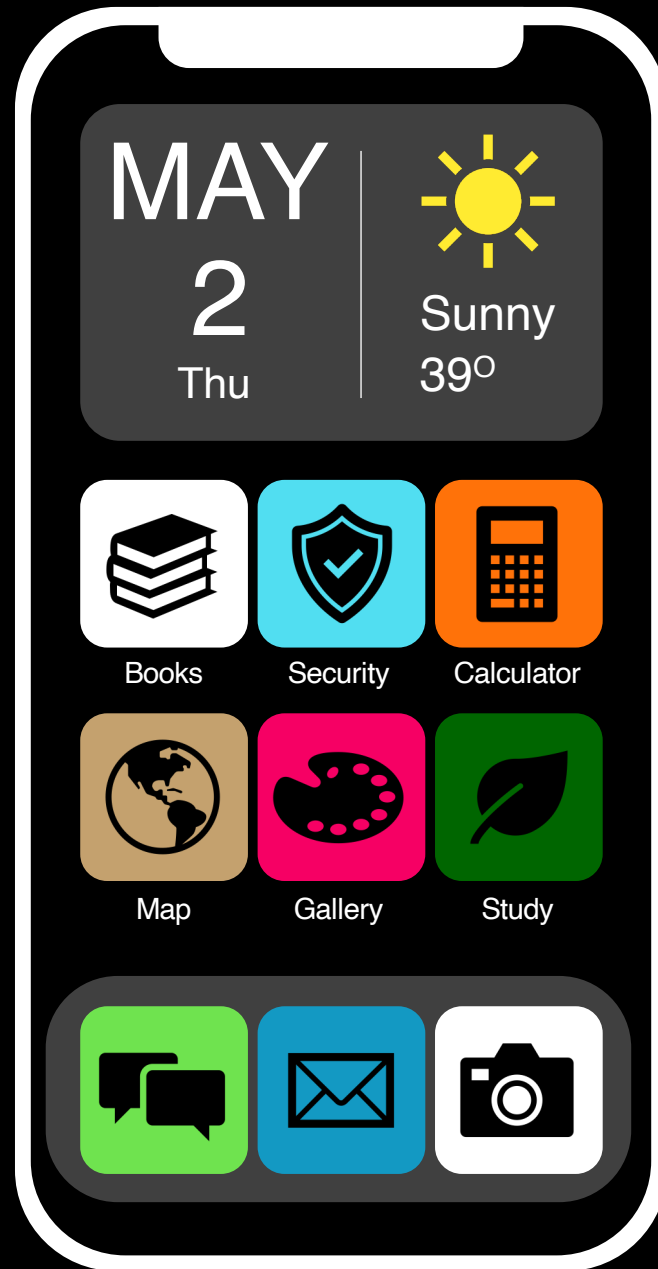


PASSWORD MANAGER



MAY

2

Thu



Sunny

39°



Books



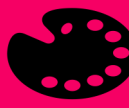
Security



Calculator



Map

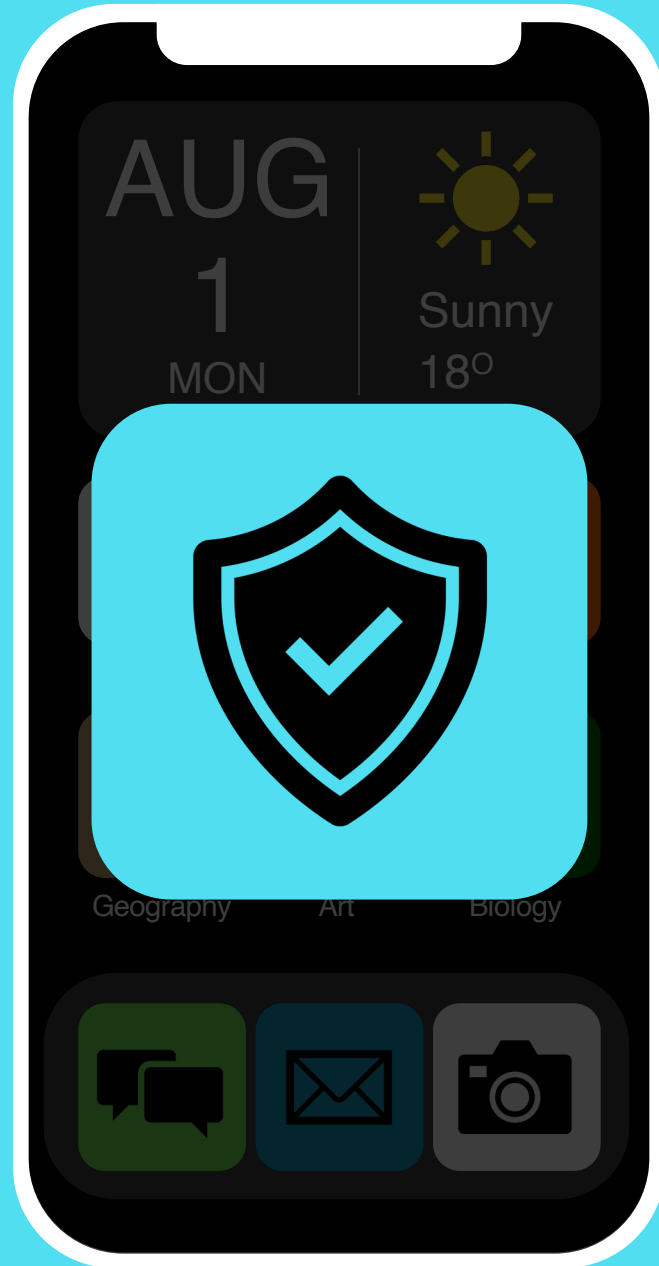


Gallery



Study







INTRODUCTION

The Password Manager System is a software solution designed to address the growing need for secure password management in today's digital world.

- User Registration and Authentication
- Secure storage of passwords
- Category based organization
- Security Questions
- User Friendly Interface



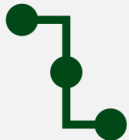
Project Implementation:

The Password Manager System will be implemented using a database management system (DBMS) to store and manage user data securely. The system will be developed using modern web development technologies, ensuring compatibility across various devices and platforms. Strong encryption algorithms will be employed to protect user passwords and sensitive information.



Entity

An entity is represented by a rectangle. It represents a real-world object or concept, such as a person, place, thing, or event. The entity's name is written inside the rectangle.



Attribute

An attribute is represented by an oval or ellipse connected to its respective entity. It represents a property or characteristic of an entity. The attribute's name is written inside the oval.



Primary Key

The primary key is an attribute (or combination of attributes) that uniquely identifies each record in an entity. It is underlined within the attribute oval.



Relationship

A relationship is represented by a diamond shape connecting two or more entities. It indicates how entities are related to each other. The type of relationship (such as one-to-one, one-to-many, or many-to-many) is often labeled near the diamond.



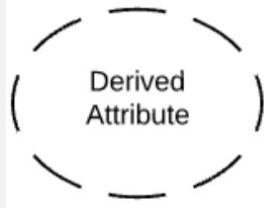
Cardinality

Cardinality describes the numerical relationship between entities in a relationship. It is represented using symbols near the relationship lines, such as "1" (for one), "M" (for many), or a range (e.g., "1..N").



Weak Entity

A weak entity is an entity that depends on another entity (called the owner entity) for its existence. It is represented by a double-bordered rectangle.



