DOCUMENTATION FOR DRINKUP APPLICATION

NAME: JULIA DOBROVODSKA
STUDENT NUMBER: 3061278
LINK TO GITHUB: HTTPS://GITHUB.COM/DZIJULIA/ANDROID-PROJECT-3061278

Milestone 1

INTRODUCTION

DrinkUp is a hydration tracking application that allows users to monitor their water intake. The application consists of four main activities: Login, Profile, Current Hydration, and History.

USER INTERFACE DESIGN

1.Login Activity

The login activity is the entry point of the application. If the user is not in the database, they will be redirected to the register page to set up their login details after registration they will be redirect straight away to their profile page to set up their details and hydration goals. If user is found in the database it will redirect the user straight to Current Hydration page with details stored from database (current intake if any)

2.Profile Activity

The profile activity allows users to set up their profile and hydration goals. This information will be stored in the database for future reference.

3.Current Hydration

The current hydration activity displays the user's current hydration status for current day. It updates in real-time as the user logs their water intake.

4.History

The history activity provides a historical view of the user's hydration data. It allows users to track their progress over time. It will be using graphs, charts or table to display the data in an easy-to- understand way.

NAVIGATION AND LAYOUT

Navigation between activities is facilitated by buttons located at the bottom of the screen. A logout button is in the top right corner, which redirects users to the login page. The layout and positioning of these widgets have been designed to support intuitive user interaction.

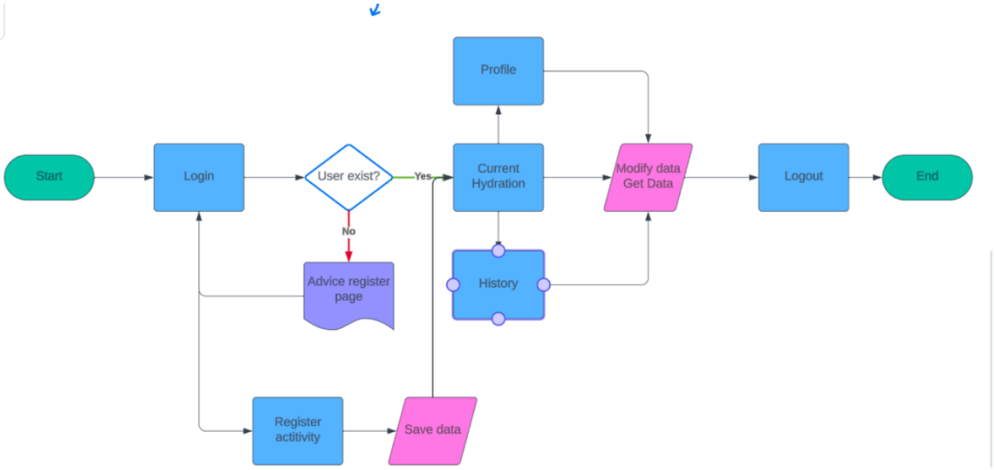
DATA SHARING BETWEEN ACTIVITIES

Data is shared between activities using intents and bundles. Here are three examples:

1. **From Login to Profile**: If a new user logs in, their data is not yet in the database. An intent is used to start the Profile activity or Register activity where they can enter their details.

2. **From Profile to Current Hydration**: Once a user has set up their profile, an intent is used to start the Current Hydration activity. The user's hydration goal is passed as an extra in the intent.

3. **From Current Hydration to History**: The user's current hydration data is passed to the History activity using an intent. This allows the History activity to display a complete record of the user's hydration data.



KOTLIN CODE DESCRIPTION

The application is written in Kotlin and makes use of custom composable for reusable UI elements. Here's an example of a custom composable function:

ISLANDSCAPE() METHOD DOCUMENTATION

```
@Composable
fun isLandscape(): Boolean {

return LocalConfiguration.current.orientation == Configuration.ORIENTATION_LANDSCAPE }
```

DESCRIPTION:

The isLandscape() method is a crucial component in the project's layout management. This method allows the application to dynamically adjust its layout based on the device's orientation. It checks the current device orientation and returns a Boolean value indicating whether the orientation is in the landscape mode.

USAGE:

To ensure optimal user experience, the **isLandscape()** method can be utilized in various aspects of the application's layout management. Its return value can be used to conditionally apply different layouts, UI components, or styles based on the device's orientation. By incorporating this method, the application can seamlessly adapt to changes in the device orientation, providing a consistent and intuitive user interface.

RETURN VALUE:

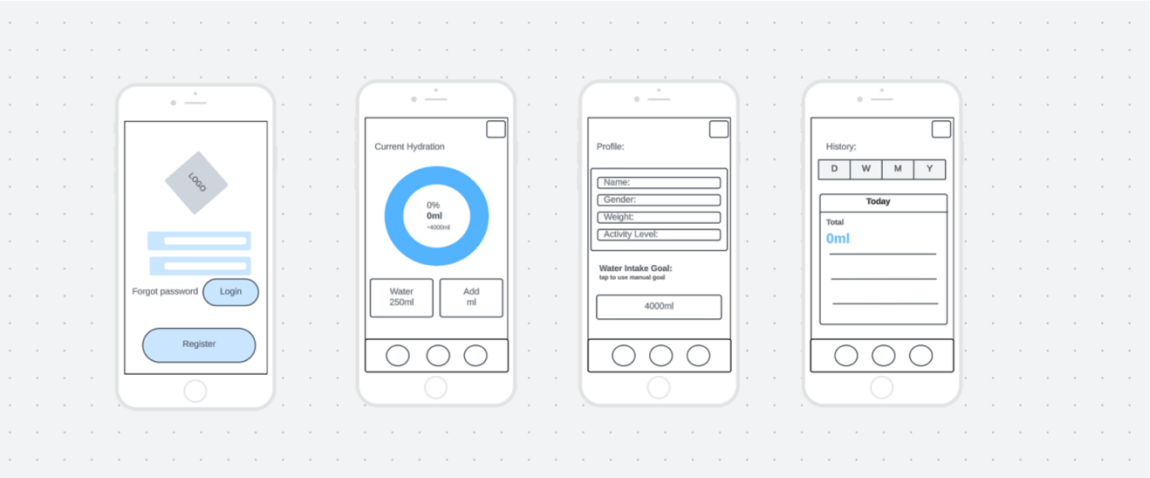
- **true** if the device is in landscape mode.

