Complete Folder Directory Structure

Server-Side Directory Structure (VPS/Remote Server)

```
zkp_biometric_voting_server/
│
├── README.md
├── requirements.txt
├── config/
│   ├── server_config.yaml
│   ├── database_config.yaml
│   └── security_keys.env
│
├── core/                          # Original voting system core
│   ├── __init__.py
│   ├── pqc_crypto.py              # Post-quantum cryptography
│   ├── zkp_module.py              # Zero-knowledge proofs
│   ├── biometric_auth.py          # Original biometric authentication
│   ├── blockchain_voting.py       # Blockchain implementation
│   └── voting_system.py           # Main voting system
│
├── server/                        # Server-specific components
│   ├── __init__.py
│   ├── server_real_data.py        # Real data handler
│   ├── server_auth.py             # Authentication server
│   ├── server_homomorphic.py      # Homomorphic processing
│   ├── api/
│   │   ├── __init__.py
│   │   ├── auth_routes.py         # Authentication API routes
│   │   ├── voting_routes.py       # Voting API routes
│   │   └── admin_routes.py        # Admin API routes
│   └── middleware/
│       ├── __init__.py
│       ├── security_middleware.py
│       └── rate_limiting.py
│
├── dataset/                       # SOCOFing dataset (server side)
│   ├── metadata/
│   │   ├── socofing_metadata.db   # SQLite database
│   │   └── dataset_info.json
│   ├── Real/                      # Only real fingerprints on server
│   │   ├── 1/
│   │   │   ├── 1_1_1.png
│   │   │   ├── 1_1_2.png
│   │   │   ├── 1_2_1.png
│   │   │   └── ...
│   │   ├── 2/
```

```
│   │   └── ...
│   └── processed_templates/        # Extracted features storage
│       ├── real_templates.pkl
│       └── template_hashes.json
│
├── storage/                        # Server storage
│   ├── blockchain_data/
│   │   ├── blocks/
│   │   └── transactions/
│   ├── voter_registry/
│   │   ├── registered_voters.db
│   │   └── voter_credentials.json
│   └── logs/
│       ├── authentication.log
│       ├── voting.log
│       └── security.log
│
├── security/                       # Security components
│   ├── __init__.py
│   ├── key_management.py
│   ├── threshold_decryption.py
│   ├── secure_storage.py
│   └── keys/
│       ├── server_private.key
│       ├── server_public.key
│       └── authority_keys/
│
├── tests/                          # Server-side tests
│   ├── __init__.py
│   ├── test_server_auth.py
│   ├── test_real_data_handler.py
│   ├── test_api_endpoints.py
│   └── integration/
│       ├── test_full_authentication.py
│       └── test_server_client_communication.py
│
├── scripts/                        # Deployment and utility scripts
│   ├── setup_server.sh
│   ├── start_server.py
│   ├── backup_data.py
│   ├── load_dataset.py
│   └── maintenance/
│       ├── cleanup_logs.py
│       └── database_maintenance.py
│
├── docs/                           # Server documentation
│   ├── server_setup.md
│   ├── api_documentation.md
│   ├── security_guidelines.md
│   └── deployment_guide.md
│
└── app.py                          # Main Flask application entry point
```

