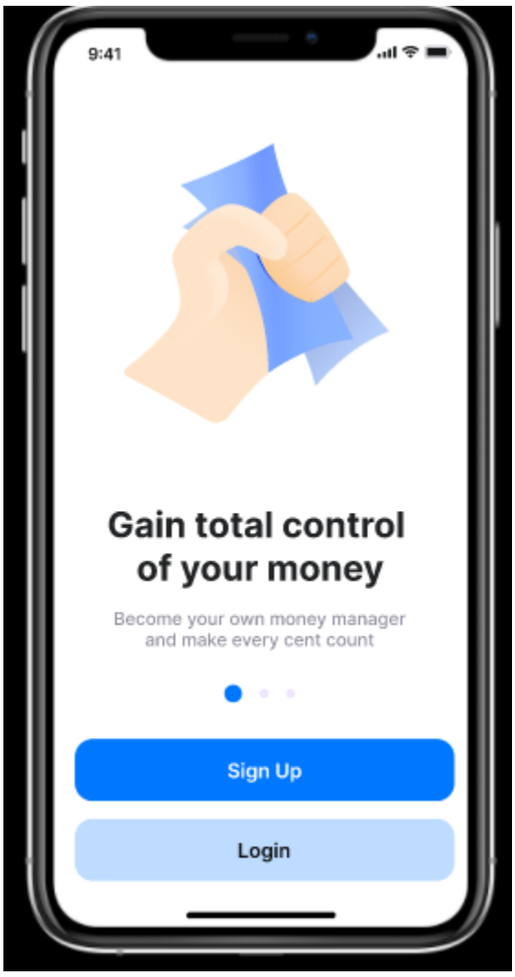
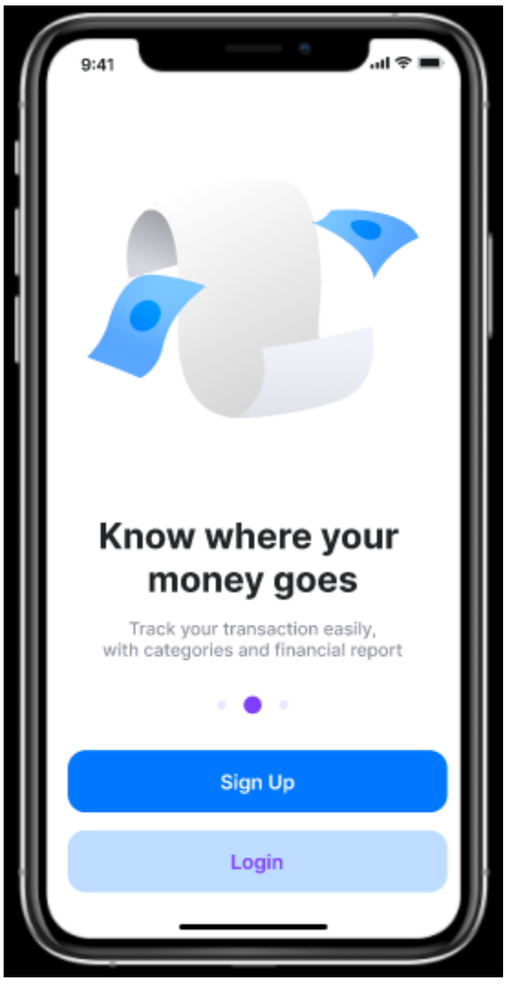
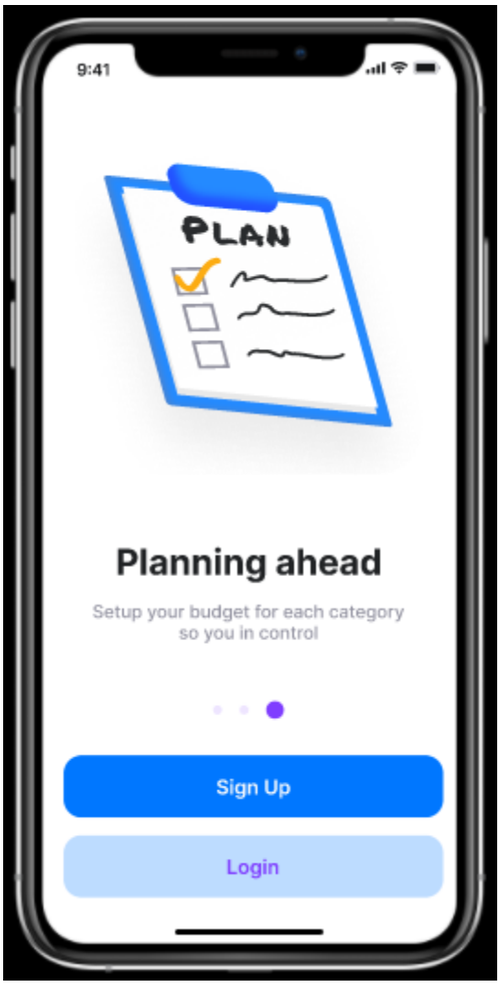
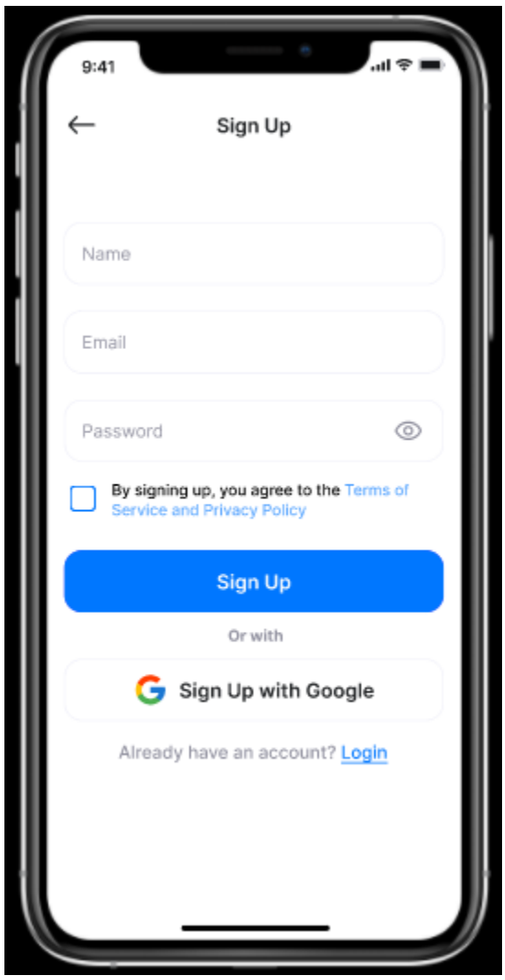
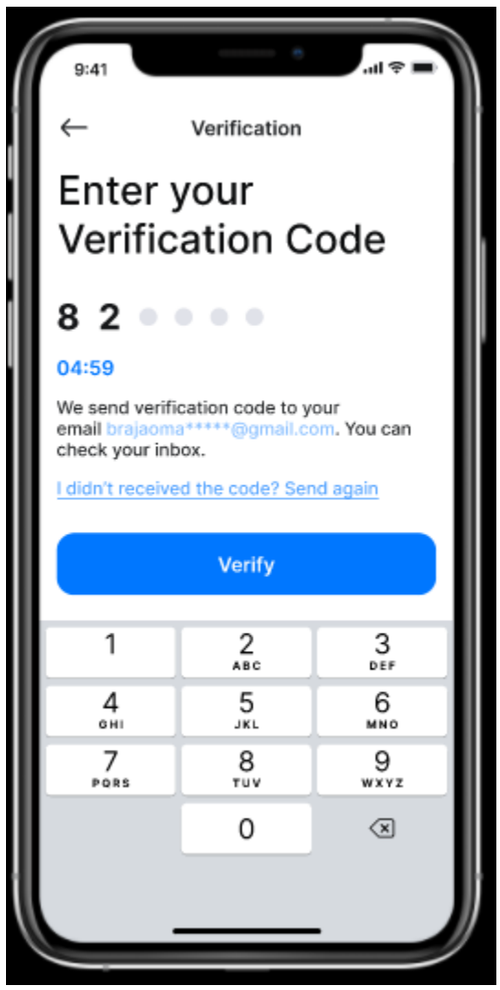

9:41
Monet UI Framework
Monet

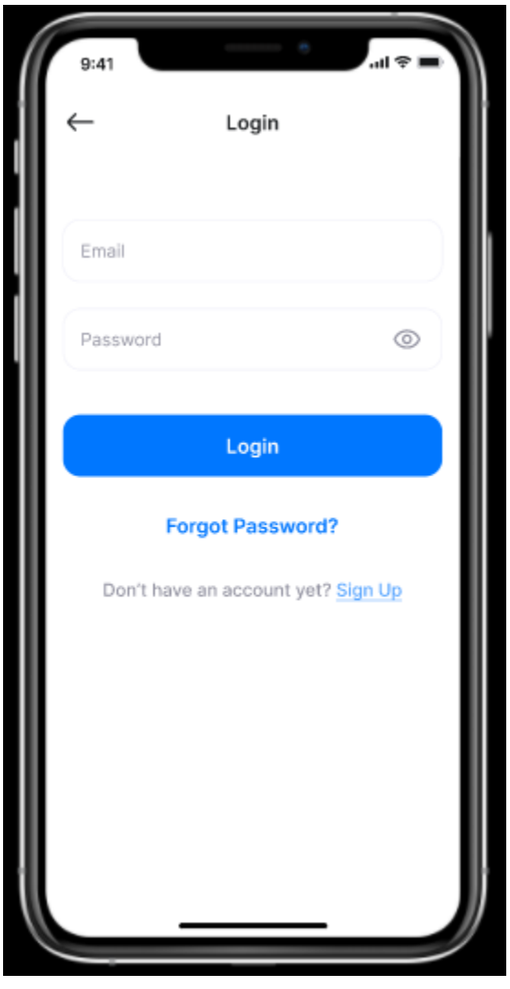

9:41
Gain total control of your money
Become your own money manager and make every cent count
Sign Up
Login

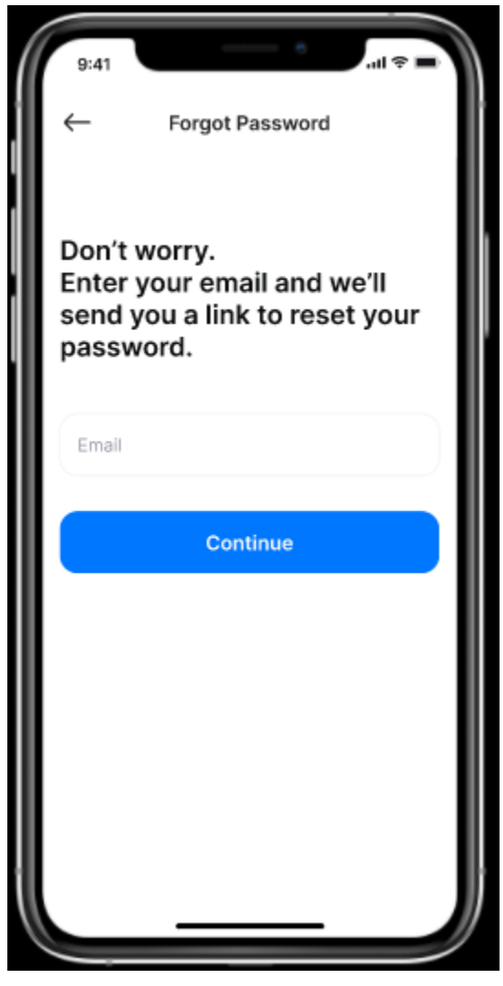

9:41
Know where your money goes
Track your transaction easily, with categories and financial report
Sign Up
Login

