# Assignment 6 6/5/2023

Implement the next release of your term project, employing a database manipulated from your application. If you want to use a database other than SQLite as described, please check with your facilitator. Otherwise, the same instructions as previously apply to this completed Word document, the gray text, the 5 page limit, appendices, JUnit tests, and a ReadMe file.

# 1 SUMMARY DESCRIPTION

*Evaluation criterion (i) applies*

One- or two-paragraph overall description of your proposed term project. Color red the parts changed from Assignment 4.

The Personal Health Information (PHI) app is a comprehensive solution designed to assist users in monitoring and managing their personal health. It allows users to create an account, securely log in, and track various health metrics such as BMI, blood pressure, and cholesterol levels. The app enables users to input and store their health data, providing a user-friendly interface for seamless data entry and updates. The app provides a user-friendly interface for adding and managing health data, making it easy for users to enter and update their information. It also offers the option to create custom health entries, where users can record specific health issues or concerns and add notes to track their progress.

With a strong emphasis on privacy and data security, the app ensures that each user has full control over their health information. Users can choose to keep their data private or share it with healthcare professionals or family members as needed. The app has the potential to not only prevent unwanted future health complications, but also act as a motivator to keep the health in track. Overall, the Health Tracker app provides a convenient and centralized platform for users to monitor and maintain their personal health records, empowering them to make informed decisions about their well-being.

### >>AI generation OR check: *I did not use AI generation here* \_X\_(Collapse section before submitting.)

PARAGRAPH DESCRIBING YOUR VALUE ADDED TO THE AI-GENERATED MATERIAL

MY PROMPT SEQUENCE (ONLY)

[1]

CODE OR TEXT PRODUCED BY THE ABOVE PROMPT SEQUENCE

chatGPT X

# 2 KEY REQUIREMENTS (FEATURES) IMPLEMENTED IN THIS RELEASE

*Evaluation criterion (i) applies*

Title and one or two sentences per requirement. Repeat requirements implemented for prior assignments if they are necessary to provide context. Make it clear which requirements are new vs. old.

## 2.1 Create User (old from week1)

The application shall allow users to register/login with their personal information, username, email, and password.

## 2.2 Adding/editing health information for individual users (old from week1)

The application shall allow users to input/edit their personal health information, including BMI, blood pressure, cholesterol levels, and any other custom health conditions.

### 2.3 Log in(old)

The app shall allow users to login using their email and password and remember the user

### >>AI generation OR check: *I did not use AI generation here* \_\_(Collapse section before submitting.)

PARAGRAPH DESCRIBING YOUR VALUE ADDED TO THE AI-GENERATED MATERIAL

The ai response was not accurate as I expected. However, it reminded me of the app’s capabilities that can be related in this week’s database incorporation to the app. The login, create user, adding, editing of the healthdata could now be stored in the database and it will take it’s full potential now. So I used the old requirements as I believe with the db, they will be fulfilled fully.

MY PROMPT SEQUENCE (ONLY)

[1]

ok for the database implementation iteration, I need to provide functional requirement for the app. Title and one or two sentences per requirement. Repeat requirements implemented for prior assignments if they are necessary to provide context. Make it clear which requirements are new vs. old. 2.1 Your title response replaces this (OLD / NEW) Your response replaces this 2.2 Your title response replaces this (OLD / NEW) Your response replaces this 2.3 Your title response replaces this (OLD / NEW) Your response replaces this 2.4 Your title response replaces this (OLD / NEW) It should be straightforward and don't mention implementation, making the requirements functional.

