



Paytm Project 11 of 20

# Building any project

Akash Kadlag OP Today at 9:18 PM

Hi @kiratsingh,

Super Excited to build End-To-End App with pretty much everything we learnt in last 16 weeks.

I already DM you the Notion Docs screenshots about Features of Projects.  
But Here I wanted to share something important...

There are many videos available about build XYZ.  
Even you also have Code With Me type Videos.  
These videos are good.

The problem starts when we try to build this kind of app by ourself.  
Here are few challenges

1. Where to Start...?
2. Design UI/UX
3. Low Level Design
4. High Level Design
5. ER Diagram
6. Tech Stack
7. How to think about new Features.
8. How much complexity is needed..?

[Back to home](#)[Jump To](#)[Prev](#)[Next](#)[Go to Top](#)

## 1. Where to start - Feature planning



Paytm Project 11 of 20

1. UX - First principles/Copy the biggest website out there
2. UI - Designer. Today there are tools but havent found any good one
3. High level Design
  1. Auth provider
  2. Database
  3. Backend Stack
  4. Frontend stack
  5. Modules you'll have (common/ui/backend)
  6. Cloud to deploy to
4. LLD
  1. Schema
  2. Route signatures
  3. Frontend Components - debatable
5. ER Diagrams -

u're a very

## 6. How to think about features

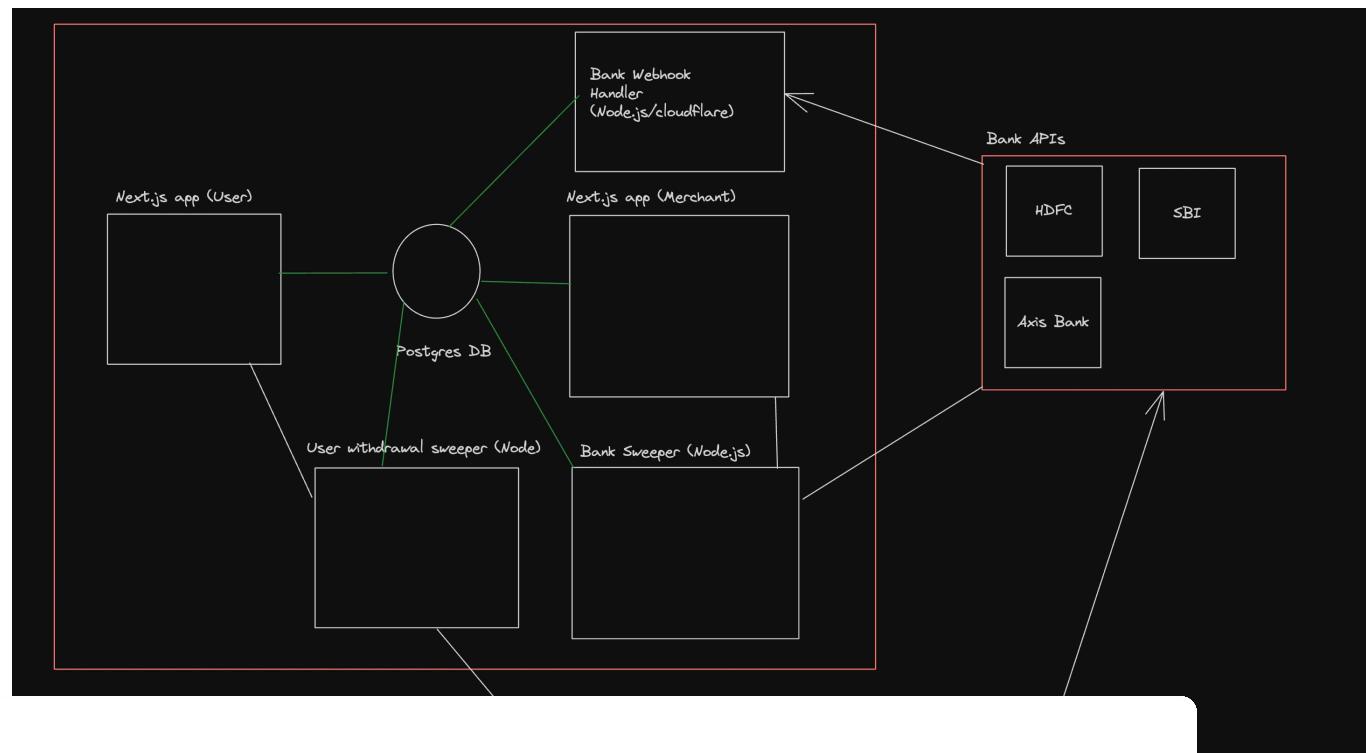


Paytm Project 11 of 20  
me from product

2. If you're a founder, then just whatever u think is right

## 7. How much complexity is needed

1. Depends on the size of the company. For a startup, whatever helps you move fast w/o tech debt. For a company there are a lot of layers of review to go through





Paytm Project 11 of 20

a voice speaker that says Paytm Received 100 rs from  
e need to add another ws service in here

We'll also be doing DB polling in the middle which should rather be  
done via a queue

# Feature planning

## User login

1. Auth (In this case, probably email/phone)
2. On ramp from bank, off ramp to bank
3. Support transfers via phone number/name
4. Support scanning a QR code for transferring to merchants

### i. Login with google

2. Generate a QR Code for acceptance



Paytm Project

11

of 20

Get an alert/notification on payment

4. Merchant gets money offramped to bank every 2 days

# UI/UX (End User)

## Login



## Paytm Project 11 of 20

The screenshot shows the Venmo login interface. At the top center is the "venmo" logo. Below it is the word "Log in". Underneath is a text input field with the placeholder "Enter email, mobile, or username". Below the input field is a blue rounded rectangular button with the word "Next" in white. Underneath the "Next" button is another blue rounded rectangular button with the words "Sign up" in white.