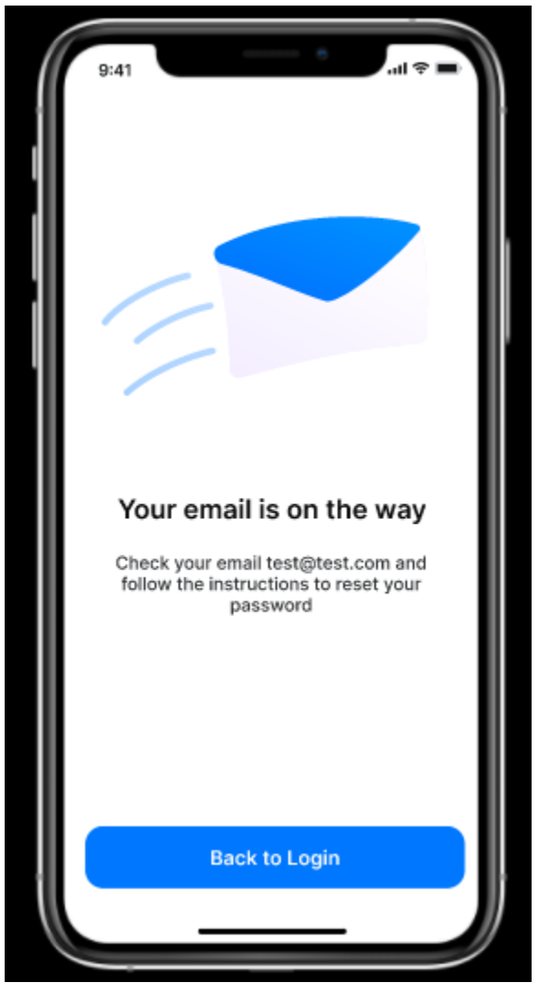

9:41
PLAN
Planning ahead
Setup your budget for each category so you in control
Sign Up
Login

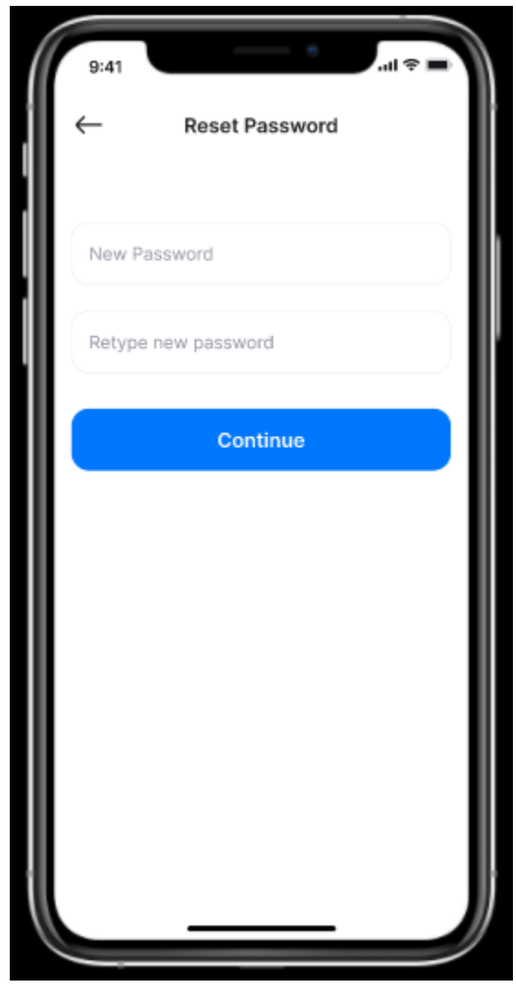

9:41
Sign Up
Name
Email
Password
By signing up, you agree to the Terms of Service and Privacy Policy
Sign Up
Or with
Sign Up with Google
Already have an account? Login

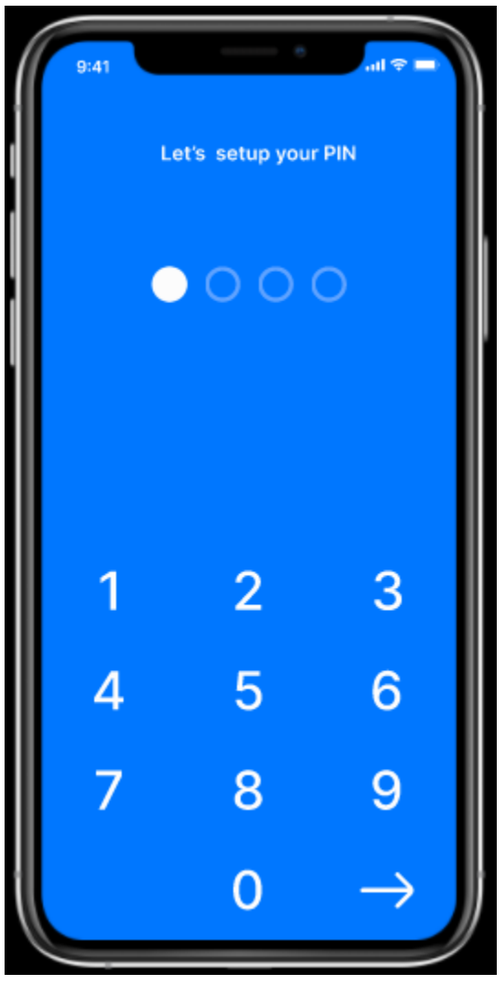

9:41
Verification
Enter your Verification Code
8 2
04:59
We send verification code to your email brajaoma*****@gmail.com. You can check your inbox.
I didn't received the code? Send again
Verify
1
2 ABC
3 DEF
4 GHI
5 JKL
6 MNO
7 PQRS
8 TUV
9 WXYZ
0


9:41
Login
Email
Password
Login
Forgot Password?
Don't have an account yet? Sign Up

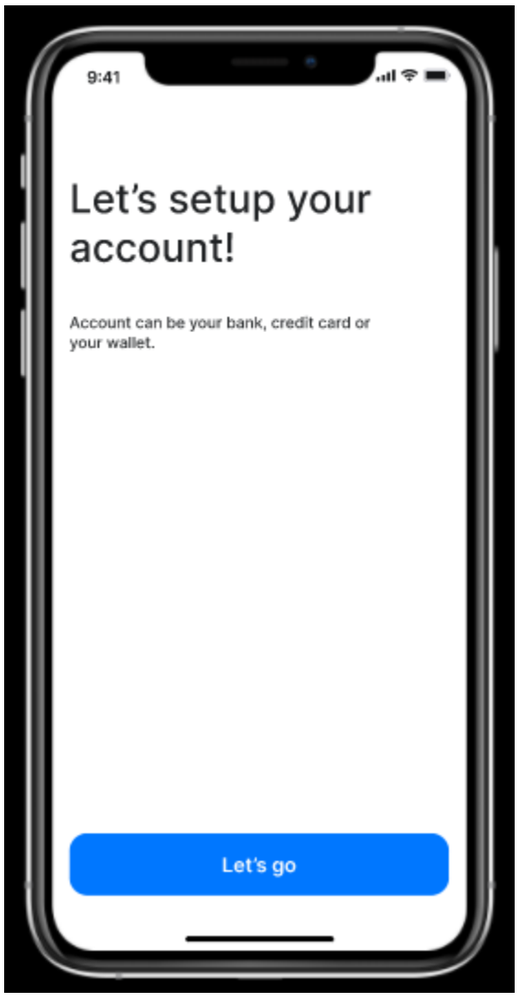

9:41
Forgot Password
Don't worry.
Enter your email and we'll send you a link to reset your password.
Email
Continue

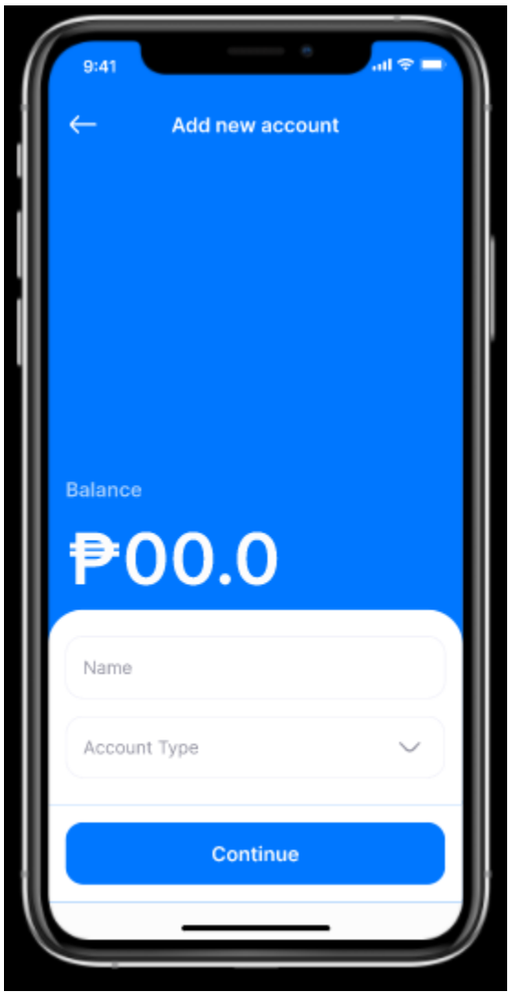

9:41
Your email is on the way
Check your email test@test.com and follow the instructions to reset your password
Back to Login

