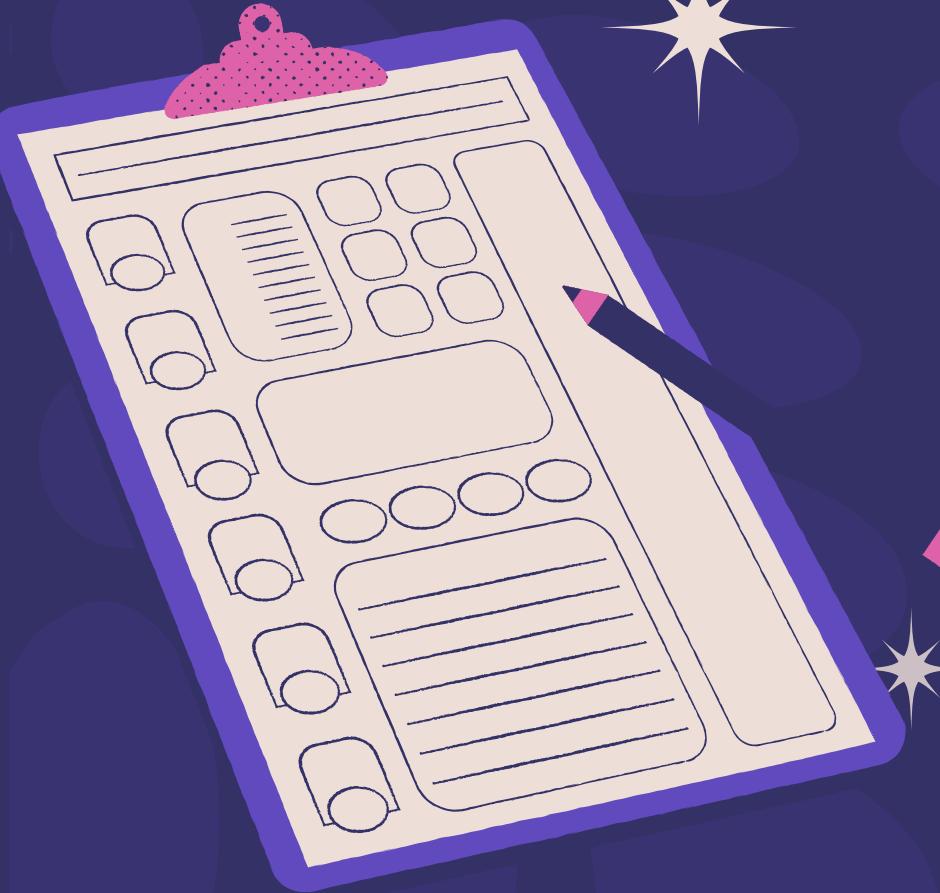


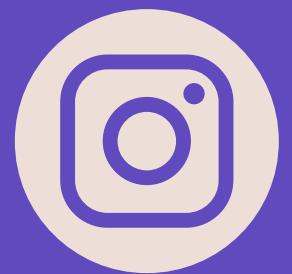
PROJECT AKHIR

laundry  
**UNBI**

FRESH - CLEAN - ENJOY



## contact



@prasashtiazz\_

@gary\_ferdinand

@adzs72

## Players



njn



emazernlt



adzs

# Topic



introduction project

concept

web presentation / demo

# introduction project



## Background case

# *Laundry Business Process*

non delivery  
service

delivery service

# NON DELIVERY SERVICE

non  
delivery  
service

delivery  
service

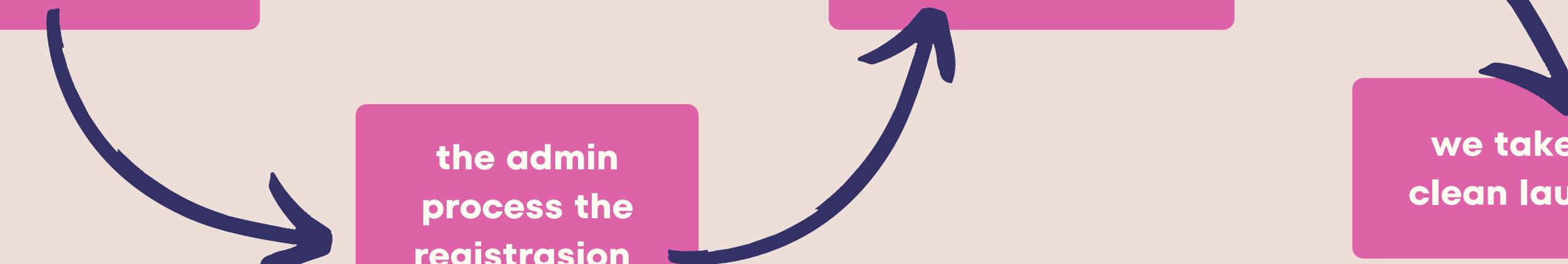
