vFlashCards

Flashcards at your fingertip.

Study aids on the go.

CISC 4900
Brooklyn College - City University of New York

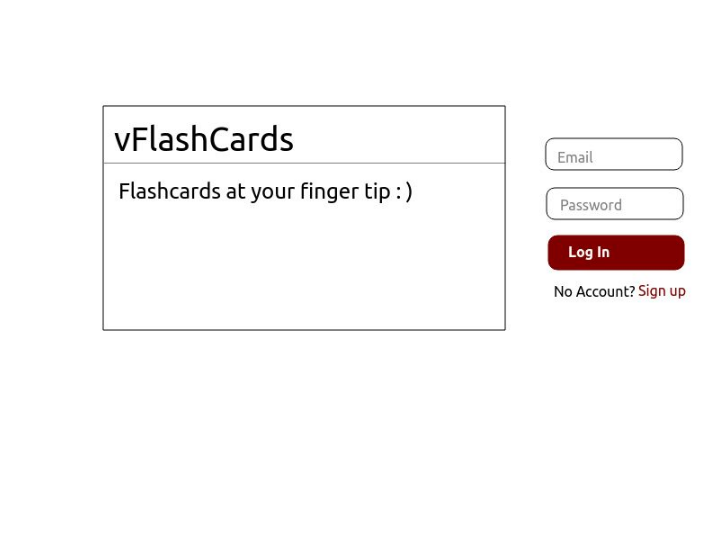
Professor Chuang
Professor Lapid

Min-Chun Lo

What & Why?

- Full stack web application virtual flashcards. Built on PostgreSQL, Express, React, Node.js.
- Because everything can be organized in one place.
- Convenience of study aids for students on the go.
- Not easily misplaced.

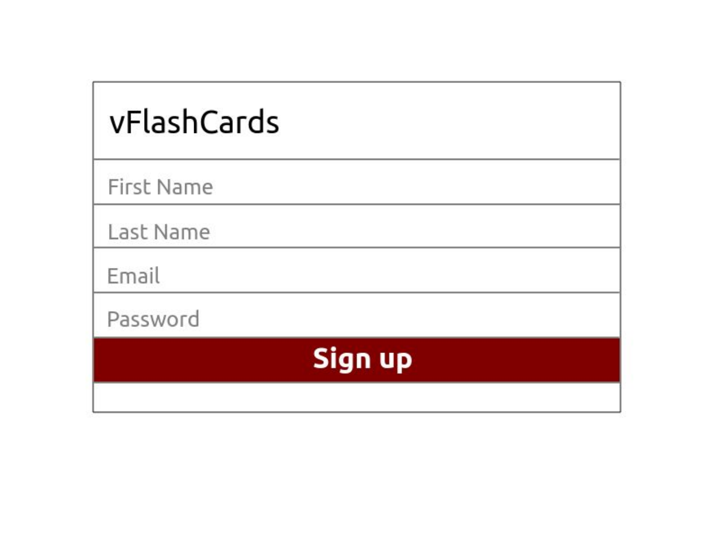
UI/Mockup



vFlashCards

Flashcards at your finger tip :)

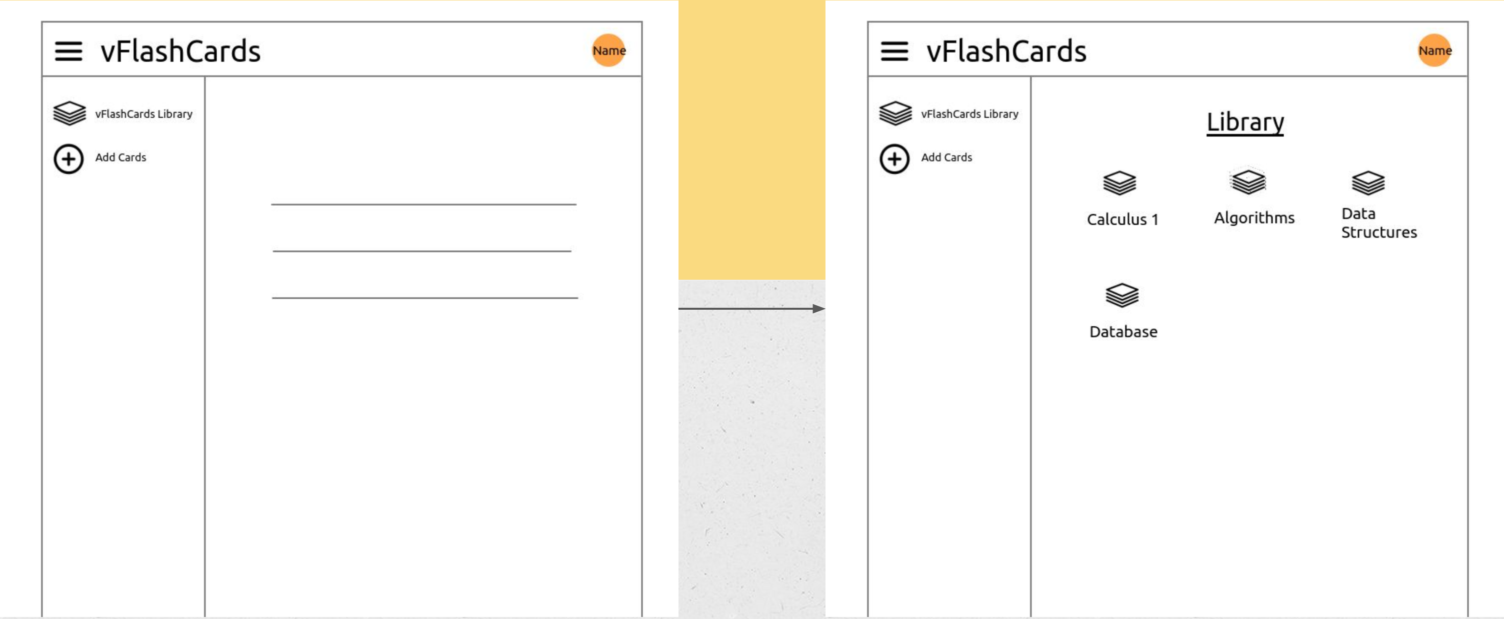
No Account? Sign up

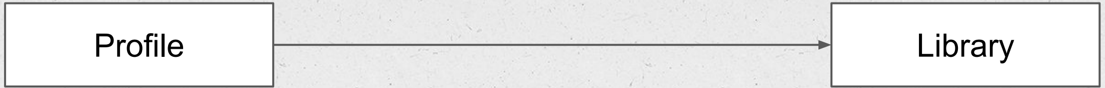


vFlashCards

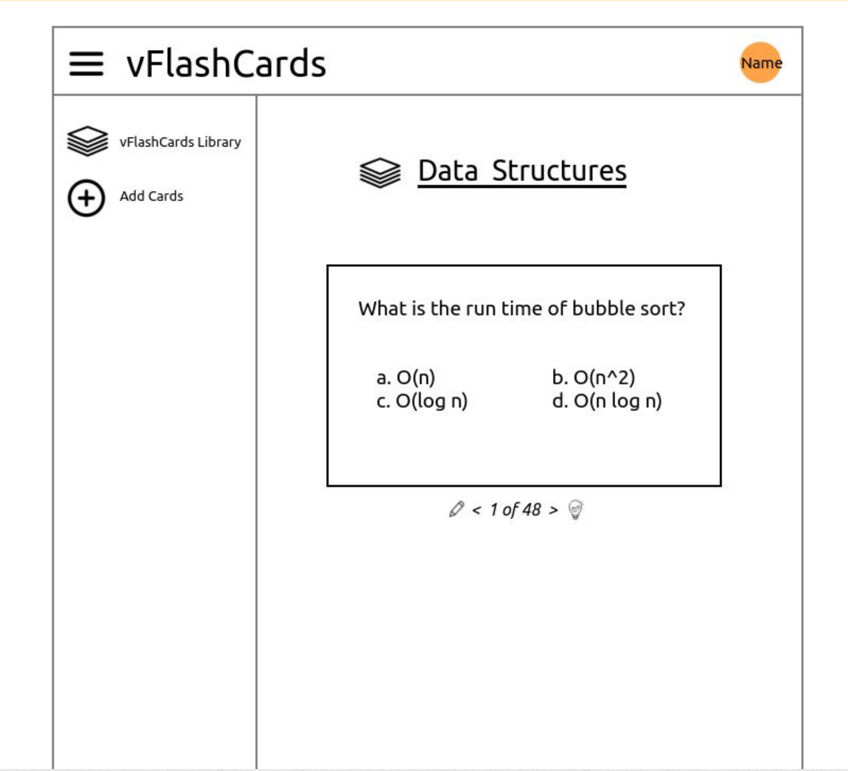
Home / Landing page
Signup

UI/Mockup


vFlashCards
Name
vFlashCards Library
Add Cards
vFlashCards
Name
vFlashCards Library
Add Cards
Library
Calculus 1
Algorithms
Data Structures
Database


