

# Mandatory I

## 01a-03a

Assignments 01a through 03a were all collected into a single project

[Data parsing server project](#)

## 04a

[Server sent event example](#)

## 04b

## Exposee documentation

The below documentation is copied from my logseq graph

### **Connect**

For my database I chose postgres so be sure to install postgresql so that you can run the cli tool used in this documentation For this exercise I will be providing you with two db users. see username below, for easy use both of their passwords are 'si-o4b'

- lasse
- troels
- As I use [Pinggy](#), the free tier only allow for random subdomains, I will be providing the host and port name to you upon request instead of in this documentation

To connect to the db instance via my TCP tunnel, use the following command, replace the placeholder with the information provided upon your request

```
psql -h <host> -p <port> -U <lasse|troels> -d si-04b
```

### **Granular control**

#### **Views**

- masked\_employee\_salary:
  - Restricts employees access to each others salary in the employee table, allows sales persons to see their own salary and managers to see all but other managers salary

#### **Roles**

- Customer

- Access level
  - SELECT (name, price) on products table
  - SELECT on customer, but only their own row
  - SELECT on purchases, but only their own
  - SELECT on discounts, but only their own
- Sales\_person
  - Access level
    - SELECT on products
    - SELECT on customers
    - SELECT, INSERT on purchases, but only with their own id
    - SELECT, INSERT on discounts, but only with their own id
    - SELECT (id, name) on employees
    - SELECT on masked\_employee\_salary
- Manager
  - Access level
    - SELECT, INSERT, UPDATE, DELETE on products
    - SELECT, INSERT, UPDATE, DELETE on customers
    - SELECT, INSERT, UPDATE, DELETE on purchases
    - SELECT, INSERT, UPDATE, DELETE on discounts
    - SELECT, INSERT, UPDATE, DELETE on employees but their own or other managers
    - SELECT on masked\_employee\_salary
    - The two users provided in the previous section have different levels of control in the database
    - The user 'lasse' has the sales\_person role
    - The user 'troels' has the manager role

## Integrator images

### *Read only table*

```
granular_access_db=> select * from read_only_data.read_only_table
granular_access_db-> ;
id | column1 | column2
----+-----+-----
 1 | du er   |      30
(1 row)

granular_access_db=> insert into re
read_only_data. restricted_data.
granular_access_db=> insert into re
read_only_data. restricted_data.
granular_access_db=> insert into read_only_data.read_only_table (column1, column2) values ('lasse er', 'cringe');
ERROR:  invalid input syntax for type integer: "cringe"
LINE 1: ...only_table (column1, column2) values ('lasse er', 'cringe');
                        ^
granular_access_db=> insert into read_only_data.read_only_table (column1, column2) values ('lasse er', 30);
ERROR:  permission denied for table read_only_table
granular_access_db=> update read_only_data.read_only_table set column1 = 'lasse er' where id = 1;
ERROR:  permission denied for table read_only_table
```

### *Write only table*

```
granular_access_db=> select * from write_only_data.write_only_table;
ERROR:  permission denied for table write_only_table
granular_access_db=> insert into write_only_data.write_only_table (id, column1, column2) values (2, 'lasse er', 30);
INSERT 0 1
```

### *Restricted table*

I could only see my own user in the table

```
granular_access_db=> select * from restricted_data.sensitive_table;
id | department_id | username | password
----+-----+-----+-----
 2 |              3 | dorohd   | ohnocringe
(1 row)
```

## 06a

### [Websocket example](#)

## **Mandatory II**

### ***08a - WebRTC example***

[Repo](#)

### ***08b - Document DLS***

For this assignment, we have chosen to document our fullstack project instead of DLS. At this point in time our DLS project mainly consists of yaml configuration files for kubernetes. Therefore we thought it would be more interesting to show documented business logic from our fullstack project.

[Repo](#)

### ***10a - Database Documentation***

[Repo](#)

### ***10b - Database Migration***

[Repo](#)

### ***11a - Auth Integration***

[Repo](#)

### ***11b - Paymnet integration***

[Repo](#)

### ***12a - Webhooks***

[Repo](#)

### ***13a - GraphQL***

[Repo](#)