## *Laundry Business Process*

we put our  
laundry by  
oursself

the admin  
process the  
registrasion

we get the receipt  
and the time  
estimation

we take the  
clean laundry

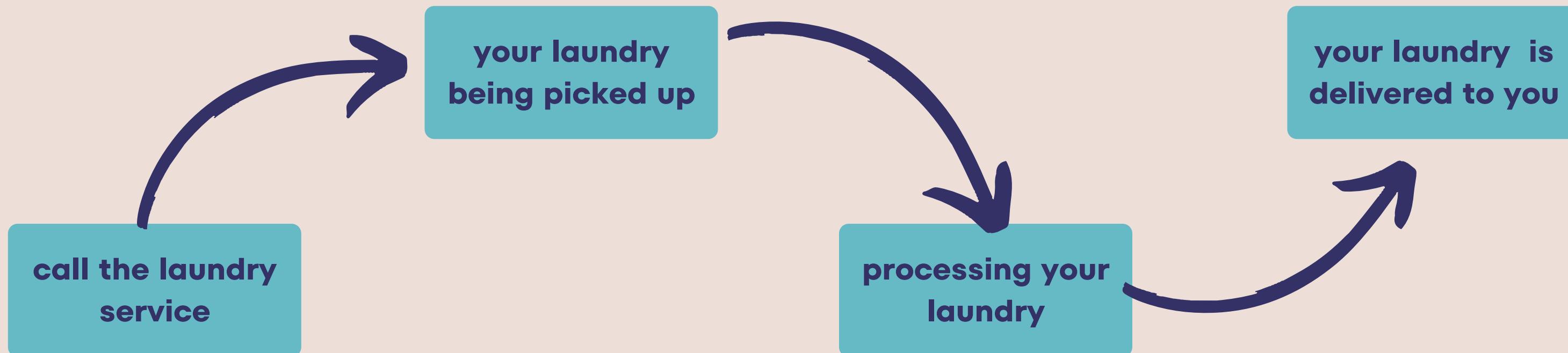


# Delivery Service

non  
delivery  
service

delivery  
service

## *Laundry Business Process*



# Formulation of the problem



non delivery service

delivery service

**cust get their receipt**

**no need to deliver the laundry**

**cust get the estimation time**

**customer