Profile
Library

UI/Mockup



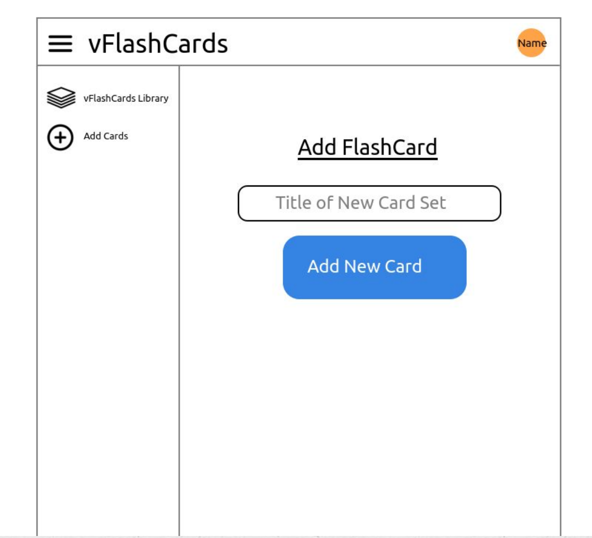
What is the run time of bubble sort?

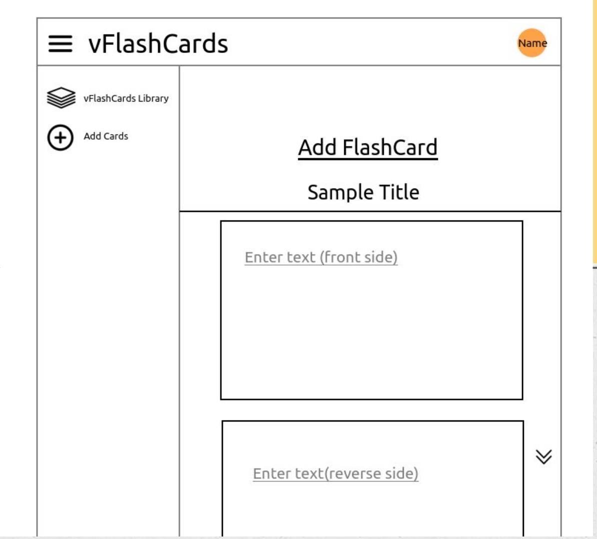
a. O(n)
b. O(n^2)
c. O(log n)
d. O(n log n)

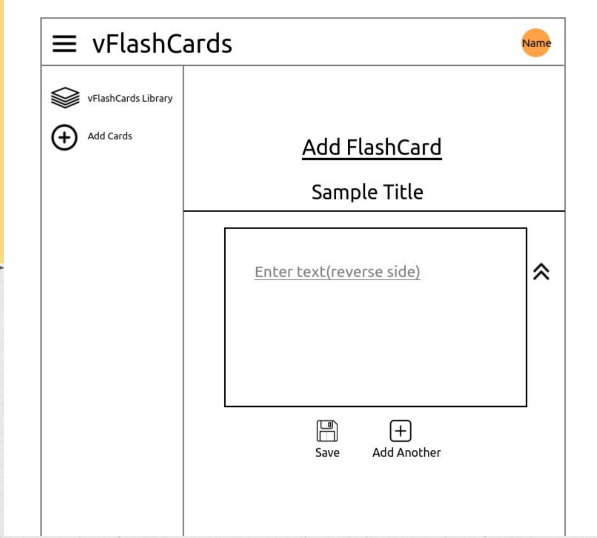
< 1 of 48 >

Flashcard

UI/Mockup







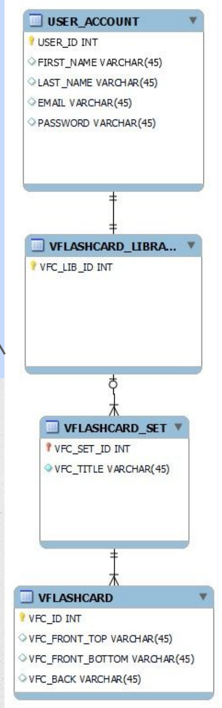
Add card set → Add question → Add answer and save

Note: Mockup created on KolourPaint (Linux). Not recommended as a first choice. Google Drawings can be a good choice due to its low learning curve and features. Frontend to be built before the end of the semester.

Behind the Scenes Backend

Database ERD

First ERD, which shows what I thought the data should be structured.



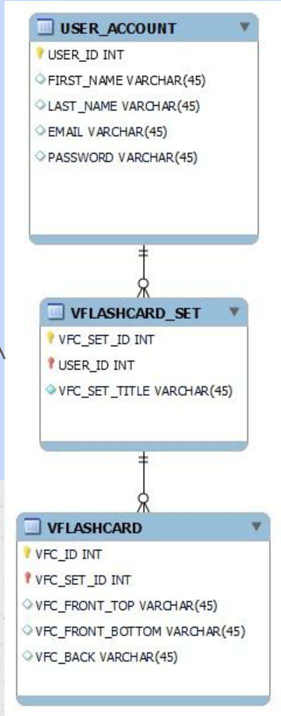
USER_ACCOUNT
USER_ID INT
FIRST_NAME VARCHAR(45)
LAST_NAME VARCHAR(45)
EMAIL VARCHAR(45)
PASSWORD VARCHAR(45)

VFLASHCARD_LIBRA...
VFC_LIB_ID INT

VFLASHCARD_SET
VFC_SET_ID INT
VFC_TITLE VARCHAR(45)

VFLASHCARD
VFC_ID INT
VFC_FRONT_TOP VARCHAR(45)
VFC_FRONT_BOTTOM VARCHAR(45)
VFC_BACK VARCHAR(45)

Deleted redundant table as per recommendation from Edgardo Molina



USER_ACCOUNT
USER_ID INT
FIRST_NAME VARCHAR(45)
LAST_NAME VARCHAR(45)
EMAIL VARCHAR(45)
PASSWORD VARCHAR(45)

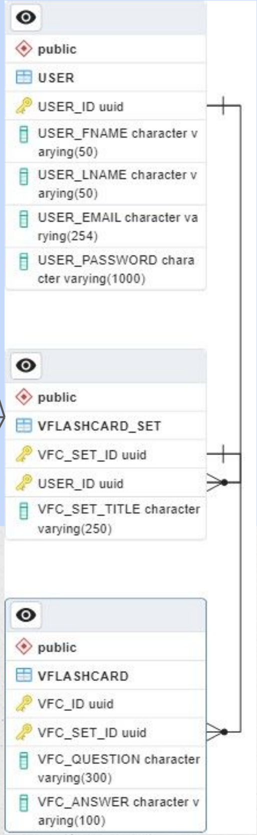
VFLASHCARD_SET
VFC_SET_ID INT
USER_ID INT
VFC_SET_TITLE VARCHAR(45)

VFLASHCARD
VFC_ID INT
VFC_SET_ID INT
VFC_FRONT_TOP VARCHAR(45)
VFC_FRONT_BOTTOM VARCHAR(45)
VFC_BACK VARCHAR(45)

Recreated ERD. Amended the column name and data type.

Generated script (via pgAdmin) required double quotes for all tables, which was unexpected..

Deleted all existing tables and recreated them scratch.



public
USER
USER_ID uuid
USER_FNAME character varying(50)
USER_LNAME character varying(50)
USER_EMAIL character varying(254)
USER_PASSWORD character varying(1000)

public
VFLASHCARD_SET
VFC_SET_ID uuid
USER_ID uuid
VFC_SET_TITLE character varying(250)

public
VFLASHCARD
VFC_ID uuid
VFC_SET_ID uuid
VFC_QUESTION character varying(300)
VFC_ANSWER character varying(100)

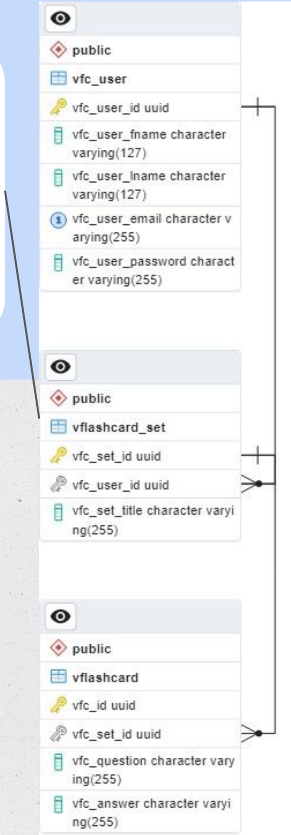
The database schema from inception, which went through iterations to arrive at the current one as seen on the next page. Also, generated scripts may not result in expected result. In the above, tables needed to be double quoted for SQL query.