- **false** if the device is not in landscape mode.

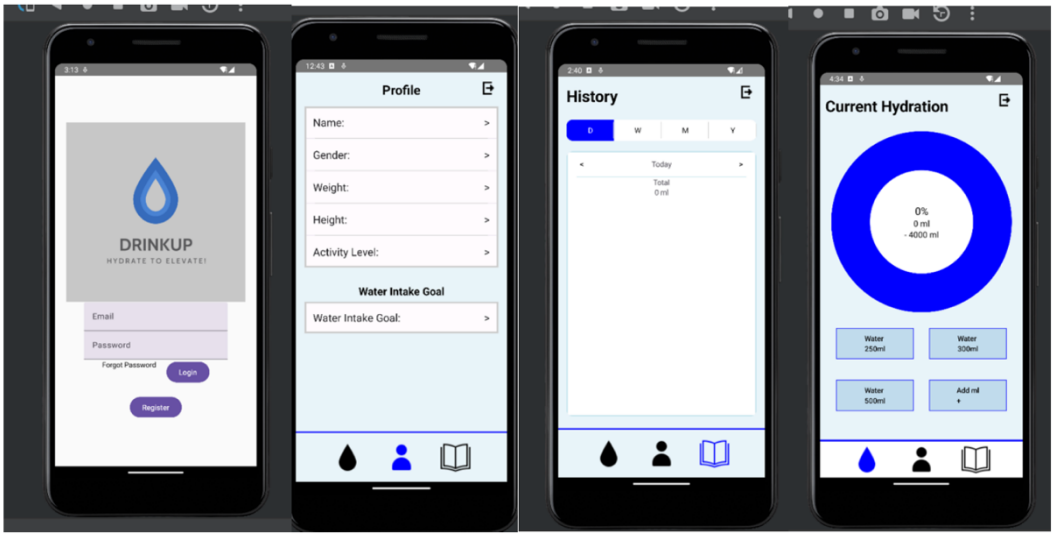
WIREFRAMES AND SCREENSHOTS AND REFERENCE

https://icons8.com/icon/set/book/fluency - Icons generator https://www.designevo.com/logo-maker/ - Logo maker
https://kotlinlang.org/docs/home.html - Kotlin documentation which I used to create my project.

WIREFRAME



SCREENSHOTS



Milestone 2

INTRODUCTION

In this milestone DrinkUp which monitor user water intake is now featuring integrated sensor capabilities for localization. With the addition of location services, DrinkUp extends beyond its original scope, offering users a comprehensive experience across four key activities: Login, Profile, Current Hydration, and History.

To empower the localization feature, the application now leverages the following permissions:

- ACCESS_FINE_LOCATION: Enables precise GPS-based location tracking.
- ACCESS_COARSE_LOCATION: Facilitates general location information retrieval.
- INTERNET: Ensures seamless connectivity for data exchange.
- ACCESS_NOTIFICATION_POLICY: Grants access to notification policies for a streamlined user experience.
- POST_NOTIFICATIONS: Allows the application to post notifications, enhancing user engagement.

Now, DrinkUp takes hydration tracking a step further by sending notifications when the weather exceeds 25 degrees Celsius. This smart feature not only keeps users informed about their water intake but also considers external factors like temperature, enhancing the app's adaptability to individual needs. Embrace the future of hydration tracking with DrinkUp as we redefine the standard for personalized and weather-informed wellness.

USER INTERFACE DESIGN

In this iteration of DrinkUp, while the visual aesthetics of the user interface remain largely consistent, significant improvements have been made to enhance user interaction and provide more accurate hydration guidance. The focal point of this upgrade lies in refining the logic and functionality behind the buttons, particularly in relation to the hydration circle.

1. Interactivity Enhancement:

• The hydration circle has undergone a transformation, allowing users to engage with it more intuitively. Now, users have full interactivity with the hydration circle, enabling a dynamic and responsive experience.

2. Localization and API requests:

• The application utilizes location-based data to send API requests. When the local temperature surpasses 25 degrees, considered a warm threshold, users are notified to increase their water intake. The weather API accommodates up to 1000 daily requests, sufficient for the current user base. However, if user engagement increases, an upgrade to unlimited requests may be necessary.

3. Improved Logic:

• The buttons within the application have been redesigned to incorporate a more sophisticated logic system. This logic is particularly evident in the hydration circle, which now adapts to user input and provides real-time feedback based on individual hydration needs.

4. Personalized Water Intake:

• One of the standout features of this update is the application's ability to offer personalized water intake recommendations. By refining the underlying algorithms, the hydration circle accurately

calculates and displays the correct water intake for each user, taking into account factors such as age, weight, and now, external temperature.

5. Dynamic Adaptability:

• The user interface now dynamically adjusts based on the user's interactions and real-time data inputs. This ensures a seamless and personalized experience, allowing users to easily track and meet their hydration goals in various scenarios.

In essence, while the visual elements of the user interface maintain familiarity, the user experience has evolved significantly. The enhanced logic behind the buttons and the interactive capabilities of the hydration circle contribute to a more intuitive and adaptive application, ensuring that users receive accurate and personalized hydration guidance. This marks a pivotal step forward in the continuous improvement of DrinkUp, aligning with our commitment to providing a user-centric and effective hydration tracking solution.

