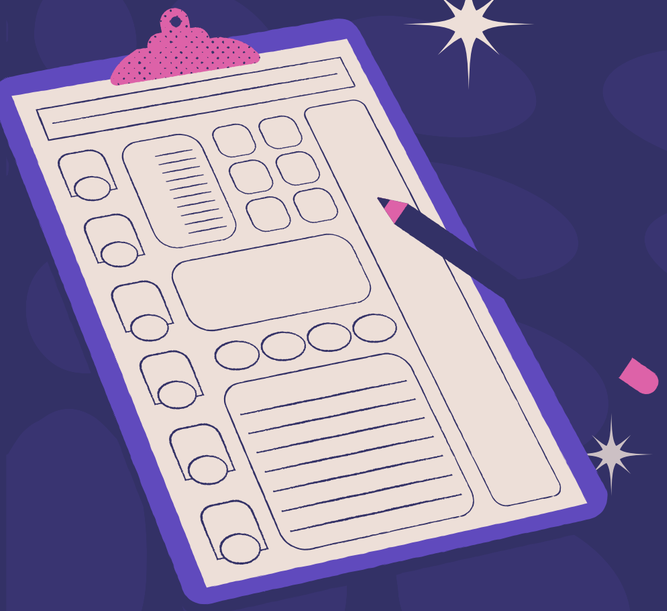
PROJECT AKHIR

laundry UNBI

FRESH - CLEAN - ENJOY



contact

@prasashtiazz_

@gary_ferdinand

@adzs72

Players

njn

emazernlt

adzs

Topic

- introduction project
- concept
- web presentation / demo

introduction project

@reallygreatsite

Background case

Laundry Business Process

non delivery service

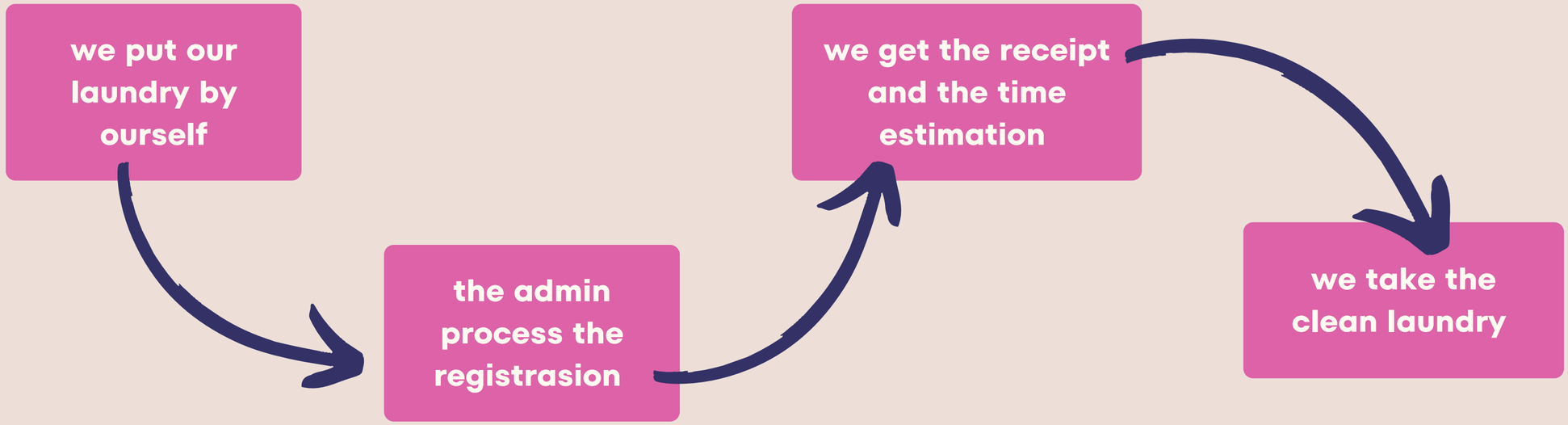
delivery service

NON DELIVERY SERVICE

non delivery service

delivery service

Laundry Business Process

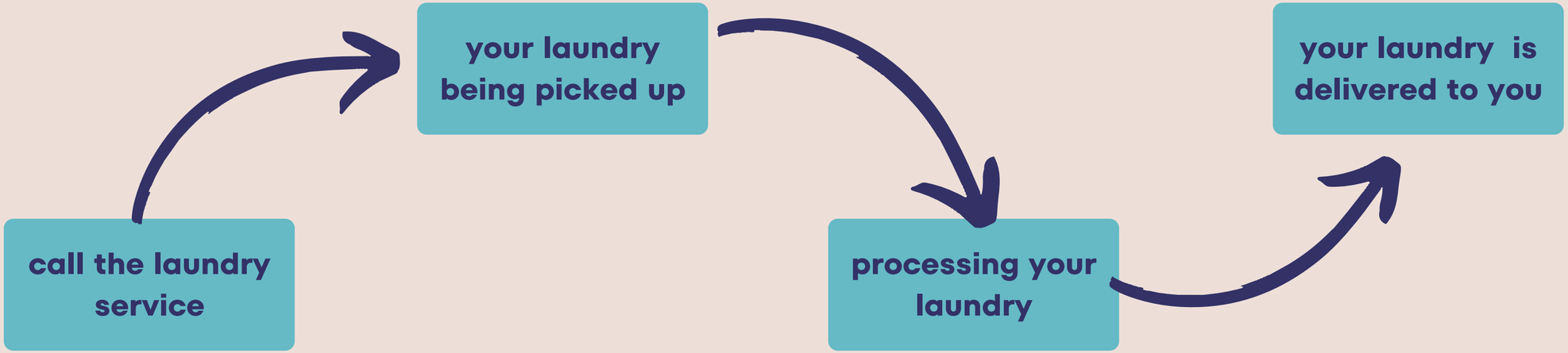


Delivery Service

non delivery service

delivery service

Laundry Business Process



Formulation of the problem

non delivery service

- cust get their receipt
- cust get the estimation time
- customer have to deliver the laundry by themselves
- the shop continuously chased by deadlines

delivery service

- no need to deliver the laundry
- customer doesn't get receipt
- has no idea when the laundry is finished

solution

UnBi
Service

WEB APP

Business Process

01 The customer register and make request on the web.

02 the laundry is picked up by staff

03 customer get their receipt. (total weight, price, status)