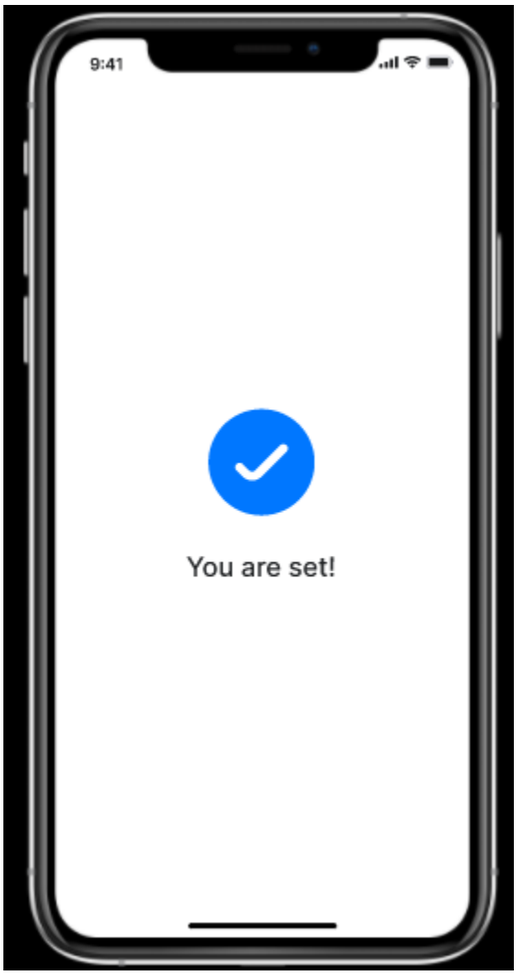

9:41
Reset Password
New Password
Retype new password
Continue

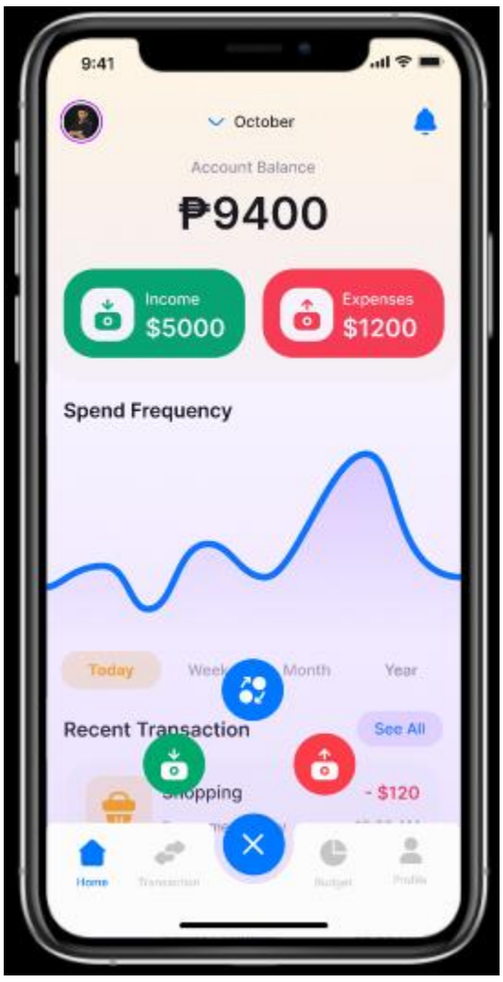

9:41
Let's setup your PIN
1 2 3
4 5 6
7 8 9
0

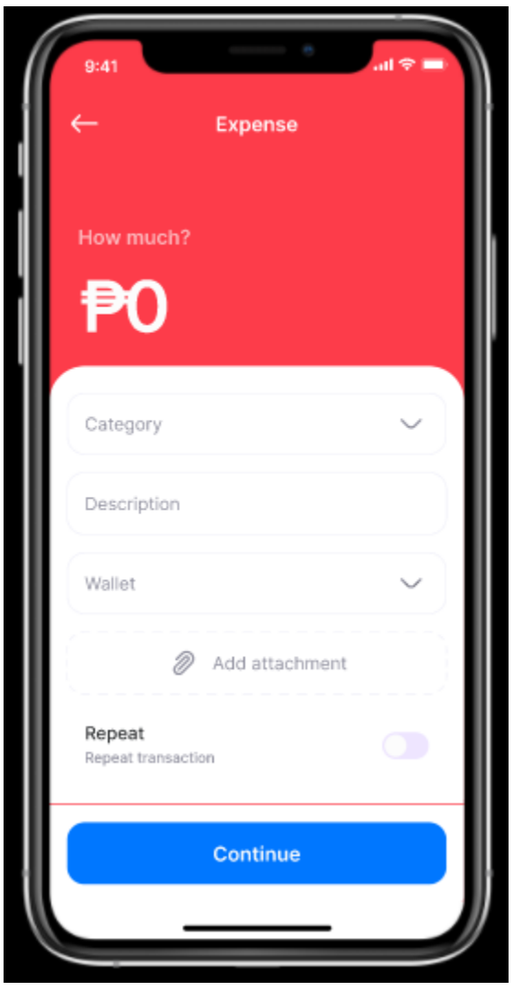

9:41
Ok. Re type your PIN again.
1 2 3
4 5 6
7 8 9
0

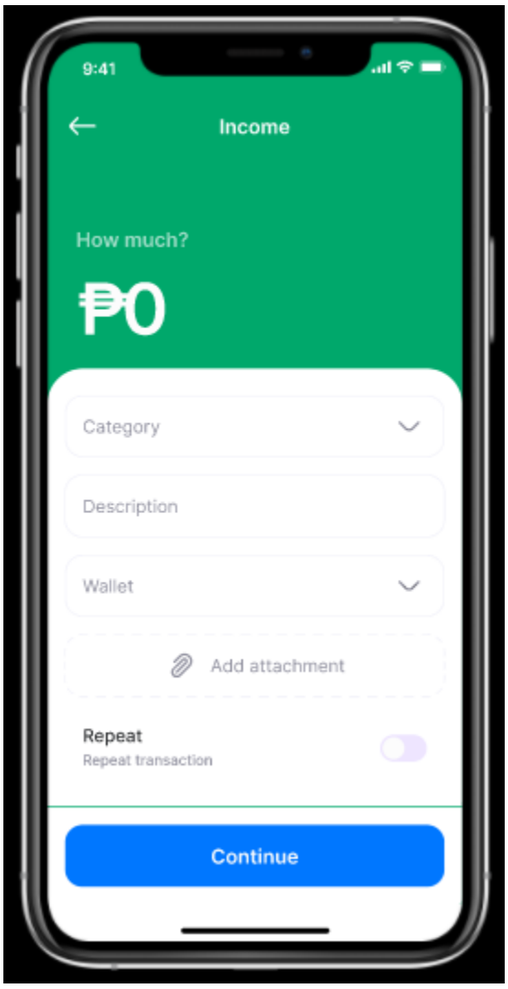

9:41
Let's setup your account!
Account can be your bank, credit card or your wallet.
Let's go

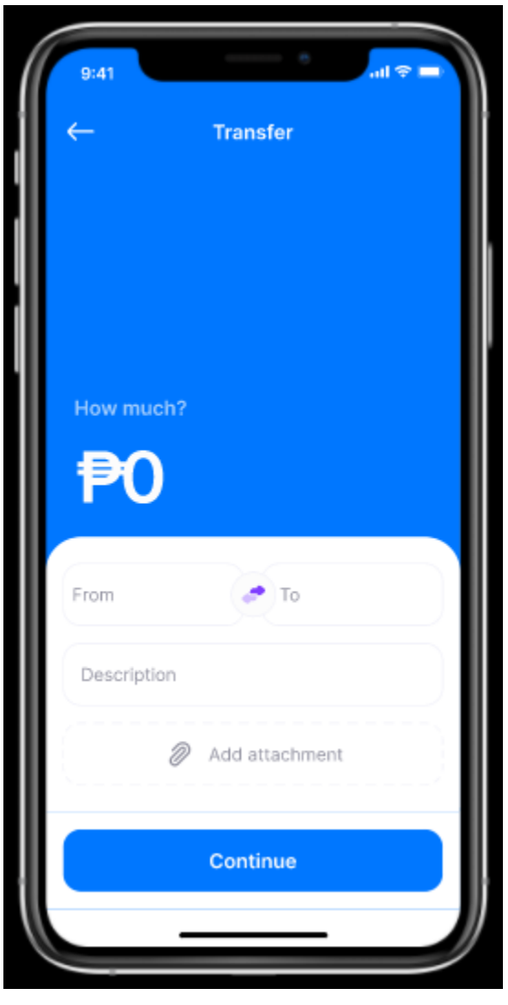

9:41
Add new account
Balance
₱00.0
Name
Account Type
Continue


9:41
You are set!

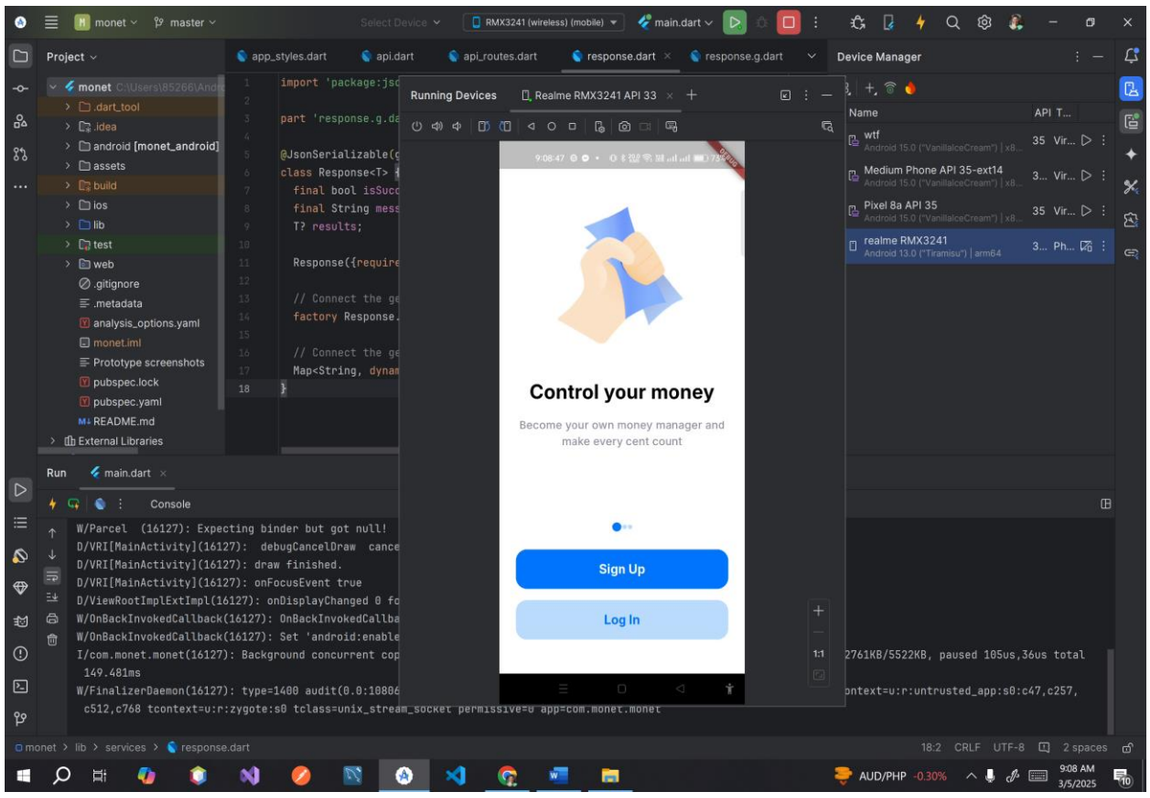

9:41
October
Account Balance
₱9400
Income
$5000
Expenses
$1200
Spend Frequency
Today
Week
Month
Year
Recent Transaction
See All
- $120
Home

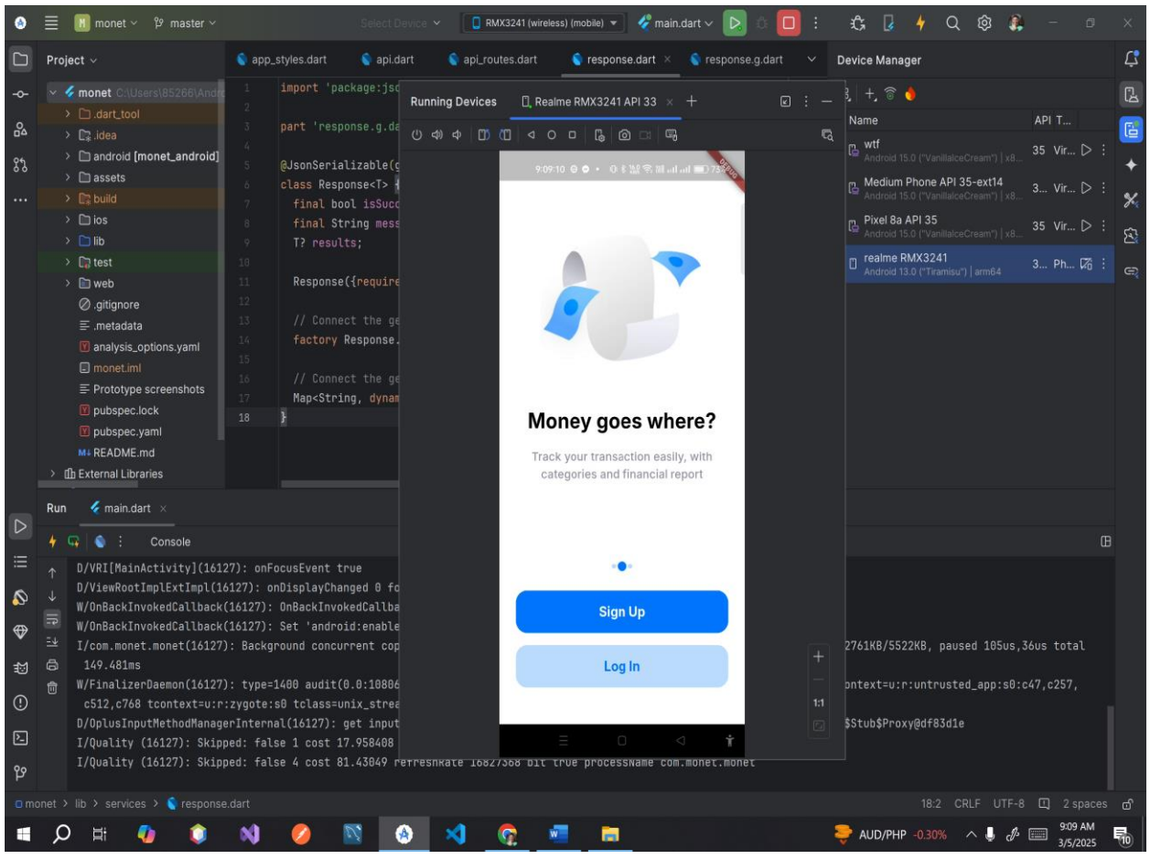

9:41
Expense
How much?
₱0
Category
Description
Wallet
Add attachment
Repeat
Repeat transaction
Continue

