**Project Functional Requirements & Step-by-Step Guide**

**1. Functional Requirements**

**1.1 User Features**

1. **RPC Selection**: Users can input and select a custom Solana RPC URL for transactions.
2. **NFT Minting**: Users can specify the number of NFTs to mint.
3. **Stage Selection**: Users can choose between different minting stages (presale or public sale).
4. **Wallet Management**: Users can add and select wallets for minting.
5. **Transaction Monitoring**: The bot displays transaction fees and compute units before and after minting.
6. **Telegram Integration**: Users interact with the system using a Telegram bot.

**1.2 Admin Features**

1. **Manage RPC Nodes**: Admins can add, update, or remove RPC nodes.
2. **Wallet Management**: Admins can add or remove user wallets.
3. **Transaction Logging**: The bot logs all transactions for auditing.

**1.3 Automation Features**

1. **RPC Health Check**: The system tests RPCs for availability before use.
2. **Gas & Compute Unit Estimation**: Estimates transaction costs before minting.
3. **Automatic RPC Fallback**: If one RPC fails, the bot selects another available RPC.

**2. Step-by-Step Guide to Building the Bot**

**Step 1: Set Up the Environment**

1. Install **Node.js** and **npm**.
2. Create a new project folder and initialize it:
3. mkdir nft-minting-bot && cd nft-minting-bot
4. npm init -y
5. Install dependencies:
6. npm install express node-telegram-bot-api @solana/web3.js mongoose dotenv axios

**Step 2: Configure Environment Variables**

1. Create a .env file in the project root:
2. BOT\_TOKEN=your\_telegram\_bot\_token
3. MONGO\_URI=your\_mongodb\_connection\_string
4. Store sensitive data securely.

**Step 3: Create the Express Server**

1. Create an index.js file and set up an Express server:
2. const express = require("express");
3. require("dotenv").config();
4. const app = express();
5. const PORT = process.env.PORT || 3000;
6. app.use(express.json());
7. app.listen(PORT, () => {
8. console.log(`Server running on port ${PORT}`);
9. });

**Step 4: Implement MongoDB Models**

1. Create a models folder and add two models:
   * **RPC Model (rpc.js)**:
   * const mongoose = require("mongoose");
   * const rpcSchema = new mongoose.Schema({
   * url: { type: String, required: true },
   * status: { type: Boolean, default: true },
   * });
   * module.exports = mongoose.model("RPC", rpcSchema);
   * **Wallet Model (wallet.js)**:
   * const mongoose = require("mongoose");
   * const walletSchema = new mongoose.Schema({
   * address: { type: String, required: true },
   * privateKey: { type: String, required: true },
   * });
   * module.exports = mongoose.model("Wallet", walletSchema);

**Step 5: Integrate Telegram Bot**

1. Create a bot.js file and set up Telegram commands:
2. const TelegramBot = require("node-telegram-bot-api");
3. require("dotenv").config();
4. const bot = new TelegramBot(process.env.BOT\_TOKEN, { polling: true });
5. bot.onText(/\/start/, (msg) => {
6. bot.sendMessage(msg.chat.id, "Welcome to the NFT Minting Bot! 🚀");
7. });
8. module.exports = bot;

**Step 6: Implement NFT Minting Logic**

1. Modify bot.js to handle minting:
2. const { Connection, Keypair } = require("@solana/web3.js");
3. bot.onText(/\/mint/, async (msg) => {
4. const chatId = msg.chat.id;
5. bot.sendMessage(chatId, "Minting NFT... ⏳");
6. });

**Step 7: Deploy and Test**

1. Start the server:
2. node index.js
3. Connect your bot to Telegram and test its features.

**Step 8: Future Enhancements**

1. Add **multi-wallet support**.
2. Integrate **automatic fee optimization**.
3. Build a **web dashboard** for monitoring.
4. Enable **whitelist presale minting**.