1. Purpose of this Guide

This Developer Guide provides all necessary instructions and steps to set up a local development environment for the CLXEND platform. It is intended for engineers working across identity, blockchain, mobile/web, infrastructure, and compliance domains.

2. Scope

This guide applies to developers involved in:

- Biometric & Digital ID Modeling
- Blockchain & Transaction Development
- Mobile/Web & API Product Engineering
- DevOps, Cloud, and Integration
- AI Risk and Governance Modeling

3. Prerequisites and Assumptions

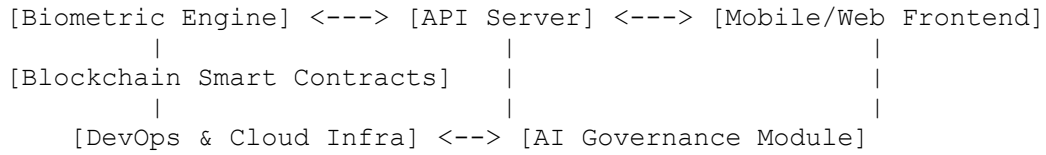
Before starting development, ensure you have:

- Basic programming knowledge in Python and JavaScript/Node.js
- Familiarity with Git and version control workflows
- Basic understanding of blockchain concepts and smart contracts
- Access to a Linux or macOS system (Ubuntu 20.04+ recommended) or Windows with WSL2
- Minimum 8GB RAM, 50GB disk space; GPU recommended for AI modules
- Installed Docker and basic familiarity with containerization

4. Project Overview and Architecture

CLXEND is a multi-component platform integrating biometric authentication, blockchain-based identity management, mobile/web client applications, and AI-driven governance.

High-Level Architecture Diagram:



Component Descriptions:

- **Biometric Engine:** Facial recognition, liveness detection, digital identity models
- **Blockchain:** Smart contracts for identity verification and transactions

- **Frontend:** Mobile and web clients for user interaction
- **API Server:** Backend REST/GraphQL APIs for data orchestration
- **DevOps:** Kubernetes infrastructure, CI/CD, monitoring tools
- **AI Governance:** Risk modeling, fraud detection, audit trails

5. Roles and Setup Instructions

5.1 AI/Identity Research Lead (FaceAuth & Digital ID)

Scope: Biometric model validation, liveness checks, DID framework exploration, privacy-first architecture.

Environment Setup:

```
sudo apt install python3 python3-venv python3-pip git
pip install virtualenv

virtualenv clxend_ai_env
source clxend_ai_env/bin/activate

pip install opencv-python torch torchvision pandas scikit-learn fastapi

git clone https://github.com/clxend/biometric-engine.git
cd biometric-engine

python -m unittest discover tests
```

5.2 Blockchain & Transaction Architect

Scope: Smart contracts, transaction schemas, DID on-chain integration.

Environment Setup:

```
curl -fsSL https://deb.nodesource.com/setup_18.x | sudo -E bash -
sudo apt install -y nodejs
npm install -g hardhat

git clone https://github.com/clxend/contracts.git
cd contracts
npm install

npx hardhat compile
npx hardhat test
```

5.3 Full-Stack Product Engineer (Mobile/Web/API)

Scope: Implement FaceAuth login, crypto transfer UI, notification stack, API integration.

Environment Setup:

```
npm install -g expo-cli

git clone https://github.com/clxend/mobile-app.git
cd mobile-app
npm install

expo start

# Backend/API setup (Python FastAPI)
git clone https://github.com/clxend/api-server.git
cd api-server
pip install -r requirements.txt
uvicorn main:app --reload
```

5.4 Cloud & Integration DevOps Engineer

Scope: K8s infrastructure, AML/KYC integration, CI/CD setup, monitoring, performance dashboards.

Environment Setup:

```
sudo apt install docker.io
curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-
linux-amd64
sudo install minikube-linux-amd64 /usr/local/bin/minikube

minikube start

curl https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3 |
bash
helm repo add prometheus-community https://prometheus-
community.github.io/helm-charts
helm install prometheus prometheus-community/prometheus
```

5.5 AI Governance & Risk Strategist

Scope: Trust signals, audit trails, fraud detection, threat modeling, compliance frameworks.

