NeuraDNS - Complete System Documentation

Deployed on Solana Devnet
Program ID: `H7azh1pVd3uySy7z4JRmQL2HpF2D9673Y9RP4yXZWfFM`

🚀 Project Overview

NeuraDNS is a decentralized domain name system built on Solana blockchain with AI-powered validation. It combines:

- **Solana Smart Contracts** for immutable domain storage
- **n8n AI Agent** (Groq) for intelligent domain validation
- **Flask Backend** serving the UI and proxying API requests
- **Node.js API** handling blockchain operations

📐 Architecture

```
┌─────────────────────────────────────────────────────────────────┐
│                          USER BROWSER                           │
│                    http://82.112.235.26:8765                    │
└─────────────────────────────────────────────────────────────────┘
                         │
                         │
                         ▼
┌─────────────────────────────────────────────────────────────────┐
│              FLASK UI SERVER (app.py)                           │
│                    Port 8765                                    │
│  ┌──────────────────────────────────────────────────────┐      │
│  │  Routes:                                              │      │
│  │  GET  /            → Serves index.html                │      │
│  │  POST /api/validate → Proxy to Node.js API           │      │
│  │  POST /api/register → Proxy to Node.js API           │      │
│  │  GET  /api/resolve  → Proxy to Node.js API           │      │
│  └──────────────────────────────────────────────────────┘      │
└─────────────────────────────────────────────────────────────────┘
                         │
                         │
                         ▼
┌─────────────────────────────────────────────────────────────────┐
│           NODE.JS API (production-api.ts)                       │
│                    Port 3007                                    │
│  ┌──────────────────────────────────────────────────────┐      │
│  │  POST /validate                                       │      │
│  │    1. Check blockchain if domain exists              │      │
│  │    2. Call n8n webhook for AI validation             │      │
│  │    3. Return validation result                       │      │
│  │                                                       │      │
│  │  POST /register                                       │      │
│  │    1. Call n8n webhook for AI validation             │      │
│  │    2. If valid, register on Solana blockchain       │       │
│  │    3. Return transaction signature                  │       │
│  │                                                      │       │
│  │  GET /resolve?domain=<name>                         │       │
│  │    → Query blockchain for domain record             │       │
│  │                                                      │       │
│  │  GET /health                                         │       │
│  │    → API status check                               │       │
│  └──────────────────────────────────────────────────────┘       │
└─────────────────────────────────────────────────────────────────┘
            │                          │
            │                          │
            │                          │
            ▼                          ▼
┌─────────────────────────┐  ┌──────────────────────────────────┐
│   N8N AI VALIDATION     │  │    SOLANA DEVNET BLOCKCHAIN      │
│  (Groq llama model)     │  │  Program ID: H7azh1pVd3u...      │
│                         │  │                                   │
│  Validates:             │  │  Smart Contract Functions:       │
│  - Domain format        │  │  1. register_request()           │
│  - IP format            │  │     - Validates domain/IP        │
│  - Well-known domains   │  │     - Creates PDA account        │
│  - Internet-registered  │  │     - Stores on-chain           │
│  - Private IPs          │  │                                   │
│  - Risk assessment      │  │  2. resolve_domain()             │
│                         │  │     - Queries domain record      │
│  URL: https://n8n.     │ │     - Returns IP/metadata       │
│  backend.lehana.in/    │ │                                  │
│  webhook/validate_     │ │  Storage: PDA-based accounts    │
│  domain                │ │  Seeds: ["domain", domain_name] │
└─────────────────────────┘  └──────────────────────────────────┘
```

🔄 Complete User Flow

1. **Check Domain Availability**

```
User enters domain + IP → Click "Check Domain"
    ↓
Browser sends POST /api/validate
    ↓
Flask proxies to Node.js API /validate
    ↓
Node.js checks Solana blockchain (getDomainPDA)
    │
    ├─ Domain exists? → Return "Already registered"
    │
    └─ Domain available → Call n8n AI validation
           ↓
       n8n Groq AI analyzes:
           - Domain format (TLD, length, pattern)
```

```
          - IP format (IPv4, public/private)
          - Well-known domain check (google.com, etc.)
          - Internet-registered likelihood
          - Risk level assessment
          ↓
      AI returns: {valid, reason, confidence, checks}
          ↓
   Node.js returns validation result to Flask
          ↓
   Flask returns to browser
          ↓
   UI displays: ✅ Available or ❌ Rejected
```