Behind the Scenes Backend

Database ERD

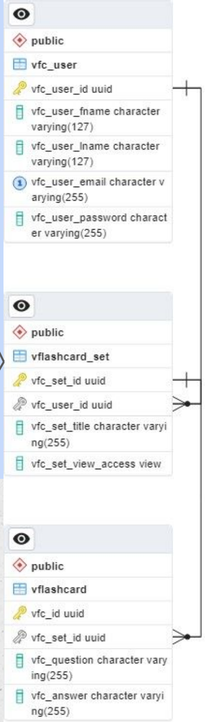
Updated the primary key to using a single attribute instead of two.

Done as a workaround for failed attempts to create the third table, vflashcard.


public
vfc_user
vfc_user_id uuid
vfc_user_fname character varying(127)
vfc_user_lname character varying(127)
vfc_user_email character varying(255)
vfc_user_password character varying(255)
public
vflashcard_set
vfc_set_id uuid
vfc_user_id uuid
vfc_set_title character varying(255)
public
vflashcard
vfc_id uuid
vfc_set_id uuid
vfc_question character varying(255)
vfc_answer character varying(255)

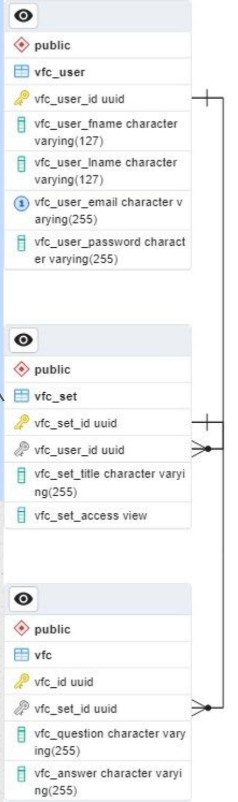
Added additional attribute under vflashcard_set for view access.

Implemented an ENUM type (public/private) as suggested by Edgardo Molina.


public
vfc_user
vfc_user_id uuid
vfc_user_fname character varying(127)
vfc_user_lname character varying(127)
vfc_user_email character varying(255)
vfc_user_password character varying(255)
public
vflashcard_set
vfc_set_id uuid
vfc_user_id uuid
vfc_set_title character varying(255)
vfc_set_view_access view
public
vflashcard
vfc_id uuid
vfc_set_id uuid
vfc_question character varying(255)
vfc_answer character varying(255)

Shorten table names for convenience.

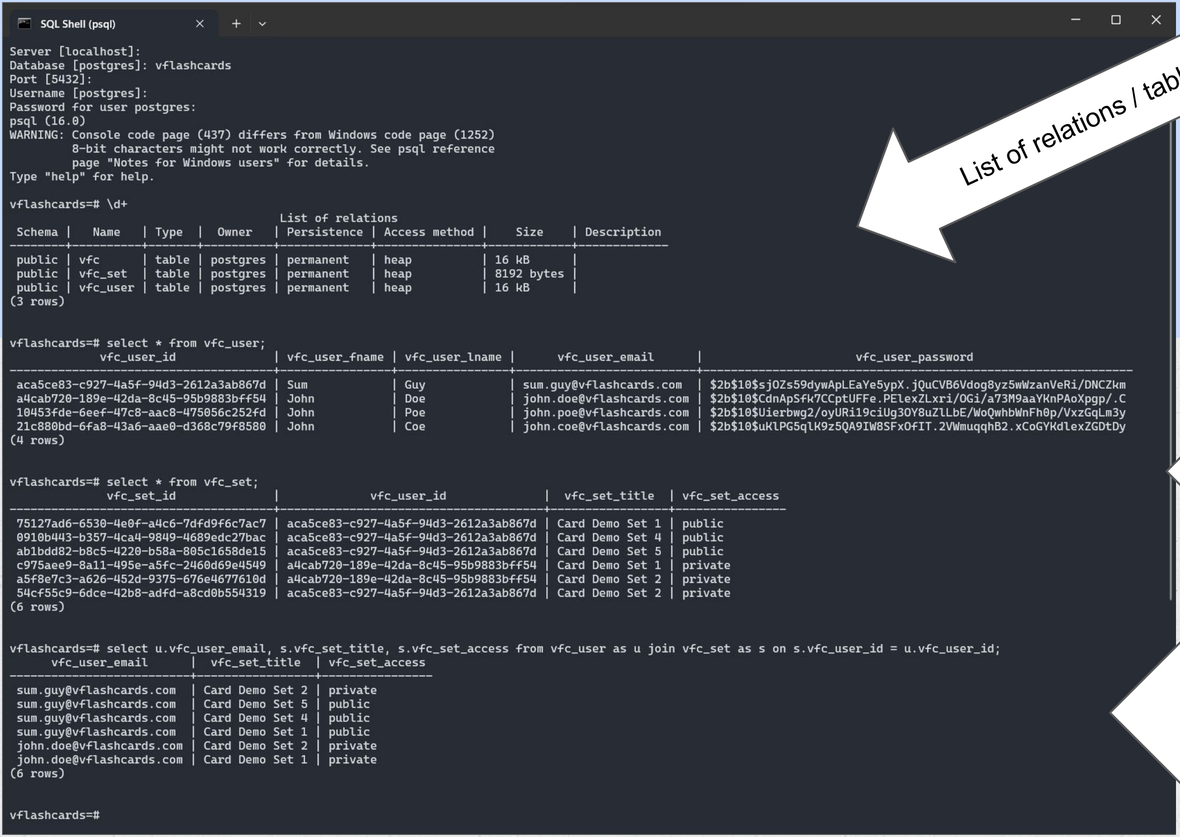
(Less typing during coding & debugging processes.)


public
vfc_user
vfc_user_id uuid
vfc_user_fname character varying(127)
vfc_user_lname character varying(127)
vfc_user_email character varying(255)
vfc_user_password character varying(255)
public
vfc_set
vfc_set_id uuid
vfc_user_id uuid
vfc_set_title character varying(255)
vfc_set_access view
public
vfc
vfc_id uuid
vfc_set_id uuid
vfc_question character varying(255)
vfc_answer character varying(255)

Note: ER diagrams were created using MySQL WorkBench (first two ERD) and pgAdmin from PostgreSQL. Both are available from the official website at no cost as of 2023-11.

Behind the Scenes Backend

PostgreSQL Database



```
Server [localhost]:
Database [postgres]: vflashcards
Port [5432]:
Username [postgres]:
Password for user postgres:
psql (16.0)
WARNING: Console code page (437) differs from Windows code page (1252)
         8-bit characters might not work correctly. See psql reference
         page "Notes for Windows users" for details.
Type "help" for help.

vflashcards=# \d+
                                      List of relations
 Schema |   Name   | Type  |  Owner   | Persistence | Access method |    Size    | Description
--------+----------+-------+----------+-------------+---------------+------------+-------------
 public | vfc      | table | postgres | permanent   | heap          | 16 kB      |
 public | vfc_set  | table | postgres | permanent   | heap          | 8192 bytes |
 public | vfc_user | table | postgres | permanent   | heap          | 16 kB      |
(3 rows)


vflashcards=# select * from vfc_user;
             vfc_user_id              | vfc_user_fname | vfc_user_lname |      vfc_user_email      |                       vfc_user_password
--------------------------------------+----------------+----------------+--------------------------+--------------------------------------------------------------
 aca5ce83-c927-4a5f-94d3-2612a3ab867d | Sum            | Guy            | sum.guy@vflashcards.com  | $2b$10$sjOZs59dywApLEaYe5ypX.jQuCVB6Vdog8yz5wWzanVeRi/DNCZkm
 a4cab720-189e-42da-8c45-95b9883bff54 | John           | Doe            | john.doe@vflashcards.com | $2b$10$CdnApSfk7CCptUFFe.PElexZLxri/OGi/a73M9aaYKnPAoXpgp/.C
 10453fde-6eef-47c8-aac8-475056c252fd | John           | Poe            | john.poe@vflashcards.com | $2b$10$Uierbwg2/oyURi19ciUg3OY8uZlLbE/WoQwhbWnFh0p/VxzGqLm3y
 21c880bd-6fa8-43a6-aae0-d368c79f8580 | John           | Coe            | john.coe@vflashcards.com | $2b$10$uKlPG5qlK9z5QA9IW8SFxOfIT.2VWmuqqhB2.xCoGYKdlexZGDtDy
(4 rows)


vflashcards=# select * from vfc_set;
              vfc_set_id              |             vfc_user_id              |  vfc_set_title  | vfc_set_access
--------------------------------------+--------------------------------------+-----------------+----------------
 75127ad6-6530-4e0f-a4c6-7dfd9f6c7ac7 | aca5ce83-c927-4a5f-94d3-2612a3ab867d | Card Demo Set 1 | public
 0910b443-b357-4ca4-9849-4689edc27bac | aca5ce83-c927-4a5f-94d3-2612a3ab867d | Card Demo Set 4 | public
 ab1bdd82-b8c5-4220-b58a-805c1658de15 | aca5ce83-c927-4a5f-94d3-2612a3ab867d | Card Demo Set 5 | public
 c975aee9-8a11-495e-a5fc-2460d69e4549 | a4cab720-189e-42da-8c45-95b9883bff54 | Card Demo Set 1 | private
 a5f8e7c3-a626-452d-9375-676e4677610d | a4cab720-189e-42da-8c45-95b9883bff54 | Card Demo Set 2 | private
 54cf55c9-6dce-42b8-adfd-a8cd0b554319 | aca5ce83-c927-4a5f-94d3-2612a3ab867d | Card Demo Set 2 | private
(6 rows)


vflashcards=# select u.vfc_user_email, s.vfc_set_title, s.vfc_set_access from vfc_user as u join vfc_set as s on s.vfc_user_id = u.vfc_user_id;
      vfc_user_email      |  vfc_set_title  | vfc_set_access
--------------------------+-----------------+----------------
 sum.guy@vflashcards.com  | Card Demo Set 2 | private
 sum.guy@vflashcards.com  | Card Demo Set 5 | public
 sum.guy@vflashcards.com  | Card Demo Set 4 | public
 sum.guy@vflashcards.com  | Card Demo Set 1 | public
 john.doe@vflashcards.com | Card Demo Set 2 | private
 john.doe@vflashcards.com | Card Demo Set 1 | private
(6 rows)


vflashcards=#
```