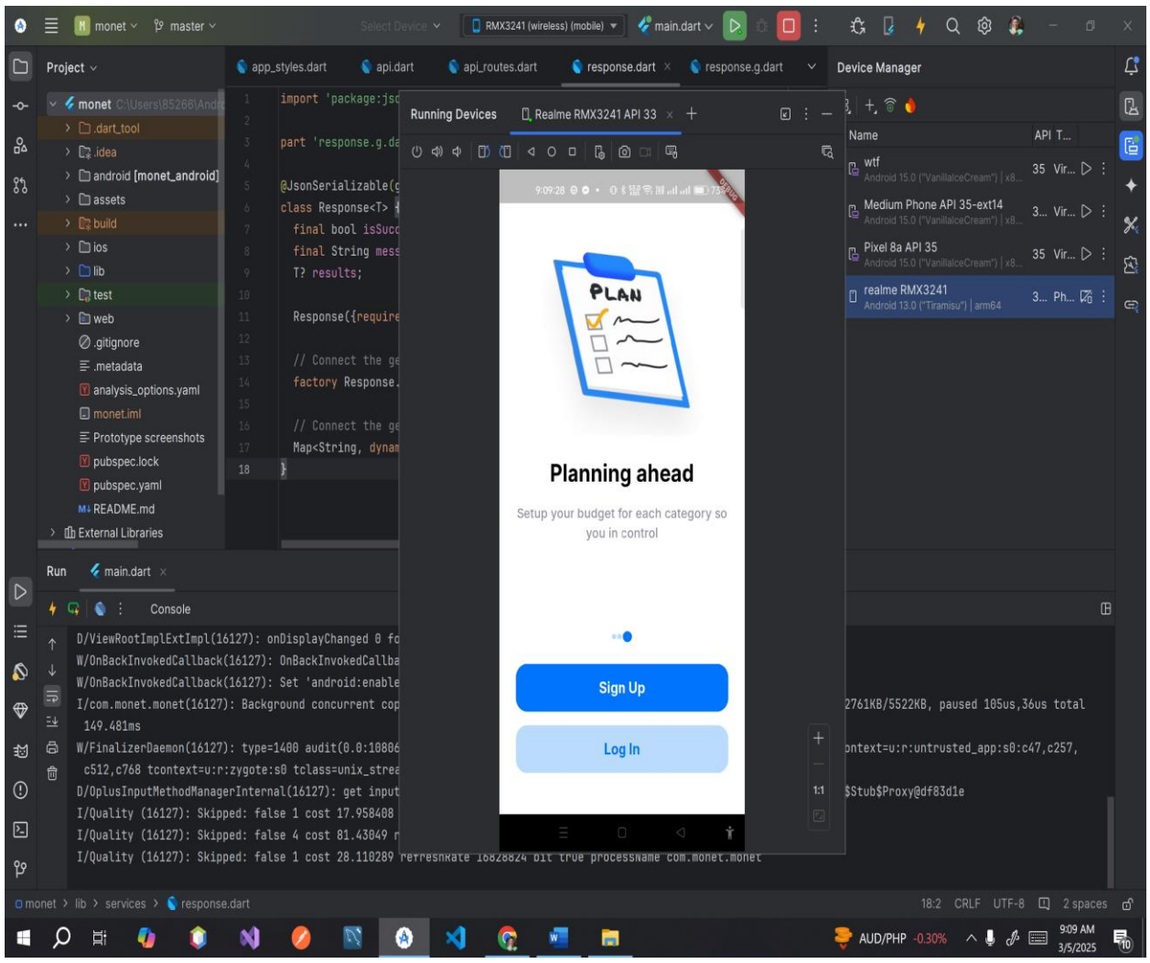

9:41
Income
How much?
₱0
Category
Description
Wallet
Add attachment
Repeat
Repeat transaction
Continue

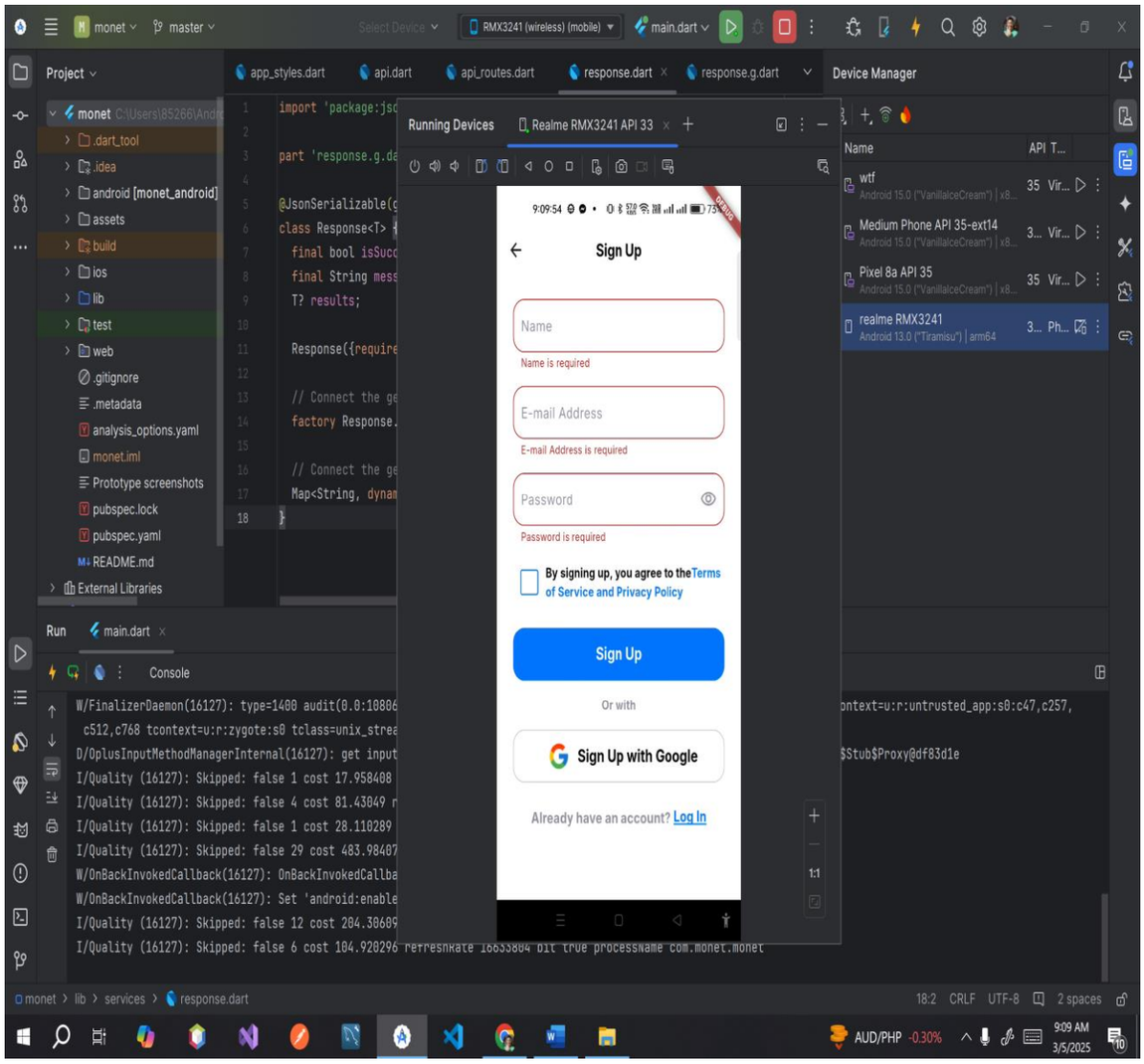

9:41
Transfer
How much?
₱0
From
To
Description
Add attachment
Continue

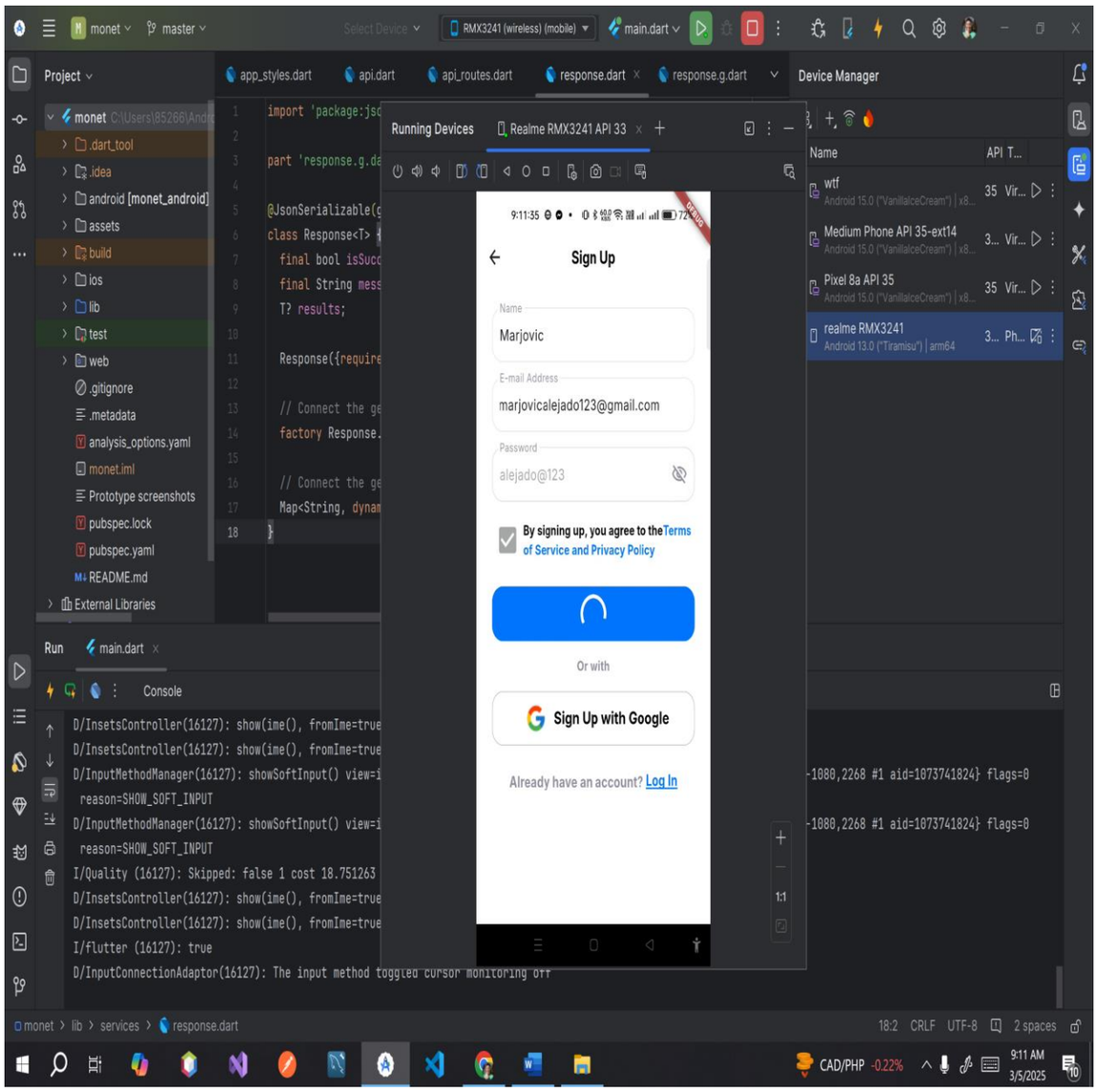
Sign up / Log In System Implementation





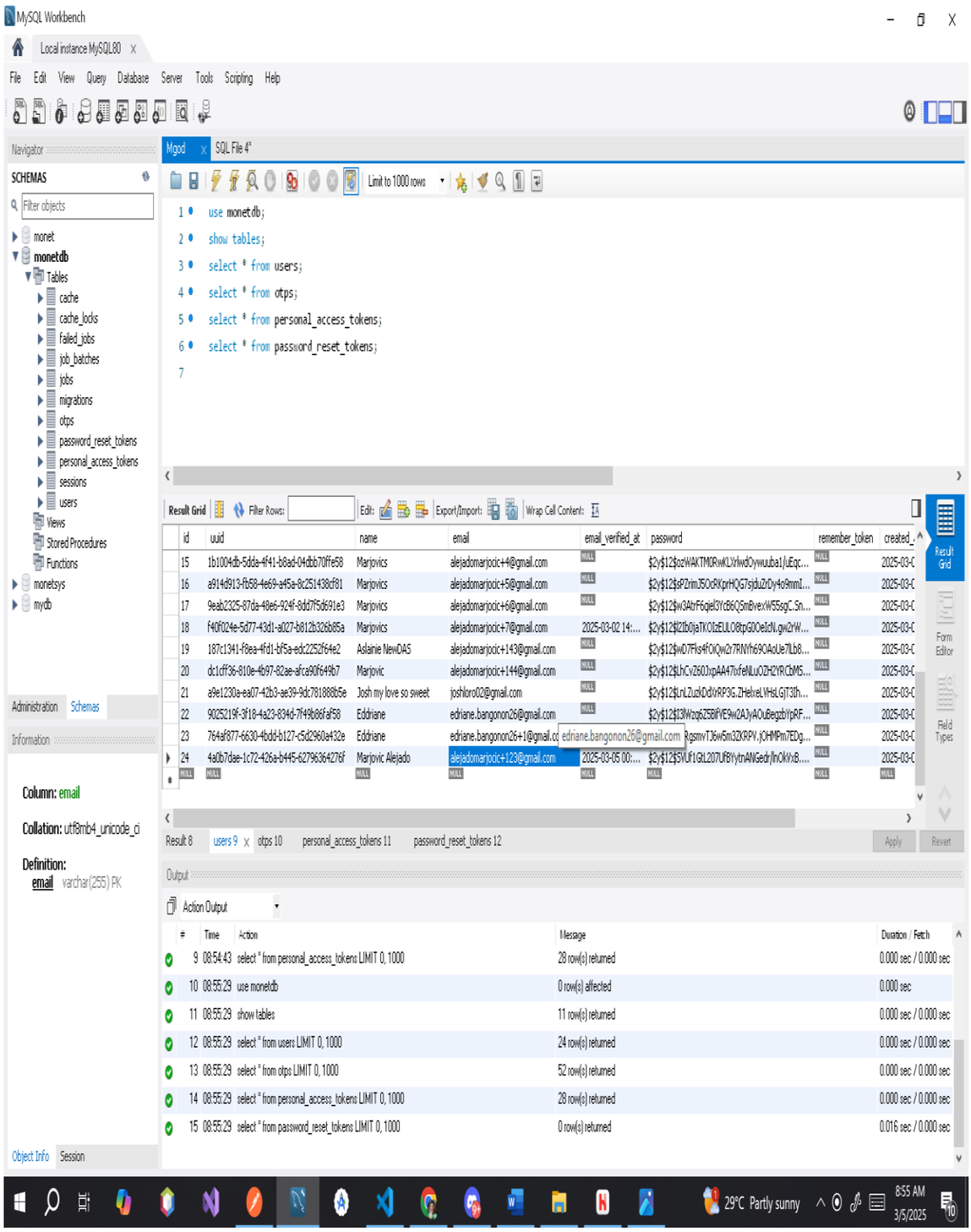

Running Devices
Realme RMX3241 API 33
Planning ahead
Setup your budget for each category so you in control
Sign Up
Log In