2. **Register Domain on Blockchain**

```
User clicks "Register on Blockchain"
    ↓
Browser sends POST /api/register
    ↓
Flask proxies to Node.js API /register
    ↓
Node.js calls n8n AI validation first
    │
    ├─ AI rejects? → Return error
    │
    └─ AI approves →
          ↓
      Create Solana transaction:
          1. Derive PDA (Program Derived Address)
             Seeds: ["domain", domain_name]
          2. Build instruction for register_request()
          3. Sign with wallet private key
          4. Send transaction to Devnet
          5. Wait for confirmation
          ↓
      Solana Smart Contract executes:
          - Validates domain/IP format
          - Creates domain account (PDA)
          - Stores: {domain, ip, authority, timestamp}
          - Returns success
          ↓
   Node.js receives transaction signature
          ↓
   Returns: {
       success: true,
       transaction: "signature...",
       explorer: "https://explorer.solana.com/tx/...",
       aiValidation: {validation_data}
   }
```

```
        ↓
    UI displays success with Solana Explorer link
```

3. **Resolve Domain**

```
User queries GET /api/resolve?domain=example.com
    ↓
Node.js derives PDA for domain
    ↓
Queries Solana blockchain account
    │
    ├─ Account exists? → Return {domain, ip, authority, timestamp}
    │
    └─ Not found? → Return "Domain not registered"
```

💻 Code Components

1. Solana Smart Contract (`programs/neura-dns/src/lib.rs`)

Functions:

`register_request(domain_name: String, record: String)`

- **Purpose**: Register a new domain on blockchain
- **Validation**:
  - Domain: Not empty, max 256 chars, must contain `.`
  - IP: Valid IPv4 format (4 octets, 0-255 each)
- **Storage**: Creates PDA account with seeds `["domain", domain_name]`
- **Data Stored**:

  ```
  {
      domain_name: String,    // e.g., "example.com"
      record: String,         // IP address "8.8.8.8"
      authority: Pubkey,      // Registrar's wallet address
      created_at: i64         // Unix timestamp
  }
  ```

`resolve_domain(domain_name: String)`

- **Purpose**: Query domain record from blockchain
- **Process**: Reads PDA account data
- **Returns**: Domain record with IP, authority, timestamp

Key Features:

- **PDA (Program Derived Address)**: Deterministic account addresses
- **No rent for PDA**: Uses Solana's rent-exempt storage
- **Immutable once created**: Cannot modify registered domains
- **Authority-based**: Only signer can register under their account

---

2. Node.js Backend API (`blockchain_dns_register/production-api.ts`)

Core Functions:

`validateWithAI(domain: string, ip: string)`

```
Purpose: Call n8n webhook for AI validation
Request: POST to https://n8n.backend.lehana.in/webhook/validate_domain
Payload: {domain, ip, requestedBy, timestamp}
Timeout: 10 seconds
Returns: {valid, reason, confidence, checks, riskLevel}
```

`getDomainPDA(domain: string)`

```
Purpose: Derive Program Derived Address for domain
Process:
  1. Create seeds: [Buffer.from("domain"), Buffer.from(domain)]
  2. Call PublicKey.findProgramAddressSync()
  3. Return [publicKey, bump]
Uses: Check if domain exists, resolve domain
```

`POST /validate`

```
1. Extract domain + IP from request
2. Check if domain already exists on blockchain
3. If exists, return "already registered"
4. If available, call validateWithAI()
5. Return validation result with AI confidence
```

`POST /register`

```
1. Validate request has domain + IP
2. Call validateWithAI() for AI approval
3. If AI rejects, return error
4. If AI approves:
   a. Derive domain PDA
   b. Get discriminator for instruction
   c. Serialize domain + IP data
   d. Build transaction instruction
   e. Sign with wallet
   f. Send to Solana Devnet
   g. Confirm transaction
5. Return transaction signature + explorer URL
```

`GET /resolve?domain=<name>`

```
1. Derive PDA for domain
2. Fetch account data from Solana
3. Parse and return domain record
4. If not found, return error
```

Helper Functions:

- `getDiscriminator(name: string)`: SHA256 hash for Anchor instruction
- `serializeString(str: string)`: Convert string to byte array with length prefix
- Wallet loading from `wallet.json`
- Connection to Solana Devnet

---

3. Flask UI Server (`blockchain_dns_register/app.py`)

Routes:

`GET /`

```
Serves: public/index.html
Purpose: Load the main UI page
```

`POST /api/validate`

```
Proxy to: http://localhost:3007/validate
Purpose: Forward validation requests to Node.js API
Returns: JSON with validation result
```

`POST /api/register`

```
Proxy to: http://localhost:3007/register
Purpose: Forward registration requests to Node.js API
Returns: JSON with transaction data
```

`GET /api/resolve?domain=<name>`

```
Proxy to: http://localhost:3007/resolve?domain=<name>
Purpose: Forward domain queries to Node.js API
Returns: JSON with domain record
```

Why Flask Proxy?