04 The customer get updates about their laundry processing

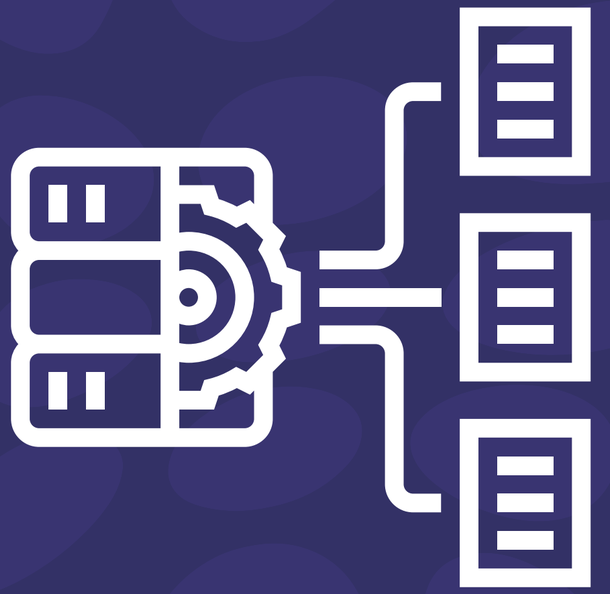
05 the laundry is delivered to the customer