Running Devices
Realme RMX3241 API 33
Sign Up
Name
Name is required
E-mail Address
E-mail Address is required
Password
Password is required
By signing up, you agree to the Terms of Service and Privacy Policy
Sign Up
Or with
Sign Up with Google
Already have an account? Log In

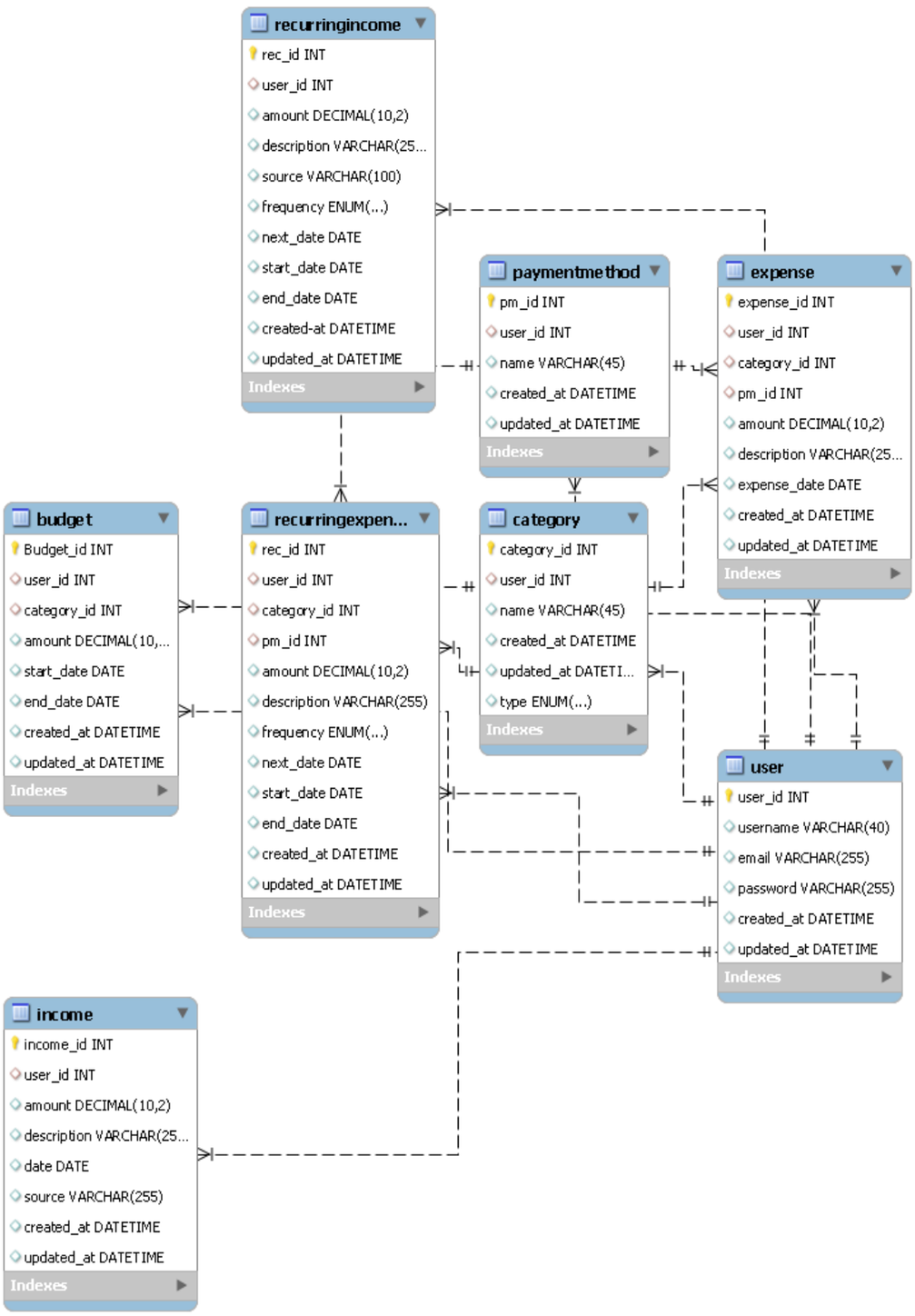

Running Devices
Realme RMX3241 API 33
Sign Up
Name
Marjovic
E-mail Address
marjovicalejado123@gmail.com
Password
alejado@123
By signing up, you agree to the Terms of Service and Privacy Policy
Or with
Sign Up with Google
Already have an account? Log In
Device Manager
Name
API T...
wtf
Medium Phone API 35-ext14
Pixel 8a API 35
realme RMX3241
Run
main.dart
Console
D/InsetsController(16127): show(ime(), fromIme=true
D/InputMethodManager(16127): showSoftInput() view=i
reason=SHOW_SOFT_INPUT
I/Quality (16127): Skipped: false 1 cost 18.751263
I/flutter (16127): true
D/InputConnectionAdaptor(16127): The input method toggled cursor monitoring off
monet › lib › services › response.dart
18:2 CRLF UTF-8 2 spaces
CAD/PHP -0.22%
9:11 AM
3/5/2025

Dashboard Implementation – ps. we were working for it

MYSQL Database

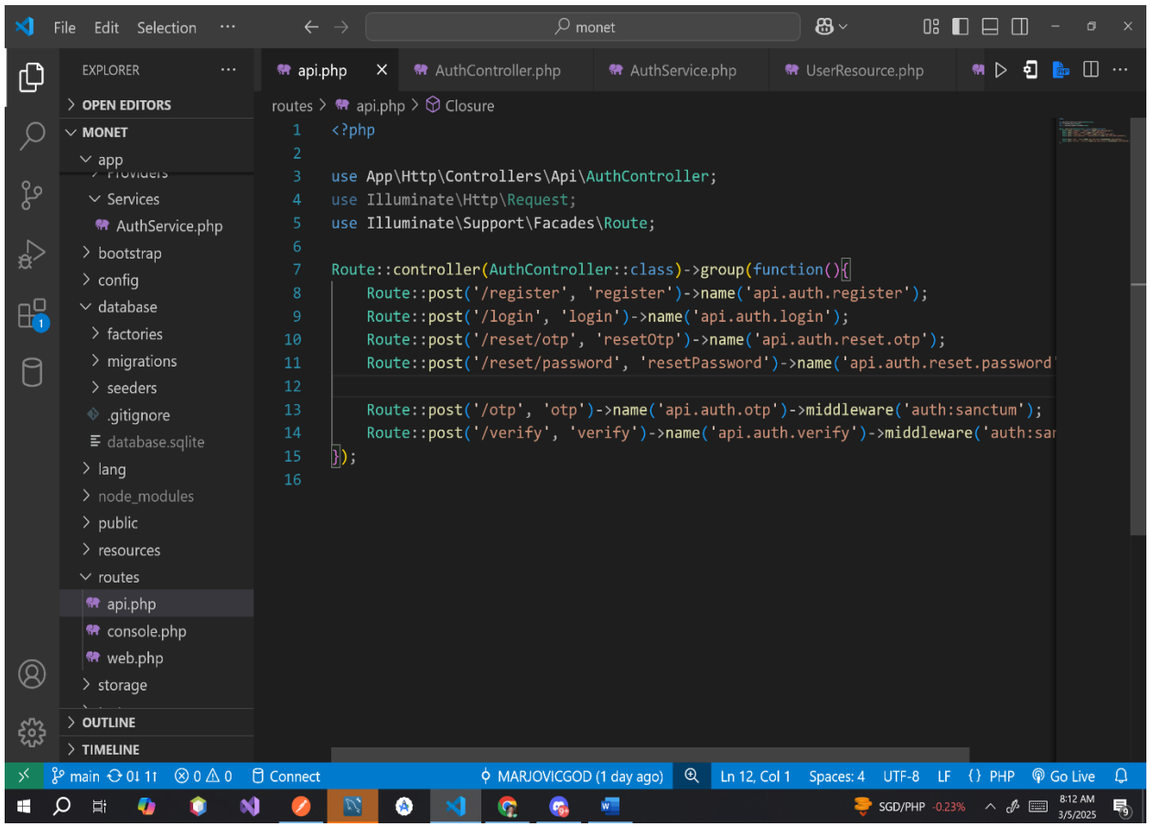


Monet Schema Diagram



Code Snippets

routes/api



Auth_Controller/Auth_Service

Register

package.json | api.php | AuthController.php | AuthService.php | Use

app > Services > AuthService.php > AuthService > register

```php
<?php

namespace App\Services;

use App\Models\Otp;
use App\Models\User;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Str;
use Nette\Utils\Random;
use Illuminate\Support\Facades\Mail;
use App\Mail\OtpMail;
use Carbon\Carbon;

class AuthService {
    public function register(object $request) : User {

        // Create user
        $user = User::create([
            'uuid' => Str::uuid(),
            'name' => $request->name,
            'email' => $request->email,
            'password' => $request->password,
        ]);

        $this->otp($user);

        return $user;
    }

    public function login(object $request) : ?User
    {

        // Create user
        $user = User::where('email', $request->email)->first();
        if($user && Hash::check($request->password, $user->password)) {
            return $user;
        }
        return null;
    }
```

Log in

```php
    public function login(Request $request) : Response {
        $request->validate([
            'email' => 'required|email|max:255',
            'password' => 'required|min:6|max:255',
        ]);
        // Login user
        $user = $this->authService->login($request);
        if(!$user) {
            return response([
                'message' => __('auth.failed'),
            ], 401);
        }
        // Create Access token
        $token = $user->createToken('auth')->plainTextToken;
        // Return
        return response([
            'message' => $user->email_verified_at ? __('app.login_success') : __('app.login_success_verify'),
            'result' => [
                'user' => new UserResource($user),
                'token' => $token
            ]
        ], 200);
    }

```

```php
    public function login(object $request) : ?User
    {

        // Create user
        $user = User::where('email', $request->email)->first();
        if($user && Hash::check($request->password, $user->password)) {
            return $user;
        }
        return null;
    }

```

Otp

```php
    public function otp(Request $request) : Response {
        /** @var User $user */
        $user = Auth::user();

        /** @var Otp $otp */
        $otp = $this->authService->otp($user);

        return response([
            'message' => __('app.otp_sent'),
        ]);
    }
```

```php
    public function otp(User $user, string $type = 'verification') : Otp {

        $tries = 3;
        $time = Carbon::now()->subMinutes(30);

        $count = Otp::where([
            'user_id' => $user->id,
            'type' => $type,
            'active' => 1
        ])->where('created_at', '>=', $time)->count();

        if($count >= $tries) {
            abort(422, __('app.to_many_requests'));
        }

        $code = random_int(100000, 999999);

        $otp = Otp::create([
            'user_id' => $user->id,
            'type' => $type,
            'code' => $code,
            'active' => 1
        ]);
```