CODE OR TEXT PRODUCED BY THE ABOVE PROMPT SEQUENCE

chatGPT X

Sure! Here are some functional requirements for the app, with clear indications of whether they are new or carried over from previous assignments:

1. (NEW) User Registration: Users should be able to create an account by providing their first name, last name, email, password, date of birth, gender, and phone number.
2. (NEW) User Login: Users should be able to log in to their account using their email and password.
3. (NEW) User Profile Management: Users should be able to view and update their profile information, including first name, last name, email, date of birth, gender, and phone number.
4. (NEW) Health Data Entry: Users should be able to enter their health data, including blood pressure, cholesterol levels, blood glucose levels, BMI, and custom notes.
5. (NEW) Health Data Visualization: Users should be able to view graphical representations or charts of their health data, allowing them to track and monitor their progress over time.
6. (OLD) Health Data Editing: Users should be able to edit previously entered health data, allowing them to correct any mistakes or update the information.
7. (OLD) Health Data Deletion: Users should be able to delete unwanted or outdated health data entries from their records.
8. (NEW) Database Integration: The application should integrate SQLite database to securely store and retrieve user data and health data entries.
9. (NEW) Data Synchronization: The application should have a mechanism to synchronize user data and health data entries across multiple devices, ensuring consistent and up-to-date information.
10. (NEW) Data Backup and Restore: Users should have the option to backup their data to external storage or cloud services and restore it when needed, ensuring data integrity and availability.
11. (NEW) Data Privacy and Security: The application should implement proper measures to protect user data, including encryption, secure authentication, and adherence to data privacy regulations.
12. (NEW) Error Handling and Notifications: The application should provide meaningful error messages and notifications to users in case of data entry errors, database connectivity issues, or any other exceptions.

Please note that the numbering above is just for reference and to indicate whether the requirement is new or carried over from previous assignments. Feel free to modify or expand upon these requirements based on your specific project needs and goals.

# 3 I/O SUPPORTING THE NEW REQUIREMENTS LISTED ABOVE

*Evaluation criterion (ii) applies*

Provide an example of input / output showing the new features of your application.

Your response replaces this

### Input File(s)

File ….

### Input / Output

Console I/O:

### Output File(s)

File …

# 4 YOUR DIRECTORY

Show a screenshot of your directory. Include all relevant files. This should include JUnit tests.

Your response replaces this

# 5 DESIGN

*Evaluation criterion (i) applies*

### 5.1 Class Model, Use Case, and Sequence Diagram

## Supply a main use case, the class model, and the sequence diagram corresponding to the use case. These should be consistent. Indicate in red where you applied the features listed below.

Your response replaces this

### >>AI generation OR check: *I did not use AI generation here* \_\_(Collapse section before submitting.)

PARAGRAPH DESCRIBING YOUR VALUE ADDED TO THE AI-GENERATED MATERIAL

MY PROMPT SEQUENCE (ONLY)

[1]

CODE OR TEXT PRODUCED BY THE ABOVE PROMPT SEQUENCE

chatGPT X

# 6 CODE SNIPPETS

*Evaluation criterion (iii) applies*

### 1 Code of any kind showing table creation

.Your response replaces this

### 2 Java code involving the insertion of data

.Your response replaces this

### 3 Java code involving the selection of records, and ordering with SQL

Your response replaces this

### 4 Java code involving selection involving at least two tables

Your response replaces this

### 5 Java code involving selection and aggregation of data

Your response replaces this

### >>AI generation OR check: *I did not use AI generation here* \_\_(Collapse section before submitting.)

PARAGRAPH DESCRIBING YOUR VALUE ADDED TO THE AI-GENERATED MATERIAL

MY PROMPT SEQUENCE (ONLY)

[1] okay, let's start with the code for log in and user creation implementation using sqllite db.

[2] So I need to create tables for these right?

[3] So in the Main class, when a user hits create user, they input their values and submit. I want the db to store that info. Next time when the user hits login, they put in their username and password to login to the user. I want the db here so it can recognize the user.

CODE OR TEXT PRODUCED BY THE ABOVE PROMPT SEQUENCE

chatGPT X

[1] Set up the SQLite Database:

Download the SQLite JDBC driver (e.g., sqlite-jdbc-3.36.0.3.jar) and add it to your project's dependencies.

Create a SQLite database file (e.g., "healthtracker.db") to store user data and health data entries.

Establish a connection to the SQLite database using JDBC.

Create the User table:

Define a schema for the User table, including columns such as id, firstName, lastName, email, password, etc.

