User manual

Lib/models/period_model.dart

This Dart code defines a PeriodData class used to model period cycle data, intended for use with Firebase Firestore. The class has fields for tracking information such as the start date of the last period, cycle length, period length, symptoms, notes, and timestamps for creation and updates.

- The PeriodData constructor initializes these fields.
- The fromFirestore factory method converts Firestore document data (stored as a Map<String, dynamic>) into a PeriodData object, converting Firestore Timestamp values into Dart DateTime objects.
- The toFirestore method converts a PeriodData object into a Map<String, dynamic>, suitable for saving into Firestore. It ensures the created_at field uses Firestore's server timestamp if not already set and updates updated_at with the current server timestamp on each update.

Lib/navigation/bottom_nav_bar.dart

This code creates a BottomNavBar widget, which is a bottom navigation bar for a Flutter app with four menu items: Home, Calendar, Notifications, and Settings. Each item displays an icon and a label. The background color is dark red, the selected item is cream-colored, and unselected items are grey. When a user taps on any item, the onTap function is triggered.

Lib/screens/notification/medicationlistscreen.dart

This Flutter code defines a medication notification list screen using Firebase Authentication and Firestore. It checks if the user is logged in, then fetches and displays their medication reminders from Firestore in a list. Each list item shows the medication name, time, and dosage. Users can tap on an item to navigate to a detailed notification screen or add a new reminder using the floating action button.

Lib/screens/notification/mednoti.dart

This Flutter code defines a MedNoti screen that allows users to view and edit details of a medication reminder from Firestore. It provides input fields for editing the medicine name, dosage, and time of notification. Users can also delete the reminder. The time input is handled using a time picker, and changes are saved back to Firestore. If the user deletes a reminder, it's removed from Firestore and the user is notified. Validation checks ensure the medicine name is not left blank, and a default value of 1 is used for the dosage if none is provided.

Lib/screens/notification/notidetail.dart

This Flutter code defines the NotiDetail screen, allowing users to create a new medication reminder in the app. It provides input fields for the medicine name, type, dosage, time of notification, and a custom message. The DropdownButtonFormField offers a selection of predefined medicine types such as birth control pills or pain relief for menstrual cramps. The time picker allows the user to set a specific notification time, and a button to save the reminder adds the data to Firestore under the logged-in user's collection.

Lib/screens/notification/ notification_screen.dart

This NotificationScreen widget helps users track their menstrual cycle and manage medication reminders. Key features include:

1. **Menstrual Cycle Tracking**: Users can toggle between *on period*, *period ended*, and *ovulating* using switches.
2. **Medication Reminders**: Displays a list of medication reminders with options to add, edit, or delete.
3. **Navigation**: Users can add new reminders via the NotiDetail screen or edit existing ones through MedNoti.

Lib/screens/settings/home_screen.dart

This Flutter code defines a **HomeScreen** for a menstrual cycle tracking app. The app fetches period data from Firebase, displays information about the user's last and predicted periods, and allows users to log a new period using a date picker. The screen uses a bottom navigation bar to switch between different features like the menstrual calendar, notifications, and settings.

Key features:

1. **Data Fetching and Logging**: The app loads period data from Firebase and allows users to log new period dates. Upon success or failure, appropriate messages are shown using snack bars.
2. **Period Tracking**: Displays the last period and calculates the next period and fertile window.
3. **UI Components**: Includes a circular card showing days until the next period, and info cards to display period and fertility data. There are also buttons for adding new period data.
4. **Navigation**: The bottom navigation allows users to navigate to a calendar, notifications, and settings.

Lib/screens/settings/login_screen.dart

This Flutter code creates a **LoginScreen** for the "HerCycle" app. Users can enter their email and password to log in, or navigate to registration and password recovery pages.

Key features:

1. **Login Validation**: When users enter their email and password, the app checks them against the Profile data passed from the registration screen. If the credentials are correct, it navigates to the home screen.
2. **Error Handling**: If the email or password is empty or incorrect, an error message is displayed using a snack bar.
3. **UI Elements**: The screen includes text fields for email and password input, a "Remember Me" checkbox (though not functional here), and buttons for login, registration, and password recovery.
4. **Navigation**: Users can navigate to the registration screen or reset their password.

Lib/screens/settings/menstrual_cycle_calendar.dart

This Flutter code defines a **MenstrualCycleCalendar** screen using the TableCalendar widget. It allows users to select multiple days to mark their menstrual cycle on a calendar.

Key features:

1. **Calendar**: The calendar displays dates between 2020 and 2030. Users can switch between different calendar formats (e.g., month, week) and select or deselect days.
2. **Day Selection**: Users can tap on a date to mark or unmark it. The selected days are stored in a set.
3. **UI and Navigation**: The app has a header with a close button to go back, and two action buttons at the bottom—Cancel (to discard changes) and Save (to store the selected days). The buttons are styled with custom colors.

Lib/screens/settings/register_screen.dart

This Flutter code creates a **RegisterScreen** for user registration using Firebase Authentication and Firestore. It includes input fields for name, email, password, and password confirmation, with validation checks. Upon successful registration, user details are stored in Firestore, and the user is redirected to the login screen. Error messages and a loading indicator ensure smooth user experience.

