CREATE TABLE users(user_id INTEGER PRIMARY KEY AUTOINCREMENT, username TEXT UNIQUE, first_name TEXT, last_name TEXT, email_address TEXT UNIQUE, password TEXT, salt TEXT);

CREATE TABLE password_history(user_id INTEGER, password TEXT, FOREIGN KEY (user_id) REFERENCES users(user_id));

I included all of a user's info in one table 'users', as these are all data points that are related to each other, and splitting them wound create redundancy. User_id serves as the primary key that is autoincremented with each user to make sure each user has a distinct id. Username must also be distinct for security and identification, hence the UNIQUE keyword. First_name and last_name store text information. Email_address is also UNIQUE so one email cannot be used to create multiple users. The password stores the hashed TEXT value, not the plaintext password, for security. Salt stores a value used to hash the password, as an extra layer of security.

The password history is stored in another table with the user_id as a foreign key referencing the user_id of users, to associate each password (current and past) with the user that created it. The password column again stores the hashed value of the password for security. There is some redundancy when storing a user's current password in both the users table and password_history table, but I chose to add the password to both for the sake of accurate documentation, and that it would be added when the password changes anyway.