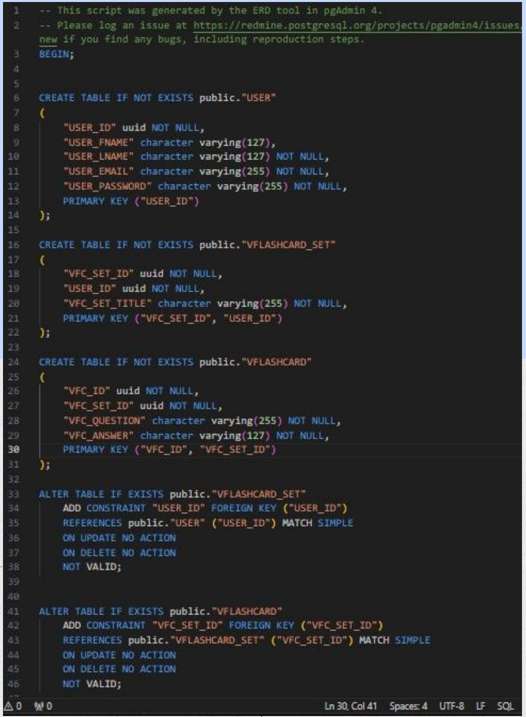
List of relations / tables

Querying user list & card sets.

Joining tables to get users who have card sets, which list email, associated card sets, and the access type.

Behind the Scenes Backend

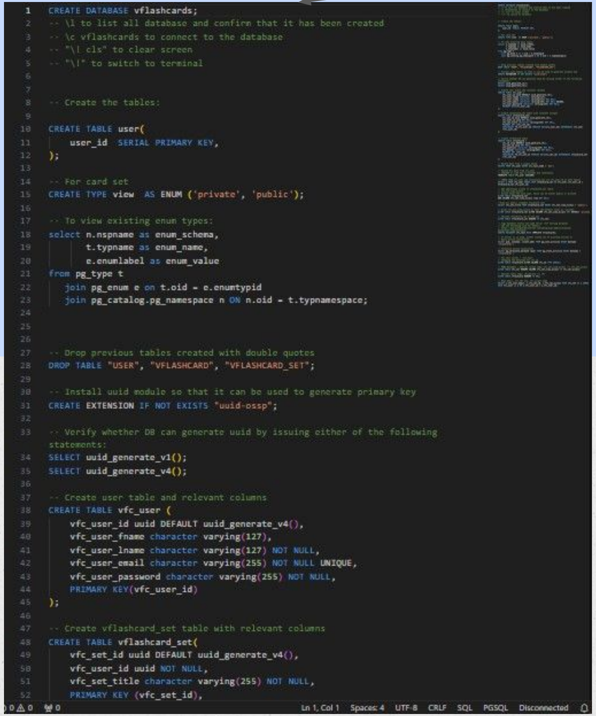
SQL Query Journal



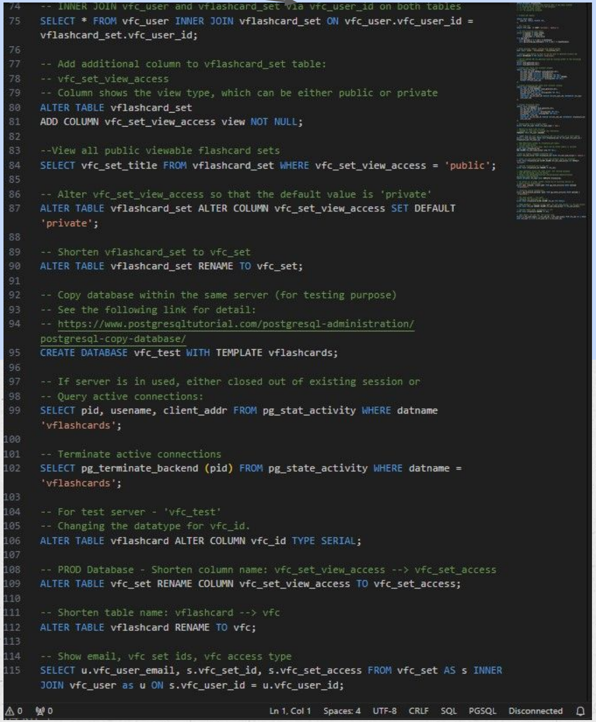
```sql
-- This script was generated by the ERD tool in pgAdmin 4.
-- Please log an issue at https://redmine.postgresql.org/projects/pgadmin4/issues/
new if you find any bugs, including reproduction steps.
BEGIN;


CREATE TABLE IF NOT EXISTS public."USER"
(
    "USER_ID" uuid NOT NULL,
    "USER_FNAME" character varying(127),
    "USER_LNAME" character varying(127) NOT NULL,
    "USER_EMAIL" character varying(255) NOT NULL,
    "USER_PASSWORD" character varying(255) NOT NULL,
    PRIMARY KEY ("USER_ID")
);

CREATE TABLE IF NOT EXISTS public."VFLASHCARD_SET"
(
    "VFC_SET_ID" uuid NOT NULL,
    "USER_ID" uuid NOT NULL,
    "VFC_SET_TITLE" character varying(255) NOT NULL,
    PRIMARY KEY ("VFC_SET_ID", "USER_ID")
);

CREATE TABLE IF NOT EXISTS public."VFLASHCARD"
(
    "VFC_ID" uuid NOT NULL,
    "VFC_SET_ID" uuid NOT NULL,
    "VFC_QUESTION" character varying(255) NOT NULL,
    "VFC_ANSWER" character varying(127) NOT NULL,
    PRIMARY KEY ("VFC_ID", "VFC_SET_ID")
);

ALTER TABLE IF EXISTS public."VFLASHCARD_SET"
    ADD CONSTRAINT "USER_ID" FOREIGN KEY ("USER_ID")
    REFERENCES public."USER" ("USER_ID") MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;


ALTER TABLE IF EXISTS public."VFLASHCARD"
    ADD CONSTRAINT "VFC_SET_ID" FOREIGN KEY ("VFC_SET_ID")
    REFERENCES public."VFLASHCARD_SET" ("VFC_SET_ID") MATCH SIMPLE
    ON UPDATE NO ACTION
    ON DELETE NO ACTION
    NOT VALID;
```

Generated script by pgAdmin. Did not produce expected result.