MARKETING TARGET

01 DORMITORY

02 EMPLOYEE

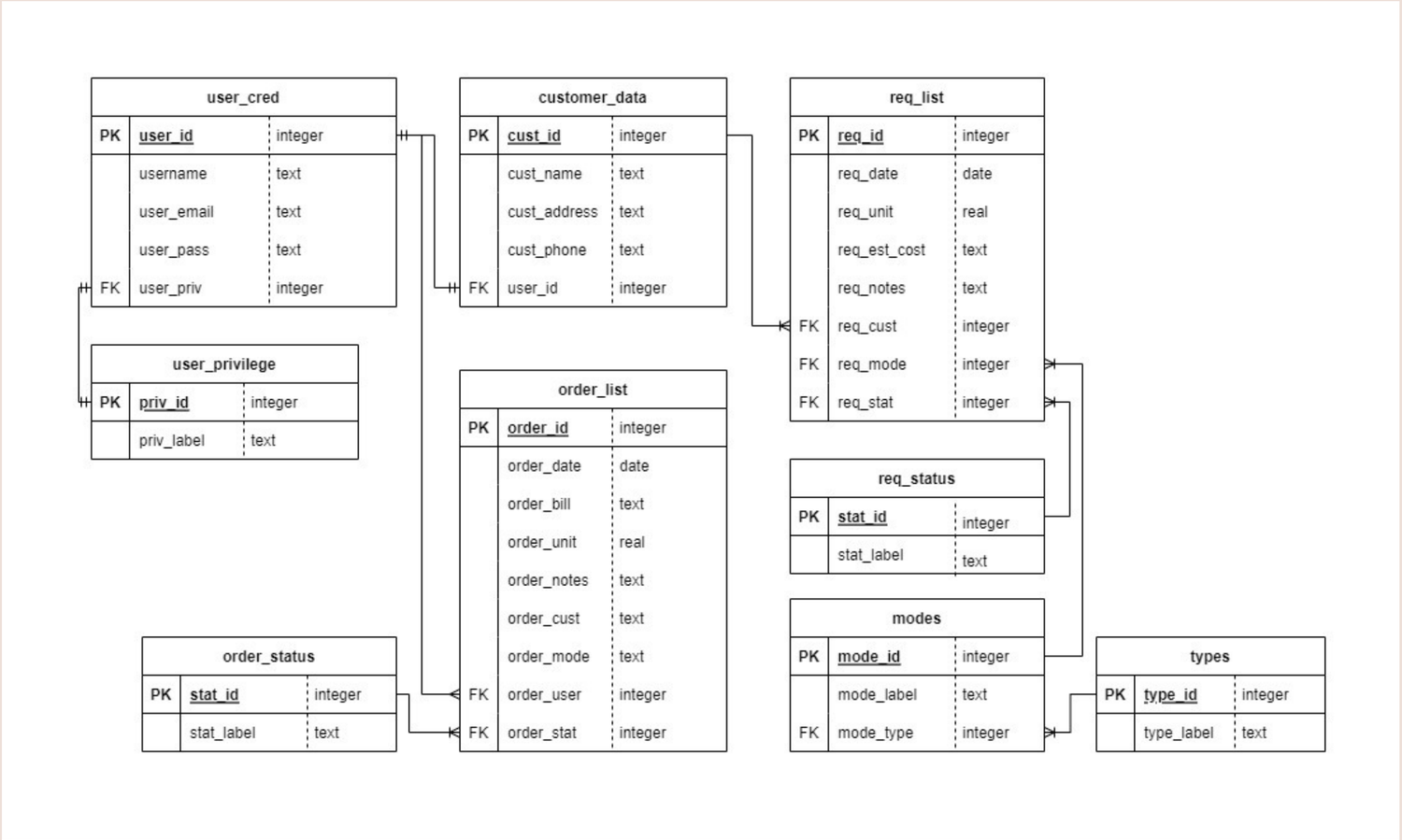
03 COLLEGE STUDENT



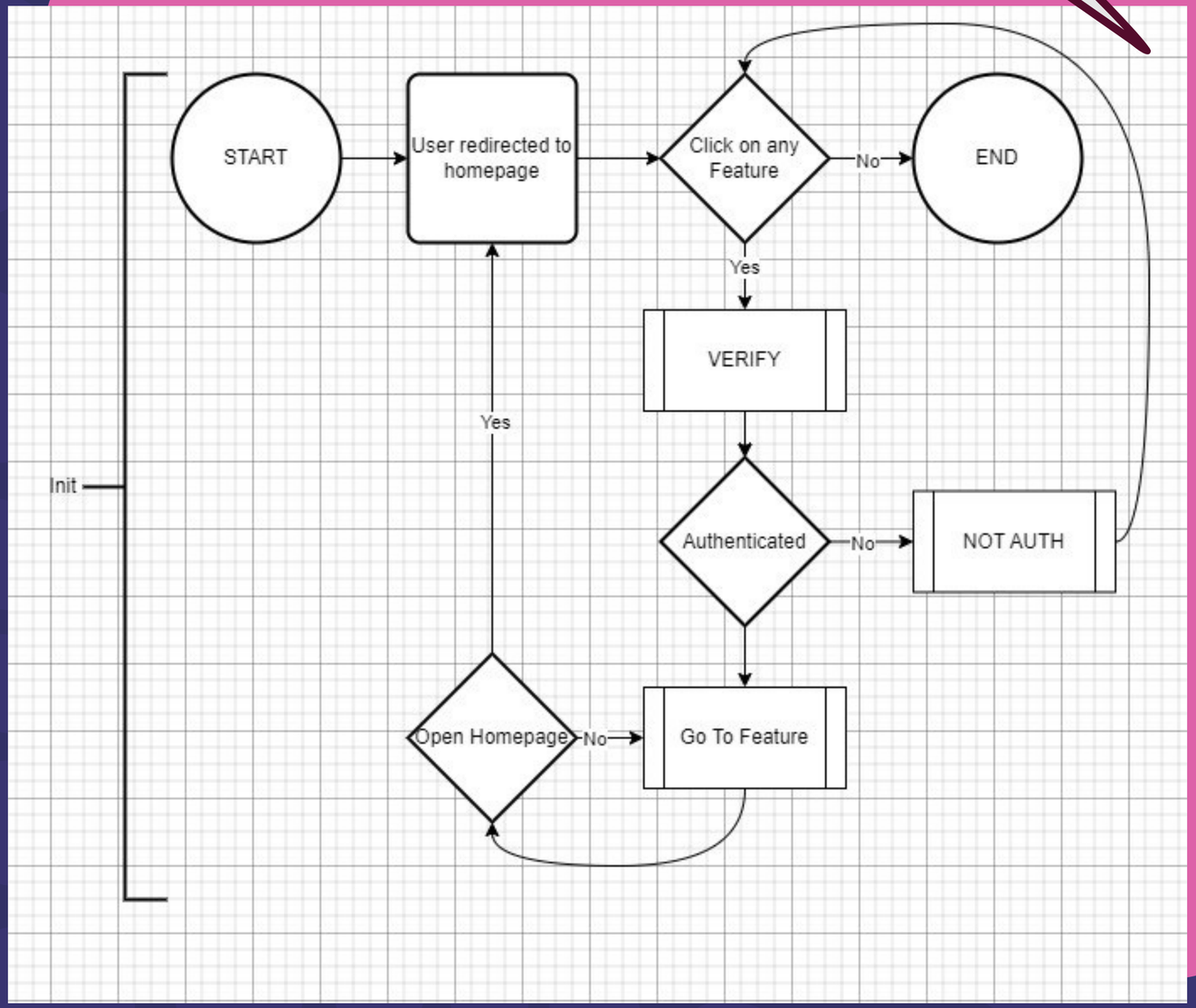
web app

concept

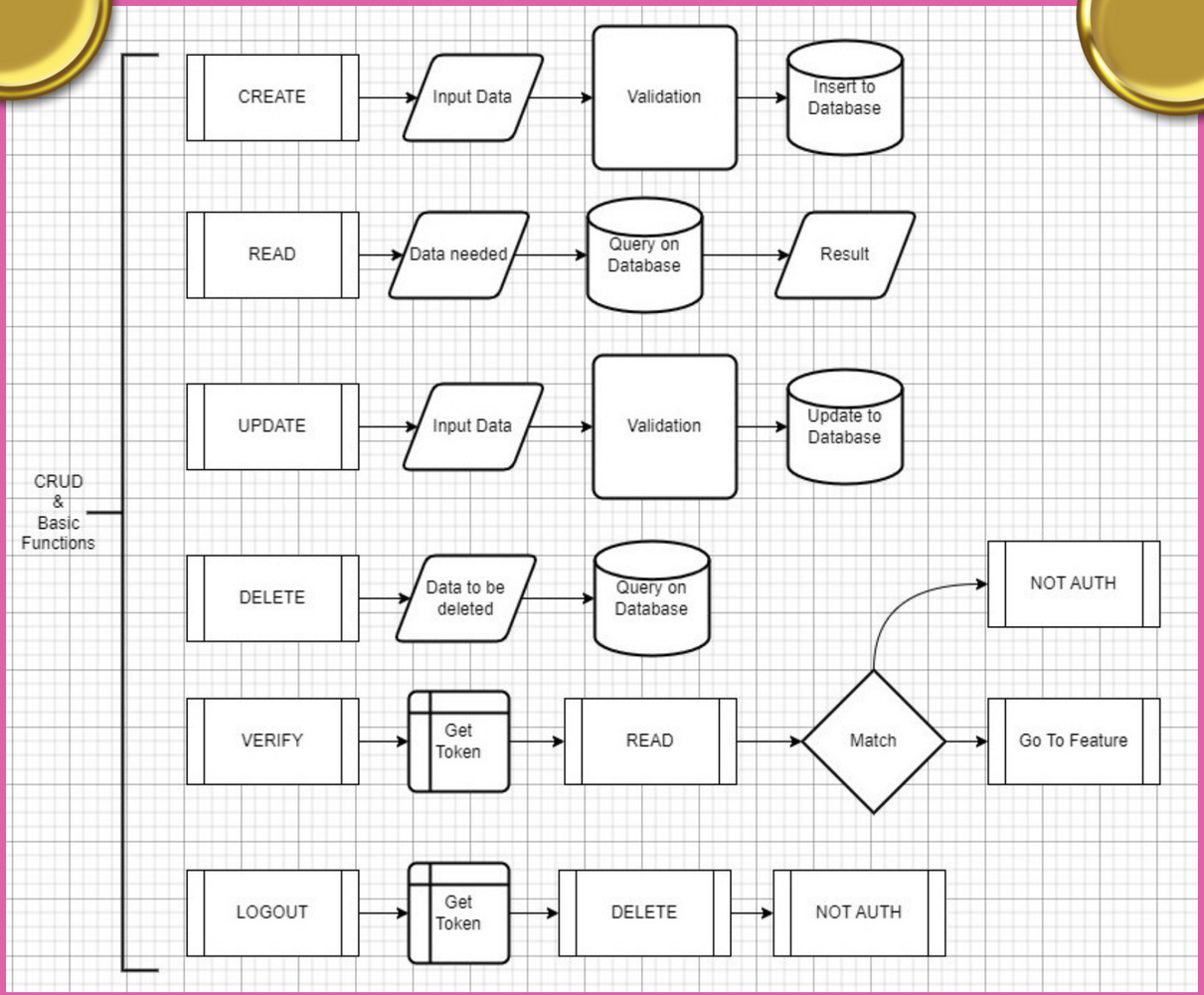