Execute a CREATE TABLE SQL statement to create the User table if it doesn't exist already.

Implement User Creation:

Modify the createCreateUserScene() method in the Main class to insert a new user into the SQLite database.

Retrieve the user input (e.g., first name, last name, email, password, etc.) from the UI components.

Generate a secure hash of the password (e.g., using bcrypt) before storing it in the database to enhance security.

Execute an INSERT INTO SQL statement to insert the user's data into the User table.

Implement Login:

Modify the createLoginScene() method in the Main class to perform the login functionality.

Retrieve the entered username and password from the UI components.

Execute a SELECT SQL statement to retrieve the user with the entered username from the User table.

Compare the entered password with the stored password hash (using bcrypt or a similar mechanism) to verify the login credentials.

If the login is successful, proceed to the next steps, such as showing the health data entry scene.

You can use session management techniques to keep the user logged in until they explicitly log out.

[2] import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

import java.sql.Statement;

public class DatabaseSetup {

public static void main(String[] args) {

// SQLite database connection

String url = "jdbc:sqlite:healthtracker.db";

try (Connection conn = DriverManager.getConnection(url);

Statement stmt = conn.createStatement()) {

// Create User table

String createUserTableSql = "CREATE TABLE IF NOT EXISTS User (" +

"id INTEGER PRIMARY KEY AUTOINCREMENT," +

"firstName TEXT NOT NULL," +

"lastName TEXT NOT NULL," +

"email TEXT NOT NULL UNIQUE," +

"password TEXT NOT NULL," +

"dateOfBirth TEXT NOT NULL," +

"gender TEXT NOT NULL," +

"phoneNumber TEXT" +

")";

stmt.execute(createUserTableSql);

// Create HealthData table

String createHealthDataTableSql = "CREATE TABLE IF NOT EXISTS HealthData (" +

"id INTEGER PRIMARY KEY AUTOINCREMENT," +

"userId INTEGER NOT NULL," +

"metric TEXT NOT NULL," +

"value REAL NOT NULL," +

"dateRecorded TEXT NOT NULL," +

"FOREIGN KEY (userId) REFERENCES User(id)" +

")";

stmt.execute(createHealthDataTableSql);

System.out.println("Tables created successfully.");

} catch (SQLException e) {

System.out.println("Error creating tables: " + e.getMessage());

}

}

}

[3] When the user hits the "Create User" button, you need to capture the input values (such as first name, last name, email, password, date of birth, gender, and phone number) from the user interface.

Once you have captured the input values, you can use an SQL INSERT statement to insert the user data into the User table in the database. Here's an example:

java

Copy code

try (Connection conn = DriverManager.getConnection(url);

Statement stmt = conn.createStatement()) {

String firstName = "John";

String lastName = "Doe";

String email = "johndoe@example.com";

String password = "password123";

String dateOfBirth = "1990-01-01";

String gender = "Male";

String phoneNumber = "1234567890";

String insertUserSql = "INSERT INTO User (firstName, lastName, email, password, dateOfBirth, gender, phoneNumber) " +

"VALUES ('" + firstName + "', '" + lastName + "', '" + email + "', '" + password + "', '" + dateOfBirth + "', '" + gender + "', '" + phoneNumber + "')";

stmt.executeUpdate(insertUserSql);

System.out.println("User created successfully.");

} catch (SQLException e) {

System.out.println("Error creating user: " + e.getMessage());

}

In this example, I have used the hardcoded values for simplicity. However, in your application, you would use the actual input values obtained from the user interface.

When the user hits the "Login" button and provides their username and password, you need to validate their credentials against the data stored in the User table.