List of queries used up to now.



```sql
CREATE DATABASE vflashcards;
-- \l to list all database and confirm that it has been created
-- \c vflashcards to connect to the database
-- "\! cls" to clear screen
-- "\!" to switch to terminal


-- Create the tables:

CREATE TABLE user(
    user_id  SERIAL PRIMARY KEY,
);

-- For card set
CREATE TYPE view  AS ENUM ('private', 'public');

-- To view existing enum types:
select n.nspname as enum_schema,
       t.typname as enum_name,
       e.enumlabel as enum_value
from pg_type t
   join pg_enum e on t.oid = e.enumtypid
   join pg_catalog.pg_namespace n ON n.oid = t.typnamespace;



-- Drop previous tables created with double quotes
DROP TABLE "USER", "VFLASHCARD", "VFLASHCARD_SET";

-- Install uuid module so that it can be used to generate primary key
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";

-- Verify whether DB can generate uuid by issuing either of the following
statements:
SELECT uuid_generate_v1();
SELECT uuid_generate_v4();

-- Create user table and relevant columns
CREATE TABLE vfc_user (
    vfc_user_id uuid DEFAULT uuid_generate_v4(),
    vfc_user_fname character varying(127),
    vfc_user_lname character varying(127) NOT NULL,
    vfc_user_email character varying(255) NOT NULL UNIQUE,
    vfc_user_password character varying(255) NOT NULL,
    PRIMARY KEY(vfc_user_id)
);

-- Create vflashcard_set table with relevant columns
CREATE TABLE vflashcard_set(
    vfc_set_id uuid DEFAULT uuid_generate_v4(),
    vfc_user_id uuid NOT NULL,
    vfc_set_title character varying(255) NOT NULL,
    PRIMARY KEY (vfc_set_id),
```



```sql
-- INNER JOIN vfc_user and vflashcard_set via vfc_user_id on both tables
SELECT * FROM vfc_user INNER JOIN vflashcard_set ON vfc_user.vfc_user_id =
vflashcard_set.vfc_user_id;

-- Add additional column to vflashcard_set table:
-- vfc_set_view_access
-- Column shows the view type, which can be either public or private
ALTER TABLE vflashcard_set
ADD COLUMN vfc_set_view_access view NOT NULL;

--View all public viewable flashcard sets
SELECT vfc_set_title FROM vflashcard_set WHERE vfc_set_view_access = 'public';

-- Alter vfc_set_view_access so that the default value is 'private'
ALTER TABLE vflashcard_set ALTER COLUMN vfc_set_view_access SET DEFAULT
'private';

-- Shorten vflashcard_set to vfc_set
ALTER TABLE vflashcard_set RENAME TO vfc_set;

-- Copy database within the same server (for testing purpose)
-- See the following link for detail:
-- https://www.postgresqltutorial.com/postgresql-administration/
postgresql-copy-database/
CREATE DATABASE vfc_test WITH TEMPLATE vflashcards;

-- If server is in used, either closed out of existing session or
-- Query active connections:
SELECT pid, usename, client_addr FROM pg_stat_activity WHERE datname
'vflashcards';

-- Terminate active connections
SELECT pg_terminate_backend (pid) FROM pg_state_activity WHERE datname =
'vflashcards';

-- For test server - 'vfc_test'
-- Changing the datatype for vfc_id.
ALTER TABLE vflashcard ALTER COLUMN vfc_id TYPE SERIAL;

-- PROD Database - Shorten column name: vfc_set_view_access --> vfc_set_access
ALTER TABLE vfc_set RENAME COLUMN vfc_set_view_access TO vfc_set_access;

-- Shorten table name: vflashcard --> vfc
ALTER TABLE vflashcard RENAME TO vfc;

-- Show email, vfc set ids, vfc access type
SELECT u.vfc_user_email, s.vfc_set_id, s.vfc_set_access FROM vfc_set AS s INNER
JOIN vfc_user as u ON s.vfc_user_id = u.vfc_user_id;
```

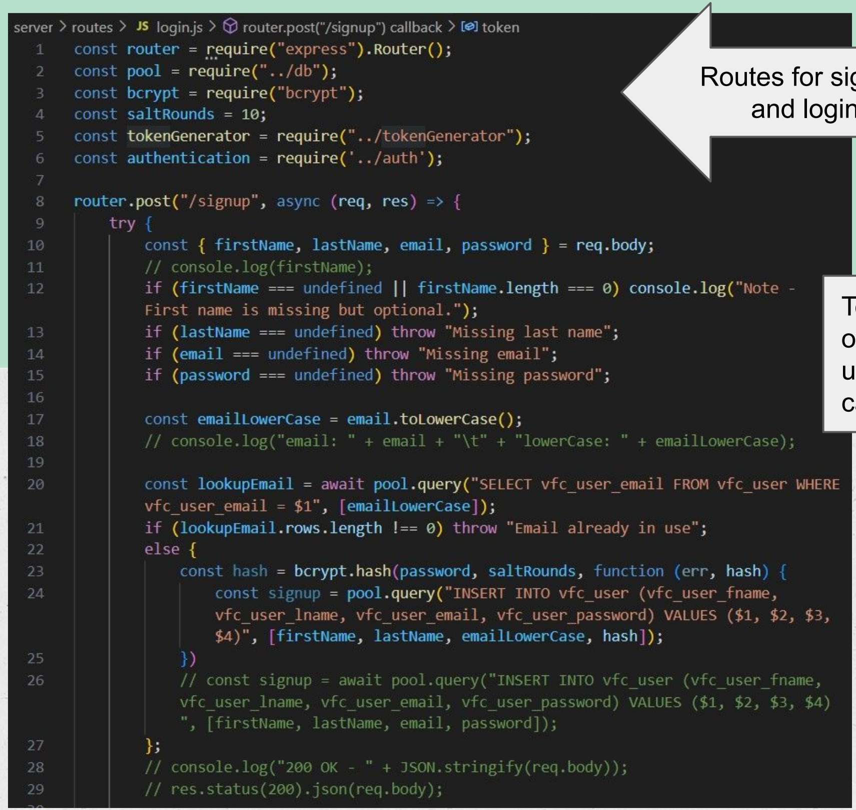
Note: Producing a journal takes time, but is helpful for record keeping. It also allows for quick references on previously used queries and can serve as a reminder as to why particular decisions took place.

Behind the Scenes Backend

Routes

Login & User Profile (Using Express & Node.js)

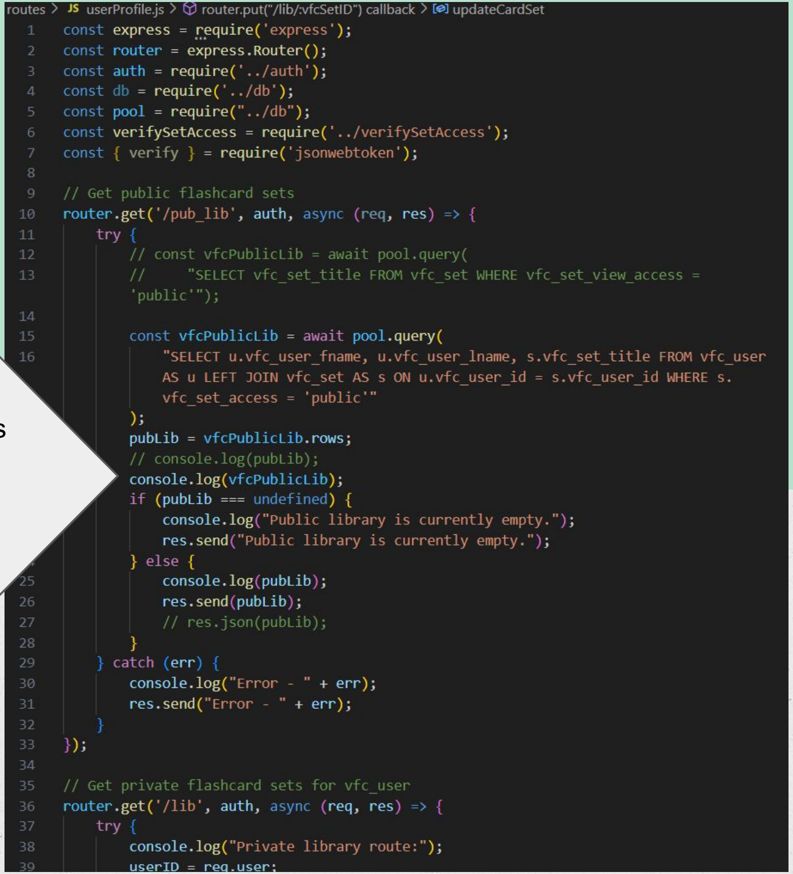
Routes for sign-up and login.



server > routes > JS login.js > router.post("/signup") callback > token

```javascript
const router = require("express").Router();
const pool = require("../db");
const bcrypt = require("bcrypt");
const saltRounds = 10;
const tokenGenerator = require("../tokenGenerator");
const authentication = require('../auth');

router.post("/signup", async (req, res) => {
    try {
        const { firstName, lastName, email, password } = req.body;
        // console.log(firstName);
        if (firstName === undefined || firstName.length === 0) console.log("Note - First name is missing but optional.");
        if (lastName === undefined) throw "Missing last name";
        if (email === undefined) throw "Missing email";
        if (password === undefined) throw "Missing password";

        const emailLowerCase = email.toLowerCase();
        // console.log("email: " + email + "\t" + "lowerCase: " + emailLowerCase);

        const lookupEmail = await pool.query("SELECT vfc_user_email FROM vfc_user WHERE vfc_user_email = $1", [emailLowerCase]);
        if (lookupEmail.rows.length !== 0) throw "Email already in use";
        else {
            const hash = bcrypt.hash(password, saltRounds, function (err, hash) {
                const signup = pool.query("INSERT INTO vfc_user (vfc_user_fname, vfc_user_lname, vfc_user_email, vfc_user_password) VALUES ($1, $2, $3, $4)", [firstName, lastName, emailLowerCase, hash]);
            })
            // const signup = await pool.query("INSERT INTO vfc_user (vfc_user_fname, vfc_user_lname, vfc_user_email, vfc_user_password) VALUES ($1, $2, $3, $4)", [firstName, lastName, email, password]);
        };
        // console.log("200 OK - " + JSON.stringify(req.body));
        // res.status(200).json(req.body);
```