The screenshot shows the Venmo login interface. At the top center is the "venmo" logo. Below it is the word "Log in". Underneath is a text input field containing the email address "harkirat@gmail.com" with a "Change" link next to it. To the right of the input field is a small circular icon with an eye symbol. Below the input field is a blue rounded rectangular button with the word "Log in" in white. Underneath the "Log in" button is another blue rounded rectangular button with the words "Sign up" in white. At the bottom left of the screen, there is a link "Forgot password?".



## Paytm Project 11 of 20

The screenshot shows the Venmo homepage. At the top, there's a navigation bar with a profile icon, the text "Paytm Project 11 of 20", and a search icon. Below the navigation is the Venmo logo. In the center, there's a large blue banner with the text "Fast, safe social payments". Below the banner, a subtext reads: "Pay, get paid, grow a business, and more. Join the tens of millions of people on Venmo." A "Get Venmo" button is located at the bottom of the banner. To the right of the banner is a photograph of three diverse young people smiling outdoors. A white callout box on the right side of the photo contains the text "You paid Trish A Picnic in the park 🍞". At the top right of the page, there are "Log in" and "Merchant login" buttons.

## User Home page

The screenshot shows the Kraken crypto exchange dashboard. At the top, there's a navigation bar with the Kraken logo, a user profile picture, and links for "Buy crypto", "Help", and "Logout". The main header reads "Paytm Project 11 of 20 afternoon, Harkirat". On the left, a sidebar has three options: "Rewards", "Transfer", and "Transactions". The central area displays a chart titled "Portfolio value" showing a flat line at \$0.00 from February 20 to March 23. Below the chart are time filters: "1W", "1M" (selected), "3M", "6M", "1Y", and "ALL". Underneath these are five buttons: "Buy" (with a plus sign), "Sell" (with a minus sign), "Convert" (with a double arrow), "Deposit" (with a downward arrow), and "Withdraw" (with an upward arrow). A callout box on the right says "Set up recurring buys" with the subtext "Schedule regular crypto purchases to balance market fluctuations" and a "Get started" button. At the bottom of the dashboard, there's a note about earning up to 10%+ APR.

## User Transfer page



## Paytm Project 11 of 20

Withdraw Transfer

Rewards

Transfer

Transactions

## Payment options

UPI Options  
Pay Directly From Your Bank AccountCredit/Debit/ATM Card  
Visa, MasterCard, Amex, RuPay And MoreBook Now Pay Later  
Tripmoney, Lazypay, Simpli, ZestMoney, ICICI, HDFCNet Banking  
All Major Banks AvailableGift Cards & e-wallets  
MMT Gift cards, Mobilwik & MoreEMI  
No Cost EMI availableGooglePay  
Pay with Google Pay

## Card Number

Enter Your Card Number Here

## Name on Card

Enter Your Name On Card

## Expiry Month &amp; Year

## Card CVV

Month Year Enter Card CVV

₹ 2,058 Due Now

PAY NOW

By continuing to pay, I understand and agree with the [privacy policy](#), the [user agreement](#) and [terms of service](#) of makemytrip.

## INR Balance

Total balance

0 BTC

Order balance

0 BTC

Staking balance

0 BTC

Available balance

0 BTC

## Funding limits

Daily Monthly

Daily deposits

\$0.00 of Unlimited USD

Daily deposit limit

Unlimited USD

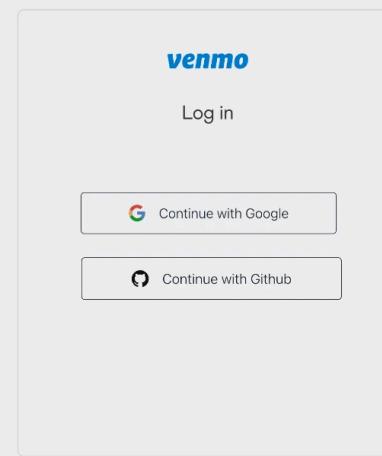
## Recent transactions

View all transactions

## UI/UX (Merchant)



## Paytm Project 11 of 20



Good afternoon, Harkirat

Portfolio value  
\$0.00

\$0.00      \$0.00

20 FEB      28 FEB      7 MAR      15 MAR      23 MAR

1W    **1M**    3M    6M    1Y    ALL

New Set up recurring buys

Schedule regular crypto purchases to balance market fluctuations.

