SOEN 6461: Software Design Methodologies

Winter 2016

Software Design Specifications

SCRAPING PROJECT

Submitted to:

Prof: Dr. Juergen Rilling,

Prepared by:

Team Scraping (Group #:3**)**

Log file:		
Student: Name	**ID**	**Main contribution(s)**
Saketh Mudrakola	27394764	Scraping coding, Documentation(Usecase-1)
Birdevinder Singh	40012584	Documentation(Usecase-2)
Taranjot Singh	27678711	Documentation(Usecase-3)

Table of Contents

Revision History

Name	Date	Reason For Changes	Version
Scraping Team 3	9/03/2016	Detailed Design	1.0
Scraping Team 3	*2/3/2016*	Change in Sequence Diagram	1.0

1. Introduction

The Scraping Tool is a software designed to assist users in retrieving various data from URL, more specifically Google Play applications, and export to XML and RDF files. It is a computer software technique of extracting information from Google Play Store. The tool is implemented using Java programming language and it uses JSOUP library to scrape the information from the web.

1.1 Purpose

This Software Design Specification serves the purpose of assisting the user in retrieving various data from URL and tool will be designed to work with a counterpart Analysis Tool software. User will use this tool to extract information such as number of downloads, reviews, etc. from Google Play Store application. The extracted information will be stored in XML file which will act as input for Analysis team. The scraped data can also be viewed in RDF file format. The system will be easy to use, user-friendly interface, and portable.

1.2 Project Scope and Product Features

The Scraping Tool will be designed for the FactRus Company in order to collect data from various applications on the Google Play store. The tool will be user friendly and efficient. The application will be compatible with the Analysis Tool being provided by another team. The application will include features for uploading a CSV file, scraping data from the URLs provided in the given CSV file, parse and organize the information, converting and writing the retrieved data to XML format, and exporting retrieved data into an RDF file.

Deliverables include:

- The Scraping Tool V1.0: The software that users can install and use to scrape data from Google Play.
- The Scraping Tool Documentation: Instructions and guidelines document designated for the intended user.

1.3 Definitions, Acronyms and Abbreviations

CSV (Comma Separated Values) :- It is a way to collect the data from the table, so that it can be conveyed as input to table-oriented application such as a relational database application.

JDK (**Java Development Kit):-** It is a software development environment used for developing Java applications and includes the Runtime Java Environment, an interpreter or loader, a compiler, an archiver, a documentation generator and other tools needed in Java development.

XML: Extensible Markup Language

RDF: Resource Description Framework

SRS: Software Requirements Specification

SDD: Software Design Document

ECLIPSE/NETBEANS: Integrated Development Environments

1.4 References

[1] *Juergen Rilling,* Software Requirement Specification for scraping tool- course material, Concordia University, Department of Computer Science.

[2] IEEE-830 – 1993 Software Requirements Specification Standard.

2. Overall Description

2.1 Product Perspective

The Google play scraping tool being developed for FactsRUS is a new self-containing product.

It has the following features:
File: Choose CSV file, Upload CSV file, Export to XML, Export to RDF.
Scraping Information: Scraped information can be stored in XML or RDF file.
Display: System displays generated XML or RDF content on web page.

2.2 Product Features

File:
1. Choose new CSV file to scrape from.
2. Upload CSV file.
3. Save scraped data to XML file.
4. Export Data in RDF format.
5. Close file.

Scraping Info:
1. Display Scraping information for CSV file.

Display:
1. Open scraped information in XML file.
2. Open scraped information in RDF file.
3. Display scraped information on web page.