Entity Relationship Diagram

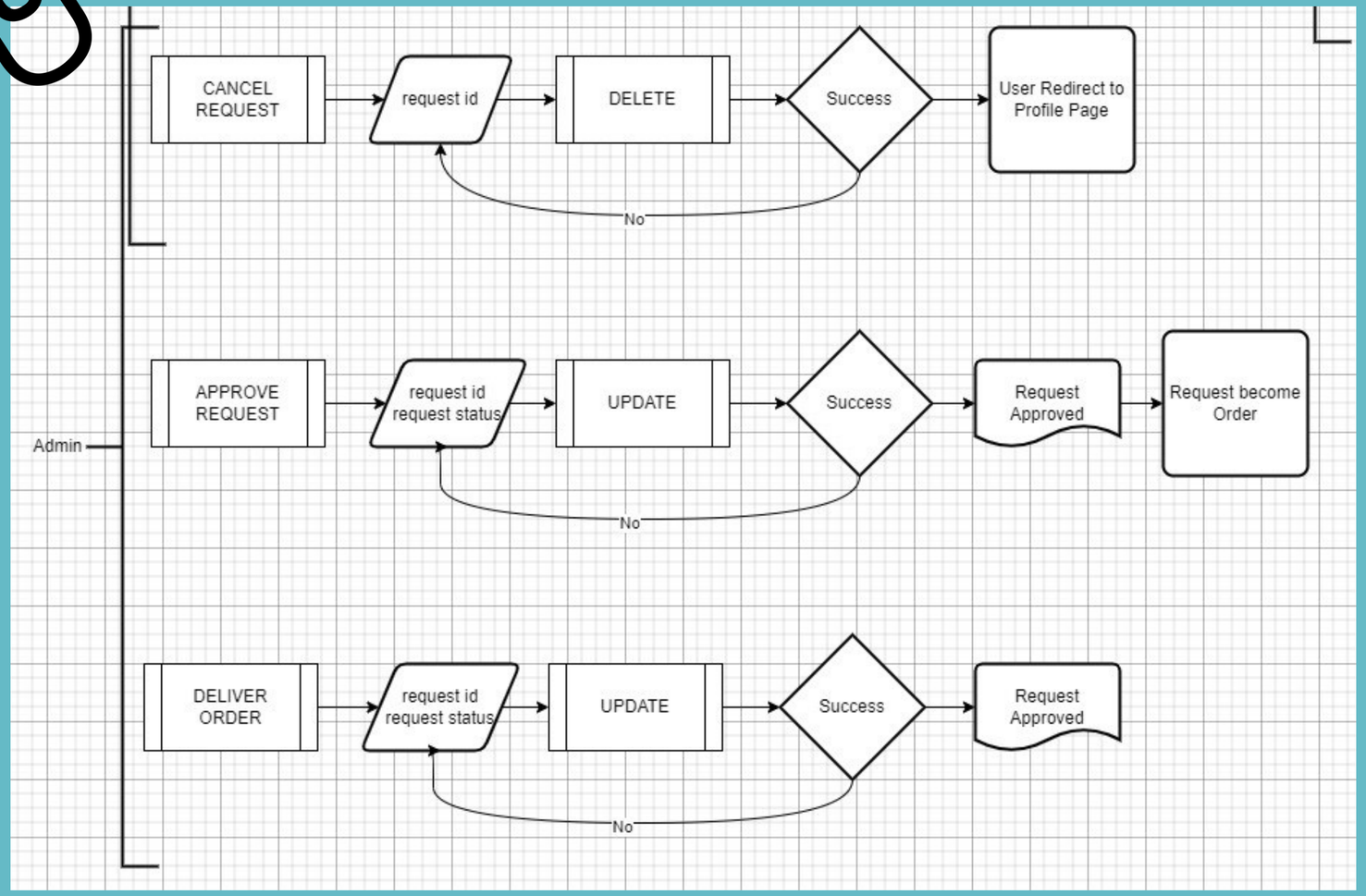


flowchart 1


START
User redirected to homepage
Click on any Feature
No
END
Yes
VERIFY
Yes
Init
Authenticated
No
NOT AUTH
Open Homepage
No
Go To Feature