Get started

Earn up to 10%+ APR

Paytm Project 11 of 20

actions

Explore

Rewards

Transfer

Transactions

History Scheduled

Transactions

Assets Types Start date End date Clear

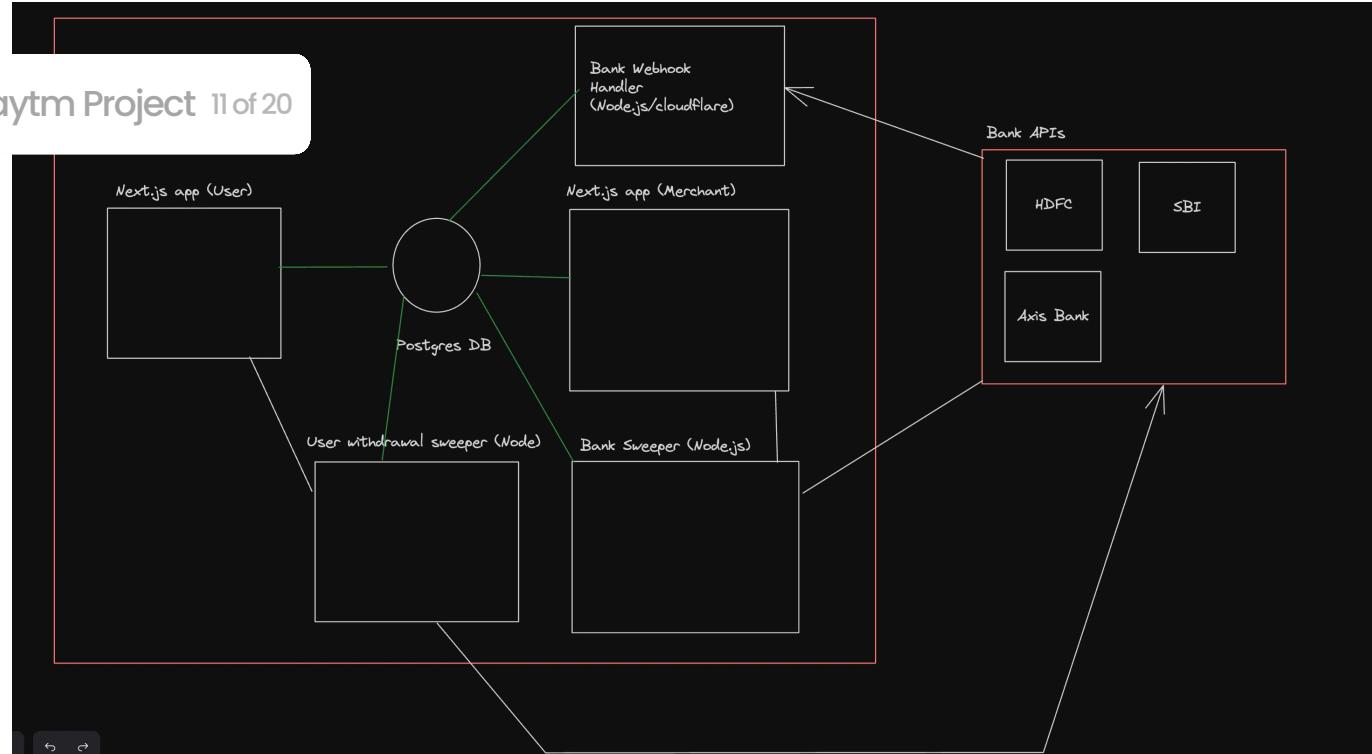
Withdrew USDC  
Feb 19, 2024, 03:18

-5,059.9477 USDC  
-\$5,060.36

# Architecture



## Paytm Project 11 of 20



## Hot paths

1. Send money to someone
2. Withdraw balance of merchant
3. Withdraw balance of user back to bank
4. Webhooks from banks to transfer in money

This is ~1 month job for a 2 engineer team.



Paytm Project 11 of 20

# Stack

1. Frontend and Backend – Next.js (or Backend)
2. Express – Auxiliary backends
3. Turborepo
4. Postgres Database
5. Prisma ORM
6. Tailwind



# Bootstrap the app

- Init turborepo

```
npx create-turbo@latest
```



- Rename the two Next apps to
  1. user-app
  2. merchant-app
- Add tailwind to it.

```
cd apps/user-app
```

```
npm install -D tailwindcss postcss autoprefixer
```

```
npx tailwindcss init -p
```



```
npm install -D tailwindcss postcss autoprefixer
```

```
css init -p
```

## Paytm Project 11 of 20



You can also use

<https://github.com/vercel/turbo/tree/main/examples/with-tailwind>

### Update tailwind.config.js

```
/** @type {import('tailwindcss').Config} */
module.exports = {
  content: [
    "./app/**/*.{js,ts,jsx,tsx,mdx}",
    "./pages/**/*.{js,ts,jsx,tsx,mdx}",
    "./components/**/*.{js,ts,jsx,tsx,mdx}",
    "../packages/ui/**/*.{js,ts,jsx,tsx,mdx}"
  ],
  theme: {
    extend: {},
  },
  plugins: [],
}
```



### Update global.css

```
@tailwind base;  
@tailwind components;
```





💡 - .. wind is working as expected  
Paytm Project 11 of 20

# Adding prisma

Ref - <https://turbo.build/repo/docs/handbook/tools/prisma>

1. Create a new **db** folder
2. Initialise package.json

```
npm init -y  
npx tsc --init
```

1. Update package.json



## Paytm Project 11 of 20

```
"version": "0.0.0",
  "dependencies": {
    "prisma": "^5.11.0"
  },
  "devDependencies": {
    "prisma": "5.11.0"
  },
  "exports": {
    "./client": "./index.ts"
  }
}
```

### 1. Update tsconfig.json

```
{
  "extends": "@repo/typescript-config/react-library.json",
  "compilerOptions": {
    "outDir": "dist"
  },
  "include": ["src"],
  "exclude": ["node_modules", "dist"]
}
```