Lib/screens/settings/settings_screen.dart

This code defines a **SettingsScreen** in Flutter, providing various options for managing app settings. The screen features an app bar with a language selection dropdown and a list of setting options, including:

1. **Profile Section**: Displays the user's profile with a button to edit information.
2. **Settings Options**: Buttons navigate to different settings such as:
    - **Report & Graphs**
    - **Cycle & Ovulation**
    - **App Settings**
    - **Access & Security**
    - **Notifications**
    - **Help**
3. **Logout Button**: Allows users to log out.

Lib/screens/settings/welcome_screen.dart

This **WelcomeScreen** is a simple Flutter UI that introduces users to the app:

- **Logo**: Displays at the top left.
- **Main Image**: A central image related to health.
- **Welcome Text**: A title and brief introduction to menstrual tracking benefits.
- **Next Button**: At the bottom right, it navigates to the login screen.

Lib/services/firebase_auth_services.dart

This Dart code defines a FirebaseAuthServices class for handling Firebase authentication in Flutter. It provides:

- **signUpWithEmailAndPassword()**: Registers a user with email and password.
- **signInWithEmailAndPassword()**: Signs in an existing user.
- **_handleAuthException()**: Handles Firebase authentication errors.
- **signOut()**: Logs out the current user.
- **getCurrentUser()**: Retrieves the currently signed-in user, if any.

Lib/services/firebase_options.dart

This Dart code defines the DefaultFirebaseOptions class, which provides platform-specific Firebase configuration options for a Flutter app. It determines the platform at runtime and retrieves the appropriate Firebase configuration.

Key points:

- **currentPlatform**: Determines the platform (e.g., Android, iOS, etc.) and returns the corresponding Firebase configuration. If the platform is unsupported or not configured, an UnsupportedError is thrown.
- **android**: Contains Firebase configuration options for the Android platform (e.g., apiKey, appId, etc.).

Lib/services/firebase_service.dart

This Dart code defines a FirebaseService class that provides several methods for interacting with Firebase services, such as Firestore and Firebase Authentication, in a Flutter app.

Key functionality includes:

- **initializeFirebase()**: Initializes the Firebase app. Handles errors related to Firebase initialization.
- **addUser()**: Adds a new user to Firestore with a specified user ID and user data.
- **getUserData()**: Retrieves user data from Firestore based on a user ID.
- **updateUserData()**: Updates existing user data in Firestore.
- **deleteUser()**: Deletes a user document from Firestore by user ID.
- **isUserLoggedIn()**: Checks if a user is currently logged in using Firebase Authentication.
- **signOut()**: Logs out the current user from Firebase Authentication.

Lib/services/period_firebase_service.dart

This Dart code manages menstrual cycle tracking with Firebase. It consists of:

1. **PeriodData Class**:
    - Holds period info (start date, cycle length, symptoms, etc.).
    - Provides JSON conversion for Firestore.

2. **PeriodFirebaseService Class**:
    - Manages Firestore operations:
        - updatePeriodData(): Updates current period data.
        - getCurrentPeriodData(): Retrieves the latest period data.
        - getPeriodHistory(): Gets past period data.
        - calculateNextPeriod(): Predicts next period.
        - calculateFertileWindow(): Estimates fertile window.
        - logNewPeriod(): Logs new period and moves old data to history.
        - streamCurrentPeriodData(): Streams current period data.

Lib/services/period_service.dart

This Dart code defines a PeriodService class to manage period data in Firestore for a logged-in Firebase user. It handles CRUD operations and error handling via custom exceptions.

Key Components:

1. **PeriodServiceException**: Custom exception for handling period service errors.
2. **PeriodService**:
    - Manages Firebase Firestore and FirebaseAuth instances.
    - **updatePeriodData()**: Updates the user's period data in Firestore.
    - **getPeriodData()**: Fetches the user's period data.
    - **streamPeriodData()**: Streams real-time updates of period data from Firestore.
    - **calculateNextPeriod()**: Calculates the next expected period start date.
    - **isCurrentlyOnPeriod()**: Checks if the user is currently on their period.

Lib/main.dart

"HerCycle" is a Flutter app for menstrual cycle tracking, using Firebase for backend services. It supports Thai and English, with a pink theme. The app initializes Firebase and Thai date formatting, then routes to various screens: welcome, login, registration, home, settings, menstrual cycle calendar, and notifications. It also includes dynamic routing for medication notifications with arguments passed for custom handling.

Lib/profile.dart

This code defines a class named Profile in Dart. The class has three final properties: name, email, and password. These properties are immutable once initialized. The constructor uses named parameters with the required keyword, meaning that when creating an instance of Profile, you must provide values for name, email, and password.

Lib/type.dart

This code defines three enum types in Dart:

1. **WeekDay**: It has two values, short and long. This enum could represent the format of a weekday, such as short (e.g., "Mon") or long (e.g., "Monday").
2. **SelectedDayPosition**: It has three values, left, right, and center. This enum could be used to specify the position of a selected day, perhaps in a calendar or timeline interface.
3. **FullCalendarScroll**: It has two values, horizontal and vertical. This enum likely represents the scroll direction in a full calendar view, determining whether the user can scroll horizontally or vertically.

Make by

65101297 Ms. Janista Jaeyalee

65103756 Ms. Thananya Wansen

65120834 Ms. Wilasinee Sirichum