doesn't get receipt**

**customer have to deliver the laundry by themselves**

**has no idea when the laundry is finished**

**the shop continuously chased by deadlines**

# 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

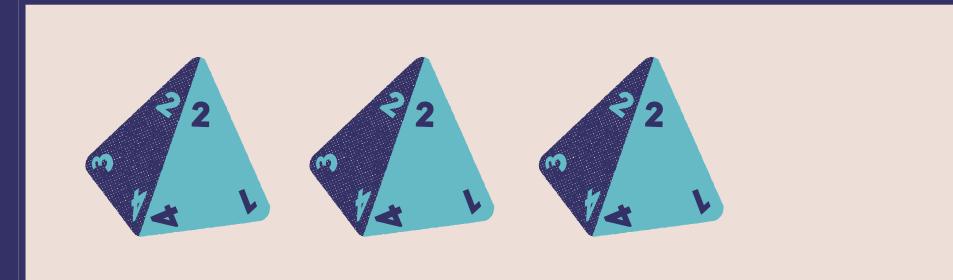
01

DORMITORY



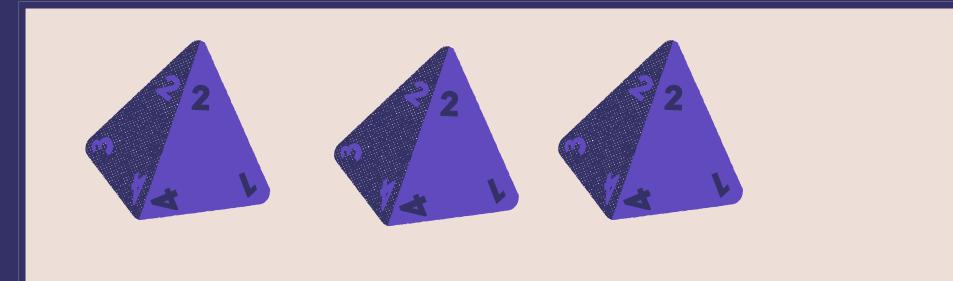
02

EMPLOYEE



03

COLLEGE  
STUDENT

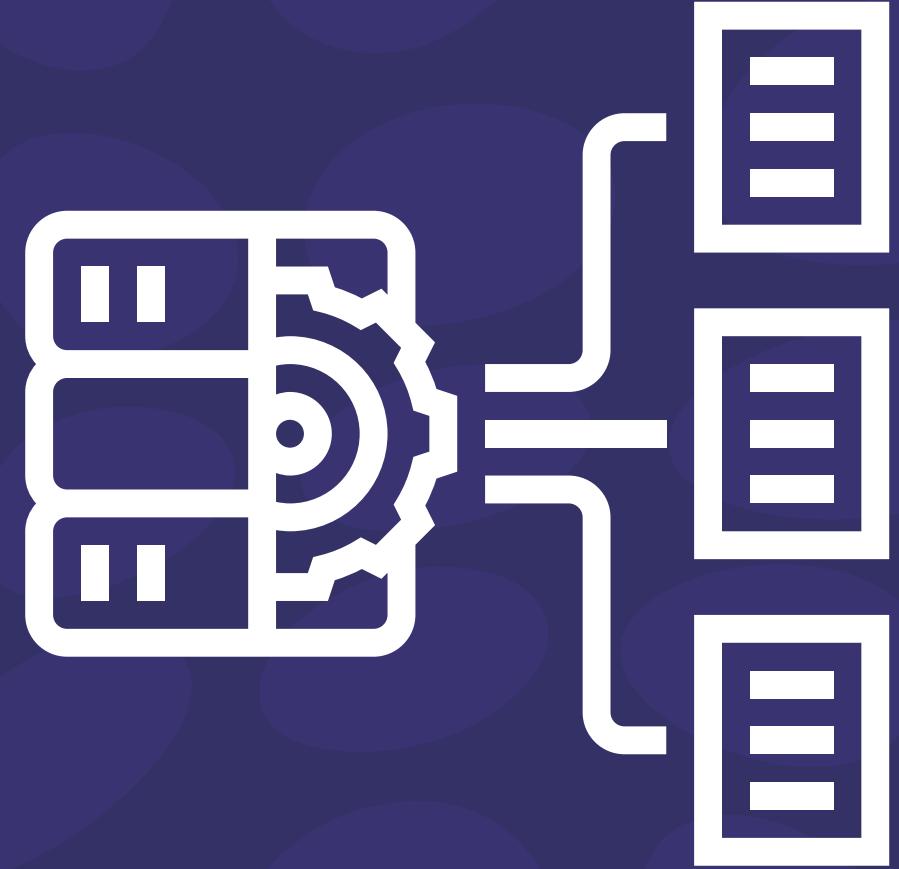


# MARKETING TARGET

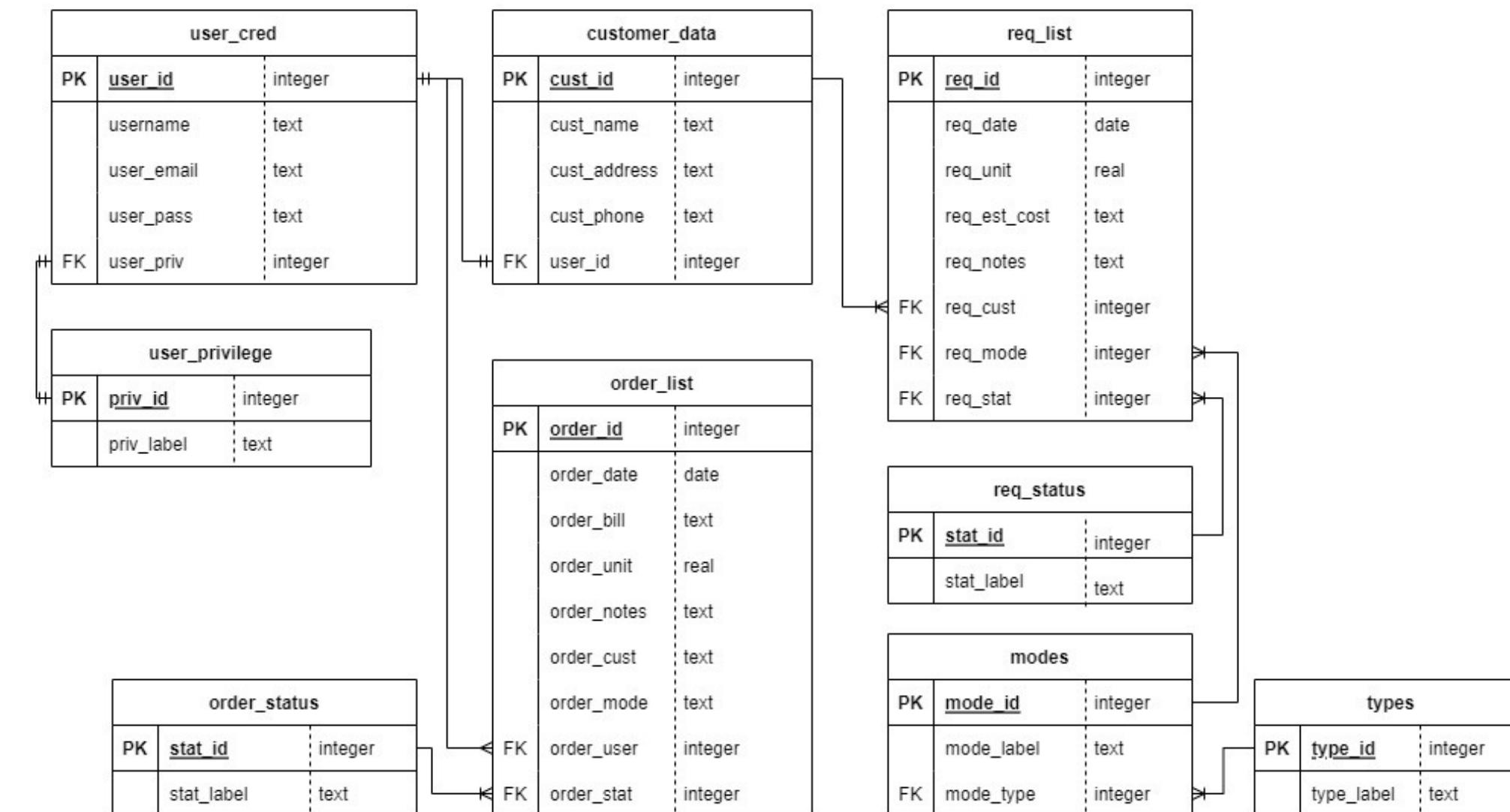
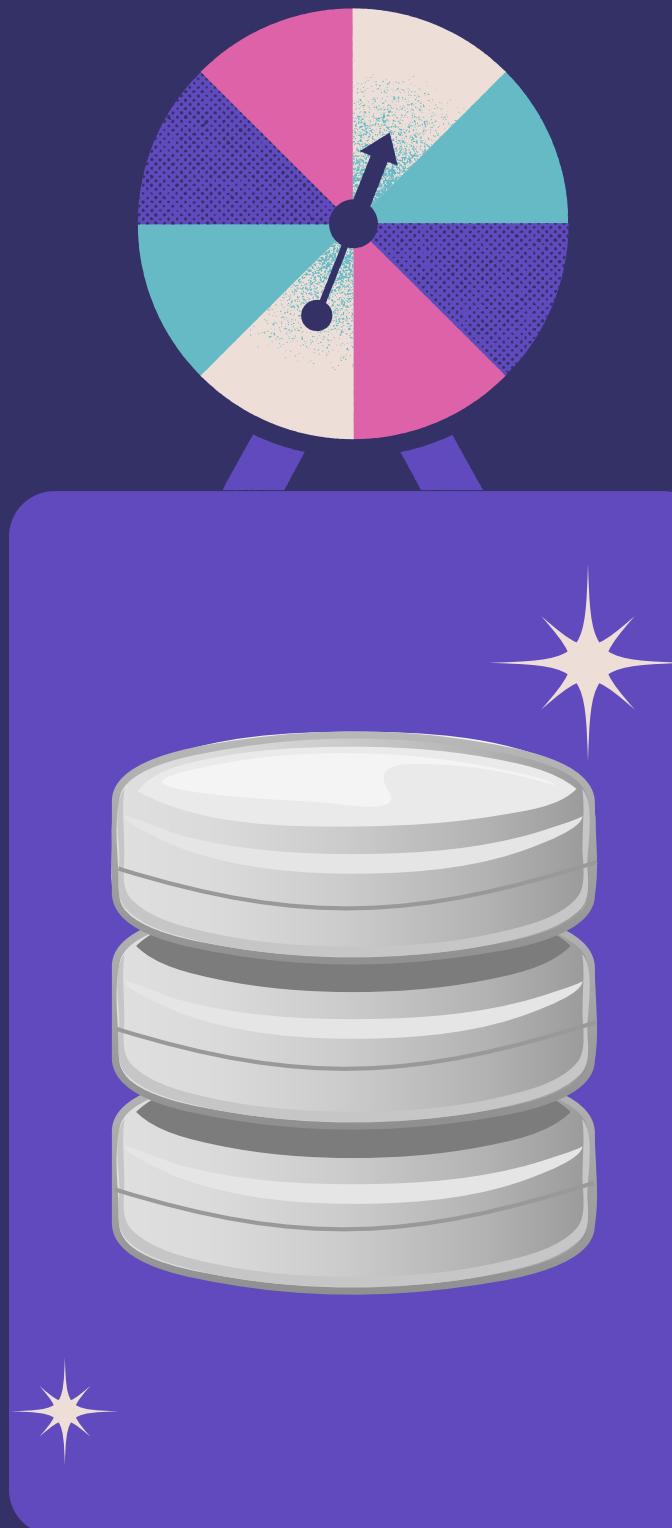