### 1. Init prisma

```
npx prisma init
```

## 2. Update .env with the new database URL



Paytm Project 11 of 20 schema

```
model User {  
    id Int @id @default(autoincrement())  
    email String @unique  
    name String?  
}
```

## 1. Migrate DB

```
npx prisma migrate
```

## 1. Update index.ts

```
export * from '@prisma/client';
```

## 1. Try adding api/user/route.ts

```
import { NextResponse } from "next/server"  
import { PrismaClient } from "@repo/db/client";  
  
const client = new PrismaClient();  
  
export const GET = async () => {  
    await client.user.create({
```

```
name: "adsads"
```



## Paytm Project 11 of 20

```
return NextResponse.json({  
  message: "hi there"  
})  
}
```

# Add a recoil/store module

- Create `store` package

```
cd packages  
mkdir store  
npm init -y
```



- Install dependencies



## Paytm Project 11 of 20



- Update tsconfig.json

```
{  
  "extends": "@repo/typescript-config/react-library.json",  
  "compilerOptions": {  
    "outDir": "dist"  
  },  
  "include": ["src"],  
  "exclude": ["node_modules", "dist"]  
}
```



- Update package.json

```
{  
  "name": "@repo/store",  
  "version": "1.0.0",  
  "description": "",  
  "main": "index.js",  

```



"recoil": "^0.7.7"



## Paytm Project 11 of 20

- Create a simple `atom` called `balance` in `src/atoms/balance.ts`

```
import { atom } from "recoil";

export const balanceAtom = atom<number>({
  key: "balance",
  default: 0,
})
```

- Create a simple `hook` called `src/hooks/useBalance.ts`

```
import { useRecoilValue } from "recoil"
import { balanceAtom } from "../atoms/balance"

export const useBalance = () => {
  const value = useRecoilValue(balanceAtom);
  return value;
}
```

- Add export to `package.json`

```
"exports": {
  "./useBalance": "./src/hooks/useBalance"
```



# Import recoil in the next.js apps

- Install recoil in them

```
npm i recoil
```



- Add a `providers.tsx` file

```
"use client"
import { RecoilRoot } from "recoil";

export const Providers = ({children}: {children: React.ReactNode}) => {
  return <RecoilRoot>
    {children}
  </RecoilRoot>
}
```



```
</RecoilRoot>
```



## Paytm Project 11 of 20

- Update `layout.tsx`

```
return (
  <html lang="en">
    <Providers>
      <body className={inter.className}>{children}</body>
    </Providers>
  </html>
);
```

- Create a simple client component and try using the `useBalance` hook in there

```
"use client";

import { useBalance } from "@repo/store/useBalance";

export default function() {
  const balance = useBalance();
  return <div>
    hi there {balance}
  </div>
}
```

If you see this, we'll try to debug it end of class. Should still work in dev mode



## Paytm Project 11 of 20

```
1  use client,
2
3  import { useBalance } from "@repo/store/balance";
4
5  export default function() {
6    const balance = useBalance();
7    return <div>
8      | hi there {balance}
9    </div>
```

{} package.json .../store U {} package.json .../ui ts page.tsx .../merchant-app/... 1, U  
Cannot find module '@repo/store/balance' or its corresponding type declarations. ts(2307)

[View Problem \(F8\)](#) No quick fixes available

# Add next-auth

## Database

Update the database schema

```
generator client {
  provider = "prisma-client-js"
```

```
datasource db {
```



## Paytm Project 11 of 20

```
provider = "postgresql"
("DATABASE_URL")
```

```
model User {
    id      Int    @id @default(autoincrement())
    email   String? @unique
    name    String?
    number  String @unique
    password String
}
```

```
model Merchant {
    id      Int    @id @default(autoincrement())
    email   String @unique
    name    String?
    auth_type AuthType
}
```

```
enum AuthType {
    Google
    Github
}
```

## User-app





## Paytm Project 11 of 20

- Initialize a simple next auth config in `lib/auth.ts`

```
import db from "@repo/db/client";
import CredentialsProvider from "next-auth/providers/credentials"
import bcrypt from "bcrypt";

export const authOptions = {
  providers: [
    CredentialsProvider({
      name: 'Credentials',
      credentials: {
        phone: { label: "Phone number", type: "text", placeholder: "1231231231" },
        password: { label: "Password", type: "password" }
      },
      // TODO: User credentials type from next-auth
      async authorize(credentials: any) {
        // Do zod validation, OTP validation here
        const hashedPassword = await bcrypt.hash(credentials.password, 10)
        const existingUser = await db.user.findFirst({
          where: {
            number: credentials.phone
          }
        });
        if (!existingUser) {
          throw new Error("User not found")
        }
        const passwordValidation = await bcrypt.compare(
          credentials.password,
          existingUser.password
        )
        if (!passwordValidation) {
          throw new Error("Incorrect password")
        }
        return {
          id: existingUser.id,
          email: existingUser.email,
          name: existingUser.name
        }
      }
    })
  ]
}
```



## Paytm Project 11 of 20

```
        return {
          id: existingUser.id.toString(),
          name: existingUser.name,
          email: existingUser.number
        }
      }
      return null;
    }

  try {
    const user = await db.user.create({
      data: {
        number: credentials.phone,
        password: hashedPassword
      }
    });

    return {
      id: user.id.toString(),
      name: user.name,
      email: user.number
    }
  } catch(e) {
    console.error(e);
  }

  return null
}
```