NAVIGATION AND LAYOUT

In this iteration of DrinkUp, we've introduced several intuitive features to the navigation and layout, optimizing user interaction and providing a more comprehensive overview of daily water intake.

1. Incremental Water Intake:

• Users can now easily track and add specific quantities of water to their daily intake. Pressing the button with a 300 ml label, for example, seamlessly increments both the percentage displayed on the hydration circle and the corresponding millilitres consumed within the day. This feature ensures a straightforward and interactive approach to monitoring hydration progress.

2. Dynamic Colour Changes:

• As users add water throughout the day, the hydration circle undergoes dynamic colour changes. The circle starts with a neutral white colour and progressively adds blue, visually representing the user's hydration progress. This visual cue offers an at-a-glance overview of how close the user is to meeting their daily water intake goal.

3. Midnight Circle Reset:

• To facilitate a fresh start each day, the hydration circle automatically resets to a pristine white colour every midnight. This reset aligns with the natural rhythm of a new day, allowing users to begin tracking their hydration anew. The reset ensures that users can easily gauge their daily progress without carryover from the previous day.

4. Improved Layout for Clarity:

• The layout has been optimized to ensure clarity and ease of use. Buttons for specific water quantities are strategically placed, making it convenient for users to quickly input their water intake. The dynamic colour changes on the hydration circle and the midnight reset contribute to a visually intuitive layout.

5. Real-time Feedback:

• The application provides real-time feedback on both the hydration circle and numerical metrics, giving users immediate insight into their hydration status. This feature empowers users to make informed decisions about their water intake throughout the day.

4. Calendar and history data:

• The calendar been improved, now the user can select Date , week, month or year from data that he will store in the DB. This will be connected to the database and pulling data accordingly for the user. Also the graph was implemented so the user can see how much water he rank in current day or current year, week or month.

These navigation and layout enhancements not only simplify the user experience but also provide a visually engaging and informative platform for tracking hydration. By incorporating dynamic colour changes, incremental water intake tracking, and a daily midnight reset, DrinkUp ensures that users have a seamless and motivating experience on their journey towards optimal hydration.

DATA SHARING BETWEEN ACTIVITIES

The application shares data related to the user's current hydration level and hydration goal. This information is utilized across three activities within the app: CurrentHydration, History, and Profile. It is Automatically updated everywhere through those activities once it is updated on Profile. Smart calculation of recommended water intake also implemented.

- The CurrentHydration activity displays the user's current hydration level and allows them to update it as they consume fluids throughout the day.
- The History activity provides a historical view of the user's hydration levels over time, offering insights into their hydration habits.
- The Profile activity allows the user to set their hydration goal, which is then reflected in the CurrentHydration and History activities.

KOTLIN CODE DESCRIPTION

The application is written in Kotlin and makes use of custom composable for reusable UI elements. Here's an example of a custom composable function:

1. fun calculateRecommendedWaterIntake(weight: Int, height: Int, activityLevel: Float, gender: String): String

Calculates the recommended water intake for a person based on their weight, height, activity level, and gender. That is why in his profile we gather information about user. Collecting name would be use for personalize application for the user himself sort of welcome message on the CurrentHydration or somewhere else.

For men/others:

water intake = ((weight/30 + height/100) * 1000 * 0.85) + activity level

For women:

water intake = ((weight/30 + height/100) * 1000 * 0.8) + activity level

In these formulas, the activity level is a multiplier that represents how active the user is. For example, it will be used 0 ml added for no activity, 100ml for low activity, 250ml for moderate activity, and 500ml for high activity. The 0.8 multiplier for women accounts for the generally lower water requirements compared to men and men 0.85

2. fun LocationUpdates(onLocationChanged: (Location) -> Unit) METHOD DOCUMENTATION

In summary, the LocationUpdates function is a comprehensive solution for obtaining location updates from the device's GPS sensor. It handles permission requests, location updates, and lifecycle management, providing a seamless interface for location-based functionality in our application.

DESCRIPTION:

1. **Function Definition**: The function LocationUpdates is defined with a single parameter, onLocationChanged. This parameter is a call-back function that is triggered whenever there is a change in the device's location.
2. **Context and Location Manager**: The function begins by obtaining the current context and the location manager from the system services. The location manager is a system service that provides access to the device's location services.
3. **Location Listener**: A location listener object is created. This object is designed to listen for updates from the location manager. It overrides several methods:

o onLocationChanged: This method is called when the device's location changes. It triggers the onLocationChanged call-back function, passing the new location as a parameter.

o onStatusChanged, onProviderEnabled, onProviderDisabled: These methods are also overridden, but are not used in this function (the first one is deprecated).

4. **Disposable Effect**: A disposable effect is used to manage the lifecycle of the location updates. It performs several actions:

o **Permission Check**: It checks if the application has the ACCESS_FINE_LOCATION permission. If the permission is not granted, it requests the permission from the user.

o **Request Location Updates**: It requests location updates from the location manager. These updates are provided by the GPS provider. The location listener object is passed as a call-back to handle these updates.

o **Clean-up**: A clean-up action is defined to remove the location listener from the location manager when the effect is disposed. This ensures that the application does not continue to receive location updates when they are no longer needed.

