ImmersivΞ Backend Brief

Sui Overflow Hackathon Project

What is this project?

- This project is **NFT minting + 3D Gallery with AR**
- Users can **log in with Google (via zkLogin)**, get a new **Sui wallet**, and **mint a free NFT** with a .glb file
- Built with **React + TypeScript (frontend)** and **Sui Move (smart contract)**

Project Structure

frontend/ React + TypeScript (UI)

contracts/ Smart Contract (Sui Move)

To run frontend:

```
cd frontend
npm install
npm run dev
```

To build + test smart contract:

```
cd contracts
sui move build      # compile smart contract
sui move test       # run unit tests
```

Backend Tasks

1. zkLogin Integration (Google Login + Wallet)

- Implement **Google Login** via zkLogin
- If user doesn't have a wallet → **create a new Sui wallet**

- **Return the new wallet address** to the frontend

---

2. Minting API (Call Smart Contract)

- Create POST /mint endpoint
  - Receives the **CID** (uploaded asset)
  - Calls the **deployed smart contract** to mint the NFT for the user
- Smart contract is ready (see contracts/ folder)

---

3. Walrus Integration (.glb + .webp + metadata)

- Upload .glb, .webp, and metadata .json to Walrus
- Get the **CID** and pass it to /mint API

---

4. API Testing

- Test all APIs (/login, /mint, etc.)
- Test full flow: **login → upload → mint NFT successfully**

---

5. Deployment

- Smart contract in contracts/ is ready to deploy
- Use the deployment script inside frontend/:

```
cd frontend
npx ts-node src/deploy.ts
```

Next Steps

Backend Dev:

- Deploy smart contract to **Sui testnet**
- Implement /login, /mint, and **Walrus upload**
- Connect backend to frontend (update Hero.tsx to trigger minting)

Frontend Dev:

- Improve UI/UX
- Test login and NFT minting with live backend

Attachments / References

- **Smart Contract**: contracts/sources/simple_nft.move
- **Unit Tests**: contracts/tests/contracts_tests.move
- **Frontend (mint button)**: frontend/src/components/Hero.tsx