## Paytm Project 11 of 20

```
    ],
    session.env.JWT_SECRET || "secret",
  {
    // TODO: can u fix the type here? Using any is bad
    async session({ token, session }: any) {
      session.user.id = token.sub

      return session
    }
  }
}
```

- Create a `/api/auth/[...nextauth]/route.ts`

```
import NextAuth from "next-auth"

const handler = NextAuth(authOptions)

export { handler as GET, handler as POST }
```

- Update `.env`

```
JWT_SECRET=test
NEXTAUTH_URL=http://localhost:3001
```

ignin



## Paytm Project 11 of 20

Create nextauth.ts

```
import GoogleProvider from "next-auth/providers/google";

export const authOptions = {
  providers: [
    GoogleProvider({
      clientId: process.env.GOOGLE_CLIENT_ID || "",
      clientSecret: process.env.GOOGLE_CLIENT_SECRET || ""
    })
  ],
}
```

- Create a /api/auth/[...nextauth]/route.ts

```
import NextAuth from "next-auth"

const handler = NextAuth(authOptions)

export { handler as GET, handler as POST }
```

- Put your google client and secret in .env of the merchant app.  
Ref <https://next-auth.js.org/providers/google>

NEXTAUTH\_URL=http://localhost:3000

ECRET=kiratsecr3t



## Paytm Project 11 of 20

- Ensure you see a signin page at <http://localhost:3001/api/auth/signin>
- Try signing in and make sure it reaches the DB

# Add auth

## Client side

- Wrap the apps around `SessionProvider` context from the next-auth package
- Go to [merchant-app/providers.tsx](#)

"use client"





## Paytm Project 11 of 20

```
Providers = ({children}: {children: React.ReactNode}) => {
  <coilRoot>
    <SessionProvider>
      {children}
    </SessionProvider>
  </RecoilRoot>
}
```

- Do the same for `user-app/providers.tsx`

## Server side

Create `apps/user-app/app/api/user.route.ts`

```
import { getServerSession } from "next-auth"
import { NextResponse } from "next/server";
import { authOptions } from "../lib/auth";

export const GET = async () => {
  const session = await getServerSession(authOptions);
  if (session.user) {
    return NextResponse.json({
      user: session.user
    })
  }
}

message: You are not logged in
```

}, {

03

Paytm Project 11 of 20

}

Ensure login works as expected

# Add an AppBar component

- Update the `Button` component in UI

```
"use client";
```

```
import { ReactNode } from "react";
```

```
interface ButtonProps {
```

```
    children: ReactNode;
```

}



## Paytm Project 11 of 20

```
Button = ({ onClick, children }: ButtonProps) => {
  return (
    <button onClick={onClick} type="button" className="text-white bg-gray-
      {children}
    </button>

  );
};
```

- Create a `ui/Appbar` component that is highly generic (doesnt know anything about the user/how to logout).

```
import { Button } from "./button";

interface AppBarProps {
  user?: {
    name?: string | null;
  },
  // TODO: can u figure out what the type should be here?
  onSignin: any,
  onSignout: any
}

export const AppBar = ({
```





onSignout  
Paytm Project 11 of 20 ps) => {  
  className="flex justify-between border-b px-4">  
    <div className="text-lg flex flex-col justify-center">  
      PayTM  
    </div>  
    <div className="flex flex-col justify-center pt-2">  
      <Button onClick={user ? onSignout : onSignin}>{user ? "Logout" : "Login"}  
    </div>  
  </div>  
}

# Checkpoint

ect-



Paytm Project 11 of 20

# On Ramping

- Allows PayTM to generate a **token** for a payment for a user for some



## Paytm Project 11 of 20

```
POST /api/transaction
{
  "user_identifier": "1",
  "amount": "59900", // Rs 599
  "webhookUrl": "http://localhost:3003/hdfcWebhook"
}
```

- PayTM should redirect the user to

```
https://bank-api-frontend.com/pay?token={token\_from\_step\_1}
```

- If user made a successful payment, **Bank** should hit the **webhookUrl** for the company

## Creating a bank\_webhook\_handler Node.js project

- Init node.js project + esbuild

```
cd apps
mkdir bank_webhook_handler
cd bank_webhook_handler
npm init -y
npx tsc --init
```

- Update tsconfig



## Paytm Project 11 of 20

```
"extends": "@repo/typescript-config/base.json",
"compilerOptions": {
  "outDir": "dist"
},
"include": ["src"],
"exclude": ["node_modules", "dist"]
}
```

- Create `src/index.ts`

```
import express from "express";

const app = express();

app.post("/hdfcWebhook", (req, res) => {
  //TODO: Add zod validation here?
  const paymentInformation = {
    token: req.body.token,
    userId: req.body.user_identifier,
    amount: req.body.amount
  };
  // Update balance in db, add txn
  // ...
}
```

- UPDATE DB SCHEMA

```
  datasource client {  
    Paytm Project 11 of 20 prisma-client-js  
    j  
  
    datasource db {  
      provider = "postgresql"  
      url = env("DATABASE_URL")  
    }  
  
    model User {  
      id Int @id @default(autoincrement())  
      email String? @unique  
      name String?  
      number String @unique  
      password String  
      OnRampTransaction OnRampTransaction[]  
      Balance Balance[]  
    }  
  
    model Merchant {  
      id Int @id @default(autoincrement())  
      email String @unique  
      name String?  
      auth_type AuthType  
    }  
  
    status OnRampStatus
```