To retrieve sets of flash cards, update/delete cards



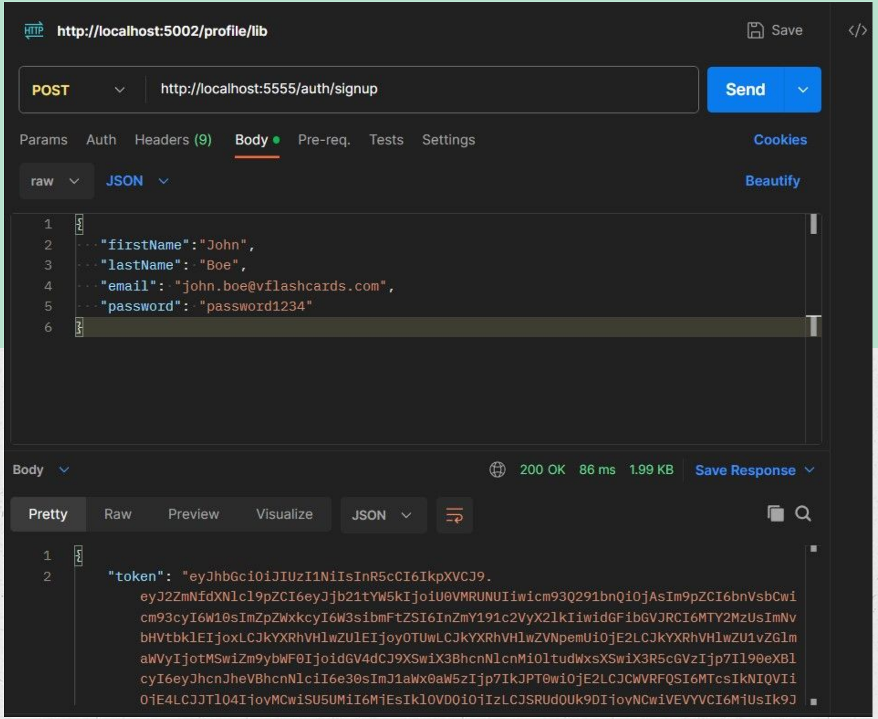
routes > JS userProfile.js > router.put("/lib/:vfcSetID") callback > updateCardSet

```javascript
const express = require('express');
const router = express.Router();
const auth = require('../auth');
const db = require('../db');
const pool = require("../db");
const verifySetAccess = require('../verifySetAccess');
const { verify } = require('jsonwebtoken');

// Get public flashcard sets
router.get('/pub_lib', auth, async (req, res) => {
    try {
        // const vfcPublicLib = await pool.query(
        //     "SELECT vfc_set_title FROM vfc_set WHERE vfc_set_view_access = 'public'");

        const vfcPublicLib = await pool.query(
            "SELECT u.vfc_user_fname, u.vfc_user_lname, s.vfc_set_title FROM vfc_user AS u LEFT JOIN vfc_set AS s ON u.vfc_user_id = s.vfc_user_id WHERE s.vfc_set_access = 'public'"
        );
        publib = vfcPublicLib.rows;
        // console.log(publib);
        console.log(vfcPublicLib);
        if (publib === undefined) {
            console.log("Public library is currently empty.");
            res.send("Public library is currently empty.");
        } else {
            console.log(publib);
            res.send(publib);
            // res.json(publib);
        }
    } catch (err) {
        console.log("Error - " + err);
        res.send("Error - " + err);
    }
});

// Get private flashcard sets for vfc_user
router.get('/lib', auth, async (req, res) => {
    try {
        console.log("Private library route:");
        userID = req.user;
```

Behind the Scenes Backend

Routes

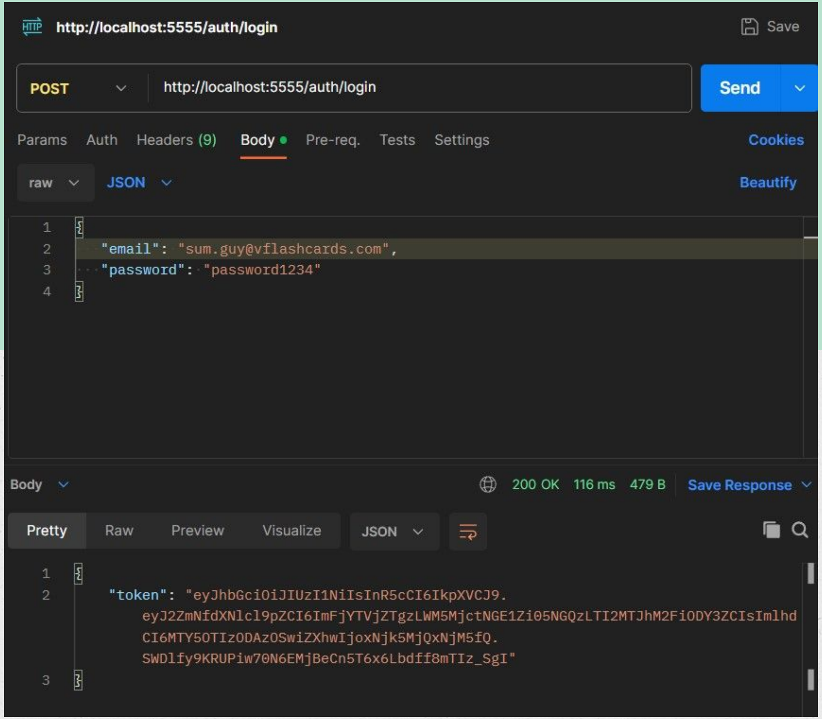
API Request & Response With Postman



```
{
    "firstName":"John",
    "lastName": "Boe",
    "email": "john.boe@vflashcards.com",
    "password": "password1234"
}
```

```
{
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJ2ZmNfdXNlcl9pZCI6eyJjb21tYW5kIjoiU0VMRUNUIiwicm93Q291bnQiOjAsIm9pZCI6bnVsbCwi
        cm93cyI6W10sImZpZWxkcyI6W3sibmFtZSI6InZmY191c2VyX2lkIiwidGFibGVJRCI6MTY2MzUsImNv
        bHVtbklEIjoxLCJkYXRhVHlwZUlEIjoyOTUwLCJkYXRhVHlwZVNpemUiOjE2LCJkYXRhVHlwZU1vZGlm
        aWVyIjotMSwiZm9ybWF0IjoidGV4dCJ9XSwiX3BhcnNlcnMiOltudWxsXSwiX3R5cGVzIjp7Il90eXBl
        cyI6eyJhcnJheVBhcnNlciI6e30sImJ1aWx0aW5zIjp7IkJPT0wiOjE2LCJCWVRFQSI6MTcsIkNIQVIi
        OjE4LCJJTlQ4IjoyMCwiSU5UMiI6MjEsIklOVDQiOjIzLCJSRUdQUk9DIjoyNCwiVEVYVCI6MjUsIk9J
```

Sign up process for John Boe.



```
{
    "email": "sum.guy@vflashcards.com",
    "password": "password1234"
}
```

