IPL Match Scraper Project Report

**Prepared by:** Duleeka Amarasena  
**Date:** 2025/05/03  
**Version:** 1.0

## 1. Executive Summary

This document outlines the **IPL Match Scraper** project, a **production-ready data pipeline** that extracts, processes, stores, and visualizes Indian Premier League (IPL) match data from **Cricbuzz**.

The project follows **modular architecture**, ensuring **scalability, maintainability, and testability**. It includes:

✅ **Web Scraping** (BeautifulSoup, Requests)  
✅ **Data Processing & Cleaning** (Pandas)  
✅ **Storage** (SQLite/PostgreSQL, CSV/JSON)  
✅ **Visualization** (Matplotlib, Seaborn)  
✅ **Error Handling & Logging**  
✅ **Unit Testing** (Pytest)  
✅ **CLI & Configuration Management**

## 2. Project Structure & Functional Breakdown

**📁 config/**

**Purpose:** Centralized configuration management.

| **File** | **Functionality** |
| --- | --- |
| settings.py | Stores project-wide settings (API timeouts, DB URLs, output formats). |
| logging.conf | Configures logging format, levels, and handlers. |

**📁 data/**

**Purpose:** Data storage in a structured manner.

| **Subdirectory** | **Functionality** |
| --- | --- |
| raw/ | Stores raw scraped HTML/JSON before processing. |
| processed/ | Cleaned & transformed data (CSV/JSON/Parquet). |
| outputs/ | Final exported reports & visualizations. |

**📁 src/**

**Purpose:** Core business logic implementation.

**📁 src/ 📂 scraper/**

**Purpose:** Web scraping components.

| **File** | **Functionality** |
| --- | --- |
| base\_scraper.py | Base class with retry logic, error handling, and HTTP requests. |
| match\_scraper.py | Extracts match details (teams, date, venue, status). |
| player\_scraper.py *(Future Expansion)* | Scrapes player stats (runs, wickets, strike rate). |

**📁 src/📂 processing/**

**Purpose:** Data transformation & cleaning.

| **File** | **Functionality** |
| --- | --- |
| data\_cleaner.py | Handles missing values, standardizes team names, extracts match numbers. |
| data\_analyzer.py *(Future Expansion)* | Computes win rates, player performance metrics. |
| transformer.py | Converts timestamps, enriches data (e.g., day/night matches). |

**📁 src/📂 storage/**

**Purpose:** Data persistence.

| **File** | **Functionality** |
| --- | --- |
| database.py | SQLAlchemy-based DB operations (SQLite/PostgreSQL). |
| cache.py | Caches HTTP responses to avoid redundant scraping. |
| file\_storage.py | Saves data in CSV/JSON/Parquet formats. |

**📁 src/📂 utils/**

**Purpose:** Shared utilities.

| **File** | **Functionality** |
| --- | --- |
| logger.py | Custom logging setup. |
| helpers.py | Helper functions (timestamp conversion, text formatting). |
| notifications.py | Email/Slack alerts for pipeline failures. |

**📁 src/📂 visualization/**

**Purpose:** Data insights & reporting.

| **File** | **Functionality** |
| --- | --- |
| plotter.py | Generates plots (matches by venue, win percentages). |
| dashboard.py *(Future Expansion)* | Interactive Dash/Streamlit dashboard. |

**📁 tests/**

**Purpose:** Unit & integration testing.

| **File** | **Functionality** |
| --- | --- |
| test\_scraper.py | Tests HTTP requests & HTML parsing. |
| test\_processing.py | Validates data cleaning logic. |
| test\_storage.py | Checks DB/file storage operations. |

**Root-Level Files**

| **File** | **Functionality** |
| --- | --- |
| requirements.txt | Python dependencies. |
| Dockerfile | Containerization for deployment. |
| .env.example | Template for environment variables. |
| README.md | Project documentation. |
| run.py | Alternative entry point (e.g., for Docker). |

## 3. Key Features & Business Value

**✅ Scalability**

* **Modular design** allows easy addition of new scrapers (e.g., player stats).
* **Database & file storage** support large datasets.

**✅ Reliability**

* **Retry logic** (Tenacity) for failed HTTP requests.
* **Comprehensive logging** for debugging.

**✅ Maintainability**

* **Separation of concerns** (scraping, processing, storage).
* **Unit tests** ensure code quality.

**✅ Extensibility**

* **Future-proof** for:
  + More cricket leagues (BBL, PSL).
  + Advanced analytics (machine learning).
  + APIs (FastAPI) for data access.

## 4. Future Enhancements (Roadmap)

| **Feature** | **Priority** | **Description** |
| --- | --- | --- |
| **Player Stats Scraper** | High | Add detailed player performance metrics. |
| **Automated Scheduling** | Medium | Run daily via Airflow/Luigi. |
| **Live Match Updates** | Medium | Real-time scraping for ongoing matches. |
| **Interactive Dashboard** | Low | Streamlit/Power BI integration. |

## 5. Conclusion

This project delivers a **robust, production-ready pipeline** for IPL match data extraction and analysis. Its **modular architecture** ensures adaptability for future requirements, making it a **valuable tool** for sports analysts, bettors, and cricket enthusiasts.

**Next Steps:**

1. Implement player stats scraping.
2. Set up CI/CD (GitHub Actions).
3. Deploy to cloud (AWS/GCP).

**Approval:**  
Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Role: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  
Date: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_