2.3 Assumptions and Dependencies

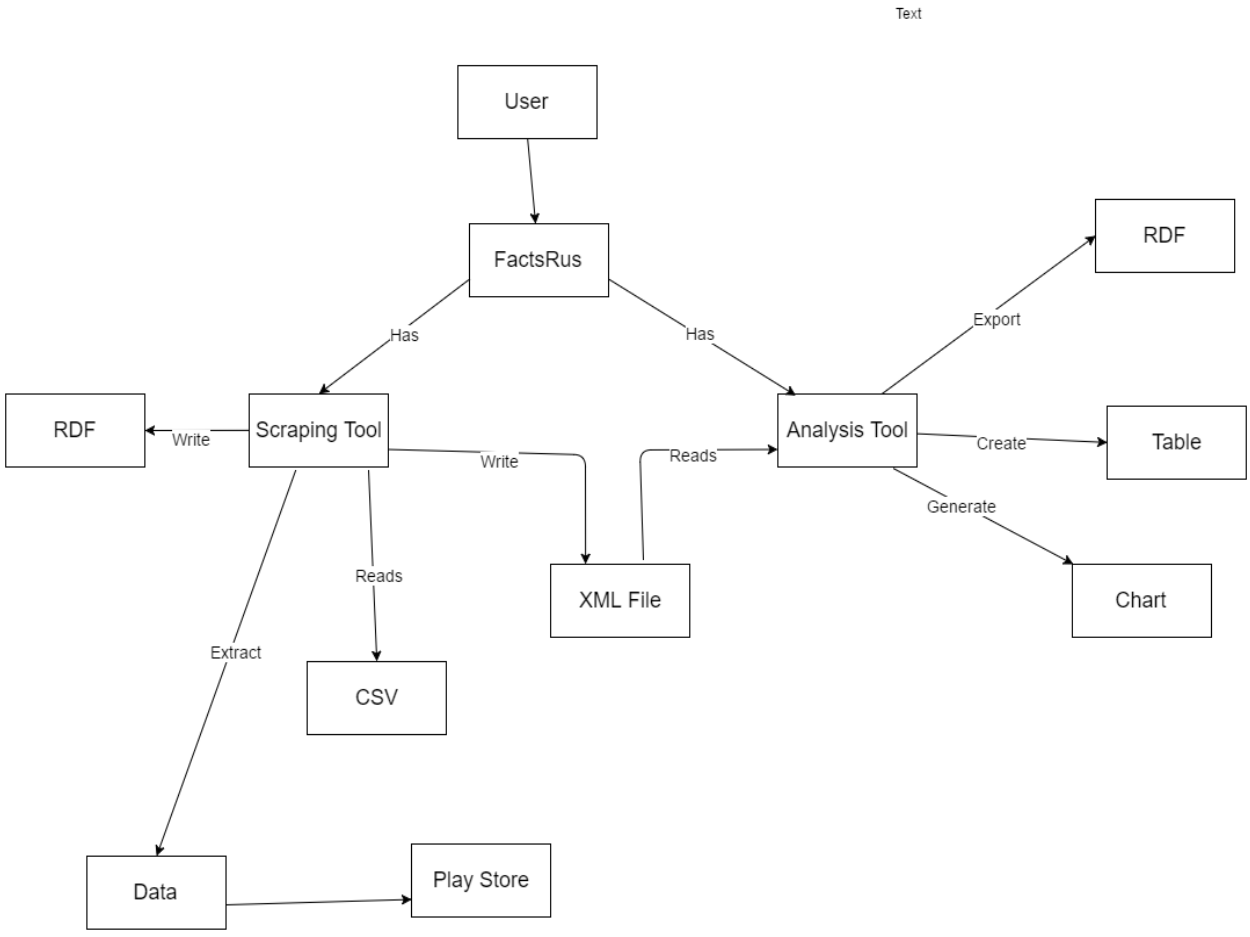
A-01. CSV file will be given to user.
A-02. CSV file will be having URLs for the Google Play application.