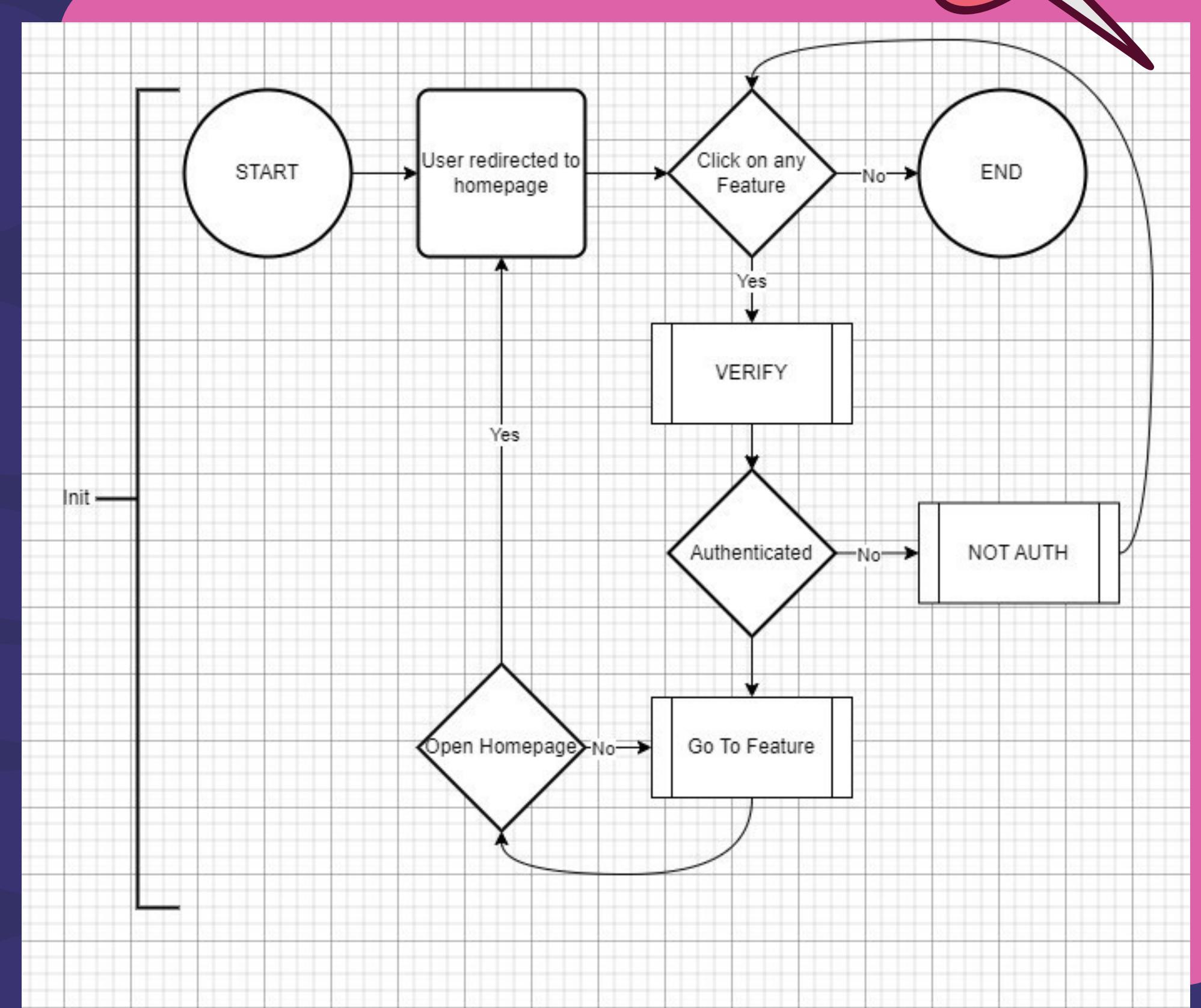
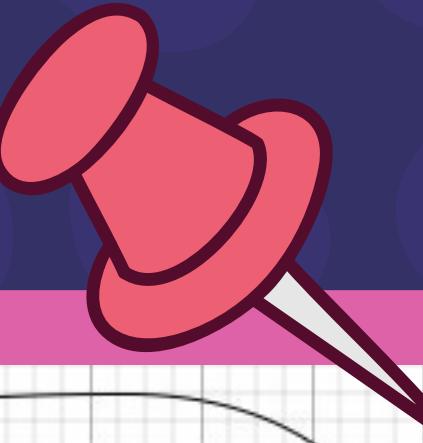
*web app*  
**concept**