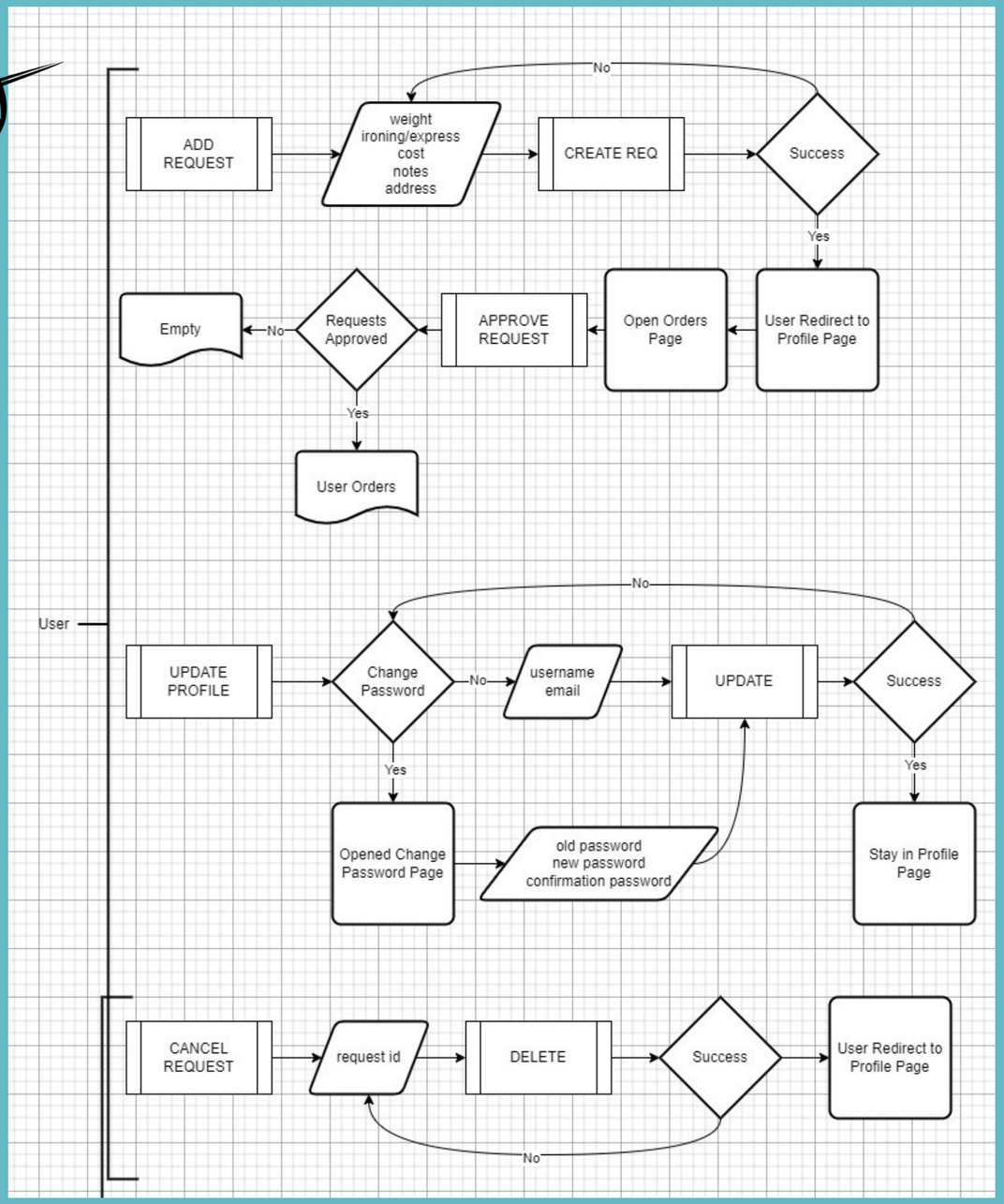
flowchart 2

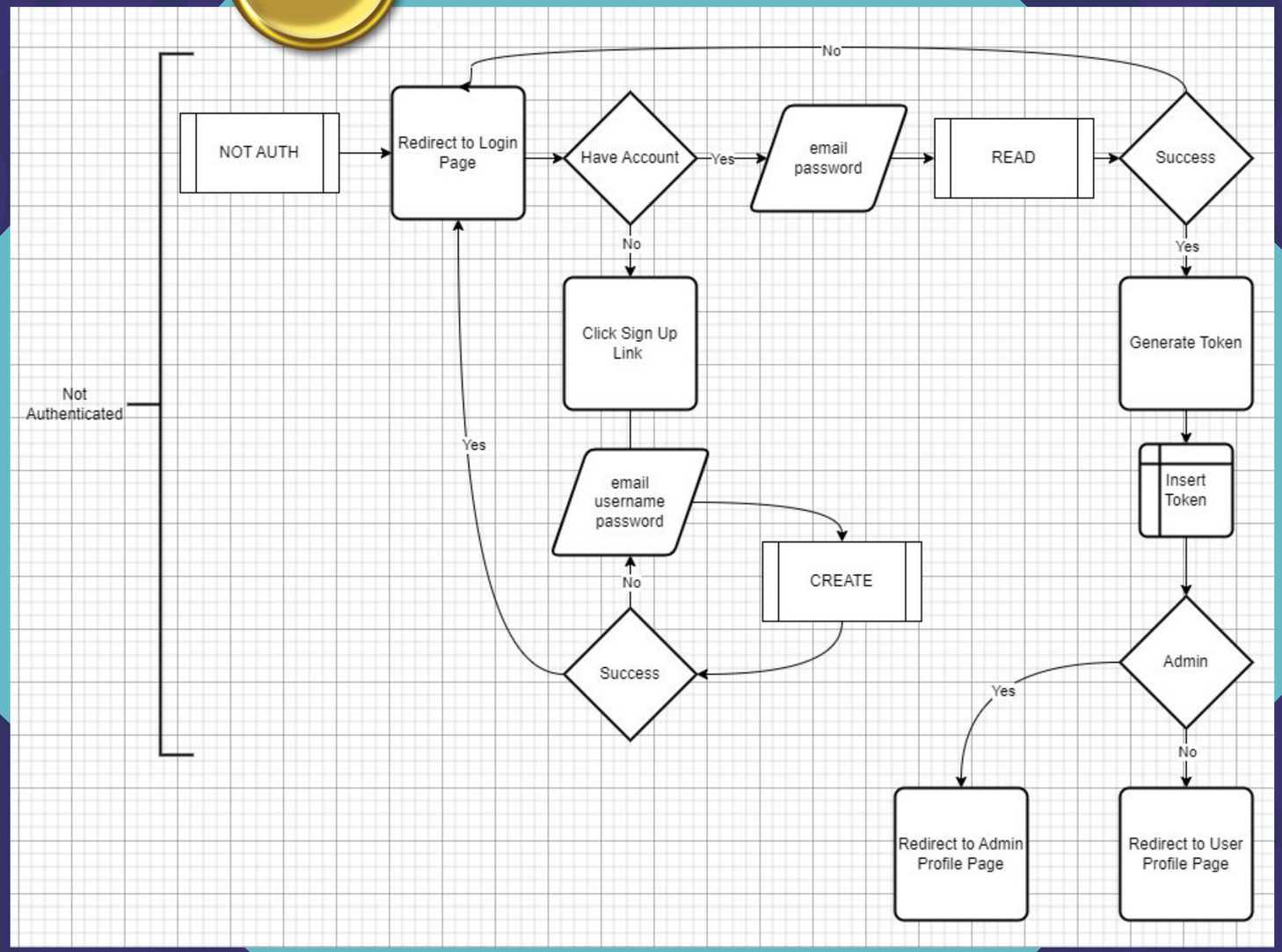

CRUD & Basic Functions
CREATE
Input Data
Validation
Insert to Database
READ
Data needed
Query on Database
Result
UPDATE
Input Data
Validation
Update to Database
DELETE
Data to be deleted
Query on Database
VERIFY
Get Token
READ
Match
NOT AUTH
Go To Feature
LOGOUT
Get Token
DELETE
NOT AUTH


CANCEL REQUEST
request id
DELETE
Success
User Redirect to Profile Page
No
Admin
APPROVE REQUEST
request id request status
UPDATE
Success
Request Approved
Request become Order
No
DELIVER ORDER
request id request status
UPDATE
Success
Request Approved
No

flowchart 3

flowchart 3


User
ADD REQUEST
weight
ironing/express
cost
notes
address
No
CREATE REQ
Success
Yes
User Redirect to Profile Page
Open Orders Page
APPROVE REQUEST
Requests Approved
No
Empty
Yes
User Orders
UPDATE PROFILE
Change Password
No
username
email
UPDATE
Success
No
Yes
Yes
Opened Change Password Page
old password
new password
confirmation password
Stay in Profile Page
CANCEL REQUEST
request id
DELETE
Success
User Redirect to Profile Page
No


Not Authenticated
NOT AUTH
Redirect to Login Page
Have Account
Yes
email password
READ
Success
No
Yes
No
Click Sign Up Link
Generate Token
Insert Token
email username password
CREATE
No
Yes
Success
Admin
Yes
No
Redirect to Admin Profile Page
Redirect to User Profile Page

flowchart 5

Resume & Demo

BACKEND

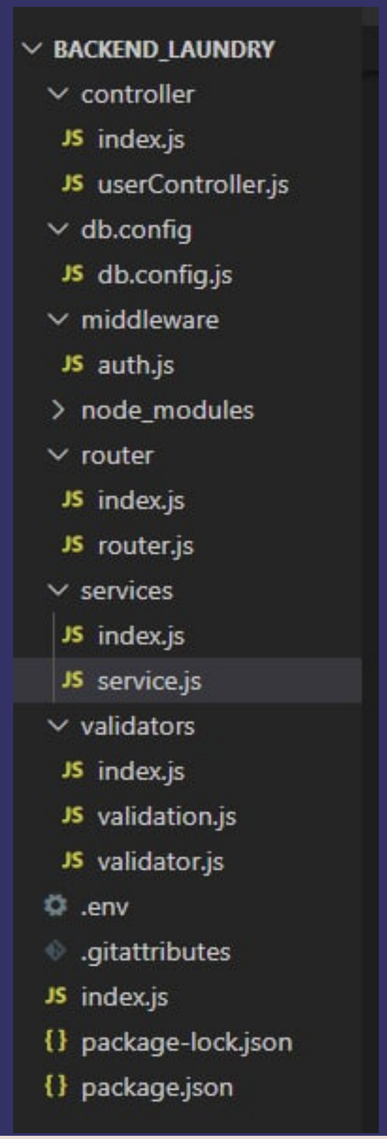
- foldering
- validasi
- crud
- connect database
- otentikasi, enkripsi, deployment