Client-Side Directory Structure (Local Testing Environment)

```
zkp_biometric_voting_client/
│
├── README.md
├── requirements.txt
├── config/
│   ├── client_config.yaml
│   ├── server_endpoints.json
│   └── test_config.yaml
│
├── core/                          # Same core components as server
│   ├── __init__.py
│   ├── pqc_crypto.py              # Identical to server
│   ├── zkp_module.py              # Identical to server
│   ├── biometric_auth.py          # Identical to server
│   ├── blockchain_voting.py       # Identical to server
│   └── voting_system.py           # Identical to server
│
├── client/                        # Client-specific components
│   ├── __init__.py
│   ├── client_biometric.py        # Client biometric processing
│   ├── client_zkp.py              # Client ZKP generation
│   ├── homomorphic_client.py      # Client homomorphic encryption
│   ├── client_test_data.py        # Test data handler
│   ├── enhanced_voting_system.py # Enhanced system with dataset
│   └── communication/
│       ├── __init__.py
│       ├── server_client.py       # Server communication
│       └── secure_channel.py      # Secure communication
│
├── dataset/                       # SOCOFing dataset (client side)
│   ├── metadata/
│   │   ├── socofing_metadata.db  # Same structure as server
│   │   └── test_metadata.json
│   ├── Altered-Easy/              # Client testing data
│   │   ├── 1/
│   │   │   ├── 1_1_1.png
│   │   │   └── ...
│   │   └── ...
│   ├── Altered-Medium/
│   │   └── ...
│   ├── Altered-Hard/
│   │   └── ...
│   ├── Synthetic/
│   │   └── ...
│   └── processed_features/        # Extracted test features
│       ├── altered_features.pkl
│       ├── synthetic_features.pkl
│       └── feature_analysis.json
│
├── testing/                       # Comprehensive testing suite
│   ├── __init__.py
│   ├── dataset_handler.py         # SOCOFing dataset handler
│   ├── test_voting_system.py      # Original test suite
│   ├── test_with_socofing.py      # Dataset integration tests
```

```
│   ├── performance_tests.py      # Performance evaluation
│   ├── security_analysis.py      # Security testing
│   ├── test_data/
│   │   ├── mock_biometrics/
│   │   ├── test_vectors/
│   │   └── expected_results/
│   └── reports/
│       ├── test_results.json
│       ├── security_analysis.html
│       └── performance_metrics.csv
│
├── ui/                           # User interface components
│   ├── __init__.py
│   ├── cli/
│   │   ├── __init__.py
│   │   ├── voter_cli.py
│   │   ├── admin_cli.py
│   │   └── test_cli.py
│   ├── gui/
│   │   ├── __init__.py
│   │   ├── main_window.py
│   │   ├── voting_interface.py
│   │   └── test_interface.py
│   └── web/
│       ├── templates/
│       ├── static/
│       └── web_client.py
│
├── results/                      # Test and analysis results
│   ├── authentication_results/
│   │   ├── real_vs_altered.json
│   │   ├── real_vs_synthetic.json
│   │   └── security_metrics.json
│   ├── voting_results/
│   │   ├── successful_votes.json
│   │   ├── failed_attempts.json
│   │   └── blockchain_analysis.json
│   ├── performance_metrics/
│   │   ├── latency_measurements.csv
│   │   ├── throughput_analysis.csv
│   │   └── resource_usage.json
│   └── reports/
│       ├── comprehensive_analysis.pdf
│       ├── security_assessment.md
│       └── dataset_evaluation.html
│
├── scripts/                      # Client utility scripts
│   ├── setup_client.sh
│   ├── download_dataset.py
│   ├── run_tests.py
│   ├── generate_reports.py
│   └── utilities/
│       ├── data_preprocessing.py
│       ├── feature_extraction.py
│       └── result_analysis.py
│
```

```
├── docs/                          # Client documentation
│   ├── client_setup.md
│   ├── testing_guide.md
│   ├── dataset_integration.md
│   └── usage_examples.md
│
└── main.py                        # Main client application entry point
```

Configuration Files

Server Configuration (`config/server_config.yaml`)

```yaml
# server_config.yaml
server:
  host: "0.0.0.0"
  port: 5000
  debug: false
  ssl_enabled: true
  ssl_cert_path: "/opt/certs/server.crt"
  ssl_key_path: "/opt/certs/server.key"

database:
  path: "/opt/zkp_voting/storage/voter_registry/registered_voters.db"
  backup_interval: 3600  # seconds
  max_connections: 100

security:
  rate_limiting:
    requests_per_minute: 60
    requests_per_hour: 1000
  authentication_timeout: 300  # seconds
  max_failed_attempts: 3

dataset:
  path: "/opt/zkp_voting/dataset"
  real_data_only: true
  template_cache_size: 10000

logging:
  level: "INFO"
  file_path: "/opt/zkp_voting/storage/logs/server.log"
  max_file_size: "10MB"
  backup_count: 5
```

Client Configuration (`config/client_config.yaml`)

```yaml
# client_config.yaml
client:
  name: "ZKP_Voting_Test_Client"
  version: "1.0.0"
  testing_mode: true
```

```
server:
  url: "https://your-vps-server.com:5000"
  timeout: 30
  retry_attempts: 3
  ssl_verify: true

dataset:
  path: "/opt/zkp_voting_client/dataset"
  categories: ["Altered-Easy", "Altered-Medium", "Altered-Hard", "Synthetic"]
  max_samples_per_category: 100

testing:
  parallel_tests: true
  max_workers: 4
  generate_reports: true
  save_intermediate_results: true

biometric:
  fingerprint:
    feature_extraction_method: "histogram_edge"
    similarity_threshold: 0.85
  iris:
    feature_extraction_method: "concentric_circles"
    similarity_threshold: 0.80

output:
  results_path: "/opt/zkp_voting_client/results"
  report_format: ["json", "html", "csv"]
  detailed_logging: true
```

