all the unit tests will run and if they all pass this stage will pass, otherwise it will fail.
5. The last part of the CI Pipeline is the sonarqube stage. This triggers the sonarqube build, so you get a sonarqube report.

3.2 CD Pipeline

This part will be added later.

4. Security Report

In this chapter the OWASP top 10 security risks will be analysed (OWASP, 2021).

Overview

	Risk	Status	
1	Broken Access Control		CSRF disabled, authentication and authorisation implemented
2	Cryptographic Failures		BCrypt is used to encrypt passwords, an access token is used. Not all personal data is encoded.
3	Injection		There is no protection against SQL injection or Cross Site Scripting.
4	Insecure Design		Error messages are vague, passwords are encrypted, unit tests are made.
5	Security Misconfiguration		CSRF is disabled, passwords for employees are default hard coded passwords with no option to change them.
6	Vulnerable and Outdated Components		Unused dependencies will be removed
7	Identification and Authentication Failures		There are no checks for password strength, employee passwords are a hard coded default, there is no way to change a password
8	Software and Data Integrity Failures		No digital signatures
9	Security Logging and Monitoring Failures		No logging and monitoring
10	Server-Side Request Forgery		No measures against SSRF are used

1. Broken Access Control

The number one security issue in 2021 was broken access control. Some weaknesses include exposure of sensitive information to an unauthorised actor, insertion of sensitive information into sent data and cross site request forgery.

To prohibit unauthorised users from accessing data they are not allowed to access, authentication and authorisation is used. Most methods, except some GET methods like getting all rooms and POST for creating a user account, require authorisation and are only accessible to certain roles. By checking the roles and user id in the access token, personal information is only available to the right user.

Sensitive data like passwords can be exposed, for example in error messages. Passwords of users are encrypted in the database. However, certain passwords, like the MySQL password are still stored in plain text in some files.

Currently the CSRF is disabled, this means that the web applications cannot verify whether the request was intentionally provided by the user who submitted the request.

2. Cryptographic failures

Number two is cryptographic failures. Some of the most common weaknesses are use of hard-coded password, broken or risky crypto algorithm and insufficient entropy.

Only necessary user information is stored in the database. Passwords are encrypted and not stored in plain text, to encode these passwords BCrypt is used. This is one of the most secure ways to store passwords, as it doesn’t use deprecated hash functions like MD5 or SHA1.

Other user information, like emails, are not encrypted. To limit who can access this information the user id of the logged in user is checked in the access token.

One example of using insufficient entropy is when an attacker could guess the random numbers that are generated to allow access to the system. For example when a user id is user for authentication/authorisation. The attacker can guess the id that is used for the session. The user id is stored in the access token, but any access token can easily be decoded. Therefore, the access token should not be stored in local storage of the browser after login, because everything that is stored in the local storage can be accessed by hackers. To read/update/delete personal data the id in the access token is checked to see if it matches the user id of the record.

3. Injection

Injection was the third biggest security risk in 2021. Cross-site scripting and SQL injection are the most common weaknesses.

The application is at risk of SQL injection, because the user-supplied data is not validated or filtered. Any user could enter a malicious string when creating an account or adding some

other data. Data is not limited, so it is possible to get all user records in case of a SQL injection. Another reason why data should be validated is because right now it is possible to enter double values for capacity and the price value is not limited to two decimals. The entities have min and max value annotations, so when someone for example enters a negative number for price it won't be saved in the database.

An example of Cross Site Scripting is displaying a message based on the url parameters, like getting the username in the profile page from the url parameters. These parameters can be changed in the url to contain a malicious script. A link with the modified url can be sent to victims and when they click on the link the script runs. This way the attacker can also send victims to a fake login page, to steal their login information. This application gets the user based on the access token and gets their information based on the id that is stored in the access token, not based on the parameters.

4. Insecure Design

Insecure design focuses on risks related to design and architectural flaws. The most common weaknesses are Generation of error messages containing sensitive information and unprotected storage of credentials.

To make sure the application functions the way it should, unit tests are made, to test all methods. A coverage of at least 80% is required. The 3 layer design is applied, which means the application is split between the repository, business and controller layers.

