# AirBnB (Clone) for Web Application Technology (UoA:DIT)

## Server Side

- ## Index
  - **Functionality**
  - **Implementation**
  - **Points worthy of mention**
  - **Database**
    - **Tables**
  - **Third party libraries used**
  - **Known mistakes and errors**
- ## Functionality
  - RESTful Web Service that utilises the Jersey JAX-RS implementation.
  - It uses a MySQL Database to store the data. More information can be found in the Database section of this file.
- ## Implementation
  - Due to the size of the application, no implementation details are written here. These can be found in the form of JavaDoc in {ROOT}/jd/index.html
- ## Points worthy of mention
  - This section describes some of the note worthy points of the application
  - When a user first signs up, the provided password is hashed (using Spring Framework Security - See Third party libraries) and that hash is stored in the database. Although only Prepared Statements have been used during access to the database, thus providing a reasonable amount of security against SQL Injections, this method helps us better secure the database and our users' information.
  - The values taken from the sentiment JavaScript framework have been "compressed" between the values 0 and 1 before being stored in the database.
  - The NNCF algorithm works in the following way:
    - If we do not have enough information on a user then we query the user's past searches and create an NxM matrix, where N is the amount of users and M the amount of houses, where Cell(i, j) = 1 if the user i has rated the house j positively (currently 0.5 or above) and 0 otherwise. We use the LSH algorithm on that matrix and find the other users that hashed into the same bucket at the current user. We then calculate the cosine similarity between the the current user and the other users, we sort those results, we find houses rated by the other top 10 users and not the current user and we return the first 4 of these.
    - If we do have enough information, we query the actual comments table (see tables) and we follow the same steps as before, with the exception that in the cosine similarity calculation, we use the actual values

of the ratings (thus getting more relevant results)

- During XML Export, due to the size of the comments table (see tables) (691,055 comments of various length at the time of writing - about 220MB) it is highly advised to comment that section out during any kind of testing on a non-server machine.

## Known mistakes (due to time constraints)

- The application is not polished.
  - Not all possible errors are addressed (most exceptions simply return a 500 error with no further explanation).
  - More checks could be made in various operations (i.e. a user being able to comment on a house only after booking and visiting a house)
- The NNCF algorithm is not optimized meaning that some unnecessary calculations are made.

## Database

- The database is designed to be in 3NF, meaning that no redundant data is held in any of the tables and each reference to existing data is in the form of foreign keys.
- Accesses to the database are managed by Apache Commons' Connection pooling.
- The following settings need to be specified in the {ROOT}/config.xml file for the server to run on a machine
  - The driver class
  - The JDBC Url
  - The username
  - The password

### _Tables

| Name | Usage | Key(s) | Foreign Keys | Additional Comments |
|---|---|---|---|---|
| bookings | Stores users' bookings | bookingID | userID(users), houseID(houses) | |
| comments | Stores each house's comments | commentID | userID(users), houseID(house) | This table is especially large. Thus, it's better to remove that section when exporting to XML |
| houses | Stores data about each registered house | houseID | ownerID(users) | |
| searches | Stores users' searches | searchID | userID(users), houseID(houses) | When we don't have enough information on a user's preferences, we keep track of house pages they've visited (later used in the sparse NNCF algorithm |
| invalidTokens | Stores invalidated tokens | tokenID | None | When a user signs out, we add their token to the blacklist and consider it invalid |
| messages | Stores user to user messages | messageID | senderID(users), receiverID(users) | |
| photographs | Intermediate table that | photoID | houseID(houses) | |

| | | | | |
|---|---|---|---|---|
| | matches houses to photographs | | | |
| users | Stores user information | userID | None | Each user's password is hashed and stored in the database |

- Third party libraries used:

| **Library** | **Version** | **Usage** | **Extra comments** |
|---|---|---|---|
| Jersey (http://jersey.github.io/) | 2.26-b09 | The JAX-RS implementation of choice | |
| MySQLConnector (https://dev.mysql.com/downloads/connector/j/5.1.html) | 5.1.6 | Connection with the MySQL Database | |
| Apache Commons Pool (https://commons.apache.org/proper/commons-pool/) | 1.4 | Database Connection Pooling | Used instead of a JPA implementation |
| Jersey Media Json-Jackson (http://jersey.github.io) | 2.26-b07 | Json Writer for jersey | |
| Jackson (https://github.com/FasterXML/jackson) | 2.9.0.pr4 | Java JSON support | |
| Json Web Token (JJWT) (https://jwt.io/) | 0.7.0 | Web Token Generator - Validator | |
| Gson (https://sites.google.com/site/gson/) | 2.8.1 | Google's JSON to POJO converter | |
| Jersey Media Multipart (jersey.github.io) | 2.17 | Picture Uploads | |
| James Murty's XMLBuilder (http://jamesmurty.com/2008/12/14/xmlbuilder-easily-build-xml-docs-in-java/) | 1.1 | Raw export of the database to XML | |
| Spring Framework Security (https://projects.spring.io/spring-security/) | 3.1.0.RELEASE | Password hashing | |
| James Murty's XMLBuilder (http://jamesmurty.com/2008/12/14/xmlbuilder-easily-build-xml-docs-in-java/) | 1.1 | XMLBuilder for database to XML export | |
| Apache Commons CSVReader (https://commons.apache.org/proper/commons-csv/) | 1.5 | Used to parse the provided CSVs and add the data to the database | |
| Univocity Parsers (https://github.com/uniVocity/univocity-parsers) | 1.0.0 | Dependency of Apache Commons | |

| | | | |
|---|---|---|---|
| | | CSVReader | |
| Debatty's LSH (https://github.com/tdebatty/java-LSH/tree/master/src/main/java/info/debatty/java/lsh) | RELEASE | LSH Implementation | Used in the NNCF algorithm |