Environment Setup:

```
sudo apt install python3-pip
pip install pandas jupyter matplotlib seaborn sklearn

git clone https://github.com/clxend/ai-governance.git
```

```
cd ai-governance

jupyter notebook
```

6. Setup Verification

Verify installations after environment setup:

- `python3 --version` (expect 3.8+)
- `node -v` (expect v18+)
- `docker --version`
- `kubectl version --client`
- `git --version`

Test component functionality:

- Biometric engine tests:
  `python -m unittest discover tests`
- Blockchain contracts tests:
  `npx hardhat test`
- Mobile app: start with `expo start` and confirm UI loads
- API server:
  `uvicorn main:app --reload` and visit http://localhost:8000/docs

7. Development Workflow

- Use feature branches: `feature/<your-feature>` or `bugfix/<issue>`
- Commit frequently with clear, descriptive messages
- Submit pull requests for code review before merging
- Follow code style guidelines (PEP8 for Python, ESLint for JavaScript)
- Write and run unit tests for new code using `pytest`, `unittest`, or `hardhat test`

8. Running and Debugging the Full System

- Start backend API server:

  ```
  cd api-server
  uvicorn main:app --reload
  ```

- Start mobile frontend:

  ```
  cd mobile-app
  expo start
  ```

- Deploy blockchain contracts locally and test:

```
npx hardhat node
npx hardhat run scripts/deploy.js --network localhost
```

- Monitor logs:
  - Use `docker logs <container_id>` for containers
  - Use `kubectl logs <pod_name>` for Kubernetes pods
- Use browser developer tools and Postman to test APIs

9. Deployment Instructions

- Securely configure environment variables (`.env` files, Kubernetes secrets)
- Implement CI/CD pipelines (e.g., GitHub Actions, Jenkins) — example configs in `devops/ci-cd/`
- Deploy backend services as Docker containers on Kubernetes clusters
- Manage releases with Helm charts found in `devops/helm`
- Publish mobile apps through Expo or native app stores after thorough testing

10. Security and Compliance Notes

- Encrypt sensitive data in transit (TLS) and at rest
- Adhere to privacy-first design principles; minimize data collection
- Comply with GDPR, HIPAA, or other relevant frameworks
- Maintain immutable audit logs and access trails
- Regularly perform threat modeling and vulnerability assessments

11. Additional Learning Resources

- Blockchain & Smart Contracts
- Biometric Authentication Basics
- Kubernetes Tutorials
- FastAPI Documentation
- Expo React Native Guide

12. FAQs and Common Issues

Q: Why does my Kubernetes pod keep restarting?
A: Check pod logs with `kubectl logs <pod>` and inspect events with `kubectl describe pod <pod>`. Common issues include misconfigured environment variables or insufficient resources.

Q: Python packages fail to install?
A: Ensure you have activated the correct virtual environment (`source clxend_ai_env/bin/activate`) before installing packages.

Q: Smart contract tests fail with "network not found"?
A: Start a local Hardhat node with `npx hardhat node` before running tests.

13. Contribution and Code of Conduct

- Fork the repository and work on feature branches
- Submit pull requests with clear descriptions referencing related issues
- Respect community guidelines and provide constructive feedback during code reviews
- Report security vulnerabilities privately to project maintainers

14. Directory Structure Recommendations

```
clxend/
├── biometric-engine/
├── contracts/
├── mobile-app/
├── api-server/
├── devops/
├── ai-governance/
```

15. Common Git Workflow

```
git checkout -b feature/<branch-name>
git add .
git commit -m "<feature implementation>"
git push origin feature/<branch-name>
```

16. Troubleshooting Tips

- Check logs in `logs/` or use `docker logs` for containerized services
- Use `kubectl get pods` and `kubectl describe pod <pod_name>` to debug Kubernetes applications
- Validate code with `pytest`, `unittest`, or `npx hardhat test`
- For GPU usage, verify with `nvidia-smi` and `torch.cuda.is_available()`

17. Contacts & Support

- **DevOps Environments:** Ranjith K
- **Identity & ML Models:** Vignesh AS
- **Smart Contracts:** E Durai Prakash
- **Mobile App:** G P Vimalashwari
- **Audit & Governance:** Ajay S