- Single port exposure (8765)
- Simplifies firewall rules
- CORS handling
- Static file serving

---

4. Frontend UI (`blockchain_dns_register/public/index.html`)

Key JavaScript Functions:

`checkDomain()`

```
1. Get domain + IP from input fields
2. Validate both fields filled
3. Show loading spinner
4. POST to /api/validate
5. Parse response:
   - exists: true → Show "Already registered"
   - valid: true → Show "Available + AI confidence"
   - valid: false → Show "Rejected + reason"
6. Display result in animated box
```

`registerDomain()`

```
1. Get domain + IP from input fields
2. Validate both fields filled
3. Show loading spinner
4. POST to /api/register with {domain, ip}
5. Parse response:
   - success: true → Show success with explorer link
   - success: false → Show error with reason
6. Display result with transaction signature
```

UI Features:

- Glassmorphism design
- Animated gradient blobs
- Typewriter effect for subtitle
- Auto-scroll to top on results
- Scrollable result boxes
- Solana/Groq/Blockchain badges

🤖 n8n AI Validation

Webhook URL: `https://n8n.backend.lehana.in/webhook/validate_domain`

Workflow Nodes:

1. **Webhook** - Receives POST requests
2. **Debug Code** - Extracts domain/IP
3. **Groq AI Agent** - Validates with llama model
4. **Code Parser** - Parses AI response
5. **Respond to Webhook** - Returns JSON

AI Validation Logic:

The AI checks:

- ✅ **Domain Format**: Valid TLD, length, special chars
- ✅ **IP Format**: Valid IPv4, public/private ranges
- ✅ **Well-Known Domains**: Blocks google.com, facebook.com, etc.
- ✅ **Internet-Registered**: Rejects short common words with .com/.net/.org
- ✅ **Accepts**: Blockchain terms, long unique names (>15 chars), custom subdomains

AI Response Format:

```
{
  "valid": true/false,
  "reason": "Explanation of decision",
  "confidence": 0.99,
  "checks": {
    "domainFormat": true,
    "ipFormat": true,
    "ipRoutable": true,
    "suspiciousPattern": false,
    "tldValid": true,
    "isWellKnownDomain": false,
    "isPrivateIP": false
  },
  "riskLevel": "low/medium/high",
  "aiProvider": "Groq",
  "processedAt": "2025-12-14T..."
}
```

🛠 Deployment

Server Configuration:

PM2 Processes:

```
# API (Node.js)
pm2 start "ts-node production-api.ts" --name neuradns-api
# Running on 0.0.0.0:3007

# UI (Flask)
pm2 start venv/bin/python --name neuradns-ui -- app.py
# Running on 0.0.0.0:8765
```

Firewall:

- Port 8765: Open (UI access)
- Port 3007: Internal only (API)

Wallet:

- Location: `/root/blockchain_dns_register/wallet.json`
- Address: `8MBPvYsG7a1SC6Jz83VxcFEVkeRs3B1PRMEqk1mxyUf7`
- Balance: 1 SOL on Devnet

📊 Data Flow Summary

```
1. USER ACTION
   Enter domain + IP → Click button

2. UI LAYER (Flask)
   Validate inputs → Show loading → Send API request

3. API LAYER (Node.js)
   Check blockchain → Call AI validation → Process result

4. AI LAYER (n8n + Groq)
   Analyze domain/IP → Apply rules → Return verdict

5. BLOCKCHAIN LAYER (Solana)
   Validate format → Create PDA → Store data → Confirm

6. RESPONSE CHAIN
   Solana → Node.js → Flask → Browser → User sees result
```

Security Features

1. **AI Validation**: Blocks malicious/well-known domains
2. **On-Chain Validation**: Smart contract validates format
3. **Immutable Storage**: Cannot modify registered domains
4. **PDA-based**: Deterministic, collision-resistant addressing
5. **Authority Control**: Only registrar can register under their key

---

Environment Variables

Node.js API (.env):

```
PORT=3007
N8N_WEBHOOK_URL=https://n8n.backend.lehana.in/webhook/validate_domain
NODE_ENV=production
```

Flask App:

- No env vars needed (proxies to localhost:3007)

Access Points

- **UI**: http://82.112.235.26:8765
- **API Health**: http://localhost:3007/health (internal)
- **Solana Explorer**: https://explorer.solana.com/address/H7azh1pVd3uySy7z4JRmQL2HpF2D9673Y9RP4yXZWfFM?cluster=devnet

---

Dependencies

Solana Program:

- anchor-lang 0.32.1

Node.js API:

- @solana/web3.js 1.95.2
- express, cors, axios
- crypto (built-in)

Flask UI:

- flask 3.0.0
- requests 2.31.0

Frontend:

- Pure HTML/CSS/JavaScript
- Space Grotesk font (Google Fonts)

🎯 Use Cases

1. **Decentralized DNS**: Store domain records on blockchain
2. **Web3 Naming**: Create custom blockchain-based names
3. **Hackathon Demo**: Showcase Solana + AI integration
4. **Learning Tool**: Understand PDA, transactions, AI APIs

📈 Future Enhancements

- [ ] Domain expiration/renewal
- [ ] Transfer ownership functionality
- [ ] Support for TXT/CNAME records
- [ ] Mainnet deployment
- [ ] ENS-style subdomain support
- [ ] Multi-chain compatibility

Built using Solana, n8n, Groq AI, and Flask