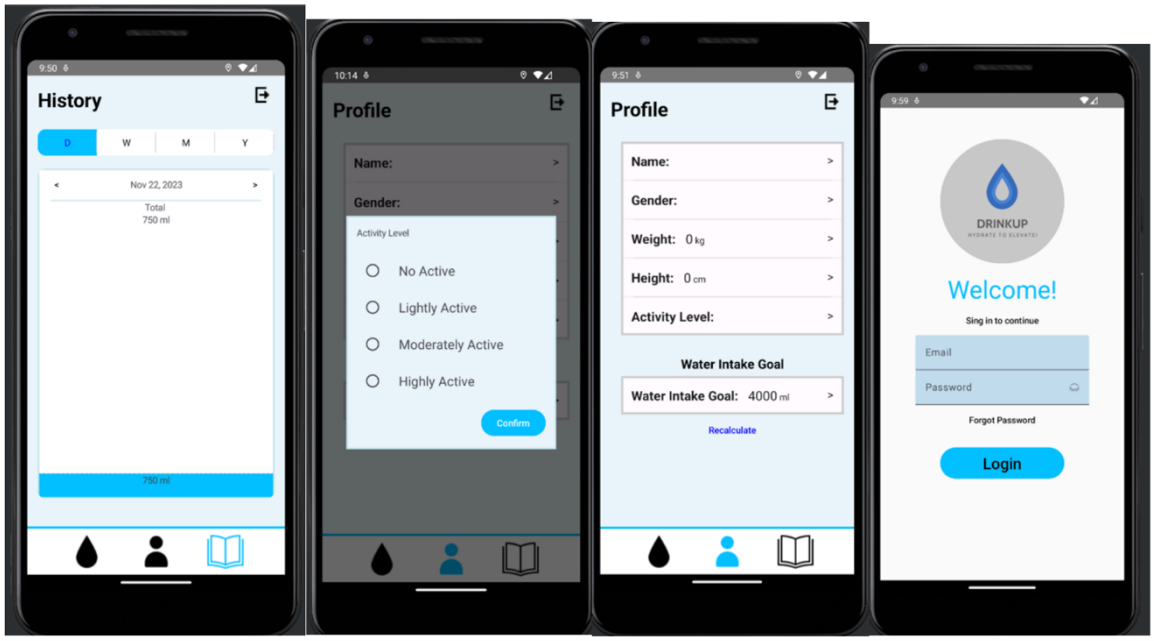
3. ADDED TESTS:

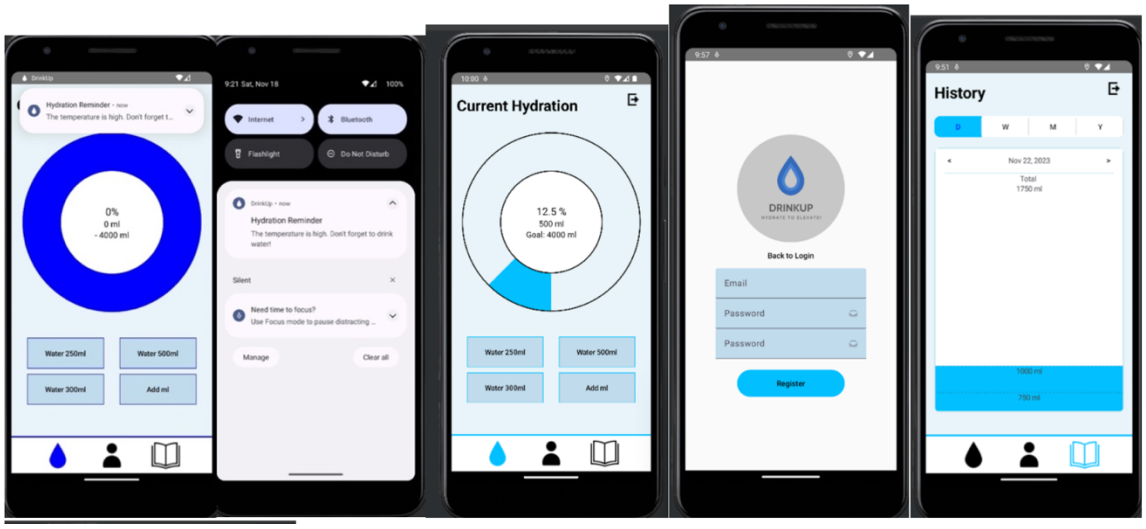
1. **Unit Testing**: To ensure the correctness of our code, I have written unit tests for functions and methods. These tests check if a particular unit of code is working as expected under various conditions.
2. **Integration Testing**: I have also added integration tests to validate behaviour of temperature check.

WIREFRAMES, SCREENSHOTS AND REFERENCE

- https://kotlinlang.org/docs/home.html - Kotlin documentation which I used to create my project.
- https://www.baeldung.com/kotlin/password-validation - Documentation about validation of password which have been used in my project
- https://www.geeksforgeeks.org/how-to-validate-textfields-in-a-login-form-in-android-using-jetpack-compose/ - validation of the input of the user which was used in my project
- https://stackoverflow.com/questions/23833765/how-to-use-drawarc - use for drawing an arc and make Hydration circle smarter. So user can see real-time progress visually
- https://learn.microsoft.com/en-us/dotnet/api/system.drawing.graphics.drawarc?view=dotnet-plat-ext-7.0 – one more for drawing the circle and inside the circle used for real time progress
- https://stackoverflow.com/questions/15448375/explaining-drawarc-method - arc and filling circle method
- https://www.mayoclinic.org/healthy-lifestyle/nutrition-and-healthy-eating/in-depth/water/art-20044256 - documentation about average water intake for adult
- https://waterproffi.com/water-calculator-by-weight-and-height/ - Documentation supporting the recommended water intake for user which the application will be using, the user however will be able to rewrite the suggested water intake. Here is where I got water intake formula with weight and height.

- https://rrtutors.com/tutorials/create-bar-chart-with-jetpack-compose - for BarChart but I had challenges importing the library so decided to go with column.
- https://www.nutrition.org.uk/healthy-sustainable-diets/hydration/?level=Consumer – for gender modification of water intake, look there for report about how much I should add for female or male
- https://www.health.harvard.edu/staying-healthy/how-much-water-should-you-drink - Information about water, intake water when I set automatically water to 3000 it is in-between recommended intake for female which is 2700 and male which is 3700. Scientific reports in this article.