Paytm Project 11 of 20

```
token String      @unique  
ring  
start Time DateTime  
userId Int  
user User      @relation(fields: [userId], references: [id])  
}  
  
model Balance {  
    id Int @id @default(autoincrement())  
    userId Int @unique  
    amount Int  
    locked Int  
    user User @relation(fields: [userId], references: [id])  
}  
  
enum AuthType {  
    Google  
    Github  
}  
  
enum OnRampStatus {  
    Success  
    Failure  
    Processing  
}
```

↗️ to the right folder (packages/db)

## Paytm Project 11 of 20 migrate dev --name add\_balance

- Add `repo/db` as a dependency to package.json

```
"@repo/db": "*"
```

- Add transaction to update the balance and transactions DB  
Ref - <https://www.prisma.io/docs/orm/prisma-client/queries/transactions>

```
import express from "express";
import db from "@repo/db/client";
const app = express();

app.use(express.json())

app.post("/hdfcWebhook", async (req, res) => {
  //TODO: Add zod validation here?
  //TODO: HDFC bank should ideally send us a secret so we know this is sent
  const paymentInformation: {
    token: string;
    userId: string;
    amount: string
  } = {
    token: req.body.token,
```

};



## Paytm Project 11 of 20

```
await db.$transaction([
  db.balance.updateMany({
    where: {
      userId: Number(paymentInformation.userId)
    },
    data: {
      amount: {
        // You can also get this from your DB
        increment: Number(paymentInformation.amount)
      }
    }
  }),
  db.onRampTransaction.updateMany({
    where: {
      token: paymentInformation.token
    },
    data: {
      status: "Success",
    }
  })
]);

res.json({
  message: "Captured"
```



Paytm Project 11 of 20

```
    console.error(e);
  }s(411).json({
    age: "Error while processing webhook"
  })
}

app.listen(3003);
```

# Create generic appbar

- Create a new `AppbarClient` component in `apps/user-app/AppbarClient.tsx`

```
import { Appbar } from "@repo/ui/appbar";
```

```
import { useRouter } from "next/navigation";
```



## Paytm Project 11 of 20 on AppbarClient()

```
const session = useSession();
const router = useRouter();

return (
  <div>
    <Appbar onSignin={signIn} onSignout={async () => {
      await signOut()
      router.push("/api/auth/signin")
    }} user={session.data?.user} />
  </div>
);
}
```

- Add `AppbarClient` to `layout.tsx`

```
import "./globals.css";
import type { Metadata } from "next";
import { Inter } from "next/font/google";
import { Providers } from "../provider";
import { AppbarClient } from "../components/AppbarClient";

const inter = Inter({ subsets: ["latin"] });

export const metadata: Metadata = {
```

};



## Paytm Project 11 of 20

```
It function RootLayout({  
  children,  
}: {  
  children: React.ReactNode;  
}): JSX.Element {  
  return (  
    <html lang="en">  
      <Providers>  
        <AppbarClient />  
        <body className={inter.className}>{children}</body>  
      </Providers>  
    </html>  
  );  
}
```

One at a time please

- Create `user-app/(dashboard)` folder



## Paytm Project 11 of 20

inside it

- `dashboard/page.tsx`
- `transactions/page.tsx`
- `transfer/page.tsx`

```
export default function() {
  return <div>
    Dashboard Page (or transfer/txn page)
  </div>
}
```

- Create `user-app/(dashboard)/layout.tsx`



Icons come from <https://heroicons.com/>  
SidebarItem will be created in the next step

```
import { SidebarItem } from "../../components/SidebarItem";

export default function Layout({
  children,
}: {
```

Paytm Project 11 of 20

```
return (
  name="flex"
  :sName="w-72 border-r border-slate-300 min-h-screen mr-4 p
    <div>
      <SidebarItem href={"/dashboard"} icon={<HomeIcon />} title="Home"
      <SidebarItem href={"/transfer"} icon={<TransferIcon />} title="Transfer"
      <SidebarItem href={"/transactions"} icon={<TransactionsIcon />} titl
    </div>
  </div>
  {children}
</div>
);
}
```

// Icons Fetched from <https://heroicons.com/>

```
function HomeIcon() {
  return <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24">
    <path stroke-linecap="round" stroke-linejoin="round" d="M2.25 12.8954-8.5 12.8954 2.25 12.8954z" />
</svg>
}
function TransferIcon() {
  return <svg xmlns="http://www.w3.org/2000/svg" fill="none" viewBox="0 0 24 24">
    <path stroke-linecap="round" stroke-linejoin="round" d="M7.5 21.3165m0 0c-1.9531 0-3.5-1.5469-3.5-3.5s1.5469-3.5 3.5-3.5m0 0v-1.3165m0 1.3165 1.3165 0m-1.3165 0-1.3165 0z" />
</svg>
}
function TransactionsIcon() {
```

&lt;/svg&gt;