3. System Features

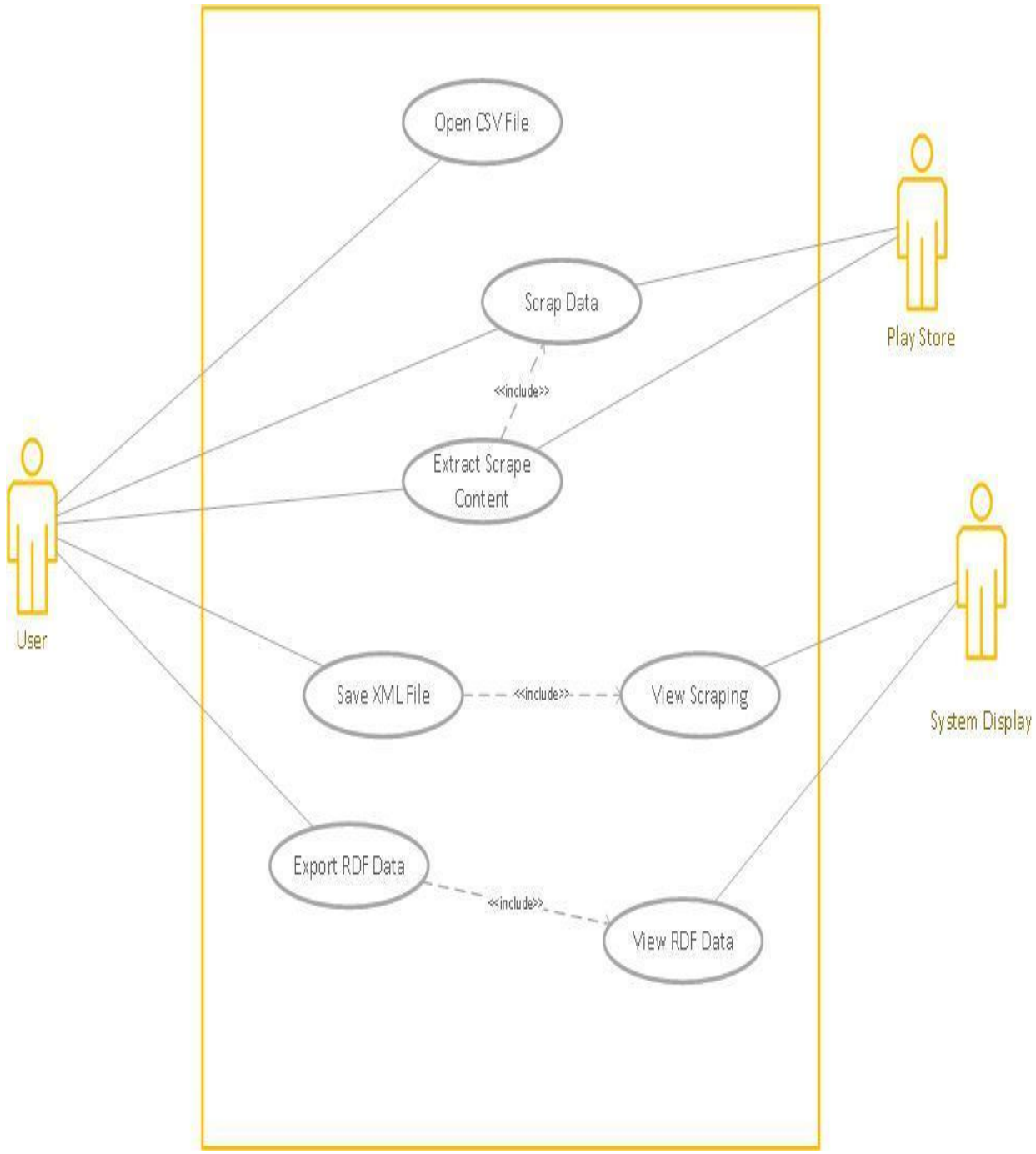
Project Features

The Scraping Tool aims to extract data from Google Play. Furthermore, it will be designed to provide the scraped data to analysis team to analyze and display the data in desired form.