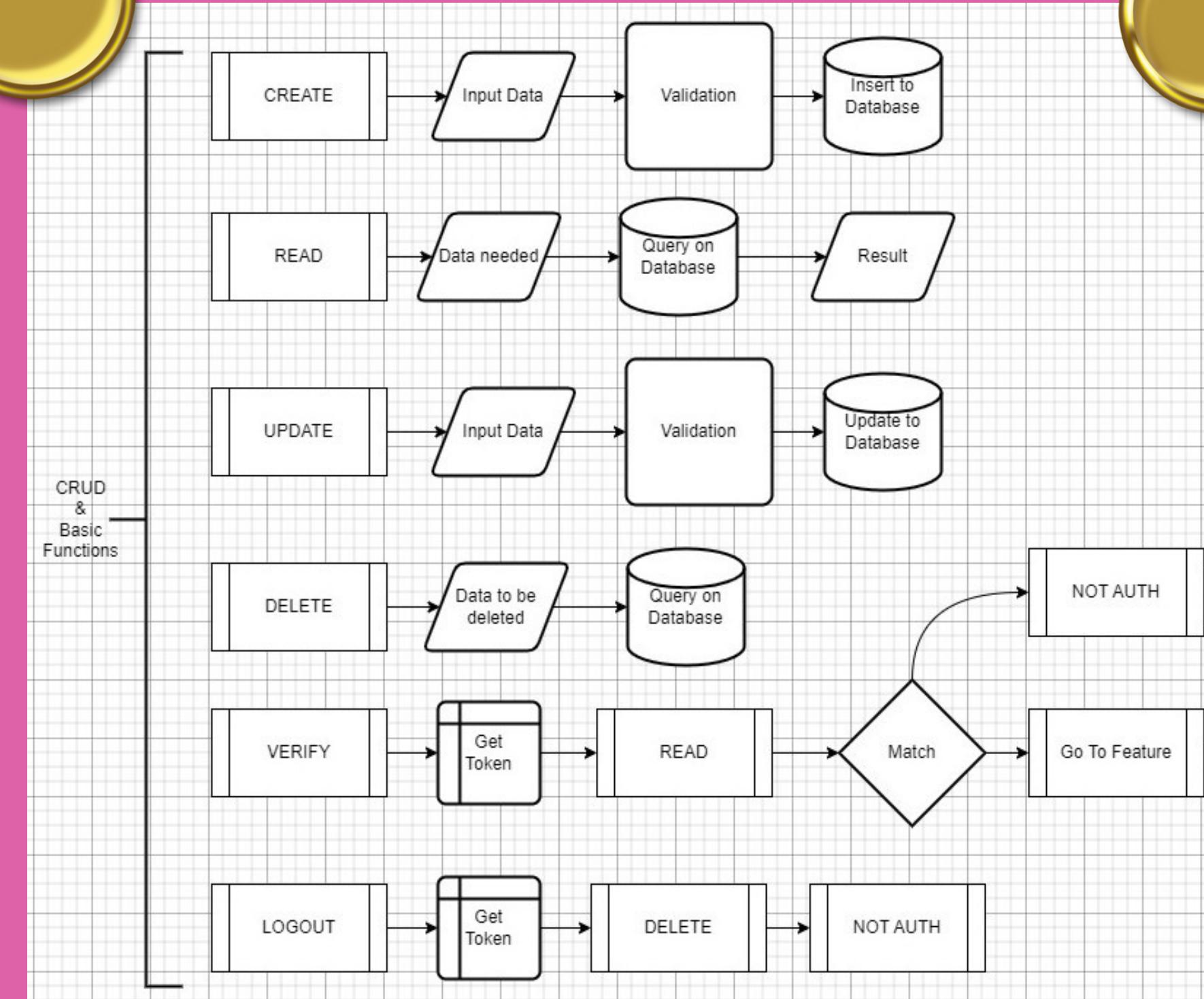
# Entity Relationship Diagram



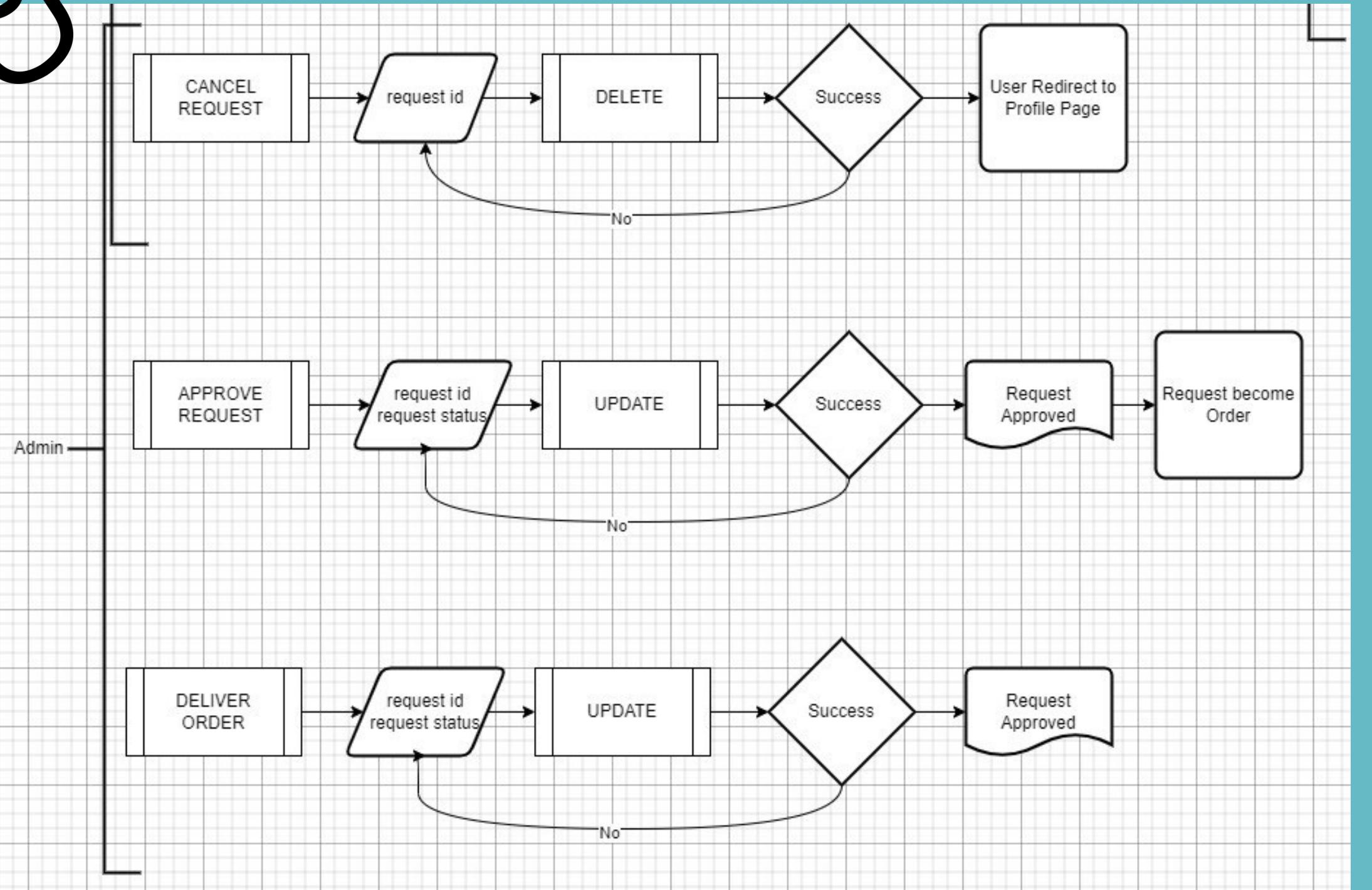
# flowchart 1



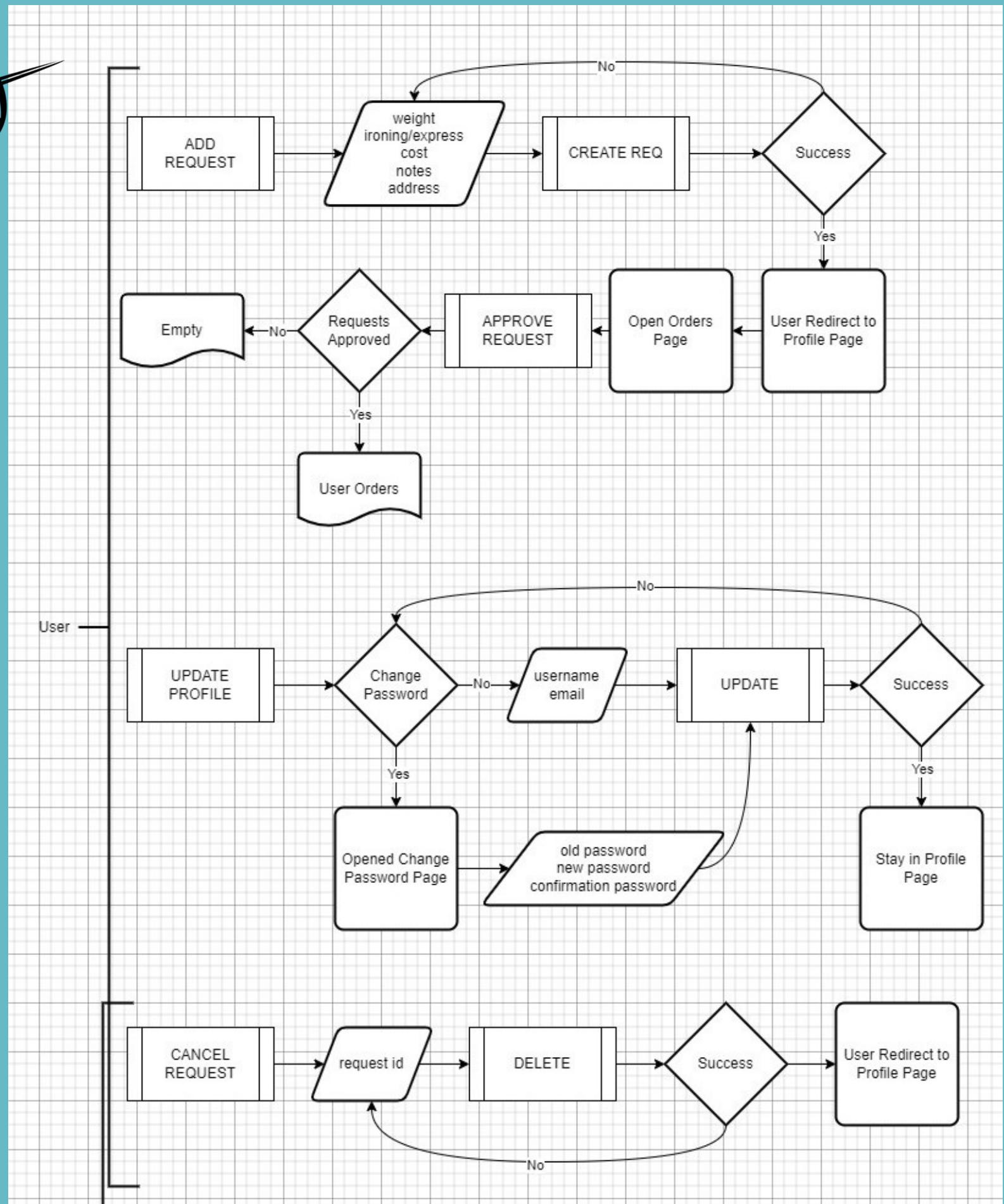
# flowchart 2

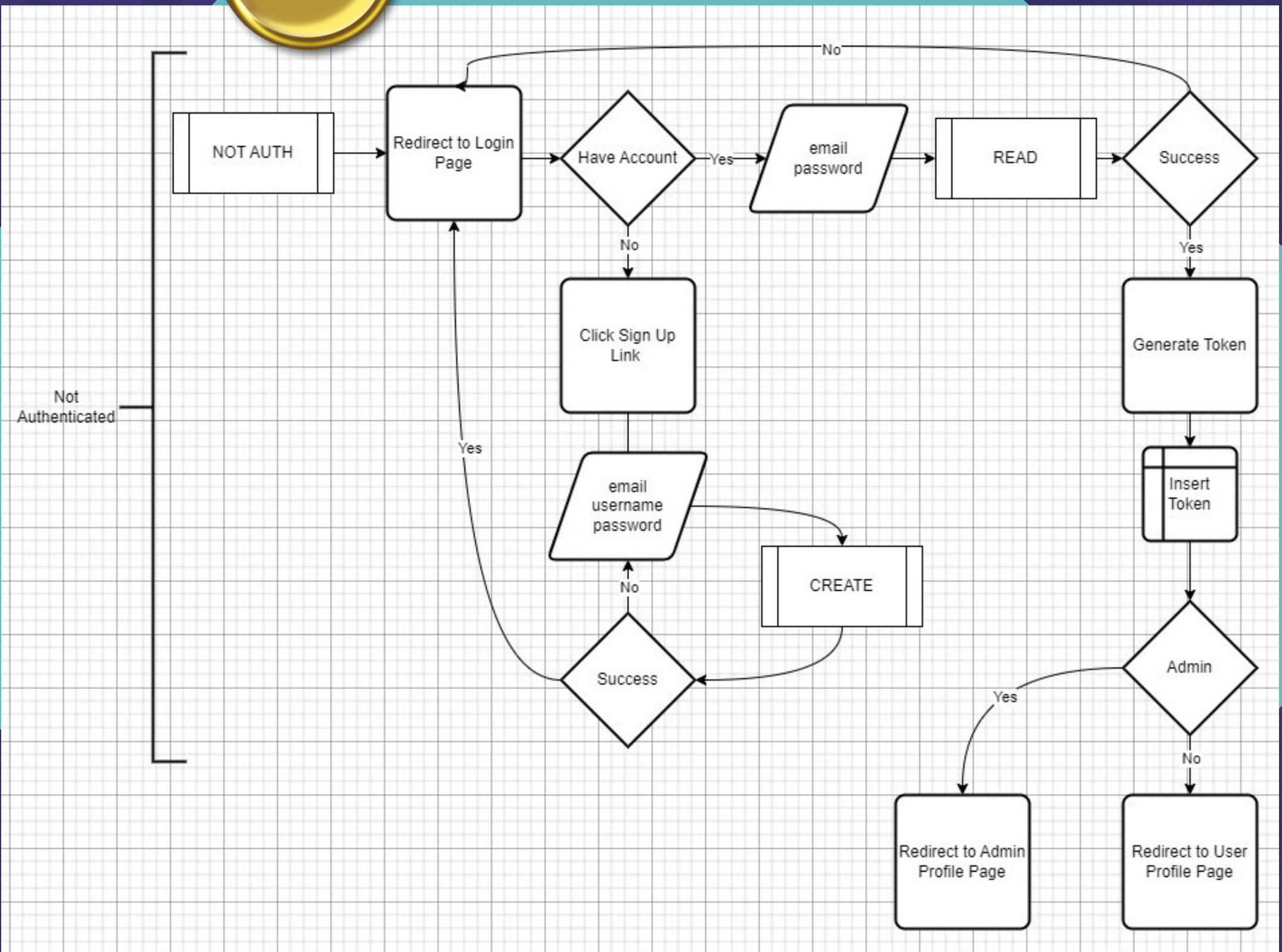


# flowchart 3



# flowchart 3





# flowchart 5

# Resume & Demo

BACKEND

FRONTEND

foldering

validasi

crud

connect database

otentikasi, enkripsi, deployment

react js

foldering

otentikasi  
dan validasi

deployment

# backend

## foldering

### BACKEND\_LAUNDRY

```
  ↘ controller
    ↗ index.js
    ↗ userController.js
  ↘ db.config
    ↗ db.config.js
  ↘ middleware
    ↗ auth.js
  > node_modules
  ↘ router
    ↗ index.js
    ↗ router.js
  ↘ services
    ↗ index.js
    ↗ service.js
  ↘ validators
    ↗ index.js
    ↗ validation.js
    ↗ validator.js
  .env
  .gitattributes
  ↗ index.js
  {} package-lock.json
  {} package.json
```

# backend

## validation

```
JS service.js   JS validation.js X  {} package.json  JS index.js
```

```
validators > JS validation.js > [x] login
1  const { param, body } = require("express-validator");