Derived Attribute

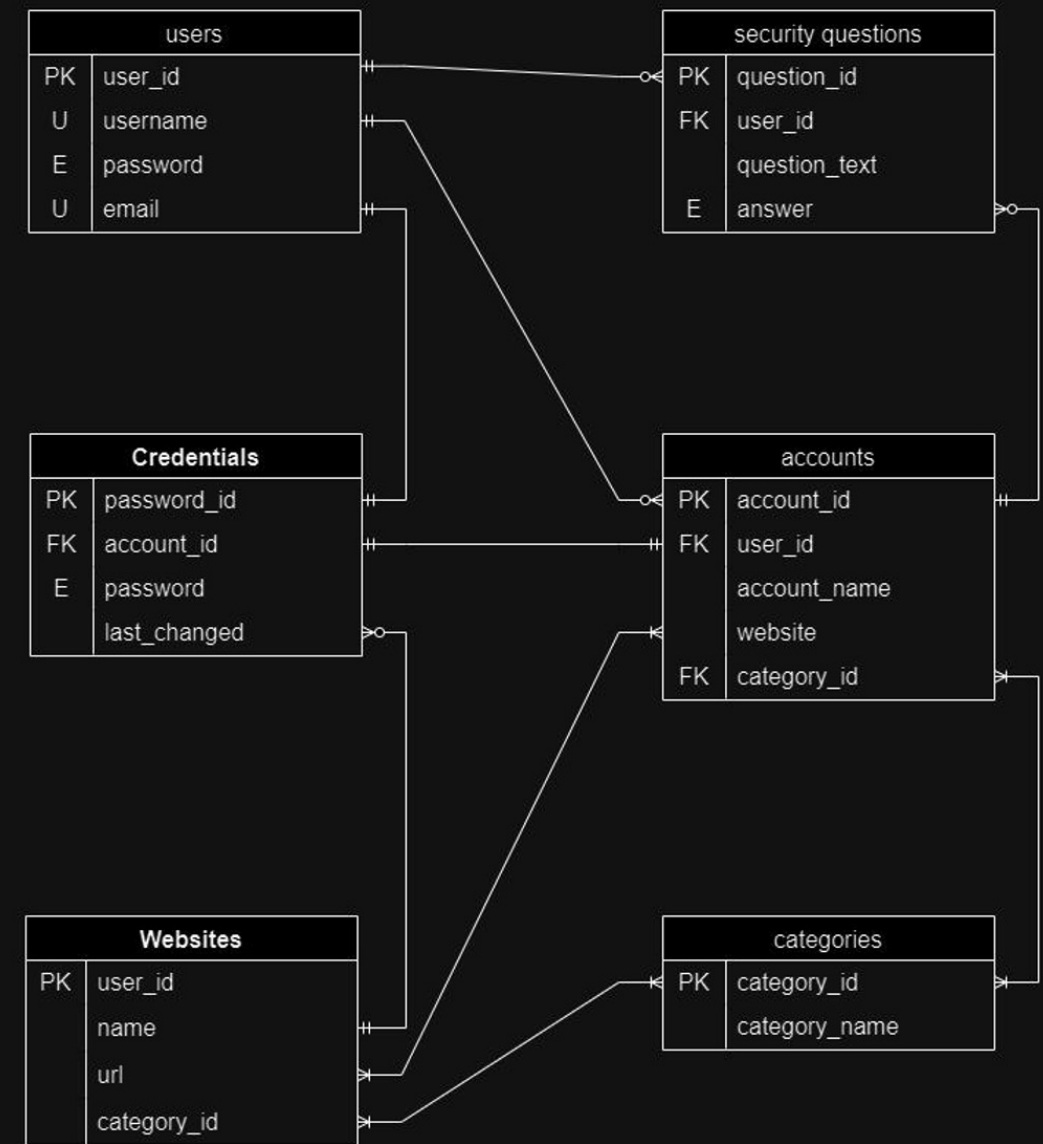
A derived attribute is an attribute whose value can be calculated from other attributes. It is represented by a dashed oval connected to its derived-from attributes.