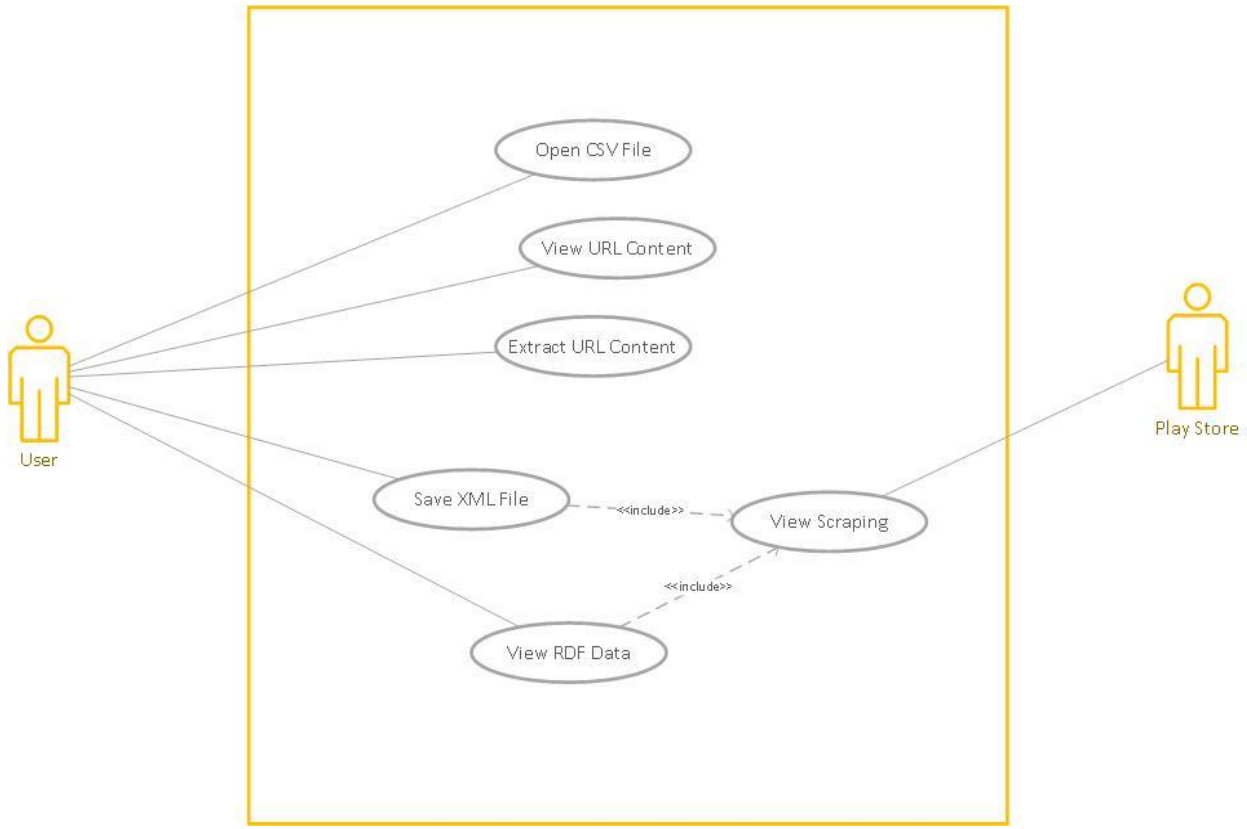
3.1 Project Domain Model



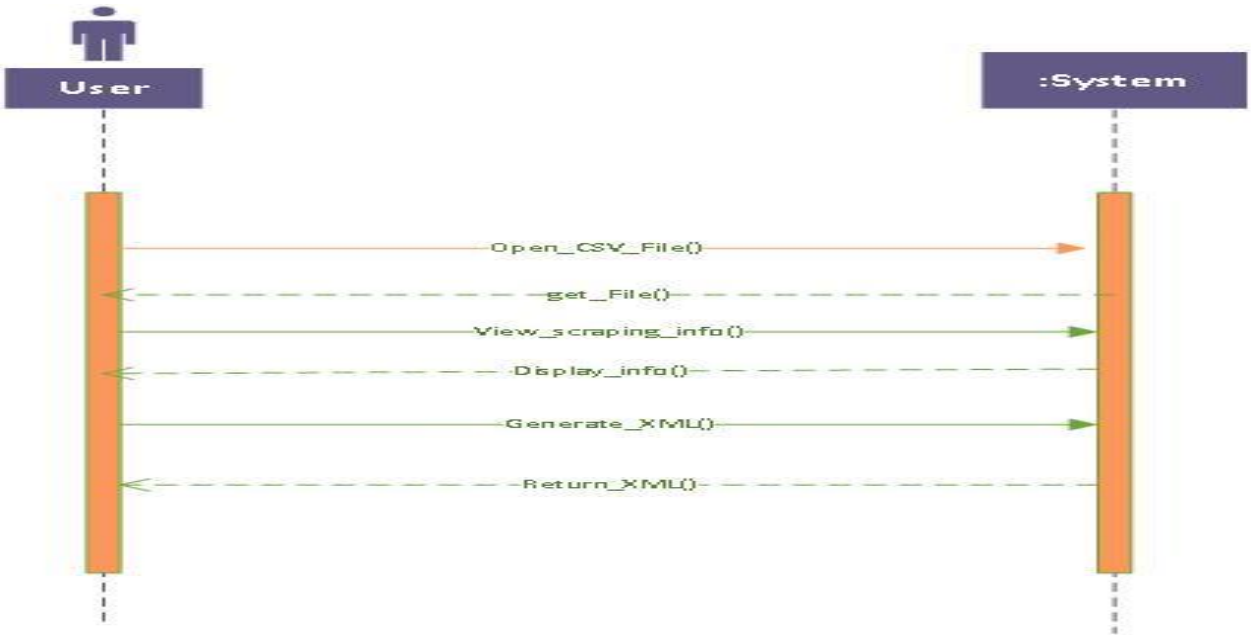
3.2 Scraping Project Use Case Model



3.3) Use case diagram for Scraping



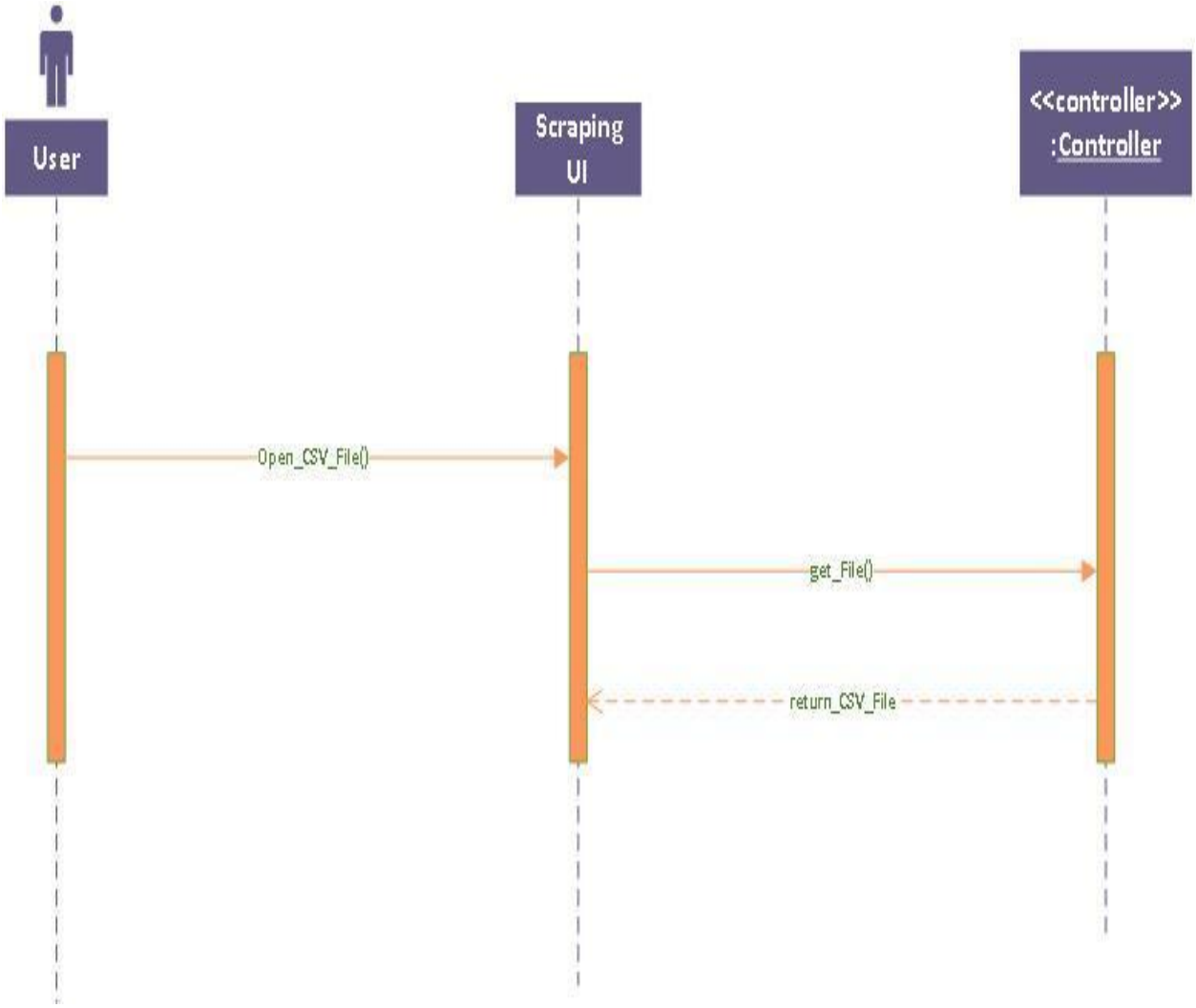
3.4)System Sequence Diagram



1) Use Case 1: Open CSV File

Number	1	
Name	Open a CSV File	
Summary	User Selects a CSV file for scraping	
Priority	1	
Preconditions	CSV file has not been chosen	
Postconditions	CSV file has been selected and is ready for scraping	
Primary Actors	User	
Secondary Actors	System	
Trigger	User clicks the “Display CSV file” button	
Main Scenario	**Step**	**Action**
	1	User selects “Open CSV”
	2	User selects the CSV file from the directory.
Extensions	**Step**	**Branching Action**
	2.a	CSV file doesn't exist. Display error.
	2.b	CSV file not compatible
Open Issues	**Issue**	**Issue Description**
	1	No Supported Environment
Requirements	**REQ #**	**Description**
	1	Java 7 or higher.
	2	Microsoft Excel