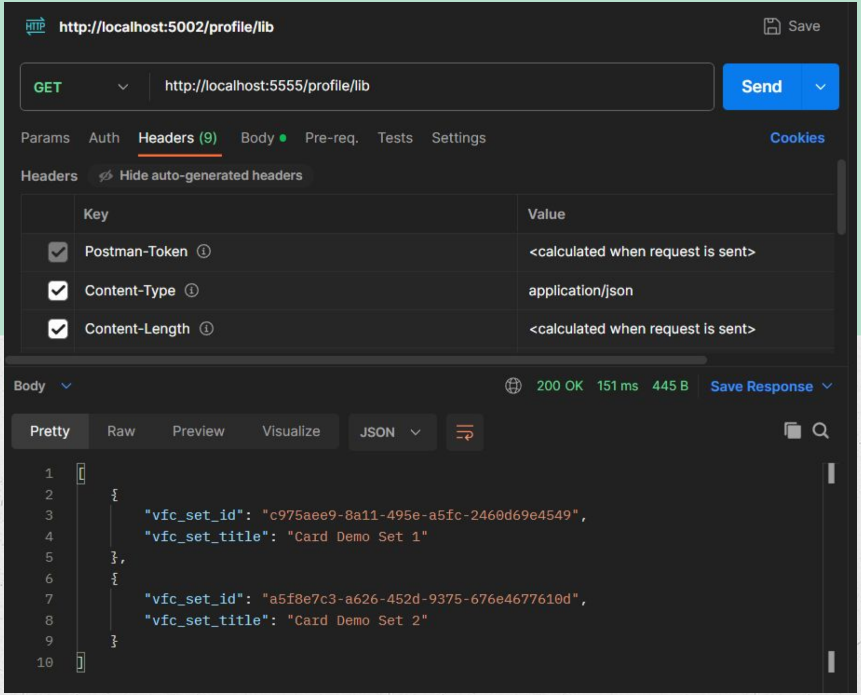
```
{
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.
        eyJ2ZmNfdXNlcl9pZCI6ImFjYTVjZTgzLWM5MjctNGE1Zi05NGQzLTI2MTJhM2FiODY3ZCIsImlhd
        CI6MTY5OTIzODAzOSwiZXhwIjoxNjk5MjQxNjM5fQ.
        SWDlfy9KRUPiw70N6EMjBeCn5T6x6Lbdff8mTIz_SgI"
}
```

Login authentication for Sum Guy.

Behind the Scenes Backend

Routes

API Request & Response With Postman

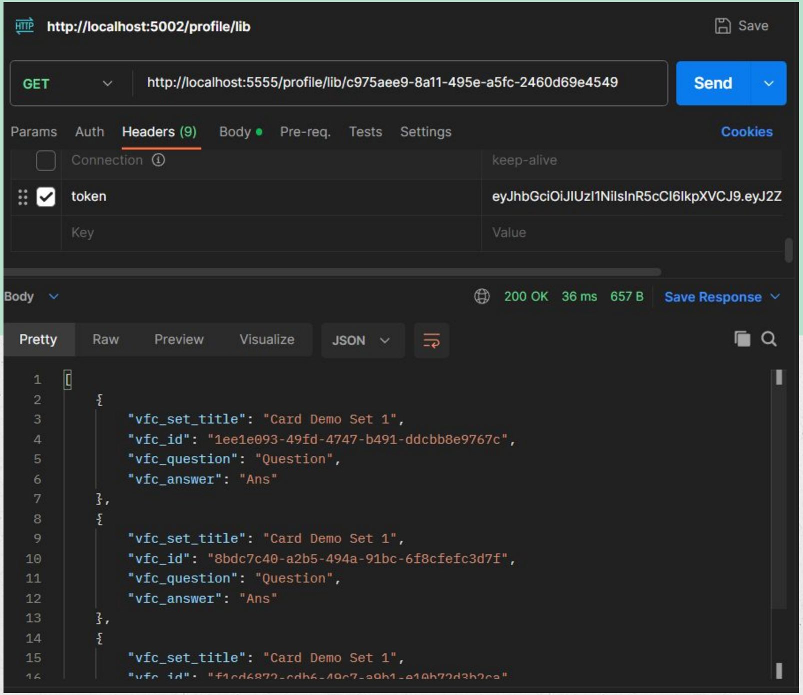


GET http://localhost:5555/profile/lib

	Key	Value
☑	Postman-Token	<calculated when request is sent>
☑	Content-Type	application/json
☑	Content-Length	<calculated when request is sent>

200 OK 151 ms 445 B

```
[
    {
        "vfc_set_id": "c975aee9-8a11-495e-a5fc-2460d69e4549",
        "vfc_set_title": "Card Demo Set 1"
    },
    {
        "vfc_set_id": "a5f8e7c3-a626-452d-9375-676e4677610d",
        "vfc_set_title": "Card Demo Set 2"
    }
]
```



GET http://localhost:5555/profile/lib/c975aee9-8a11-495e-a5fc-2460d69e4549

	Key	Value
□	Connection	keep-alive
☑	token	eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ2Z
	Key	Value

200 OK 36 ms 657 B

```
[
    {
        "vfc_set_title": "Card Demo Set 1",
        "vfc_id": "1ee1e093-49fd-4747-b491-ddcbb8e9767c",
        "vfc_question": "Question",
        "vfc_answer": "Ans"
    },
    {
        "vfc_set_title": "Card Demo Set 1",
        "vfc_id": "8bdc7c40-a2b5-494a-91bc-6f8cfefc3d7f",
        "vfc_question": "Question",
        "vfc_answer": "Ans"
    },
    {
        "vfc_set_title": "Card Demo Set 1",
        "vfc_id": "f1cd6872-cdb6-49c7-a9b1-e10b72d3b2ca"
```

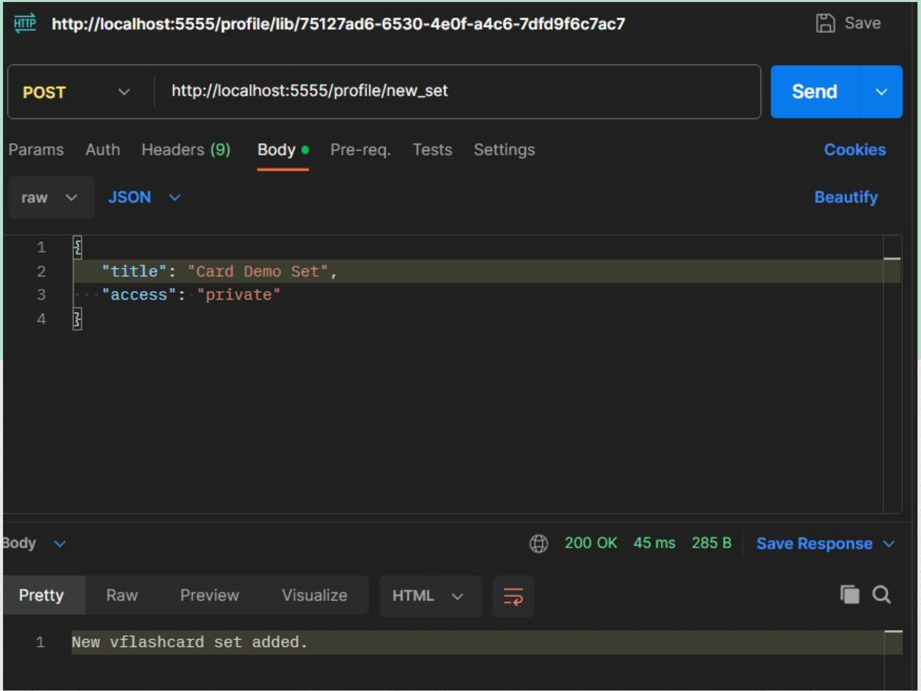
Getting the library set of cards.

Then a list of cards from a set.

Behind the Scenes Backend

Routes

API Request & Response With Postman



http://localhost:5555/profile/lib/75127ad6-6530-4e0f-a4c6-7dfd9f6c7ac7

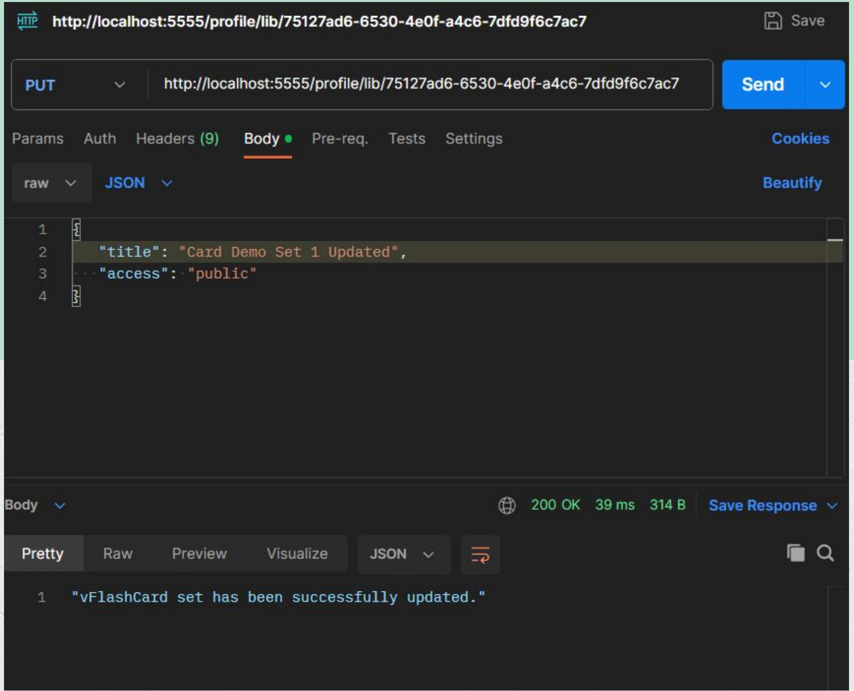
POST http://localhost:5555/profile/new_set

```
{
    "title": "Card Demo Set",
    "access": "private"
}
```