Milestone 3

INTRODUCTION

In this milestone, the DrinkUp application, which monitors user water intake, has been significantly enhanced with the integration of database capabilities for persistent data storage and the use of Kotlin coroutines for non-blocking network operations. This development marks a pivotal point in the evolution of the application, as it now offers a more robust and comprehensive user experience.

The DrinkUp application now incorporates SQLite, a powerful open-source SQL database engine, to manage user data. This integration has led to the introduction of three tables within the application's database structure. These tables, with their clearly defined relationships, form the backbone of the data management system, ensuring seamless data storage, retrieval, and manipulation.

The use of SQLite not only provides efficient data management but also ensures data persistence. This means that user interactions and data are securely stored and readily available, even when the application is not in active use. This persistent storage capability significantly enhances the user experience, making the DrinkUp application more reliable and user-friendly.

In addition to the database integration, the application now uses Kotlin coroutines for non-blocking network operations. **The CoroutineScope(Dispatchers.IO).launch { ... }** block is used to fetch weather data from the OpenWeatherMap API without blocking the main thread, which keeps the UI responsive. The fetched weather data is then stored in a variable, which is observed in the composable function. When the data changes (i.e., when new weather data is fetched), the composable function recomposes, and the UI is updated with the new data. This results in a smooth and responsive user experience.

Furthermore, the DrinkUp application has been updated to include altitude data from the localization sensor in its hydration recommendation calculations. This means that the application now takes into account the user's current altitude when determining how much water they should consume. This is particularly useful for users who are at high altitudes, where the body tends to dehydrate faster. By incorporating this data, the DrinkUp application can provide even more accurate and personalized hydration recommendations, further enhancing the user experience. This feature, combined with the other enhancements, continues to position DrinkUp as a leader in its field. Also using localization to change colour of menu if the user is in temperature above 23 degrees not only notification be send but also the colour of menu will change.

In summary, the integration of SQLite and the use of coroutines for network operations significantly enhance the DrinkUp application's functionality and user experience. The application now provides a more personalized, data-driven hydration tracking experience, making DrinkUp a leader in its field

USER INTERFACE DESIGN

The database, implemented using SQLite, serves as the backbone for data storage and management within the application. It consists of three tables: users, profile, and hydration_for_day.

1.User Table

The users table is designed to store essential user information. It includes the following fields:

- **id:** A unique identifier for each user.
- **username:** The user's chosen username.
- **hashed_password:** The user's password, which is hashed and salted for security.
- **salt:** A unique salt associated with each user to further secure the password.
- **created_at:** A timestamp indicating when the user account was created.
- **deleted_at:** A nullable timestamp used for soft deletes, allowing user accounts to be deactivated without permanently removing them from the database.

2. Profile Table

The profile table stores detailed profile information for each user. Each user can have one profile. It includes the following fields:

- id: A unique identifier for each profile.
- name: The user's name.
- gender: The user's gender.
- activity_level: The user's activity level.
- height: The user's height.
- weight: The user's weight.
- user_id: A foreign key referencing the id in the users table, linking each profile record to a specific user.
- deleted_at: A nullable timestamp for soft deletes.
- recalculate : Boolean defaulted to false

3. Hydration For Day Table

The hydration_for_day table stores daily hydration data for each user. Each user can have multiple records in this table. It includes the following fields:

- date: The date for which the hydration information is recorded.
- value_of_day: The actual hydration value for the day.
- goal: The hydration goal for the day.
- user_id: A foreign key referencing the id in the users table, linking each hydration record to a specific user.
- deleted_at: A nullable timestamp for soft deletes.

The integration of these tables into the DrinkUp application allows for efficient data management and a seamless user experience. The use of SQLite ensures data persistence, meaning that user data and interactions are securely stored and readily available, even when the application is not in active use. This integration significantly enhances the functionality of the application, making it more reliable and user-friendly. It also allows for more sophisticated hydration calculations by storing and analyzing sensor data, providing a more personalized hydration tracking experience.

NAVIGATION AND LAYOUT

The DrinkUp application now includes a secure login and registration system. Users are required to register with a valid email address and create a password. The application verifies the email and password via the SQLite database, ensuring that only registered users can access the application.

Once the user is logged in, they are transitioned to the 'Current Hydration' activity. Here, users can see their hydration progress for the day. The application fetches the user's hydration data from the SQLite database and displays it in an easy-to-understand format.

The 'Profile' activity allows users to update their profile information. This includes their hydration goals, weight, and other personal details. The updated information is stored in the SQLite database, allowing for a personalized user experience.

When a user navigates to the history activity, the application fetches the relevant hydration data from the SQLite database based on the user's selected period (day, week, month, or year). This data is then used to generate a comprehensive overview of the user's hydration progress over the selected period.

In addition to viewing their daily hydration data, users can now see their cumulative hydration goal achievement over the selected period. This feature provides users with a broader perspective on their hydration habits, helping them understand their progress towards their hydration goals over time.

In summary, the enhancements in navigation and layout, the application now provides a more personalized, data-driven hydration tracking experience, making DrinkUp a leader in its field.

DATA SHARING BETWEEN ACTIVITIES

1. Seamless Transition to History Activity:

- Users can seamlessly navigate to the 'History' activity, triggering a dynamic data retrieval process.
- Upon entry, the application intelligently fetches relevant hydration data from the SQLite database, aligning with the user's selected period (day, week, month, or year).

2. Comprehensive Data Visualization:

- The retrieved data powers a comprehensive overview of the user's hydration progress over the chosen period.
- Users can now visually interpret their daily hydration data, gaining insights into their evolving hydration habits.