2  const { validator } = require("./validator");
3
4  //at least lowercase, 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

```
❶ .env
❷ #db.config
❸ user      = 'postgres'
❹ host      = 'containers-us-west-134.railway.app'
❺ database  = 'railway'
❻ password  = 'q3yRnjtlfg9ETZmd208u'
❼ db_port   = '5554'

❽ #app
❾ PORT      = '1200'
❿ SECRET    = 'ambiltoken'
```

```
❶ const { Client } = require('pg');
❷ require('dotenv').config()
❸
❹ const db = new Client({
❺   user: process.env.user,
❻   host: process.env.host,
❽   database: process.env.database,
❾   password: process.env.password,
❿   port: process.env.db_port,
❾ });
❾
❿ module.exports = db;
```

# backend

## otentikasi

```
middleware > JS auth.js > [o] Auth > [o] 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. Lakukan 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 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

The screenshot displays the Railway app interface, featuring two main service pages: PostgreSQL and backend\_laundry.

**PostgreSQL Service:**

- Overview:** Shows the PostgreSQL service was created on 2022-12-06. It includes a GitHub deployment log from 8 hours ago via GitHub.
- Tables:** A grid of icons representing tables: customer\_data, modes, order\_list, order\_status, req\_list, req\_status, types, user\_cred, and user\_privilege.
- Create table:** A purple button at the bottom right of the table section.

**backend\_laundry Service:**

- Deployment Status:** Shows "Deployments Paused" due to an unverified account. It lists a recent deployment to unbi-backend.up.railway.app via GitHub.
- History:** A list of deployment events:
  - Removed (Deployed a day ago via GitHub) - Update auth.js
  - Removed (Deployed a day ago via GitHub) - First Commit

# frontend

react js

> react

> react-app-polyfill

> react-dev-utils

> react-dom

> react-error-overlay

> react-is

> react-refresh

> react-scripts

# 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>
```

## Sign Up

Username:

user4567456

Email:

rty

Please ! Please include an '@' in the email address. 'rty' is missing an '@'.

.....

Submit

# frontend

foldering

```
LAUNDRY_BOOTSTRAP
  node_modules
  public
  src
    component
      account
        DetailAddress.js
        DetailOrders.js
        DetailPassword.js
        DetailRequests.js
      Carousel.js
      Footer.js
      Process.js
      Util.js
    image
      404error2.jpg
      about.jpg
      jumbo1.jpg
      jumbo2.jpg
      jumbo3.jpg
      UnBI.png
    pages
      404.js
      Admin.js
      Login.js
      Main.js
      Pricing.js
      Profile.js
      Register.js
      Request.js
      App.css
      App.js
      App.test.js
      index.css
      index.js
      logo.svg
      reportWebVitals.js
```

# frontend

deployment

laundry-bootstrap

Production Deployment

The deployment that is available to your visitors.

DEPLOYMENT

laundry-bootstrap-mbqj2xnet-emazer01.vercel.app

DOMAINS

laundry-bootstrap.vercel.app +2

STATUS CREATED 31m ago by Emazer01

BRANCH master

Add 404 Not Found

# link

frontend

backend

**[https://github.com/Emazer01/laundry\\_bootstrap](https://github.com/Emazer01/laundry_bootstrap)**

**[https://github.com/Emazer01/backend\\_laundry](https://github.com/Emazer01/backend_laundry)**

WWW.THECASE.COM  
CASE CLOSED