200 OK 45 ms 285 B

```
New vflashcard set added.
```

Create a new card set.



http://localhost:5555/profile/lib/75127ad6-6530-4e0f-a4c6-7dfd9f6c7ac7

PUT http://localhost:5555/profile/lib/75127ad6-6530-4e0f-a4c6-7dfd9f6c7ac7

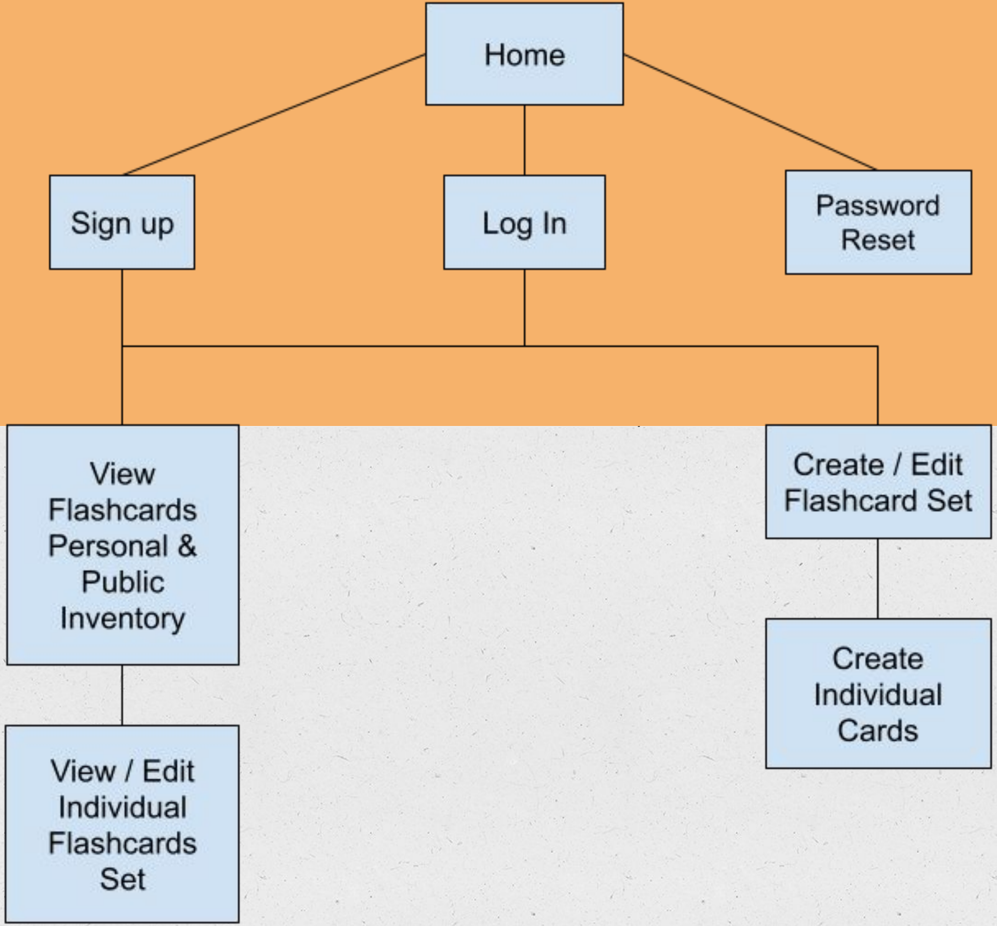
```
{
    "title": "Card Demo Set 1 Updated",
    "access": "public"
}
```

200 OK 39 ms 314 B

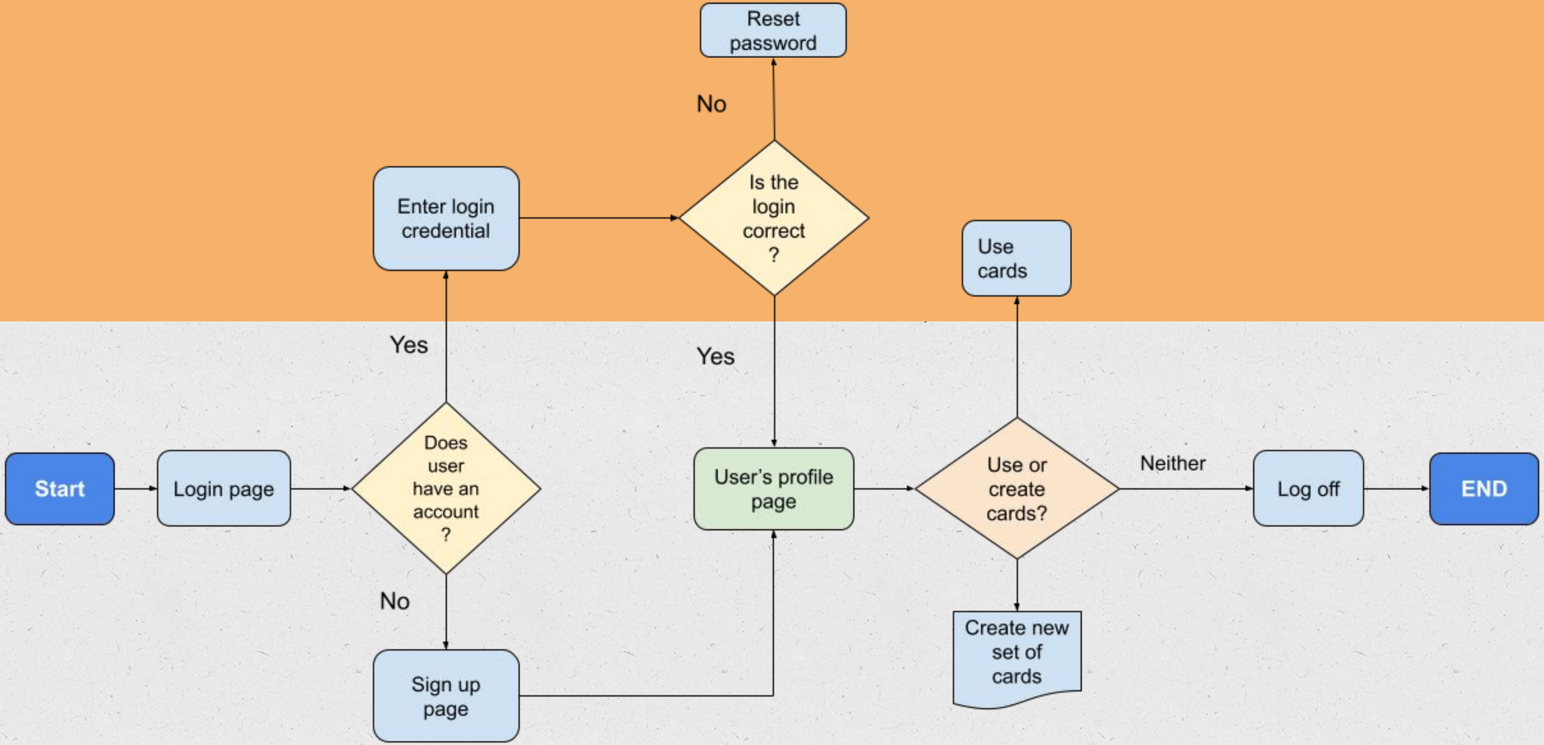
```
"vFlashCard set has been successfully updated."
```

Update title and access for a card set.

Site Map


Home
Sign up
Log In
Password Reset
View Flashcards Personal & Public Inventory
View / Edit Individual Flashcards Set
Create / Edit Flashcard Set
Create Individual Cards

User Flow Chart



Challenges

- Learning, testing, implementing components such as but not limited to
  - Bcrypt (for salt & hashing password)
  - Routes via Express
  - JSONWebToken
- Testing & debugging
  - JSONWebToken - Took hours to understand the mechanic before implementation and more time to debug.
- Database
  - Had to augment existing knowledge from the college database course. This includes server setup (PostgreSQL), database implementation, running different types of queries on a database, figuring out the appropriate data type, etc.
- Working solo
  - Responsible for all aspects from ideation, planning, designing, coding, debugging, documentation, and understanding all technical concepts.
  - Lack of sounding board, which can be useful when one wants additional perspective.
  - Tasks can take significantly more time than expected. One-man team means relying only by oneself.

Project Repository

GitHub Repo: https://github.com/not-x/vFlashCards

Time Log: https://github.com/not-x/vFlashCards/blob/main/Time_Log.csv

Project Management Board: https://github.com/users/not-x/projects/1