You can use an SQL SELECT statement to query the User table based on the provided username (email) and password. If a matching record is found, it means the login credentials are valid. Here's an example:

java

Copy code

try (Connection conn = DriverManager.getConnection(url);

Statement stmt = conn.createStatement()) {

String email = "johndoe@example.com";

String password = "password123";

String selectUserSql = "SELECT \* FROM User WHERE email = '" + email + "' AND password = '" + password + "'";

ResultSet resultSet = stmt.executeQuery(selectUserSql);

if (resultSet.next()) {

// Login successful

int userId = resultSet.getInt("id");

String firstName = resultSet.getString("firstName");

// ... Retrieve other columns as needed

System.out.println("Login successful. User ID: " + userId);

System.out.println("Welcome, " + firstName + "!");

} else {

// Login failed

System.out.println("Invalid username or password. Please try again.");

}

} catch (SQLException e) {

System.out.println("Error querying user: " + e.getMessage());

}

In this example, I have used the hardcoded values for simplicity. In your application, you would replace the hardcoded values with the actual input values provided by the user.

These examples demonstrate the basic flow of creating a user and validating their login credentials against the database. Remember to properly handle exceptions and sanitize user inputs to prevent SQL injection attacks.

# 7 YOUR CODE

*Evaluation criterion (iii) applies*

Unless your facilitator arranges another method, copy your Eclipse project to your file system, zip it, and attach it to the Blackboard submission. Please contact your facilitator in advance if you want to request an exception.

Your response replaces this

### >>AI generation OR check: *I did not use AI generation here* \_\_(Collapse section before submitting.)

PARAGRAPH DESCRIBING YOUR VALUE ADDED TO THE AI-GENERATED MATERIAL

I honored the chatgpt response this time only because I agree with it on the fact that Hibernate will require more training and at this stage of the course near the final. So I stuck with the SQLite which I know would be simpler in the sense that I don’t have to learn many new terminologies, libraries, etc.

MY PROMPT SEQUENCE (ONLY)

[1] to implement the following requirements,

2.1 Create User (old from week1)

The application shall allow users to register/login with their personal information, username, email, and password.

2.2 Adding/editing health information for individual users (old from week1)

The application shall allow users to input/edit their personal health information, including BMI, blood pressure, cholesterol levels, and any other custom health conditions.

2.3 Log in(old)

The app shall allow users to login using their email and password and remember the user

Do you think sqllite will be easy to implement or less work than hibernate?

Based on the summary and requirements, how should I start?

CODE OR TEXT PRODUCED BY THE ABOVE PROMPT SEQUENCE

chatGPT X

SQLite is generally considered easier to implement and requires less setup compared to Hibernate. Hibernate is a full-fledged ORM framework that provides advanced features and abstractions for managing database operations. It requires configuring entity mappings, session management, and dealing with complex object-relational mapping concepts.

On the other hand, SQLite is a lightweight, embedded database engine that does not require a separate server installation. It stores the database as a file on the local disk, making it easier to set up and manage. SQLite provides a simple and efficient way to interact with the database using SQL queries.

Based on your project requirements, it seems that implementing SQLite would be a suitable choice. SQLite can handle the data storage needs of your app, including user registration, health information input, and login functionality. It provides a good balance between simplicity and functionality.

To start implementing SQLite in your app, you can follow these steps:

Choose a SQLite library: You'll need to select a SQLite library or wrapper that provides a convenient API for interacting with the SQLite database in your programming language. For Java, you can use libraries like SQLite JDBC or Room Persistence Library.

Set up the database: Create a SQLite database file that will store your application's data. You can define the necessary tables and schema for storing user information, health data, and any other relevant data.

Establish database connection: Initialize the SQLite library and establish a connection to the database file. This will allow your application to interact with the database.

Implement CRUD operations: Implement the necessary database operations for creating, reading, updating, and deleting data. This includes functionalities like user registration, login, and managing health information.

Integrate with your application: Modify your existing codebase to use the SQLite database instead of the previous data storage mechanism. Update the necessary parts of your code to interact with the SQLite database for user management and health data operations.

Test thoroughly: Write unit tests to ensure the correctness of your database operations. Test various scenarios, including valid and invalid inputs, to validate the behavior of your SQLite implementation.

By following these steps, you should be able to start implementing SQLite in your application and leverage its features for storing and managing user and health data. Remember to consult the documentation of the chosen SQLite library for specific implementation details and best practices.

# 9 Evaluation