FRONTEND

- react js
- foldering
- otentikasi dan validasi
- deployment

backend

foldering



backend

validation

service.js | validation.js | package.json | index.js

validators > validation.js > login

```
1   const { param, body } = require("express-validator");
2   const { validator } = require("./validator");
3
4   //at least lowcase, uppercase, number
5   const passRegex = /^(?=.*[a-z])(?=.*[A-Z])(?=.*\d)[a-zA-Z\d]/;
6
7   //username required min 6
8   //email required isEmail
9   //password required min 8 match passRegex
10  //cust_name required isString
11  //cust_phone required min 12
12  //cust_address required isString
13
14  const register = [
15      body("username").isLength({ min: 6 }).notEmpty(),
16      body("email").isEmail().notEmpty(),
17      body("password").isLength({ min: 8 }).matches(passRegex).notEmpty(),
18      validator
19  ];
20
21  const login = [
22      body("email").isEmail().notEmpty(),
23      body("password").isLength({ min: 8 }).notEmpty(),
24      validator
25  ];
26
27  const updateprofile = [
28      body("user").isLength({ min: 6 }).notEmpty(),
29      body("email").isEmail().notEmpty(),
30      validator
31  ];
32
```

backend

crud

```
const addrequest = async (req_unit,req_cust,req_mode,req_notes,req_est) => {
    try {
        const date = new Date()
        const query = `INSERT INTO req_list(req_id,req_date,req_unit,req_cust,
            req_stat,req_mode,req_notes,req_est_cost)
            VALUES(DEFAULT,$1,$2,$3,DEFAULT,$4,$5,$6)`
        await db.query(query, [date,req_unit,req_cust,req_mode,req_notes,req_est])
        return('berhasil request')
    } catch (error) {
        return error
    }
}
```

```
const orders = async (id) => {
    try {
        if (id == '') {
            const query = `SELECT a.order_id, a.order_date, a.order_bill, a.order_unit, a.order_cust, a.order_mode, a.order_notes, b.stat_label
                FROM order_list a INNER JOIN order_status b ON a.order_stat=b.stat_id ORDER BY a.order_id`
            const requests = (await db.query(query)).rows
            return(requests)
        } else {
            const query = `SELECT a.order_id, a.order_date, a.order_bill, a.order_unit, a.order_cust, a.order_mode, a.order_notes, b.stat_label
                FROM order_list a INNER JOIN order_status b ON a.order_stat=b.stat_id WHERE a.order_user=$1 ORDER BY a.order_id`
            const requests = (await db.query(query, [id])).rows
            return(requests)
        }
    } catch (err){
        return Error
    }
}
```

backend

crud

```
const updateorder = async (order_stat,order_id) => {
    try {
        const query = `UPDATE order_list SET order_stat=$1 WHERE order_id = $2`
        await db.query(query, [order_stat,order_id])
        return('berhasil update')
    } catch (err){
        return Error
    }
}
```

```
const deleterequest= async (id) => {
    try {
        const query = `DELETE FROM req_list WHERE req_id = $1`
        await db.query(query, [id])
        return('berhasil cancel')
    } catch (err){
        return Error
    }
}
```

backend

connect database

```
const { Client } = require('pg');
require('dotenv').config()

const db = new Client({
    user: process.env.user,
    host: process.env.host,
    database: process.env.database,
    password: process.env.password,
    port: process.env.db_port,
});

module.exports = db;
```

.env

```
#db.config
user        = 'postgres'
host        = 'containers-us-west-134.railway.app'
database    = 'railway'
password    = 'q3yRnjtlfg9ETZmd208u'
db_port     = '5554'

#app
PORT    = '1200'
SECRET  = 'ambiltoken'
```

backend

otentikasi

middleware > auth.js > Auth > verifyToken

```
1   const jwt = require('jsonwebtoken');
2   require("dotenv").config();
3   SECRET = process.env.SECRET
4
5   const Auth = {
6       verifyToken(req, res, next){
7           const token = req.body.token
8           if (!token) {
9             return res.status(403).send("A token is required for authentication");
10          }
11          // 12. Lalukan jwt verify
12          try{
13            var verified = jwt.verify(token, SECRET);
14            req.user = verified
15            return next()
16          } catch(err){
17            res.status(403).send('Youre not authenticated, please login first')
18            console.log('Youre not authenticated');
19          }
20      }
21  }
22
23  module.exports = Auth;
```