## Paytm Project 11 of 20

- Create `SidebarItem` component

```
"use client"
import { usePathname, useRouter } from "next/navigation";
import React from "react";

export const SidebarItem = ({ href, title, icon }: { href: string; title: string; icon: string }) => {
  const router = useRouter();
  const pathname = usePathname()
  const selected = pathname === href

  return <div className={`${`flex ${selected ? "text-[#6a51a6]" : "text-slate-500"}`}`}
    onClick={() => router.push(href)}
  >
    <div className="pr-2">
      {icon}
    </div>
    <div className={`${`font-bold ${selected ? "text-[#6a51a6]" : "text-slate-500"}`}`}
      >{title}</div>
  </div>
}

}
```





## Paytm Project 11 of 20 • 3 pages that look like this

The screenshot shows a web application interface for a PayTM-like service. At the top left is a circular profile picture of a man with dark hair and a beard, wearing a blue shirt. To its right is the text "Paytm Project 11 of 20 • 3 pages that look like this". The main header "PayTM" is at the top center, followed by a sub-header "Dashboard". On the far right is a "Logout" button. The left sidebar contains three menu items: "Home" (with a house icon), "Transfer" (with a transfer icon), and "Transactions" (with a clock icon). The main content area is currently empty.



PayTM

Logout

Paytm Project 11 of 20

# Create transfer page

The screenshot shows the PayTM Transfer page. On the left, there's a sidebar with links: Home, Transfer (which is selected), and Transactions. The main content area has a title 'Transfer'.

The page is divided into three main sections:

- Add Money**: A form with fields for 'Amount' (with a placeholder 'Amount') and 'Bank' (set to 'HDFC Bank'). Below the form is a 'Add Money' button.
- Balance**: Displays 'Unlocked balance' (0 INR), 'Total Locked Balance' (0 INR), and 'Total Balance' (0 INR).
- Recent Transactions**: A section stating 'No Recent transactions'.

Red arrows and labels point to specific components:

- An arrow points from the text 'AddMoneyComponent' to the 'Add Money' form.
- An arrow points from the text 'BalanceComponent' to the 'Balance' section.
- An arrow points from the text 'OnRampTransaction component' to the 'Recent Transactions' section.

- Add a better card component to packages/ui



## Paytm Project 11 of 20

```
import React from "react";
export function Card({
  title,
  children,
}: {
  title: string;
  children?: React.ReactNode;
}): JSX.Element {
  return (
    <div
      className="border p-4"
    >
      <h1 className="text-xl border-b pb-2">
        {title}
      </h1>
      <p>{children}</p>
    </div>
  );
}
```

- Add a **Center** component that centralizes (both vertically and horizontally) the children given to it (in **packages/ui** )



Make sure to export it in **package.json**





Paytm Project 11 of 20

```
export const Center = ({ children }: { children: React.ReactNode }) => {
  return (
    <div className="flex justify-center flex-col h-full">
      <div styleName="flex justify-center">
        {children}
      </div>
    </div>
  )
}
```

- Add a `Select` component to `packages/ui` (Make sure to export it)

```
"use client"
export const Select = ({ options, onSelect }: {
  onSelect: (value: string) => void;
  options: {
    key: string;
    value: string;
  }[];
}) => {
  return <select onChange={(e) => {
    onSelect(e.target.value)
  }} className="bg-gray-50 border border-gray-300 text-gray-900 text-sm rounded-md py-2 px-3 w-full">
    {options.map(option => <option value={option.key}>{option.value}</option>)}
  </select>
}
```



- Add `TextInput` component to `packages/ui`

## Paytm Project 11 of 20

```
"use client"

export const TextInput = ({  
    placeholder,  
    onChange,  
    label  
}: {  
    placeholder: string;  
    onChange: (value: string) => void;  
    label: string;  
) => {  
    return <div className="pt-2">  
        <label className="block mb-2 text-sm font-medium text-gray-900">{  
            <input onChange={(e) => onChange(e.target.value)} type="text" id="fir  
        </div>  
    }  
}
```



- Create `user-app/components/AddMoneyCard.tsx`

```
"use client"  
  
import { Button } from "@repo/ui/button";  
import { Card } from "@repo/ui/card";  
import { Center } from "@repo/ui/center";  
import { Select } from "@repo/ui/select";  
import { useState } from "react";
```

```
const SUPPORTED_BANKS = [{
```



## Paytm Project 11 of 20

```
name: "HDFC Bank",
      : "https://netbanking.hdfcbank.com"

      name: "Axis Bank",
      redirectUrl: "https://www.axisbank.com/"
    }];

export const AddMoney = () => {
  const [redirectUrl, setRedirectUrl] = useState(SUPPORTED_BANKS[0]?.redirectUrl);
  return <Card title="Add Money">
    <div className="w-full">
      <TextInput label={"Amount"} placeholder={"Amount"} onChange={() =>
        setRedirectUrl((e) => e.target.value)
      } />
      <div className="py-4 text-left">
        Select Bank
      </div>
      <Select onSelect={(value) => {
        setRedirectUrl(SUPPORTED_BANKS.find(x => x.name === value)?.redirectUrl);
      }} options={SUPPORTED_BANKS.map(x => ({
        key: x.name,
        value: x.name
      }))} />
      <div className="flex justify-center pt-4">
        <Button onClick={() => {
          window.location.href = redirectUrl || "";
        }}>
```

&lt;/div&gt;



## Paytm Project 11 of 20

{}

- Create [user-app/components/BalanceCard.tsx](#)