3. Cumulative Goal Achievement Analysis:

- A notable addition is the cumulative hydration goal achievement feature.
- Users can track their progress toward hydration goals over the selected period, providing a holistic understanding of their hydration journey.

4. Enhanced User Engagement:

- This milestone elevates user engagement by offering a more immersive and informative experience.
- The application not only presents data but contextualizes it, empowering users to make informed decisions about their hydration goals.

5. Real-time Data Synchronization:

- The DrinkUp application ensures real-time synchronization between different activities.
- Any updates or changes made in the 'Current Hydration' or 'Profile' activities are immediately reflected in the 'History' activity, creating a seamless and synchronized user interface.

6. Holistic Progress Tracking:

- Users can now track their daily hydration as well as observe their overall progress, fostering a more holistic approach to health and wellness.

7. User-Centric Data Interaction:

- The application places a strong emphasis on user-centric data interaction, providing a personalized and tailored experience.
- Users can effortlessly navigate between activities, gaining deeper insights into their hydration journey.

8.EFFICIENT DATA GENERATION WITH COROUTINES:

- As part of the recent improvements, we have also implemented a mechanism for creating demo data using Kotlin’s coroutines.
- This approach allows us to populate the SQLite database with test user profiles and hydration data in an efficient and non-blocking manner.
- By marking the data insertion functions as suspend, we can call these functions within a coroutine scope, which runs on a background thread.
- This ensures that the main thread, which is responsible for updating the UI, is not blocked while the database operations are being performed.
- As a result, the application remains responsive, enhancing the overall user experience.
- Moreover, this approach allows us to easily generate a large amount of test data, which can be useful for testing the application’s performance and functionality.

In summary, the Milestone 3 advancements in data sharing and interactivity redefine the DrinkUp application's user experience. The seamless transition between activities, coupled with insightful data visualization and cumulative goal tracking, positions DrinkUp as a frontrunner in delivering a user-centric, data-driven hydration tracking experience.

KOTLIN CODE DESCRIPTION

1. Object DatabaseManagerSingleton {}

This is a singleton object for managing database operations. It provides a single global point of access to the DatabaseManager instance, ensuring that only one instance of DatabaseManager is created and shared among other classes in the application. This is particularly useful for resources that are expensive to create, like database connections. The getInstance(context: Context): DatabaseManager function returns the singleton instance of DatabaseManager, creating it if necessary. The database is named "drink_up.db".

2. private fun generateSalt(): String {}

- @RequiresApi(Build.VERSION_CODES.O): This annotation indicates that this function requires API level O (Oreo, 8.0, API level 26) or higher. This is because the function uses Base64.getEncoder(), which is available from API level 26 onwards.
- private fun generateSalt(): String: This is the function declaration. It's marked as private, meaning it can only be accessed within the same file. The function doesn't take any parameters and returns a String.
- val random = SecureRandom(): This line creates a new instance of SecureRandom, which provides a cryptographically strong random number generator.
- val salt = ByteArray(16): This line initializes a byte array of length 16 to store the salt.
- random.nextBytes(salt): This line generates random bytes and places them into the byte array salt.
- return Base64.getEncoder().encodeToString(salt): This line converts the byte array to a Base64-encoded string. This is useful because it allows the binary salt to be represented as a string of ASCII characters.

In summary, this function generates a random salt, which can be used for cryptographic operations such as hashing or encryption. The salt is returned as a Base64-encoded string.

3. onStop() in Login Activity

In the onStop() function, which is called when the activity is no longer visible to the user, we launch a coroutine to insert the new user and their profile into the database if a new user has registered. If the insertion is successful, we also insert a default user profile for the new user. If an exception occurs during this process, we print the stack trace.

4. LogOutButton(modifier: Modifier = Modifier): Unit

This composable function displays a logout button. When clicked, it shows an alert dialog confirming the logout. After a delay of 2 seconds, it resets all the user-related variables in the `AppVariables` object and navigates back to the login screen. The reset is done using the `AppVariables.resetAllValues()` function, which sets all the variables to their initial states. This ensures that no user data is retained after logout.

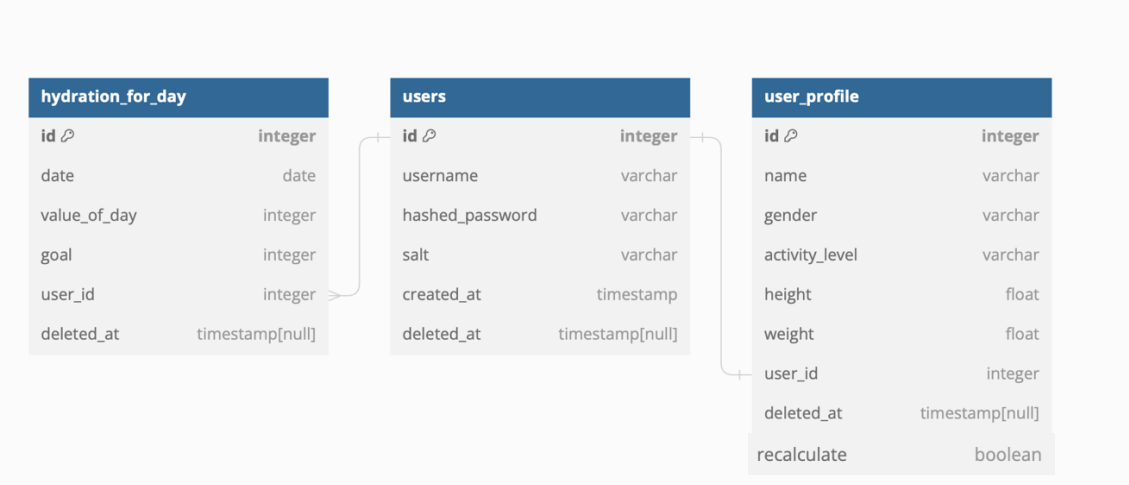
Difficulties during development proccess