Verify

```php
    public function verify(Request $request) : Response {

        // Validate the request
        $request->validate([
            'otp' => 'required|numeric',
        ]);

        /** @var User $user */
        $user = Auth::user();

        // Verify the OTP
        /** @var Otp $otp */
        $user = $this->authService->verify($user, $request);

        return response([
            'message' => __('app.verification_success'),
            'result' => [
                'user' => new UserResource($user)
            ]
        ]);
    }
```

```php
    public function verify(User $user, object $request) : User {
        $otp = Otp::where([
            'user_id' => $user->id,
            'code' => $request->otp,
            'active' => 1
        ])->first();

        if(!$otp) {
            abort(422, __('app.invalid_otp'));
        }

        // Update OTP
        $user->email_verified_at = Carbon::now();
        $user->update();

        $otp->active = 0;
        $otp->updated_at = Carbon::now();
        $otp->update();

        return $user;
    }
```

Reset using OTP with Password

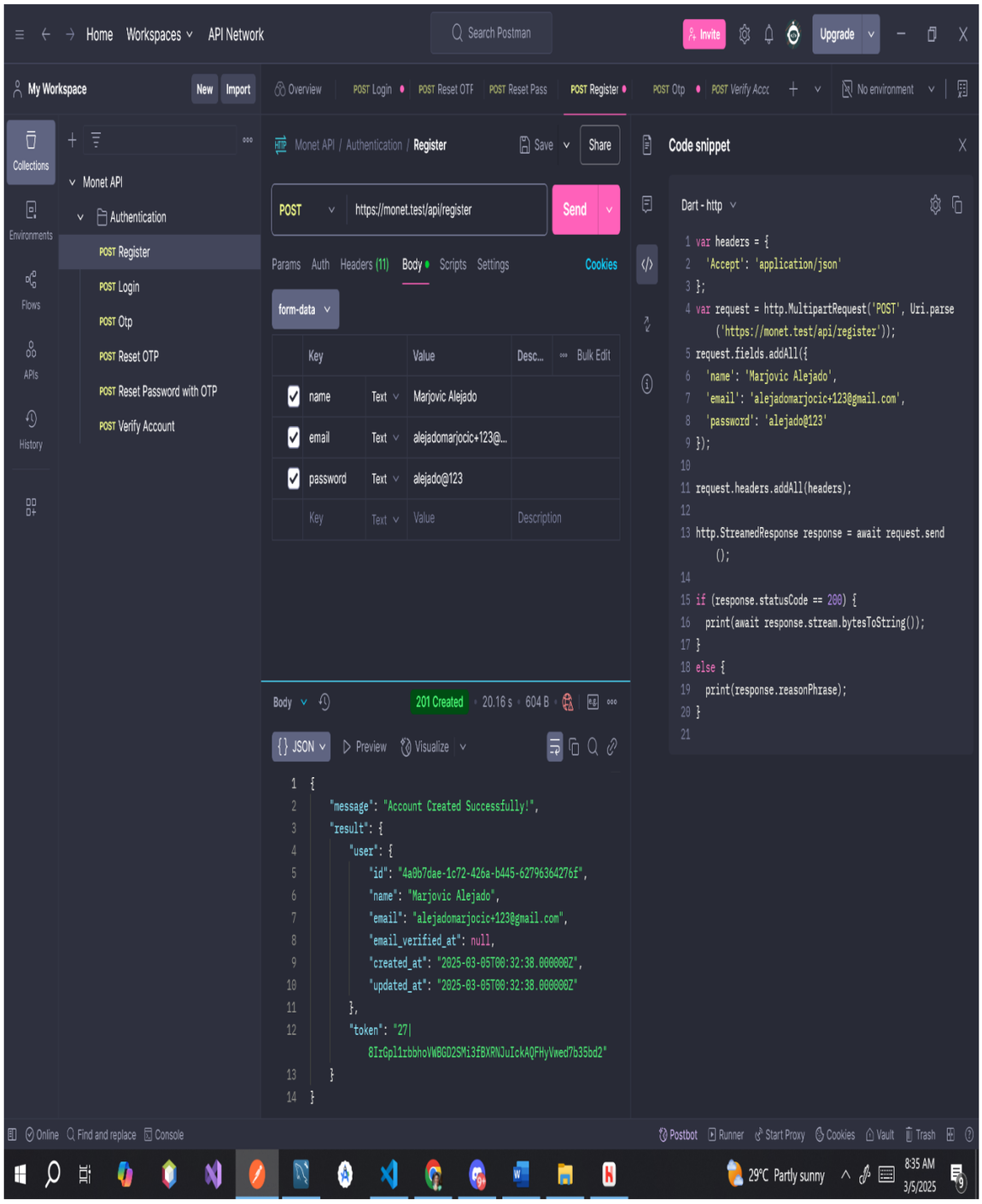
```php
    public function resetOtp(Request $request) : Response {

        // Validate request
        $request->validate([
            'email' => 'required|email|max:255|exists:users,email',
        ]);

        // Get user
        $user = $this->authService->getUserByEmail($request->email);

        /** @var Otp $otp */
        $otp = $this->authService->otp($user, 'password-reset');

        return response([
            'message' => __('app.otp_sent'),
        ]);
    }

```

```php
    public function resetPassword(Request $request) : Response {

        // Validate request
        $request->validate([
            'email' => 'required|email|max:255|exists:users,email',
            'otp' => 'required|numeric',
            'password' => 'required|min:6|max:255|confirmed',
            'password_confirmation' => 'required|min:6|max:255',

        ]);

        // Get user
        $user = $this->authService->getUserByEmail($request->email);

        $user = $this->authService->resetPassword($user, $request);

        return response([
            'message' => __('app.password_reset_success'),
        ]);
    }
```

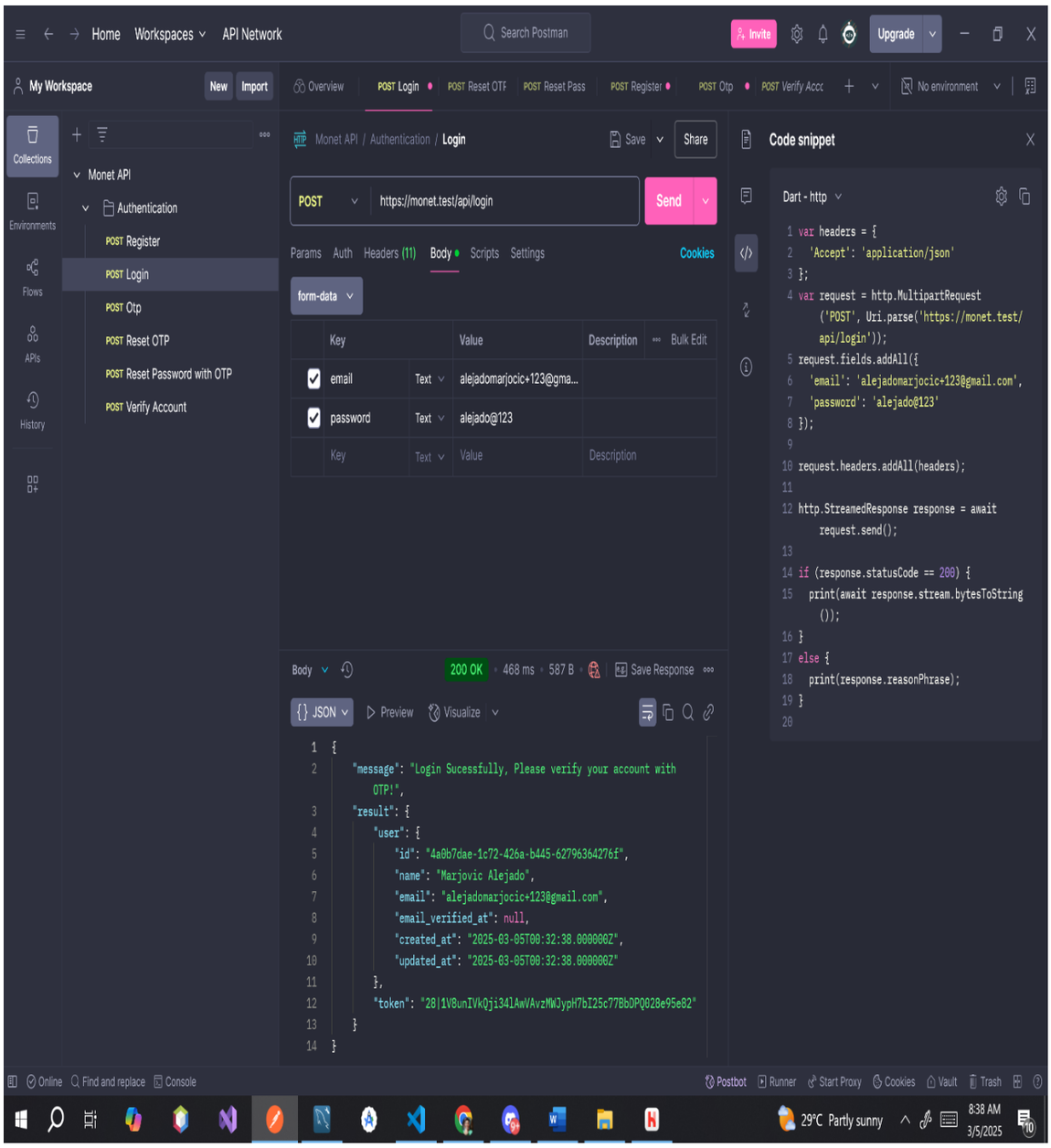
```php
    public function getUserByEmail(string $email): User
    {
        return User::where('email', $email)->first();
    }

    public function resetPassword(User $user, object $request): User
    {
        $otp = Otp::where([
            'user_id' => $user->id,
            'code' => $request->otp,
            'active' => 1,
            'type' => 'password-reset'
        ])->first();

        if (!$otp) {
            abort(422, __('app.invalid_otp'));
        }

        // Update
        $user->password = $request->password;
        $user->updated_at = Carbon::now();
        $user->update();

        $otp->active = 0;
        $otp->updated_at = Carbon::now();
        $otp->update();

        return $user;
    }
}
```

