# Product Requirements Document (PRD)

## Project Title: eCourts Cause List Scraper & Downloader

## 1. Overview

The goal of this project is to develop a web-based application that allows users to fetch, view, and download cause lists from the official eCourts India website (https://services.ecourts.gov.in/ecourtindia\_v6/?p=cause\_list/) in real time. Users should be able to select a State, District, Court Complex, and Court Name, or only a Court Complex, and retrieve the cause list PDF for a specific date.

## 2. Objectives

* Enable real-time scraping of cause lists directly from the eCourts website.
* Provide a simple, responsive UI for user interaction.
* Allow users to download cause lists in PDF format.
* Optionally support batch downloading of all court PDFs under a selected Court Complex.
* Ensure efficient error handling and data validation.

## 3. Key Features

|  |  |  |
| --- | --- | --- |
| Feature | Description | Priority |
| Real-Time Scraping | Fetch cause list data dynamically from eCourts (no cached/sample data). | High |
| Input Parameters (UI) | User inputs: State Name, District, Court Complex, Court Name, and Date. | High |
| PDF Download | Automatically download the cause list in PDF format for the selected court/date. | High |
| Court Complex-Wide Fetch | If only Court Complex is selected, fetch cause lists for all courts within it. | Medium |
| Real-Time Validation | Validate input values with eCourts’ live dropdown data. | High |
| Error Handling | Handle scenarios like missing data, invalid date, or unavailable cause list gracefully. | High |
| Storage | Optionally save PDFs and metadata locally (organized by date and court). | Medium |
| Logging | Maintain logs for fetched data and errors. | Medium |
| CLI Integration (Optional) | Command-line support for developers (--today, --tomorrow, --causelist). | Low |

## 4. Functional Requirements

### 4.1 User Interface

* Dropdowns for: State, District, Court Complex, Court Name
* Date Picker (to select cause list date)
* Buttons: Fetch Cause List, Download PDF
* Display loading spinner during data fetch
* Alerts or error messages on invalid inputs or failed fetch

### 4.2 Backend Logic

* Fetch metadata in real time using eCourts dynamic dropdowns.
* Use requests/BeautifulSoup/Selenium for data scraping.
* Parse HTML to locate cause list links for the selected criteria.
* Download the corresponding PDF file(s) to a structured folder: /downloads/{state}/{district}/{court\_complex}/{court\_name}/{date}.pdf
* Return response metadata (status, filename, URL) to frontend.

## 5. Non-Functional Requirements

|  |  |
| --- | --- |
| Parameter | Requirement |
| Performance | Real-time fetch (<5s average response time) |
| Scalability | Support multiple simultaneous users |
| Reliability | Graceful handling of network or scraping failures |
| Security | Validate and sanitize user inputs |
| Compatibility | Cross-browser support (Chrome, Edge, Firefox) |
| Maintainability | Modular, well-documented code with clear function separation |

## 6. Technical Stack

|  |  |
| --- | --- |
| Layer | Technology |
| Frontend (UI) | HTML, CSS, JavaScript (React or vanilla JS) |
| Backend | Python (Flask/FastAPI) |
| Scraping Tools | requests, BeautifulSoup4, or Selenium |
| File Handling | Python os / pathlib |
| Storage Format | PDF + optional JSON metadata |
| Version Control | GitHub repository for submission |
| Deployment (Optional) | Localhost or simple hosting (e.g., Render/Heroku) |

## 7. Workflow

1. User opens UI → selects State, District, Court Complex, Court, and Date.
2. System fetches real-time data from eCourts API or dropdowns.
3. On submission, backend scrapes the live cause list.
4. If available, the cause list PDF is downloaded.
5. If Court Complex only is provided, system loops through all courts and downloads all PDFs.
6. UI shows download status and completion summary.

## 8. Deliverables

* Functional Python + Web-based UI application.
* Source code hosted on GitHub with clear README.
* README must include setup, dependencies, and usage guide.
* Optionally include CLI commands for testing (e.g., python scraper.py --today).