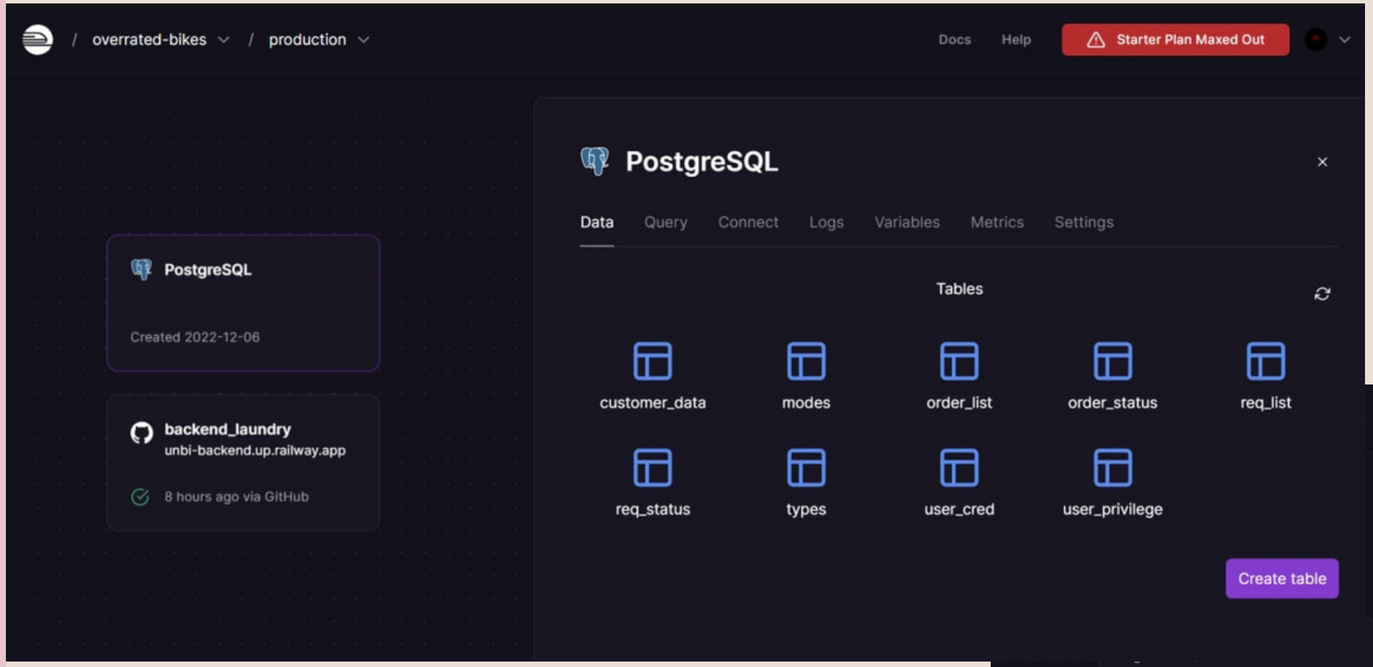
backend

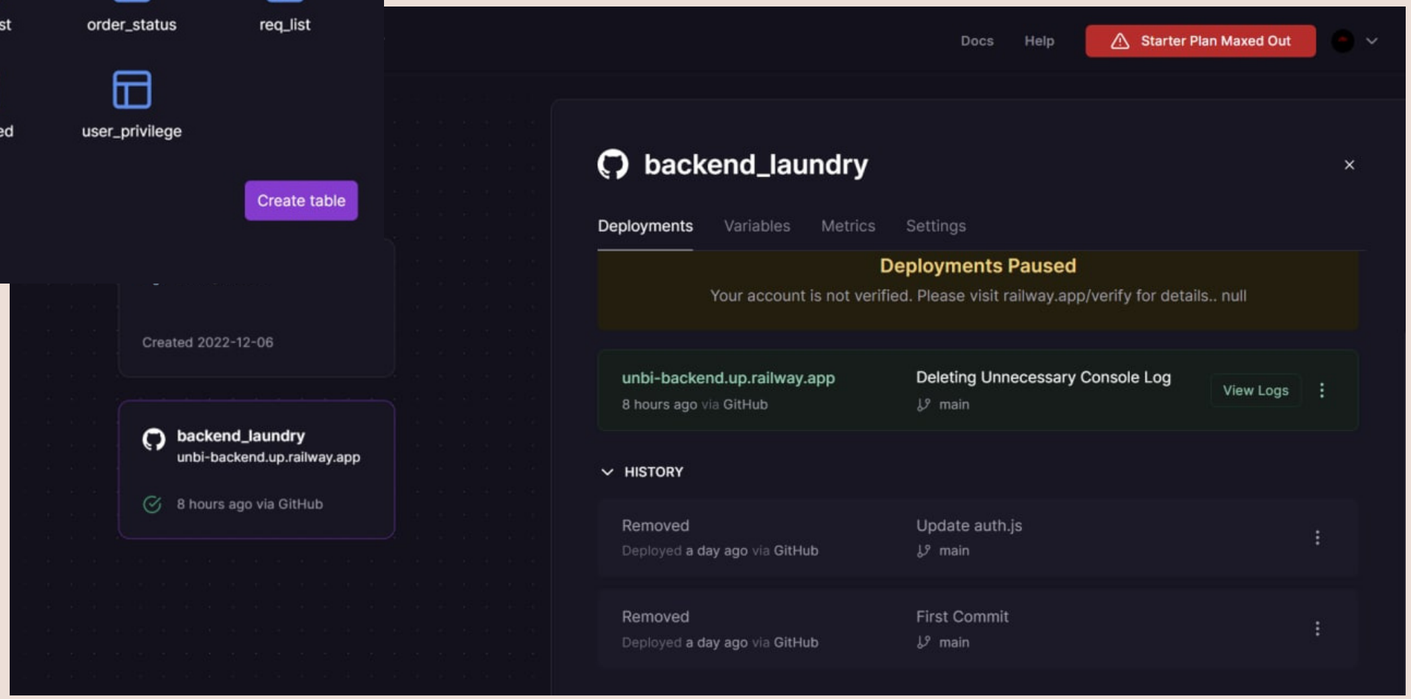
enkripsi

```
const login = async (email,password) => {
    try {
        const query = `SELECT * FROM user_cred WHERE user_email=$1`
        const user = await db.query(query, [email])
        var id = user.rows[0]['user_id']
        var username = user.rows[0]['username']
        var hash = user.rows[0]['user_pass']
        var hasil = await bcrypt.compare(password, hash)
        if (hasil == true) {
            // 10. Generate token menggunakan jwt sign
            let data = {
                id: id,
                username: username,
                email: email,
                password: hash
            }
            const token = jwt.sign(data, SECRET);
            const result = {
                id:id,
                username:username,
                email:email,
                token:token
            }
            return (result)
        } else {
            return('Password Salah');
        }
    } catch (error) {
        return error;
    }
}
```