Some error messages may contain sensitive information like passwords or the location of files that store credentials. Most error messages in this application are fully logged in the console, meaning everyone can access them. Because these are the default error messages, it could be possible that sensitive information is displayed.

Storing sensitive information like passwords in plain text is also a big risk. As mentioned above, this application does not store passwords in plain text, but instead encrypts the passwords. Other sensitive information like email addresses, phone numbers and other personal data are stored in plain text. This may cause a security risk.

5. Security Misconfiguration

Number five is security misconfiguration. This can happen when important security features or configuration are disabled, or unnecessary features are enabled. For example unnecessary user roles or ports.

Right now the CRFS is disabled, because this will be taught in semester 6.
The error messages that are sent to the user inform them of the error, but are not too informative. For example when a user enters a wrong username or password it does not explicitly tell them which one was wrong, or if the username exists or not. It just tells the user they entered the wrong credentials.

When adding an employee to the system, a user account for that employee is automatically created. The username consists of their name and the password is a default password, that is the same for every single user. Right now it is not possible to change this password in the account settings, which means that every user can guess another user's username and enter the same default password, which gives them access to that user's account and their privileges. When guests create their account they can choose their own username and password, however they cannot change these values.

6. Vulnerable and Outdated Components

Vulnerable and outdated components is number six on the OWASP top 10 list. Using third party components or libraries that are unmainted can form a risk.

This application has a few unused dependencies in both the front end and back end. These should be removed, because they can cause issues and security risks. It also has some components and classes that are no longer used and replaced with newer ones, but they have not been deleted yet. These should be deleted or updated.

Users of the application can use outdated operating systems or web browsers, this can also be a risk.

7. Identification and Authentication Failures

Risk number seven is identification and authentication failures.

The application does not check the strength of the passwords when creating an account. Default or basic passwords are permitted, it does not check if a password contains a number, symbol and lowercase or uppercase letters. It does have a minimum of 8 characters and a maximum of 50 characters. The password that is given to employees when creating their user account is a very basic and easily guessable password, that is hard coded and cannot be changed. All passwords are the same for every employee. Users can choose their own password, but they cannot change it and there is no password recovery.

As mentioned above, the passwords are not stored in plain text, but are encrypted using BCrypt, a modern and maintained encryption tool.

The access token of the users expires after 30 minutes, a refresh token is currently not used. When the user refreshes or closes the browser, they have to log in again.

8. Software and Data Integrity Failures

This risk focuses on software updates, critical data and CI/CD pipelines without verifying data.

This application does not use digital signatures to verify the source of the software and whether or not it has been altered. Malicious users can alter data sent to the application.

There is no code review process, because only one person is working on the application. Whenever a new change is made it is pushed to git without reviewing the code. A CI/CD pipeline is set up.

9. Security Logging and Monitoring Failures

Number nine is security logging and monitoring failures. Detecting and responding to breaches is critical. Without logging and monitoring, breaches cannot be detected.

Logins and failed logins should be logged. When a lot of failed logins happen at the same time it can become clear someone is trying to take advantage of the system. These logs should not only be stored locally. These logs should be encoded correctly, to prevent injections or attacks on the logging or monitoring systems. The application should also be able to detect and alert for active attacks.

This application does not do any logging or monitoring, making it impossible to detect any breaches or malicious attacks.

10. Server-Side Request Forgery (SSRF)

The last risk covered by the OWASP top 10 is Server-Side Request Forgery. SSRF risks happen when a web application fetches a remote resource without validating the user supplied URL. It can be prevented by segmenting remote resource access functionality in separate networks to reduce the impact of SSRF and enforcing a “deny by default” firewall policy. In the application all user supplied data should be validated, responses sent back to the clients should not be raw, HTTP redirections should be disabled.

This application uses none of the measures to prevent SSRF.

References

OWASP. (2021). *OWASP Top Ten Web Application Security Risks* | *OWASP*. OWASP Foundation. Retrieved June 14, 2022, from https://owasp.org/www-project-top-ten/