Setup Scripts

Server Setup Script (`scripts/setup_server.sh`)

```
#!/bin/bash
# setup_server.sh

echo "Setting up ZKP Biometric Voting Server..."

# Create directory structure
mkdir -p /opt/zkp_voting_server/{config,core,server,dataset,storage,security,tests,script
mkdir -p /opt/zkp_voting_server/dataset/{Real,metadata,processed_templates}
mkdir -p /opt/zkp_voting_server/storage/{blockchain_data,voter_registry,logs}
mkdir -p /opt/zkp_voting_server/security/keys/authority_keys
mkdir -p /opt/zkp_voting_server/server/{api,middleware}

# Install system dependencies
sudo apt update
sudo apt install -y python3-pip python3-venv sqlite3 nginx

# Create virtual environment
python3 -m venv /opt/zkp_voting_server/venv
source /opt/zkp_voting_server/venv/bin/activate
```

```
# Install Python dependencies
pip install -r requirements.txt

# Set permissions
sudo chown -R $USER:$USER /opt/zkp_voting_server
chmod +x /opt/zkp_voting_server/scripts/*.py

# Initialize database
python3 /opt/zkp_voting_server/scripts/load_dataset.py

echo "Server setup completed!"
```

Client Setup Script (`scripts/setup_client.sh`)

```
#!/bin/bash
# setup_client.sh

echo "Setting up ZKP Biometric Voting Client..."

# Create directory structure
mkdir -p zkp_voting_client/{config,core,client,dataset,testing,ui,results,scripts,docs}
mkdir -p zkp_voting_client/dataset/{Altered-Easy,Altered-Medium,Altered-Hard,Synthetic,me
mkdir -p zkp_voting_client/testing/{test_data,reports}
mkdir -p zkp_voting_client/results/{authentication_results,voting_results,performance_met
mkdir -p zkp_voting_client/ui/{cli,gui,web}

# Create virtual environment
python3 -m venv zkp_voting_client/venv
source zkp_voting_client/venv/bin/activate

# Install dependencies
pip install -r requirements.txt

# Set permissions
chmod +x zkp_voting_client/scripts/*.py

echo "Client setup completed!"
```

Requirements Files

Server Requirements (`requirements.txt`)

```
# Server requirements.txt
flask==2.3.3
flask-cors==4.0.0
opencv-python==4.8.1.78
numpy==1.24.3
scikit-learn==1.3.0
cryptography==41.0.4
sqlite3
requests==2.31.0
tenseal==0.3.12
```

```
web3==6.9.0
hashlib2==1.0.0
secrets
gunicorn==21.2.0
nginx
```

Client Requirements (`requirements.txt`)

```
# Client requirements.txt
opencv-python==4.8.1.78
numpy==1.24.3
scikit-learn==1.3.0
cryptography==41.0.4
requests==2.31.0
tenseal==0.3.12
matplotlib==3.7.2
pandas==2.0.3
pillow==10.0.0
tkinter
flask==2.3.3  # For local web interface
pytest==7.4.2
pytest-asyncio==0.21.1
```

Usage Instructions

1. Server Deployment

```
# On VPS/Server
cd /opt
git clone <your-repo> zkp_voting_server
cd zkp_voting_server
chmod +x scripts/setup_server.sh
./scripts/setup_server.sh

# Start server
python app.py
```

2. Client Setup and Testing

```
# On local machine
git clone <your-repo> zkp_voting_client
cd zkp_voting_client
chmod +x scripts/setup_client.sh
./scripts/setup_client.sh

# Download SOCOFing dataset to dataset/ folder
# Run tests
python main.py --mode testing --comprehensive
```

3. Running Tests

```
# Client side comprehensive testing
cd zkp_voting_client
python scripts/run_tests.py --all
python scripts/generate_reports.py
```

This directory structure provides complete separation between server and client components while maintaining the original voting system integrity and adding comprehensive dataset-based testing capabilities.