API

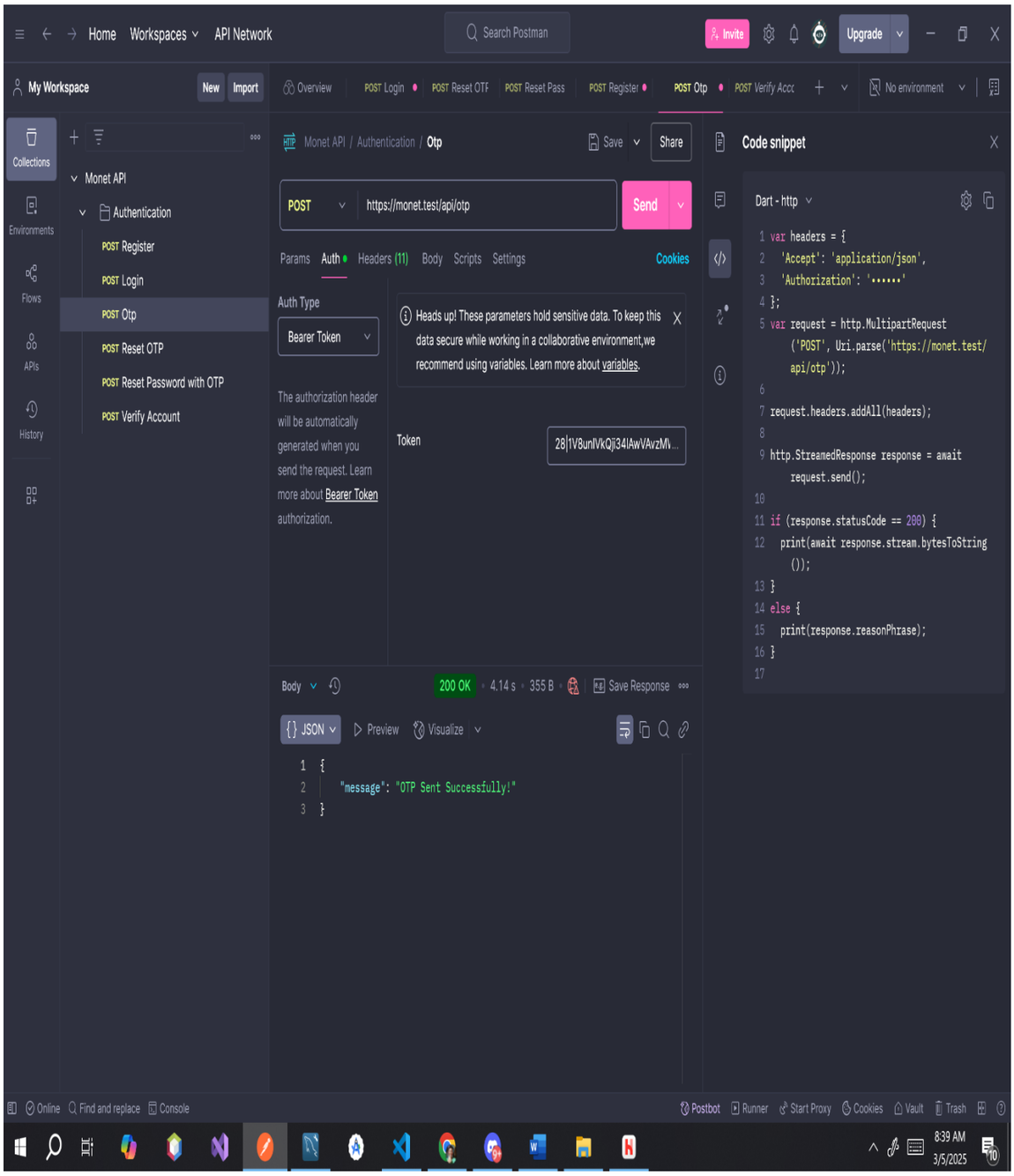
Register

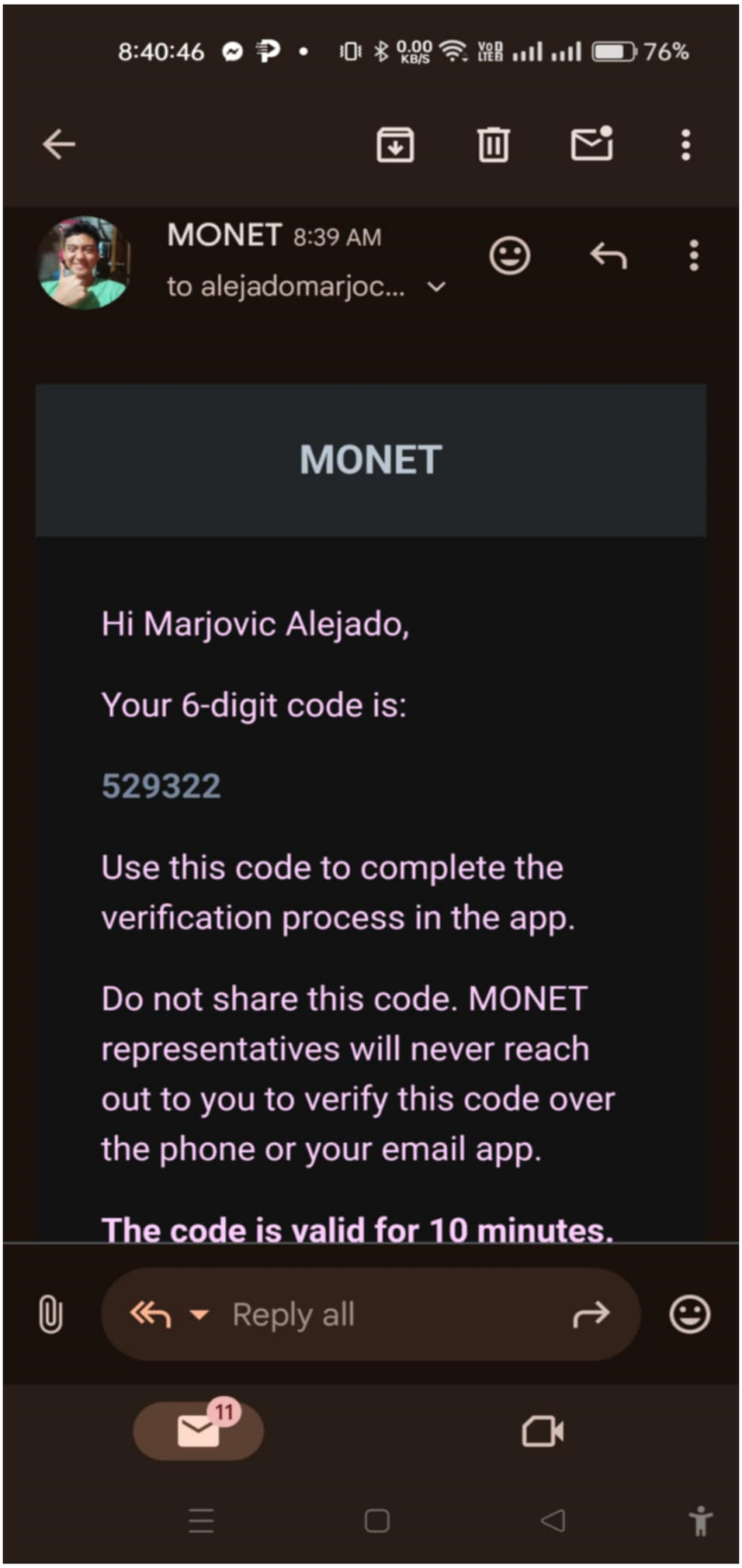


Log In

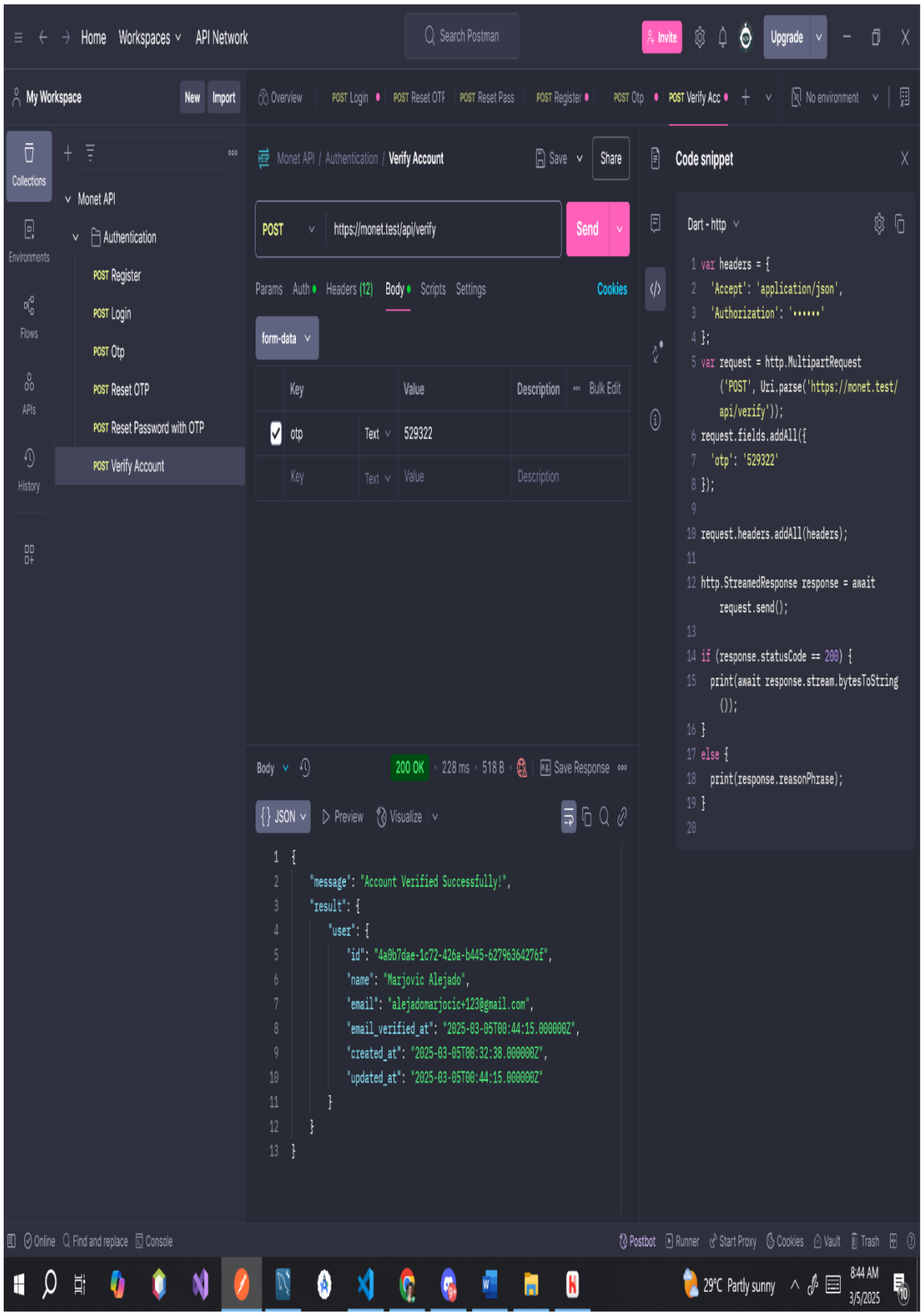

Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
Overview
POST Login
POST Reset OTP
POST Reset Pass
POST Register
POST Otp
POST Verify Acc
No environment
Collections
Environments
Flows
APIs
History
Monet API
Authentication
POST Register
POST Login
POST Otp
POST Reset OTP
POST Reset Password with OTP
POST Verify Account
Monet API / Authentication / Login
Save
Share
POST
https://monet.test/api/login
Send
Params
Auth
Headers (11)
Body
Scripts
Settings
Cookies
form-data
Key
Value
Description
Bulk Edit
email
Text
alejadomarjocic+123@gma...
password
alejado@123
Code snippet
Dart - http
var headers = {
'Accept': 'application/json'
};
var request = http.MultipartRequest
('POST', Uri.parse('https://monet.test/
api/login'));
request.fields.addAll({
'email': 'alejadomarjocic+123@gmail.com',
'password': 'alejado@123'
});
request.headers.addAll(headers);
http.StreamedResponse response = await
request.send();
if (response.statusCode == 200) {
print(await response.stream.bytesToString
());
}
else {
print(response.reasonPhrase);
}
Body
200 OK
468 ms
587 B
Save Response
JSON
Preview
Visualize
{
"message": "Login Sucessfully, Please verify your account with
OTP!",
"result": {
"user": {
"id": "4a0b7dae-1c72-426a-b445-627963642761",
"name": "Marjovic Alejado",
"email": "alejadomarjocic+123@gmail.com",
"email_verified_at": null,
"created_at": "2025-03-05T00:32:38.000000Z",
"updated_at": "2025-03-05T00:32:38.000000Z"
},
"token": "28|1V8unIVkQji34lAwVAvzMWJypH7bI25c77BbDPQ828e95e82"
}
}
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Vault
Trash
29°C Partly sunny
8:38 AM
3/5/2025