backend

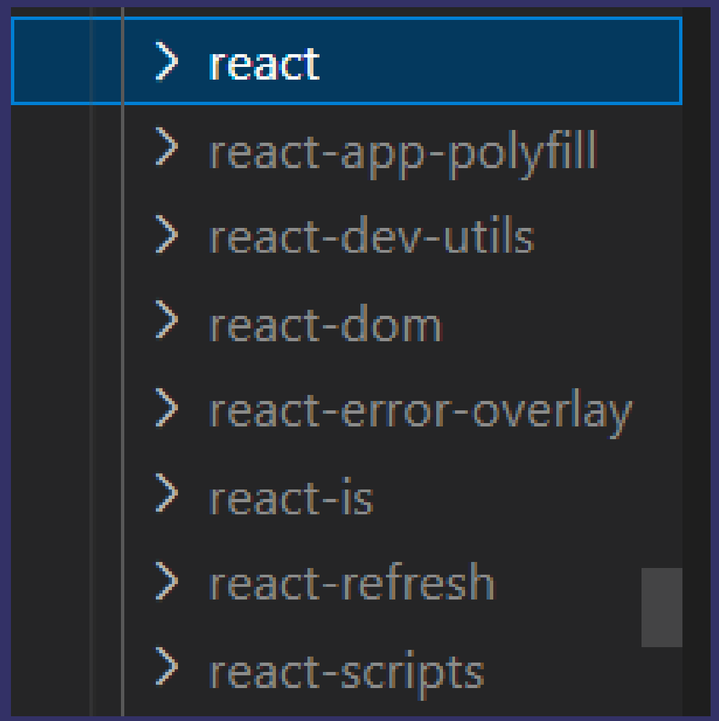
deployment





frontend

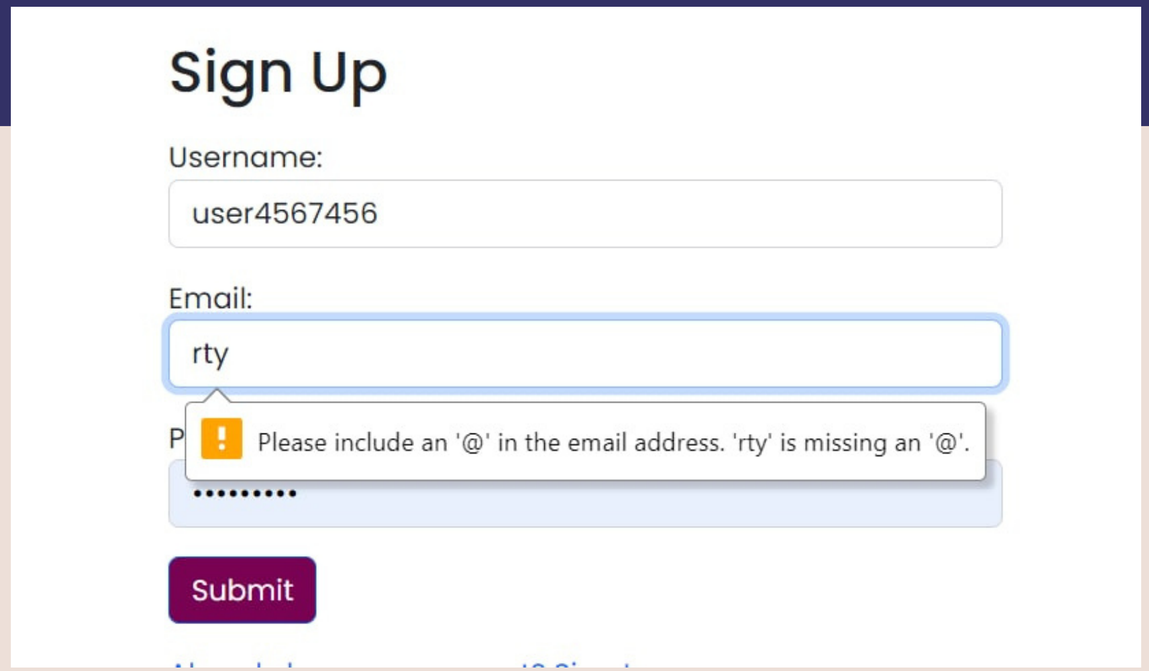
react js



frontend

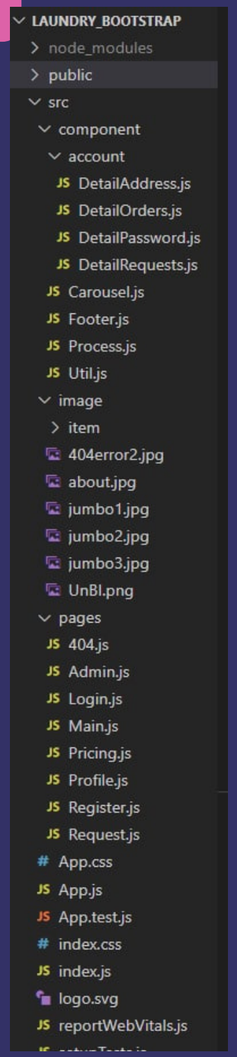
validasi

```
<div class="mb-3">
    <label for="email">Email:</label>
    <input type="email" class="form-control" id="email" placeholder="Enter email" name="email"/>
</div>
```



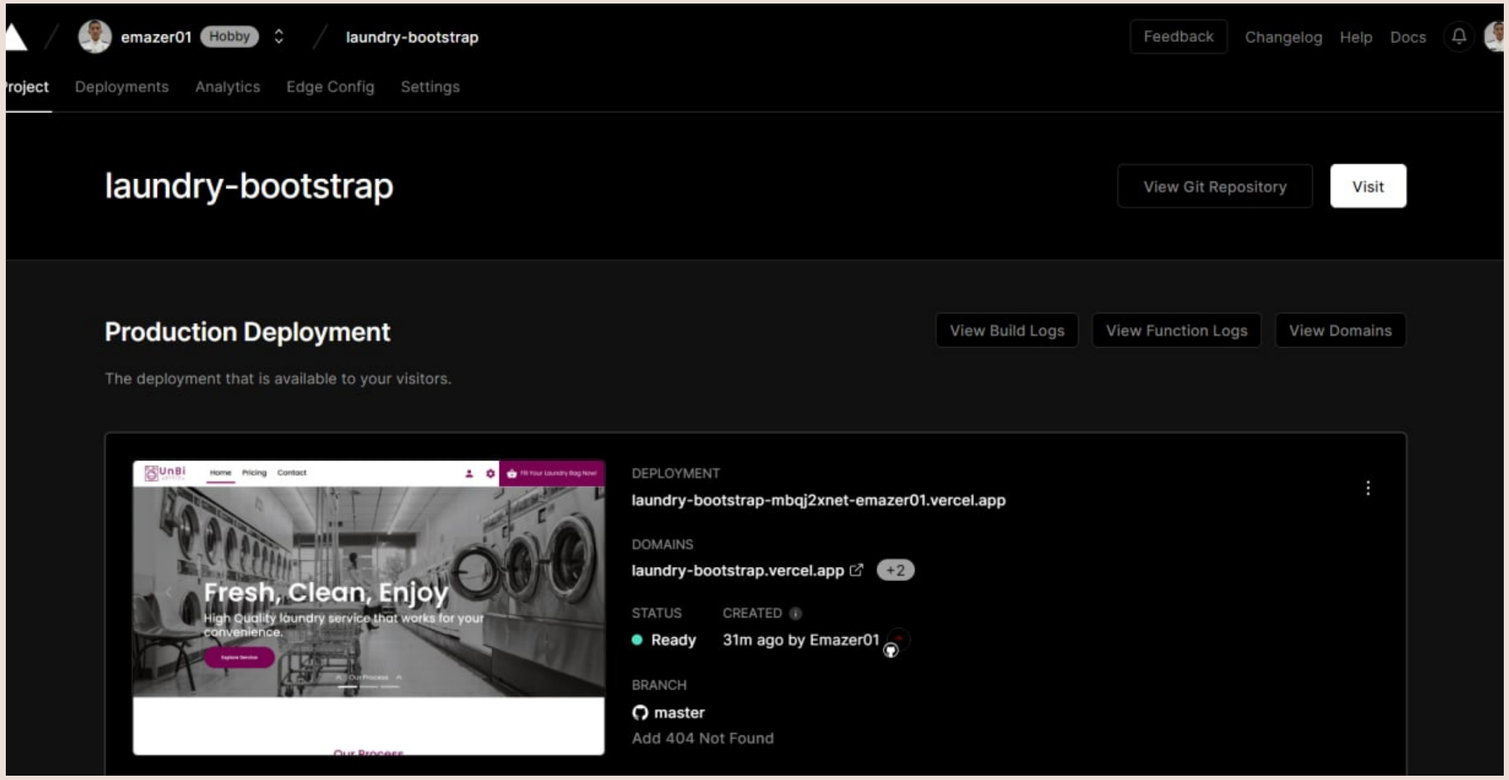
frontend

foldering



frontend

deployment



link

frontend

https://github.com/Emazer01/laundry_bootstrap

backend

https://github.com/Emazer01/backend_laundry

demo

https://laundry-bootstrap.vercel.app/


2020
GARY PERDINAND
AYUMAS SONITA