Affiliate App detailed description:

Admin View:

1.1. Admin generates a unique QR code sticker through the app.
1.2. The QR code sticker is printed and distributed to affiliates.
1.3. Admin configures the commission percentage for each purchase made using the QR code.
1.4. Admin manages affiliate accounts, resolves errors, customer complaints, and tracks affiliate sales through the admin dashboard.
1.5. Admin can batch payments to affiliates and initiate payouts.

Affiliate View:

2.1. Affiliate signs up for an account in the app using OTP (One-Time Password) authentication.

- Can be done by twilio or firebase authentication system.
- Flutter firebase authentication help: https://www.youtube.com/watch?v=rWamixHIKmQ , https://www.youtube.com/watch?v=u8H652UY-L8

2.2. Affiliate links their account to a unique printed QR code by scanning it using the app.

- Generate a Unique QR Code: When an affiliate signs up for an account, generate a unique QR code associated with their account. You can use a library like qr_flutter in Flutter to generate QR codes based on unique identifiers (e.g., affiliate ID).

- Print the QR Code:Provide an option for the affiliate to print the generated QR code.

- The affiliate can then distribute the printed QR code on various marketing materials.

- Implement QR Code Scanning:In your Flutter app, implement a screen or functionality where affiliates can scan the printed QR code. Use a QR code scanning library like flutter_barcode_scanner or qr_code_scanner to capture the QR code's content. Associate QR Code with Affiliate Account:Once the QR code is scanned, obtain the content (e.g., affiliate ID) from the QR code. Send this information to your backend server for verification and association with the affiliate's account.The backend server should store the association between the QR code and the affiliate account in the database.

2.3. Affiliates track their earnings through the app and can view the history of their commissions.
-Update Affiliate's Account: On the frontend, update the affiliate's account in your Flutter app to reflect the successful association with the QR code. You can update a flag or store a reference to the associated QR code in the affiliate's account details.

2.4. Affiliates enter their banking details in the app to receive payouts.

- Payment method.

2.5. Affiliate receives email notifications for each commission earned, payout details, and any important updates.

-1. Sign up for an Email Service Provider:
    - Sign up for an account with an ESP that offers an API for sending emails programmatically. SendGrid is one such provider, but you can choose any provider that meets your requirements.
    - Obtain the necessary API credentials (API key, SMTP settings, etc.) from the ESP.
2. Set Up Flutter Email Package:
    - In your Flutter app's **pubspec.yaml** file, add the email package dependency. For example, if you choose SendGrid, you can use the **flutter_sendgrid** package.
    - Run **flutter pub get** to fetch the package and update your project.
3. Implement Email Notification Functionality:
    - In your Flutter app, create a service or utility class to handle email notifications.
    - Use the SendGrid Flutter SDK (or the SDK provided by your chosen ESP) to send emails.
    - Implement functions to send different types of email notifications, such as commission earned, payout details, and important updates.
    - Customize the email templates and content based on your requirements.
4. Integrate Email Notifications in the App:
    - Identify the appropriate triggers in your app to send email notifications, such as when a commission is earned or a payout is initiated.
    - Call the respective email notification function from your service or utility class when these triggers occur.
    - Pass the necessary data (e.g., commission details, payout information) to the email notification function.
5. Testing and Error Handling:
    - Test the email notification functionality thoroughly to ensure emails are sent correctly.
    - Implement proper error handling and logging to handle failures during the email sending process.
    - Monitor and troubleshoot any issues related to email delivery and adjust your implementation if needed.

Customer Purchase Tracking:

3.1. Customer makes a purchase on the Shopify storefront.
3.2. Shopify storefront includes a field where the customer can enter the unique code from the QR sticker during checkout.
3.3. Shopify sends a purchase confirmation to the app's backend, including the unique code used in the purchase.
3.4. App's backend processes the purchase confirmation and identifies the affiliate associated with the unique code.
3.5. App's backend calculates the commission based on the configured percentage and updates the affiliate's earnings.

1. Integration with Shopify Storefront:
    - Integrate your app with the Shopify API to access order data and receive notifications about new purchases made on the Shopify storefront.
    - Follow Shopify's API documentation to set up authentication and establish a connection with the Shopify store.
2. Unique Code Field in Shopify Checkout:
    - Modify the Shopify storefront's checkout page to include a field where customers can enter the unique code from the QR code sticker.
    - Customize the Shopify checkout process to capture the unique code along with other order information.
3. Order Notification to the App's Backend:
    - Set up a webhook or polling mechanism to receive order notifications from Shopify whenever a purchase is made.
    - Shopify provides webhooks that can send real-time notifications to your app's backend when specific events occur, such as a new order placement.
    - Configure the webhook endpoint URL to receive the order data in a format suitable for your app's backend.
4. Processing Order Data in the App's Backend:
    - Receive the order data from Shopify and extract the relevant information, including the unique code entered by the customer during checkout.
    - Validate and sanitize the received data to prevent any security vulnerabilities or data integrity issues.
    - Match the unique code with the corresponding affiliate account to attribute the purchase to the correct affiliate.
5. Commission Calculation and Earnings Update:
    - Retrieve the commission percentage associated with the affiliate from the database based on the unique code used.
    - Calculate the commission amount based on the total purchase value and the commission percentage.
    - Update the affiliate's earnings in the database by adding the calculated commission amount to their existing earnings.
    - Ensure proper error handling and logging in case of any issues during the commission calculation or earnings update process.

6. Email Notifications to Affiliates:
    - Send email notifications to the affiliate whenever a purchase is made using their unique code.
    - Include details such as the purchase amount, commission earned, and any other relevant information in the email.
    - Personalize the email with the affiliate's name and other appropriate details to provide a clear and transparent notification.
7. Reporting and Analytics:
    - Store the order and commission data in a structured format for reporting and analytics purposes.
    - Generate reports or provide a dashboard where affiliates and admins can view sales, commissions, and other relevant statistics.
    - Implement filters, sorting, and search functionality in the reporting interface to enable users to explore and analyze the data effectively.

Payouts:

4.1. App's backend tracks the affiliate's earnings and checks if they have reached the minimum threshold for a payout.
4.2. If the earnings exceed the minimum threshold, the app's backend initiates a payout to the affiliate's provided banking details.
4.3. App's backend updates the payout status in the affiliate's account and sends an email notification regarding the payout.
4.4. If the earnings do not reach the minimum threshold, the affiliate's earnings carry over to the next payout cycle.