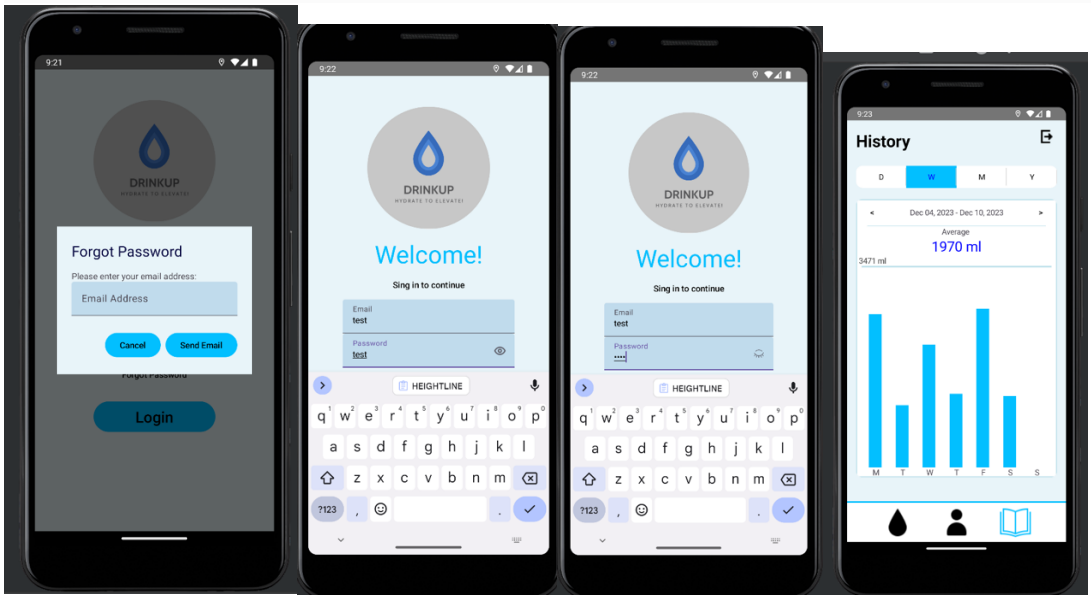
During the development process, we encountered some:

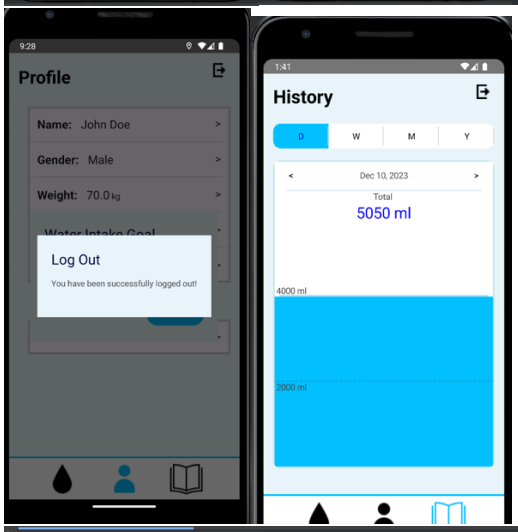
- related to inserting null values into the database. This happened when the user did not exist (i.e., getUserIdByEmail() returned null). To handle this, we used the ?: return syntax to return from the function early if the user ID is null. This prevented null values from being inserted into the database. However, it also meant that if a user ID was not found, the rest of the function wouldn't be executed. This was why the user profile was not being created when a new user was inserted. To solve this, we ensured that a profile is created as soon as a user is created.
- an issue where a new user registering on a device would not have a profile created. This was because the email of the last logged-in user was remembered by the device. To solve this, we made sure to clear all user data, including the email, when logging out. This is done in the `AppVariables.resetAllValues()` function.

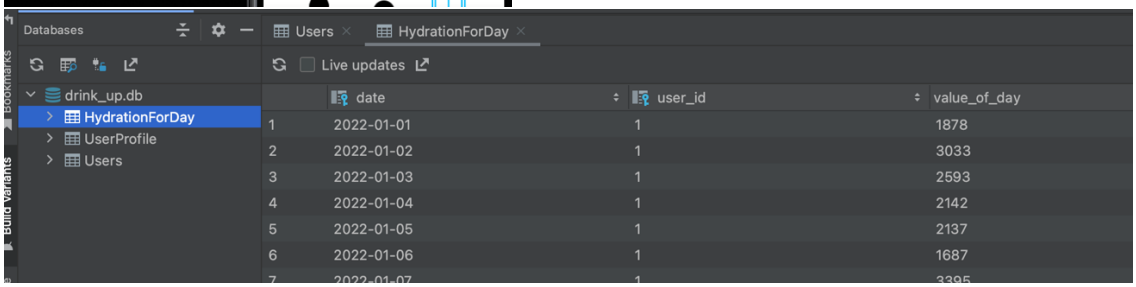
WIREFRAMES, SCREENSHOTS AND REFERENCE

- https://dbdiagram.io/d - used for drawing a relation diagram of my database and structure
- https://kotlinlang.org/docs/home.html - Kotlin documentation which I used to create my project.
- https://www.geeksforgeeks.org/how-to-read-data-from-sqlite-database-in-android-using-jetpack-compose/ - how to get the data out of database
- https://developer.android.com/training/data-storage/sqlite - saving data into database
- https://developer.android.com/kotlin/coroutines - documentation about coroutines.
- https://github.com/SlothLabs/kotlin-mail - for my mailer
- https://www.lifewire.com/md-file-4143558 - about files we had to exclude due my build failure as it seem to be a duplicates
- https://rgbcolorcode.com/color/converter/ - for color convertions
- https://stackoverflow.com/questions/72933305/android-kotlin-alertdialog-builder-usage-and-context-this-not-working - For alert builder which we tried but it didn't work for. Instead using
- https://support.google.com/accounts/answer/185833?hl=en&authuser=1# - google documentation about setting up an app email
- https://stackoverflow.com/questions/35347269/javax-mail-authenticationfailedexception-535-5-7-8-username-and-password-not-ac - debugging of authentitcation of email