We're diving my 100 because we store in `paise` denomination in the db

```
import { Card } from "@repo/ui/card";

export const BalanceCard = ({amount, locked}: {
    amount: number;
    locked: number;
}) => {
    return <Card title={"Balance"}>
        <div className="flex justify-between border-b border-slate-300 pb-2">
            <div>
                Unlocked balance
            </div>
            <div>
                {amount / 100} INR
            </div>
        </div>
        0 py-2">
```

## Total Locked Balance



## Paytm Project 11 of 20

```
{locked / 100} INR
      </div>
    </div>
  <div className="flex justify-between border-b border-slate-300 py-2">
    <div>
      Total Balance
    </div>
    <div>
      {(locked + amount) / 100} INR
    </div>
  </div>
</Card>
}
```

- Create [user-app/components/OnRampTransaction.tsx](#)

```
import { Card } from "@repo/ui/card"

export const OnRampTransactions = ({
  transactions
}: {
  transactions: {
    time: Date,
  }
}) =>
```



## Paytm Project 11 of 20

```
status: string,  
        : string  
    }) => {  
    if (!transactions.length) {  
        return <Card title="Recent Transactions">  
            <div className="text-center pb-8 pt-8">  
                No Recent transactions  
            </div>  
        </Card>  
    }  
    return <Card title="Recent Transactions">  
        <div className="pt-2">  
            {transactions.map(t => <div className="flex justify-between">  
                <div>  
                    <div className="text-sm">  
                        Received INR  
                    </div>  
                    <div className="text-slate-600 text-xs">  
                        {t.time.toDateString()}  
                    </div>  
                </div>  
                <div className="flex flex-col justify-center">  
                    + Rs {t.amount / 100}  
                </div>  
            </div>)  
        </div>)  
    }  
}
```

&lt;/Card&gt;



## Paytm Project 11 of 20

- Create user-app/app/(dashboard)/transfer/page.tsx

```
import prisma from "@repo/db/client";
import { AddMoney } from "../../components/AddMoneyCard";
import { BalanceCard } from "../../components/BalanceCard";
import { OnRampTransactions } from "../../components/OnRampTransactions";
import { getServerSession } from "next-auth";
import { authOptions } from "../../lib/auth";

async function getBalance() {
  const session = await getServerSession(authOptions);
  const balance = await prisma.balance.findFirst({
    where: {
      userId: Number(session?.user?.id)
    }
  });
  return {
    amount: balance?.amount || 0,
    locked: balance?.locked || 0
  }
}

async function getOnRampTransactions() {
  const session = await getServerSession(authOptions);
```

```
userId: Number(session?.user?.id)
```



## Paytm Project 11 of 20

```
return txns.map(t => ({
  time: t.startTime,
  amount: t.amount,
  status: t.status,
  provider: t.provider
}))
}

export default async function() {
  const balance = await getBalance();
  const transactions = await getOnRampTransactions();

  return <div className="w-screen">
    <div className="text-4xl text-[#6a51a6] pt-8 mb-8 font-bold">
      Transfer
    </div>
    <div className="grid grid-cols-1 gap-4 md:grid-cols-2 p-4">
      <div>
        <AddMoney />
      </div>
      <div>
        <BalanceCard amount={balance.amount} locked={balance.locked}>
          <div className="pt-4">
            <OnRampTransactions transactions={transactions} />
          </div>
        </BalanceCard>
      </div>
    </div>
  </div>
}
```



# Add some seed data

- Go to `packages/db`
- Add `prisma/seed.ts`

```
import { PrismaClient } from '@prisma/client'
const prisma = new PrismaClient()

async function main() {
  const alice = await prisma.user.upsert({
    where: { number: '9999999999' },
    update: {},
    create: {
      number: '9999999999',
      password: 'alice',
      name: 'alice',
      OnRampTransaction: {
        create: {
          startTime: new Date(),
          status: "Success",
          amount: 20000
        }
      }
    }
  })
}

main().catch((e) => {
  console.error(e)
})
```

},



## Paytm Project 11 of 20

```
)  
const bob = await prisma.user.upsert({  
  where: { number: '9999999998' },  
  update: {},  
  create: {  
    number: '9999999998',  
    password: 'bob',  
    name: 'bob',  
    OnRampTransaction: {  
      create: {  
        startTime: new Date(),  
        status: "Failure",  
        amount: 2000,  
        token: "123",  
        provider: "HDFC Bank",  
      },  
    },  
  },  
})  
console.log({ alice, bob })  
}  
main()  
.then(async () => {  
  await prisma.$disconnect()
```



Paytm Project 11 of 20  
console.error(e)  
na.\$disconnect()  
it(1)  
})

- Update package.json

```
"prisma": {  
  "seed": "ts-node prisma/seed.ts"  
}
```

- Run command to seed db

```
npx prisma db seed
```

- Explore db

```
npx prisma studio
```



# Make landing page redirect

Paytm Project 11 of 20

The user should go to either the `signin page` or the `dashboard page` based on if they are logged in

Update root `page.tsx`

```
import { getServerSession } from "next-auth";
import { redirect } from 'next/navigation'
import { authOptions } from "./lib/auth";

export default async function Page() {
  const session = await getServerSession(authOptions);
  if (session?.user) {
    redirect('/dashboard')
  } else {
    redirect('/api/auth/signin')
  }
}
```

Final codebase - <https://github.com/100xdevs-cohort-2/week-17-final->



Paytm Project 11 of 20