**OWASP report**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Likelihood** | **Impact** | **Risk** | **Actions possible** | **Planned** |
| A1: Broken  access  control | Medium | Severe | Medium | CORS misconfiguration could be an issue | Yes |
| A2: Cryptographic Failures | Low | Medium | Low | N/A | No |
| A3: Injection | Medium | Severe | Medium | Not every single form is checked for injection | Yes |
| A4: Insecure Design | Medium | Medium | Low | More service classes that have checks | No |
| A5: Security Misconfiguration | Low | Severe | Low | There could still be logs in the console and they should be removed | Yes |
| A6: Vulnerable and Outdate components | Low | Medium | Low | N/A | No |
| A7: Identification and Authentication Failures | Medium | Severe | Medium | N/A | Yes |
| A8: Software and Data Integrity Failures | Low | Severe | Low | Security tools could be used | No |
| A9: Security Logging and Monitoring Failures | Low | Severe | Medium | More logging could be added | No |
| A10: Server-Side Request Forgery | Low | Severe | Low | N/A | No |

A1: The likelihood of this risk is not so high on the food delivery website, since it has authorization for every request (with the exception of a few get requests), modifying the URL is a check for every page and if the role is not right, the user is sent to the home page, the jwt token is saved as a session because if it would get intercepted somehow, it will expire in 30 minutes nonetheless, and is not prone to attacks like local storage is, and the user will be logged out if it expires. The problem could be the CORS configuration, since I lack knowledge on this topic, and it is set to all (\*).

A2: Data is transmitted and received only as JSON, header options are set on every page and passwords are encrypted with BCrypt, which is currently considered as safe.

A3: User supplied data is validated because every piece of data that interacts with the database is following a format declared by request classes. Also, some of the queries use LIMIT, so in case that data would get destroyed, it would only destroy one row at a time.

A4: The web application contains some service classes that offer some validation, such as checking for emails when creating a user, but they could always be improved.

A5: Old accounts that were used for testing are deleted or edited to be secure, every request (that requires and has logic) uses headers and the application uses secure settings from the Spring security package. One problem could always be that the package would be outdated, and it would result in a risk.

A6: The security components are up to date at the time of the release.

A7: As explained on the first risk, the roles and the jwt token are checked on every page and invalidated when the jwt expires. One risk could be that checks for passwords are not too strong.

A8: The risk is low because all of the security packages are from trusted sources. A tool like OWASP CycloneDX could be used to check components for vulnerabilities.

A9: The application could contain more logging with more detailed information in order to determine if a breach is being executed.

A10: The application does not use external or user supplied URLs.

**Conclusion**

The application has a decent level of security, excelling at dealing with jwt tokens, authentication and authorization, also using headers for all requests, but does not offer strong security in the means of injection. Also, this application would have to have better CORS configuration and it is not ready for an attack, so it is not ready to be deployed, but it offers a good foundation.