Recursive Relationship

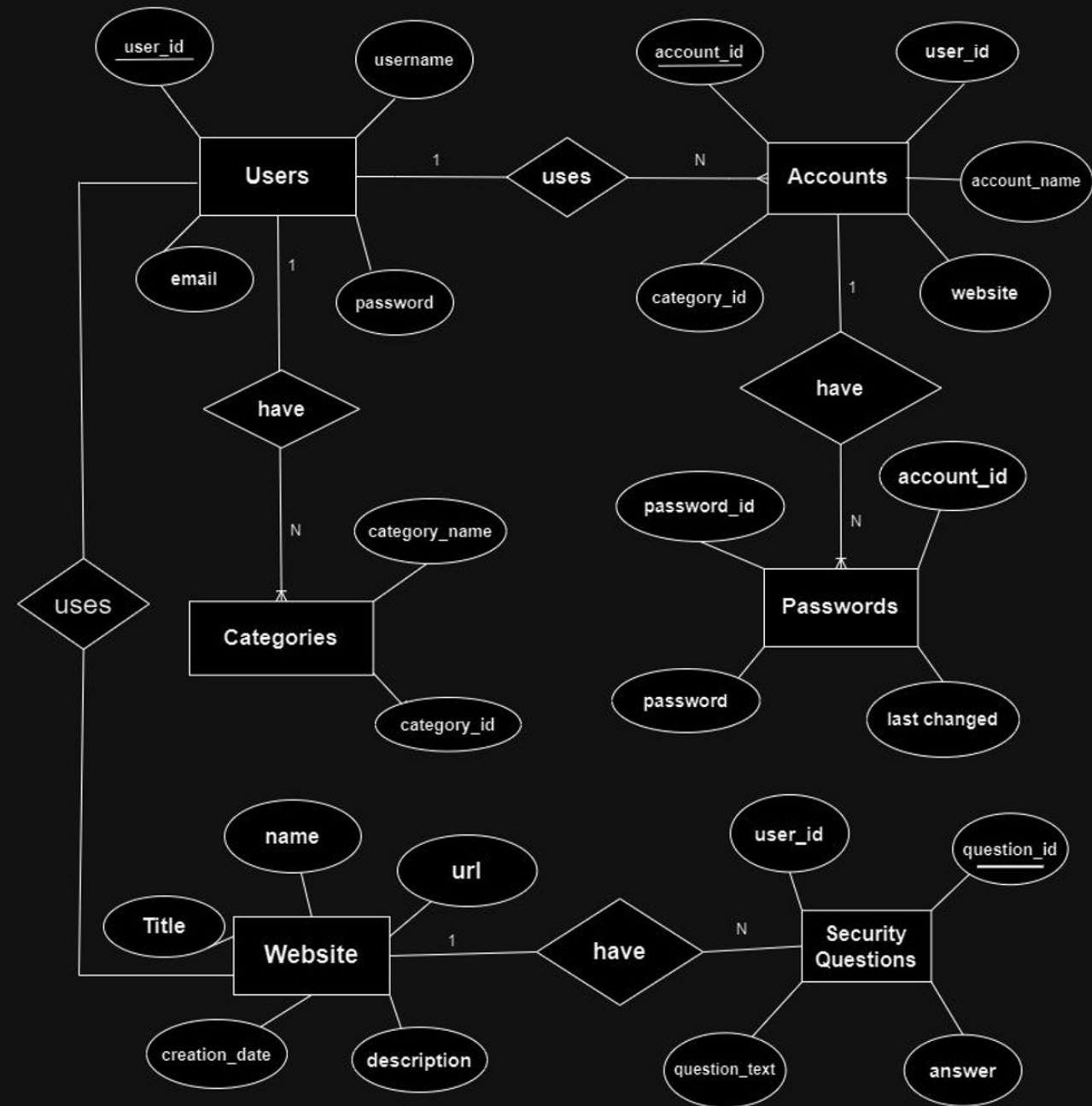
A recursive relationship occurs when an entity is related to itself. It is represented similarly to other relationships but with the same entity on both ends of the relationship line.

Schemas Visual Diagram



PK - Primary Key
FK - Foreign Key
E - Encrypted

Entity Relationship Visual Diagram



Conversion of ER Diagrams into Tables

User Table:

Column Name	Data Type	Description
user_id	INT	Primary Key
username	VARCHAR 50)	Unique
password	VARCHAR(50)	Encrypted
email	VARCHAR(100)	Unique

Password Table:

Column Name	Data Type	Description
password_id	INT	Primary Key
account_id	INT	Foreign Key (Accounts)
password	VARCHAR(50)	Encrypted
last_changed	DATE	

Categories Table:

Column Name	Data Type	Description
category_id	INT	Primary Key
category_name	VARCHAR(50)	

Security Questions Table:

Column Name	Data Type	Description
question_id	INT	Primary Key
user_id	INT	Foreign Key (Accounts)
question_text	VARCHAR(255)	
answer	VARCHAR(255)	Encrypted

Creation of Data in the tables

We will populate the table with sample data to demonstrate how the system works. This will include at least 5 records for each table, representing different users, accounts, passwords, categories, and security questions .

Users Table:

user_id	username	password	email
1	User 1	*****	user1@gmail.com
2	User 2	*****	user2@gmail.com

Accounts Table:

account_id	user_id	account_name	website	category_id
1	1	Gmail	www.gmail.com	1
2	1	Facebook	www.facebook.com	2
3	2	Linkedin	www.linkedin.com	2

Passwords Table:

password_id	account_id	password	last_changed
1	1	*****	01-04-2024
2	2	*****	28-03-2024
3	3	*****	25-03-2024

Categories Table:

category_id	category_name
1	Email
2	Social Media

Security Questions Table:

question_id	user_id	question_text	answer
1	1	What is your pet's name?	01-04-2024
2	1	What is your mother's maiden name?	28-03-2024
3	2	What is the name of your first school?	25-03-2024

Websites:

name	url	description	Title	Creation Date
google	www.google.com	google browser	GOOGLE	01-04-2024
facebook	www.facebook.com	social media platform	FACEBOOK	28-03-2024

SQL Queries

• Creating relations using SQL:

Users:

```
CREATE TABLE Users (  
    UserID INT AUTO INCREMENT PRIMARY KEY,  
    Email VARCHAR(255) UNIQUE NOT NULL,  
    Username VARCHAR(100) UNIQUE NOT NULL,  
    Password VARCHAR(255) NOT NULL  
);
```

Accounts:

```
CREATE TABLE Accounts (  
    AccountID INT PRIMARY KEY AUTO_INCREMENT,  
    UserID INT,  
    AccountName VARCHAR(255),  
    CategoryID INT,  
    Website VARCHAR(255),  
    FOREIGN KEY (UserID) REFERENCES Users(UserID),  
    FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID)  
);
```

Categories:

```
CREATE TABLE Categories (  
    category_id INT AUTO INCREMENT PRIMARY KEY,  
    category_name VARCHAR(255) NOT NULL  
);
```

Passwords:

```
CREATE TABLE Passwords (  
    PasswordID INT PRIMARY KEY AUTO_INCREMENT,  
    AccountID INT,  
    Password VARCHAR(255),  
    LastChanges TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDATE  
    CURRENT_TIMESTAMP  
);
```

Security Questions:

```
CREATE TABLE SecurityQuestions (  
    UserID INT,  
    QuestionID INT PRIMARY KEY,  
    QuestionText VARCHAR(255),  
    Answer VARCHAR(255)  
);
```

Websites:

```
CREATE TABLE Websites (  
    WebsiteID INT PRIMARY KEY AUTO_INCREMENT,  
    Title VARCHAR(255),  
    Name VARCHAR(255),  
    URL VARCHAR(255),  
    CreationDate DATE,  
    Description TEXT  
);
```

- **Storing data in relations using SQL:**

Users:

```
INSERT INTO Users (UserID, Email, Username, Password)
VALUES (1, 'example@example.com', 'example_username',
'example_password');
```

Accounts:

```
INSERT INTO Accounts (AccountID, UserID, AccountName,
CategoryID, Website)
VALUES (1, 101, 'Savings', 1, 'www.example.com');
```

Categories:

```
INSERT INTO Categories (CategoryName, CategoryID)
VALUES ('CategoryNameValue', CategoryIDValue);
```

Passwords:

```
INSERT INTO passwords (password_id, account_id, password,
last_changes)
VALUES (1, 1, 'example_password', '2024-05-01 12:00:00');
```

Security Questions:

```
INSERT INTO SecurityQuestions (UserID, QuestionID,
QuestionText, Answer)
VALUES
(1, 1, 'What is your mother''s maiden name?', 'Smith'),
(1, 2, 'What is the name of your first pet?', 'Fluffy'),
(2, 1, 'What city were you born in?', 'New York');
```

Websites:

```
INSERT INTO Websites (Title, Name, URL, CreationDate,
Description)
VALUES ('Example Website', 'Example',
'http://www.example.com', '2024-05-01', 'This is an example
website.');
```

1. Retrieve all accounts of a specific user

```
SELECT * FROM Accounts WHERE user_id = 1;
```

2. Find the account with strongest password

```
SELECT * FROM Passwords ORDER BY LENGTH(password) DESC LIMIT 1;
```

3. Update the password for a particular account

```
UPDATE Passwords SET password = 'new_password' WHERE account_id = 1;
```

4. Identify accounts with passwords expiring soon

```
SELECT * FROM Passwords WHERE DATEDIFF(CURDATE(), last_changed) > 90;
```

5. Counts the number of accounts in each category

```
SELECT c.category_name, COUNT(*) AS account_count  
FROM Accounts a  
JOIN Categories c ON a.category_id = c.category_id  
GROUP BY c.category_name;
```


Creation of 5 view using the tables

1. Display all accounts and their passwords

```
CREATE VIEW AccountPasswords AS
SELECT a.account_name, p.password
FROM Accounts a
JOIN Passwords p ON a.account_id = p.account_id;
```

2. Show users and their security questions

```
CREATE VIEW UserSecurityQuestions AS
SELECT u.username, sq.question_text
FROM Users u
JOIN SecurityQuestions sq ON u.user_id = sq.user_id;
```

3. List accounts sorted by password strength

```
CREATE VIEW StrongPasswords AS
SELECT a.account_name, p.password
FROM Accounts a
JOIN Passwords p ON a.account_id = p.account_id
ORDER BY LENGTH(p.password) DESC;
```