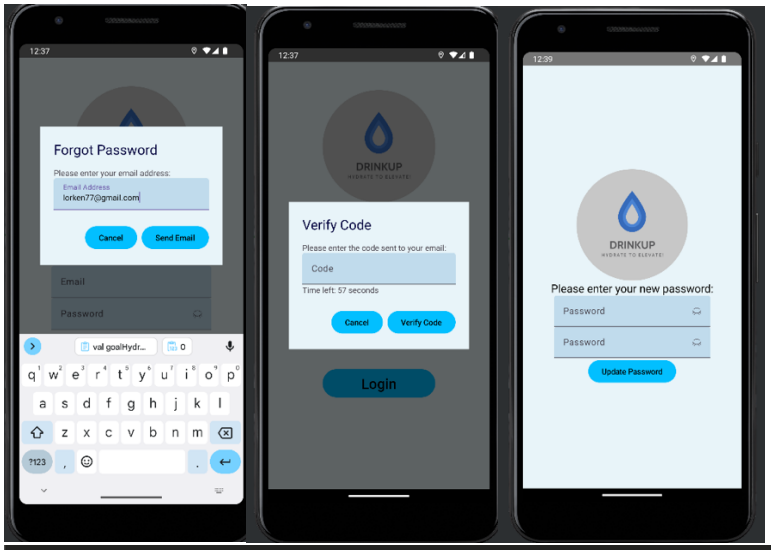


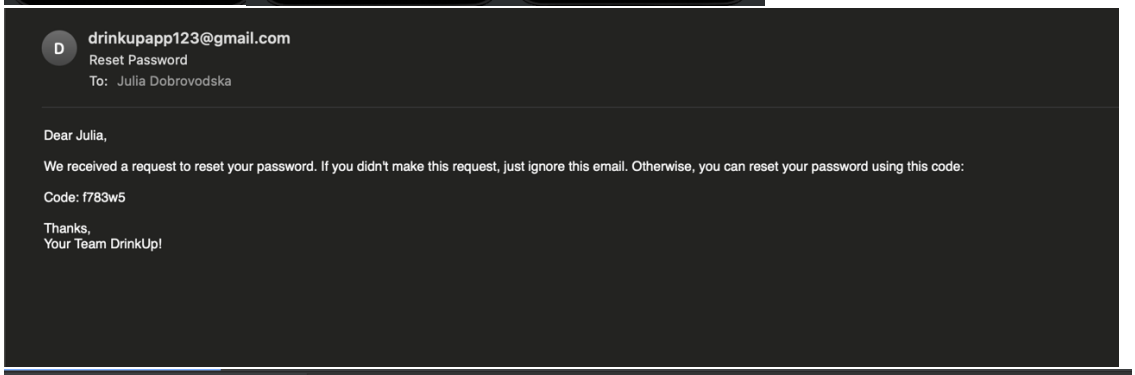


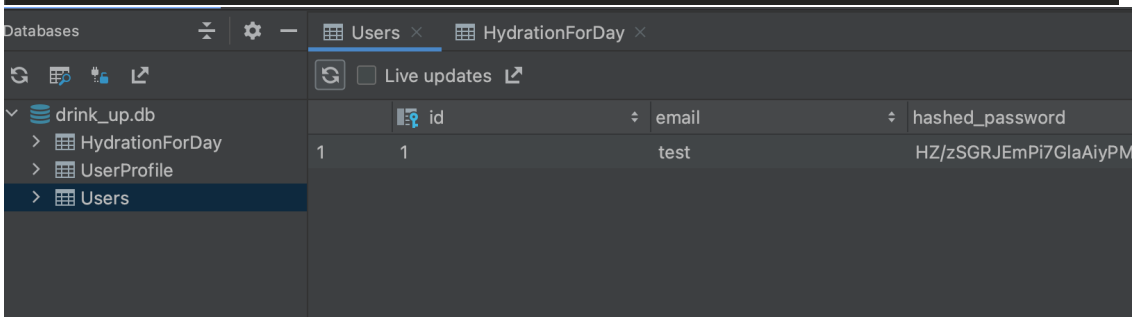




	date	user_id	value_of_day
1	2022-01-01	1	1878
2	2022-01-02	1	3033
3	2022-01-03	1	2593
4	2022-01-04	1	2142
5	2022-01-05	1	2137
6	2022-01-06	1	1687
7	2022-01-07	1	3395


12:37
Forgot Password
Please enter your email address:
Email Address
lorken77@gmail.com
Cancel
Send Email
Email
Password
12:37
DRINKUP
Verify Code
Please enter the code sent to your email:
Code
Time left: 57 seconds
Cancel
Verify Code
Login
12:39
DRINKUP
Please enter your new password:
Password
Password
Update Password


D
drinkupapp123@gmail.com
Reset Password
To: Julia Dobrovodska
Dear Julia,
We received a request to reset your password. If you didn't make this request, just ignore this email. Otherwise, you can reset your password using this code:
Code: f783w5
Thanks,
Your Team DrinkUp!


Databases
Users
HydrationForDay
Live updates
drink_up.db
HydrationForDay
UserProfile
Users
id
email
hashed_password
1
1
test
HZ/zSGRJEmPi7GlaAiyPM