Request OTP

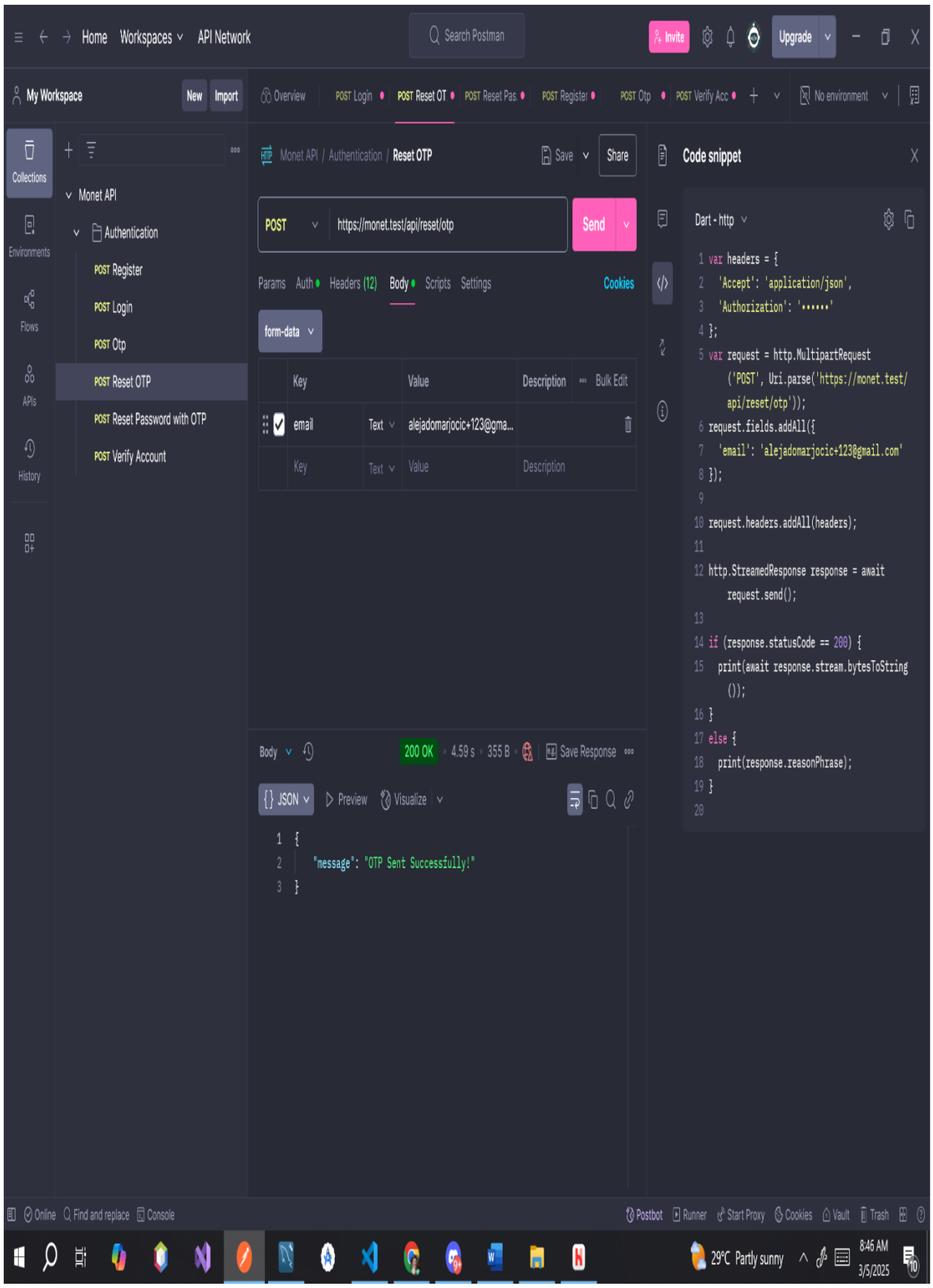

Home
Workspaces
API Network
Search Postman
Invite
Upgrade
My Workspace
New
Import
Overview
POST Login
POST Reset OTP
POST Reset Pass
POST Register
POST Otp
POST Verify Acc
No environment
Collections
Environments
Flows
APIs
History
Monet API
Authentication
POST Register
POST Login
POST Otp
POST Reset OTP
POST Reset Password with OTP
POST Verify Account
Monet API / Authentication / Otp
Save
Share
POST
https://monet.test/api/otp
Send
Params
Auth
Headers (11)
Body
Scripts
Settings
Cookies
Auth Type
Bearer Token
Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment,we recommend using variables. Learn more about variables.
The authorization header will be automatically generated when you send the request. Learn more about Bearer Token authorization.
Token
28|1V8unlVkQjj34IAwVAvzM\...
Code snippet
Dart - http
1 var headers = {
2 'Accept': 'application/json',
3 'Authorization': '••••••'
4 };
5 var request = http.MultipartRequest
('POST', Uri.parse('https://monet.test/
api/otp'));
6
7 request.headers.addAll(headers);
8
9 http.StreamedResponse response = await
request.send();
10
11 if (response.statusCode == 200) {
12 print(await response.stream.bytesToString
());
13 }
14 else {
15 print(response.reasonPhrase);
16 }
17
Body
200 OK
4.14 s
355 B
Save Response
JSON
Preview
Visualize
1 {
2 "message": "OTP Sent Successfully!"
3 }
Online
Find and replace
Console
Postbot
Runner
Start Proxy
Cookies
Vault
Trash
8:39 AM
3/5/2025


8:40:46
76%
MONET 8:39 AM
to alejadomarjoc...
MONET
Hi Marjovic Alejado,
Your 6-digit code is:
529322
Use this code to complete the verification process in the app.
Do not share this code. MONET representatives will never reach out to you to verify this code over the phone or your email app.
The code is valid for 10 minutes.
Reply all
11

Verify email using OTP



Request OTP for change password



8:48:09 • 0.05 KB/S 75%

MONET 8:46 AM
to alejadomarjoc...

MONET

Hi Marjovic Alejado,

Your 6-digit code is:

502247

Use this code to reset your password in the app.

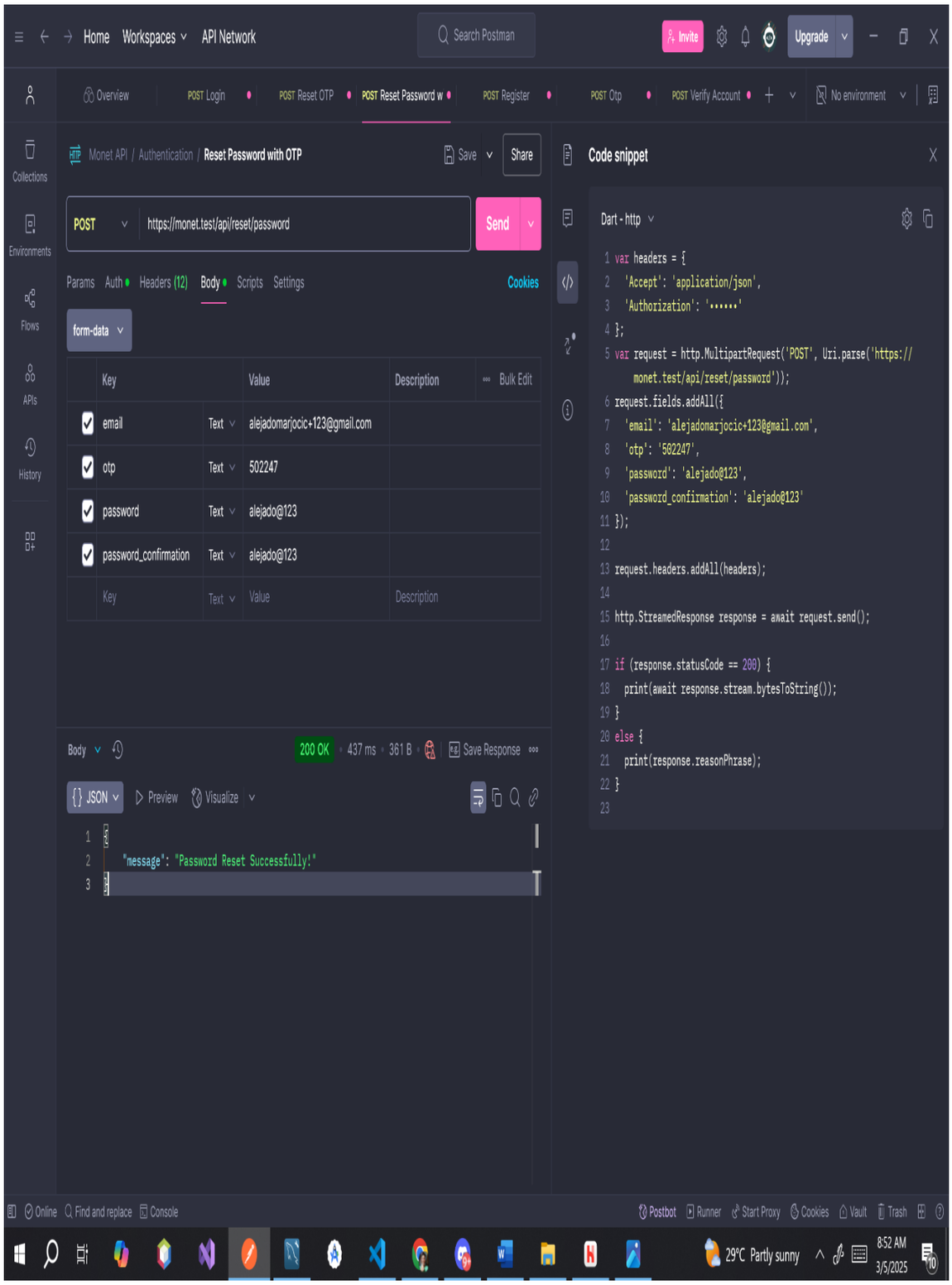
Do not share this code. MONET representatives will never reach out to you to verify this code over the phone or your email app.

The code is valid for 10 minutes.

Thanks

Reply all

Reset Password with OTP



Challenges and Solutions Applied

Developing a **Money Expense Tracker System** using **Laravel** for the backend, **Flutter** for the UI, and **GitHub** for version control can be a rewarding but challenging project.

Below are some common challenges you might face and solutions to fix them:

1. Backend Development Challenges (Laravel)

Challenges:

- **Complex Database Relationships:** Managing connections between tables (like users, expenses, and categories) can be hard.
- **API Development:** Making RESTful APIs that are safe, can grow, and are easy to understand.
- **Validation and Error Handling:** Checking user inputs properly and showing errors in a nice way.
- **Authentication and Authorization:** Setting up secure logins and controlling what users can do.
- **Performance Optimization:** Handling lots of data and making database searches fast.

Solutions:

- Use Laravel's Eloquent ORM to set up and manage relationships (like hasMany and belongsTo).
- Use Laravel's API resources and controllers to organize your APIs. Test them with tools like Postman.
- Use Laravel's built-in rules to check inputs and make custom error messages.
- Use Laravel Passport or Sanctum for secure logins and JWT tokens.
- Make things faster with Laravel's query builder, caching (like Redis), and database indexing.

2. Frontend Development Challenges (Flutter)

Challenges:

- **State Management:** Keeping track of the app's data (like expenses and user settings) efficiently.
- **UI/UX Design:** Building an interface that’s easy to use and works on all devices.
- **API Integration:** Connecting the Flutter app to the Laravel backend and handling responses.
- **Cross-Platform Compatibility:** Making sure the app works well on Android and iOS.
- **Data Synchronization:** Storing data offline and updating the backend when online.

Solutions:

- Use tools like Provider, Riverpod, or Bloc to manage the app’s data.
- Use Flutter’s widget library and tools like FlutterFlow to design the interface. Follow Material Design rules.
- Use the http or dio package to call APIs and manage responses. Add loading and error screens.
- Test the app on Android and iOS devices or emulators. Add special code if needed.
- Use Hive or SQLite to save data offline and sync it with the backend when online.

3. Version Control Challenges (GitHub)

Challenges:

- **Branch Management:** Handling different code versions for features, fixes, and releases.
- **Merge Conflicts:** Fixing problems when combining code changes.
- **Collaboration:** Working with a team and following good practices.
- **CI/CD Integration:** Setting up automatic testing and deployment.

Solutions:

- Use GitFlow or a similar plan. Name branches clearly (like feature/add-expense or bugfix/login-issue).
- Pull updates often and fix conflicts early. Use GitHub's tools to help.
- Use GitHub Issues and Pull Requests to track work and review code. Set coding rules for everyone.
- Set up GitHub Actions to test and deploy code automatically.

4. Integration Challenges

Challenges:

- **Backend-Frontend Communication:** Making sure the backend and frontend work together smoothly.
- **Data Consistency:** Keeping data the same in the app and on the server.
- **Security:** Protecting private info (like passwords and expenses) during transfer and storage.

Solutions:

- Use JSON for API responses and keep the data structure the same on both sides.
- Handle errors and check data on both ends. Use timestamps or versions for updates.
- Use HTTPS for APIs, encrypt sensitive data, and add secure login methods.

5. Testing and Debugging

Challenges:

- **Backend Testing:** Making sure the Laravel backend has no bugs and works with many users.
- **Frontend Testing:** Testing the Flutter app for ease of use, speed, and device support.
- **End-to-End Testing:** Checking the whole system to ensure everything works together.

Solutions:

- Use PHPUnit to test Laravel's units and features. Test APIs with Postman.
- Use Flutter's testing tools for unit, widget, and integration tests.
- Do manual and automatic tests to check the full system.

6. Deployment Challenges

Challenges:

- **Backend Deployment:** Putting the Laravel app on a server (like AWS, DigitalOcean, or Heroku).
- **Frontend Deployment:** Publishing the Flutter app to Google Play Store and Apple App Store.
- **Environment Configuration:** Managing different setups (like development, staging, and production).

Solutions:

- Use tools like Laravel Forge or Envoyer to deploy easily. Set up .env files for each environment.
- Follow Flutter's steps to build and release the app on Android and iOS.
- Use environment variables and separate config files for each setup.

7. Documentation and Maintenance

Challenges:

- **Code Documentation:** Writing clear code with good explanations.
- **User Documentation:** Making guides and tutorials for app users.
- **Maintenance:** Updating the app to fix bugs and add features.

Solutions:

- Use PHPDoc for Laravel and DartDoc for Flutter. Add helpful comments and README files.
- Create a user manual or in-app tutorials with Flutter.

- Plan regular updates and track issues with GitHub Issues.