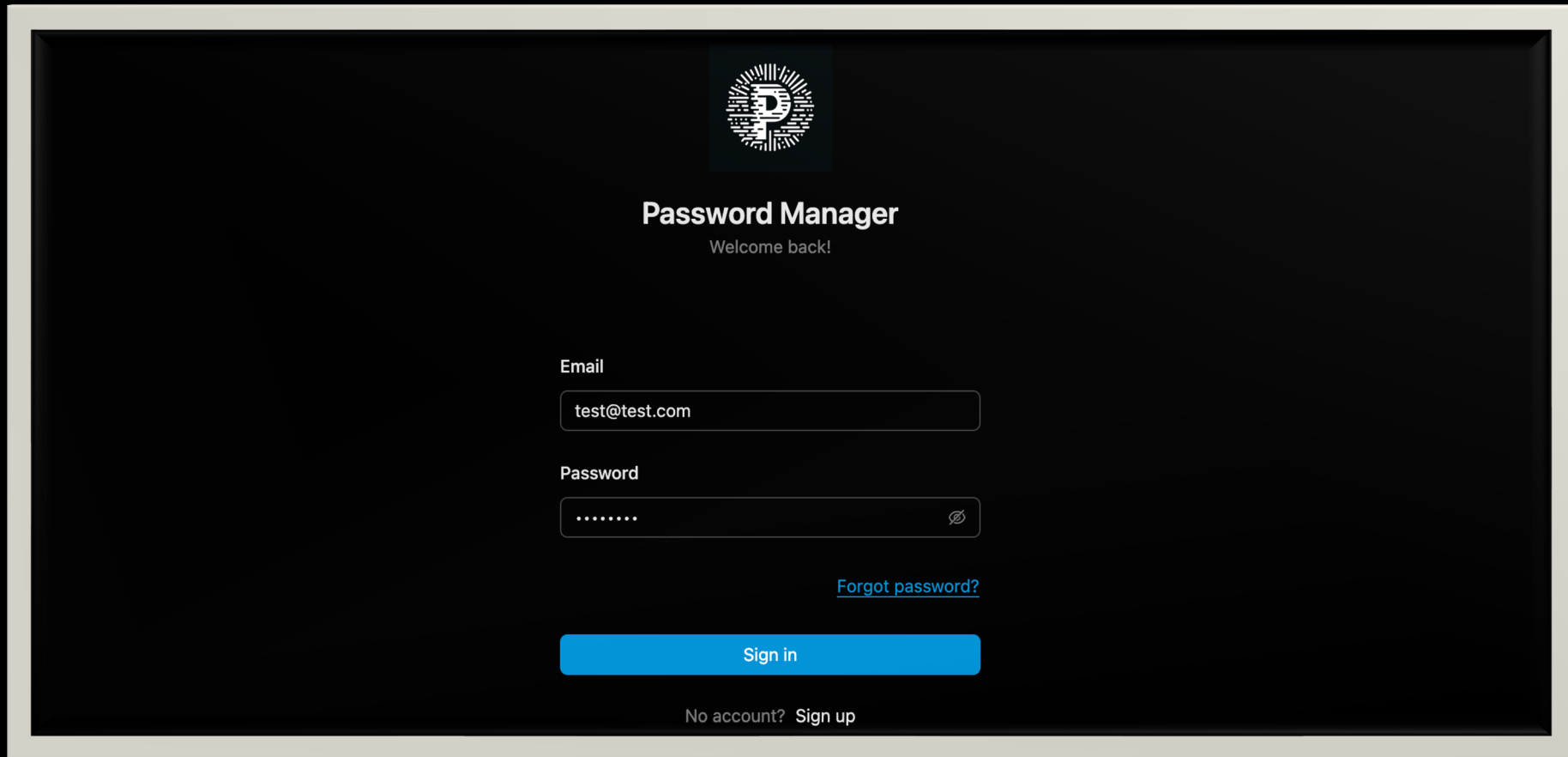
4. Display accounts expiring with a specified period


```
CREATE VIEW ExpiringPasswords AS
SELECT a.account_name, p.last_changed
FROM Accounts a
JOIN Passwords p ON a.account_id = p.account_id
WHERE DATEDIFF(CURDATE(), p.last_changed) > 90;
```

5. Show accounts grouped by category

```
CREATE VIEW AccountsByCategory AS
SELECT c.category_name, a.account_name
FROM Accounts a
JOIN Categories c ON a.category_id = c.category_id;
```

fully functional web app






Password Manager
Welcome back!

Email

Password



[Forgot password?](#)

[Sign in](#)

No account? [Sign up](#)



THANK YOU