CIRF Framework Analysis - Project Setup Guide

Project Structure

Create the following directory structure for your CIRF analysis project:

```
cultural_entrepreneurship_analysis/
├── README.md
├── requirements.txt
├── setup.py
├── .env.example
├── .gitignore
├── cirf_analysis.py              # Main implementation file
├── run_analysis.py               # Command-line runner
├── dashboard.py                  # Streamlit dashboard
├── src/
│   ├── __init__.py
│   ├── data_collection/
│   │   ├── __init__.py
│   │   ├── scrapers/
│   │   │   ├── __init__.py
│   │   │   ├── scholar_scraper.py
│   │   │   ├── researchgate_scraper.py
│   │   │   └── base_scraper.py
│   │   ├── apis/
│   │   │   ├── __init__.py
│   │   │   ├── academic_apis.py
│   │   │   └── government_apis.py
│   │   └── extractors/
│   │       ├── __init__.py
│   │       ├── pdf_extractor.py
│   │       └── text_extractor.py
│   ├── analysis/
│   │   ├── __init__.py
│   │   ├── nlp_analysis.py
│   │   ├── cirf_scoring.py
│   │   ├── statistical_analysis.py
│   │   └── pattern_recognition.py
│   ├── database/
│   │   ├── __init__.py
│   │   ├── models.py
│   │   ├── operations.py
│   │   └── migrations/
│   ├── visualization/
│   │   ├── __init__.py
│   │   ├── dashboard.py
│   │   ├── charts.py
│   │   └── maps.py
│   └── utils/
│       ├── __init__.py
│       ├── logging_config.py
│       ├── validators.py
│       └── helpers.py
├── data/
│   ├── raw/
│   │   ├── papers/
│   │   ├── pdfs/
│   │   └── archives/
│   ├── processed/
│   │   ├── cleaned/
│   │   ├── analyzed/
│   │   └── validated/
│   └── results/
│       ├── reports/
│       ├── visualizations/
│       └── exports/
├── notebooks/
│   ├── 01_data_exploration.ipynb
│   ├── 02_cirf_analysis.ipynb
│   ├── 03_statistical_analysis.ipynb
│   ├── 04_validation_analysis.ipynb
│   └── 05_results_visualization.ipynb
├── config/
│   ├── __init__.py
│   ├── database_config.py
│   ├── api_keys.py
│   ├── scraping_config.py
│   └── analysis_config.py
├── tests/
│   ├── __init__.py
│   ├── test_data_collection.py
│   ├── test_analysis.py
│   ├── test_database.py
│   └── test_visualization.py
├── docs/
│   ├── methodology.md
│   ├── api_documentation.md
│   ├── user_guide.md
│   └── technical_specifications.md
├── scripts/
│   ├── setup_environment.sh
│   ├── run_collection.sh
│   ├── run_analysis.sh
│   └── backup_database.sh
└── outputs/
    ├── reports/
    ├── charts/
    ├── data_exports/
    └── thesis_materials/
```

Installation and Setup

1. Environment Setup

```bash
# Create virtual environment
python -m venv cirf_env

# Activate virtual environment
# On Windows:
cirf_env\Scripts\activate
# On macOS/Linux:
source cirf_env/bin/activate

# Install dependencies
pip install -r requirements.txt

# Install spaCy language model
python -m spacy download en_core_web_sm

# Install additional NLP resources
python -c "import nltk; nltk.download('punkt'); nltk.download('vader_lexicon'); nltk.download('stopwords')"
```

2. Configuration

Create a `.env` file in the project root:

```bash
```

```
# Database Configuration
DATABASE_PATH=./data/cirf_analysis.db
BACKUP_PATH=./data/backups/

# API Keys (if available)
GOOGLE_SCHOLAR_API_KEY=your_key_here
JSTOR_API_KEY=your_key_here
RESEARCHGATE_API_KEY=your_key_here

# Scraping Configuration
MAX_CONCURRENT_REQUESTS=5
REQUEST_DELAY=2
SELENIUM_TIMEOUT=30
CHROME_DRIVER_PATH=./drivers/chromedriver

# Analysis Configuration
CONFIDENCE_THRESHOLD=0.7
MIN_EVIDENCE_QUALITY=2
MAX_ANALYSIS_BATCH_SIZE=100

# Logging Configuration
LOG_LEVEL=INFO
LOG_FILE=./logs/cirf_analysis.log

# Dashboard Configuration
DASHBOARD_HOST=localhost
DASHBOARD_PORT=8501
```

3. Database Setup

The system will automatically create the SQLite database on first run. For manual setup:

```python
from cirf_analysis import CIRFDatabase

# Initialize database
db = CIRFDatabase("./data/cirf_analysis.db")
print("Database initialized successfully")
```

4. Chrome Driver Setup

For web scraping functionality:

1. Download ChromeDriver from https://chromedriver.chromium.org/
2. Place it in `./drivers/chromedriver` (or update path in .env)

3. Ensure it's executable: `chmod +x ./drivers/chromedriver`

Usage Instructions

Command Line Usage

bash

```bash
# Full analysis pipeline
python cirf_analysis.py --mode full --queries 50

# Data collection only
python cirf_analysis.py --mode collect --queries 30

# Analysis only (requires existing data)
python cirf_analysis.py --mode analyze

# Launch dashboard
python cirf_analysis.py --mode dashboard
# or
streamlit run cirf_analysis.py
```

Programmatic Usage

python

```python
from cirf_analysis import CIRFResearchSystem

# Initialize system
system = CIRFResearchSystem()

# Run data collection
collection_results = system.run_data_collection(max_queries=50)
print(f"Collected {collection_results['successfully_processed']} cases")

# Run analysis
analysis_results = system.run_comprehensive_analysis()

# Generate report
report = system.generate_report(analysis_results)
print(report)

# Export data
filename = system.export_data('excel')
print(f"Data exported to {filename}")
```

Dashboard Usage

```bash
# Launch Streamlit dashboard
streamlit run cirf_analysis.py

# Or launch with specific configuration
streamlit run cirf_analysis.py --server.port 8501 --server.headless true
```

Data Collection Strategy

Search Terms and Queries

The system automatically generates search queries combining:

- **Primary terms**: Cultural entrepreneurship failure, indigenous business closure, etc.
- **Geographic modifiers**: Canada, Australia, New Zealand, etc.
- **Sector modifiers**: Tourism, crafts, heritage, etc.

Data Sources

1. **Google Scholar** - Academic papers and citations
2. **ResearchGate** - Research publications and preprints
3. **JSTOR** - Academic journal articles (API access required)
4. **Government repositories** - Policy documents and reports
5. **Custom sources** - Can be added via the scraper framework

Quality Assurance

- Duplicate detection based on title and URL
- Evidence quality scoring (1-3 scale)
- Confidence scoring for CIRF analysis
- Manual validation protocols

CIRF Analysis Methodology

Component Scoring

Each of the 13 CIRF components is scored on a scale of 0-1:

- **0.0**: Component clearly violated/absent

- **0.5**: Mixed or unclear evidence
- **1.0**: Component clearly satisfied/present

NLP Analysis Pipeline

1. **Text preprocessing**: Tokenization, cleaning, normalization
2. **Keyword extraction**: Component-specific keyword matching
3. **Sentiment analysis**: Context-aware sentiment scoring
4. **Pattern recognition**: Failure type and cause identification
5. **Confidence scoring**: Reliability assessment

Statistical Analysis

- Frequency analysis of component violations
- Geographic and sector pattern identification
- Correlation analysis between components
- Cluster analysis for failure patterns
- Temporal trend analysis

Validation and Quality Control

Automated Validation

- Cross-reference validation across sources
- Outlier detection and flagging
- Missing data identification
- Bias detection algorithms

Manual Validation

- Random sample validation (recommended 10%)
- Expert review protocols
- Inter-rater reliability testing
- Confidence threshold adjustments

Export and Integration

Academic Output Formats

- **LaTeX tables** for thesis integration
- **APA citations** for reference lists
- **Statistical summaries** for methodology sections
- **Visualizations** for publication

Data Export Options

- CSV for spreadsheet analysis
- Excel with multiple sheets and formatting
- JSON for API integration
- SQLite database for direct access

Performance Optimization

Scalability Considerations

- Batch processing for large datasets
- Parallel execution for web scraping
- Database indexing for faster queries
- Memory-efficient data structures

Monitoring and Logging

- Comprehensive logging at all levels
- Progress tracking for long-running processes
- Error handling and recovery
- Performance metrics collection

Troubleshooting

Common Issues and Solutions

1. **Web scraping failures**
    - Check internet connection
    - Verify ChromeDriver installation

    - Adjust request delays
    - Check for website changes

2. **Database errors**
    - Verify file permissions
    - Check disk space
    - Validate database schema
    - Check for corruption

3. **Analysis errors**
    - Verify NLP model installation
    - Check data quality
    - Adjust confidence thresholds
    - Review component keywords

4. **Dashboard issues**
    - Check Streamlit installation
    - Verify port availability
    - Check browser compatibility
    - Review firewall settings

Getting Help

- Check logs in `./logs/cirf_analysis.log`
- Review error messages and stack traces
- Validate configuration settings
- Test with smaller datasets first

Research Ethics and Compliance

Data Collection Ethics

- Respect robots.txt files
- Implement reasonable rate limiting
- Cite sources appropriately
- Avoid overloading servers

Academic Integrity

- Maintain clear data provenance
- Document methodology thoroughly
- Enable reproducible research
- Share code and data appropriately

Contributing and Customization

Extending the Framework

- Add new data sources via scraper modules
- Extend CIRF components with custom scoring
- Add new analysis methods
- Customize visualization themes

Component Customization

- Modify keyword lists for better accuracy
- Adjust scoring algorithms
- Add domain-specific patterns
- Integrate expert knowledge

This setup guide provides a comprehensive foundation for your doctoral research on the Cultural Innovation Resilience Framework. The modular design allows for easy customization and extension as your research evolves.