Sequence Diagram for UC1:

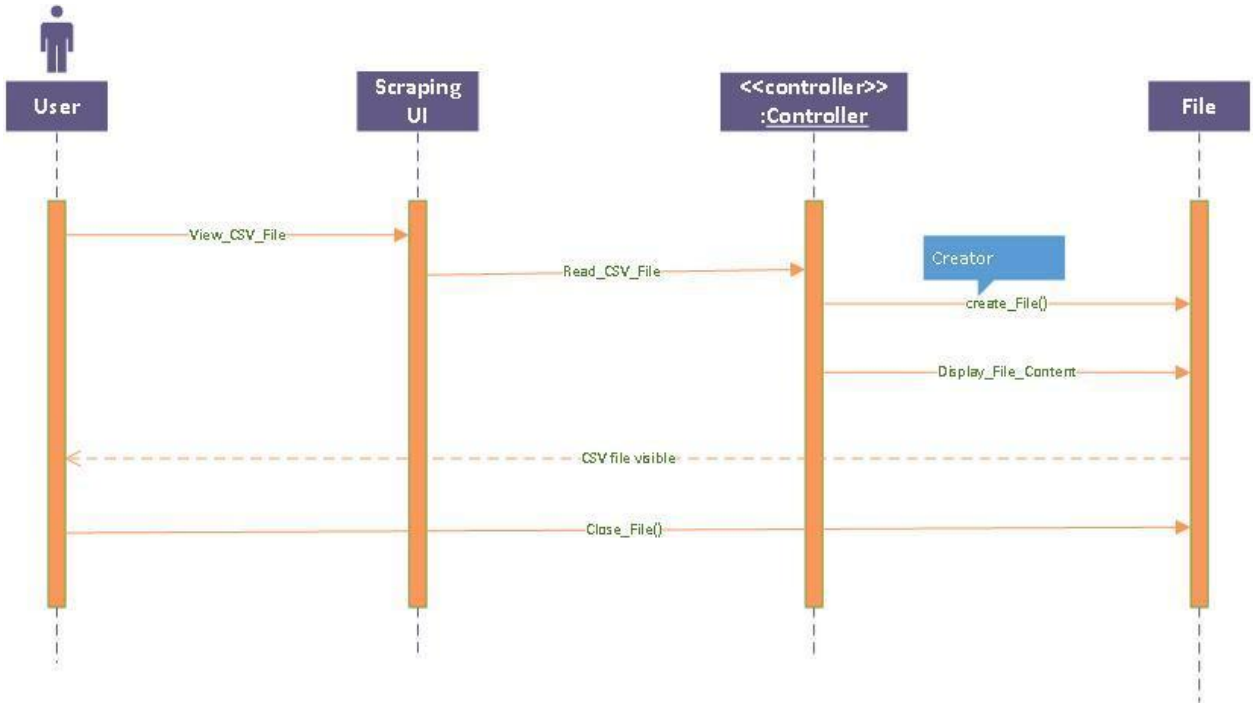


2) Use Case 2: View Content of CSV File

Number	2	
Name	Display Content of the CSV file on screen.	
Summary	Open the CSV file in an Excel document worksheet.	
Priority	5	
Preconditions	CSV file must have been chosen	
Postconditions	All information from CSV file must be displayed	
Primary Actors	User	
Secondary Actors	System	
Trigger	User clicks the “Display CSV file” button.	
Main Scenario	**Step**	**Action**
	1	User selects CSV file.
	2	User clicks on “Display CSV file” button.
	3	CSV file is opened in Excel.
Extensions	**Step**	**Branching Action**
	1.a	Display list of possible CSV files.
	1.b	Chosen CSV file is corrupt or missing.
	1.c	User is prompted to choose another CSV file.
Open Issues	**Issue #**	**Issue Description**
	1	URL doesn’t Exist
	2	Enter Wrong URL
Requirements	**REQ #**	**Description**
	1	Java 7 or higher

	2	Eclipse JDK Environment

Sequence Diagram for UC2:

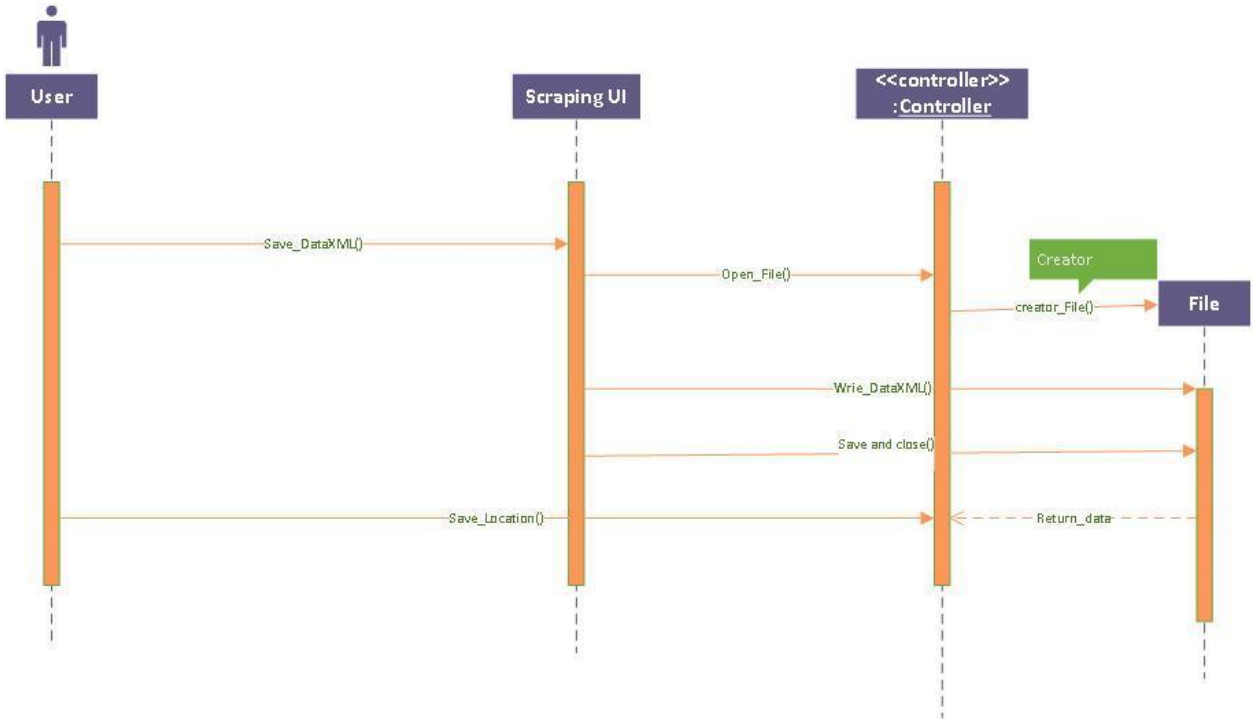


3) Use Case 3: Save Scraped Data as XML

Number	3	
Name	Save Scrapped Results to XML.	
Summary	Stores scraped data from previously defined URLs in an XML file.	
Priority	1	
Preconditions	Data has been Scraped from URLs.	
Postconditions	XML file is generated and saved for Analysis team.	
Primary Actors	User	
Secondary Actors	System	
Trigger	Scraping is done.	
Main Scenario	**Step**	**Action**
	1	Open file for writing.
	2	Write scrape results in XML format.
	3	Close file.
	4	Prompt user for a save location.
	5	Save XML file to desired location.
Extensions	**Step**	**Branching Action**
	1.a	File cannot be opened for writing, warn user via error message.
	1.b	User Selects and invalid file name (containing illegal characters.). Warn the user via error message.
	1.c	User selects file name that already exists. Prompt user to replace old file or pick a new name.

	2.a	Retrieve data from XML
	2.b	Pass arguments and data elements
	2.c	Display the result
	5.a	Store the result
Open Issues	**Issue #**	**Issue Description**
	1	User selects an invalid location (i.e. Admin folders, which user may not have access to). Display an error message.
	2	User
Requirements	**REQ #**	
	1	Java 7 or higher.
	2	Eclipse JDK Environment
	3	Apache Tomcat
	4	Spring Framework

Sequence Diagram for UC3:

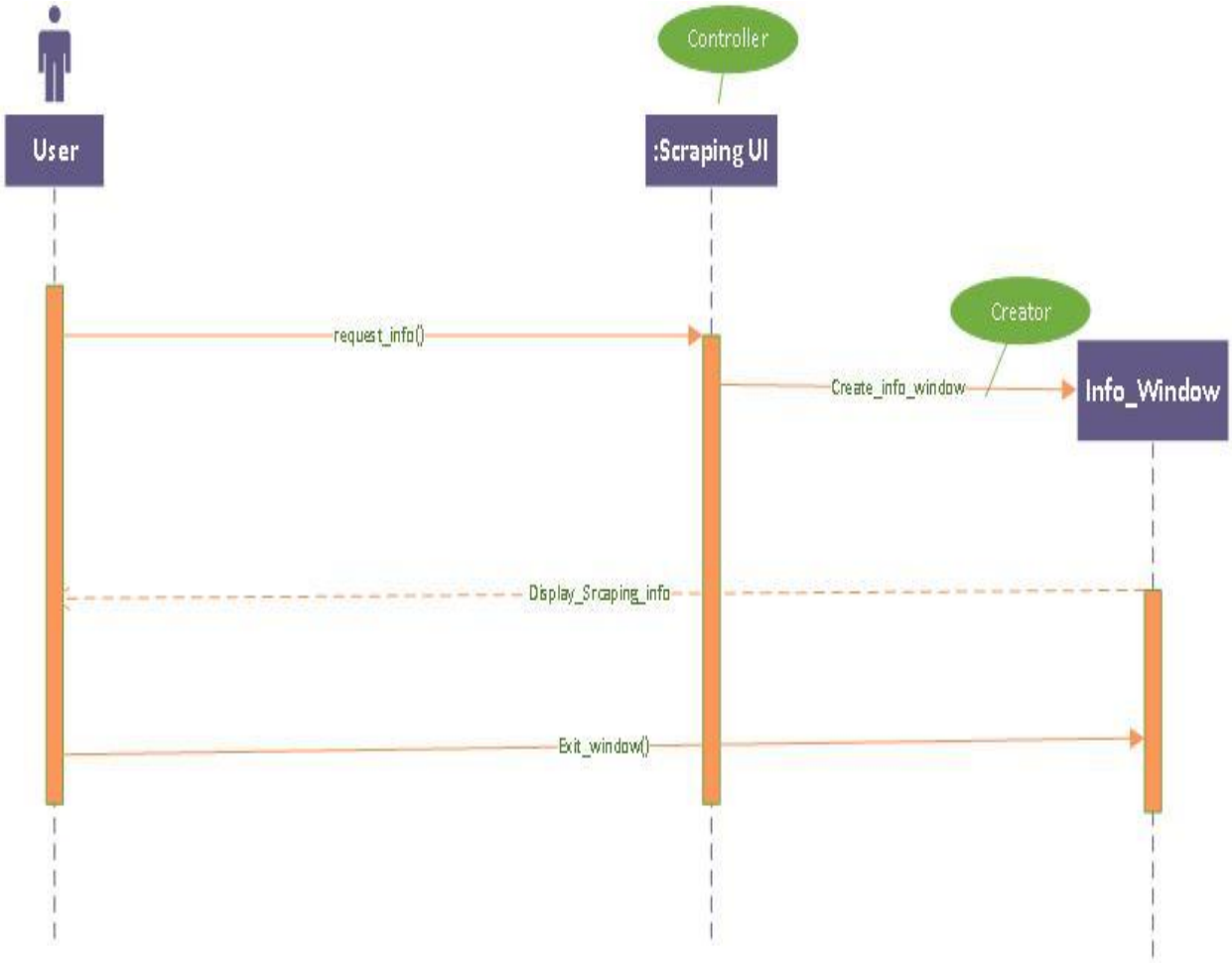


4) Use Case 4: Extract Scrape Content

Number	4	
Name	Display Scraping Information.	
Summary	Display Scraping information in a new window.	
Priority	4	
Preconditions	Data has been scraped, or is being scraped, from URLs.	
Postconditions	Window is created displaying information about the current/previous scrape.	
Primary Actors	User	
Secondary Actors	None.	
Trigger	User click “Scraping Info” button.	
Main Scenario	**Step**	**Action**
	1	User clicks “Scraping Info”.
	2	New window is created displaying various data about the current scrape. (i.e. URLs completed, time elapsed, etc.)
Extensions	**Step**	**Branching Action**
	2.a	Scraping is complete, in which case display total information only. (i.e. Total number of URLs scraped, completion time, etc.)
	4.c	User selects file name that already exists. Prompt user to replace old file or pick a new name.
Open Issues	**Issue #**	**Issue Description**
Requirements	**REQ #**	**Description**
	1	Java 7 or higher.
	2	Eclipse JDK Environment
	3	Apache Tomcat

	4	Spring Framework

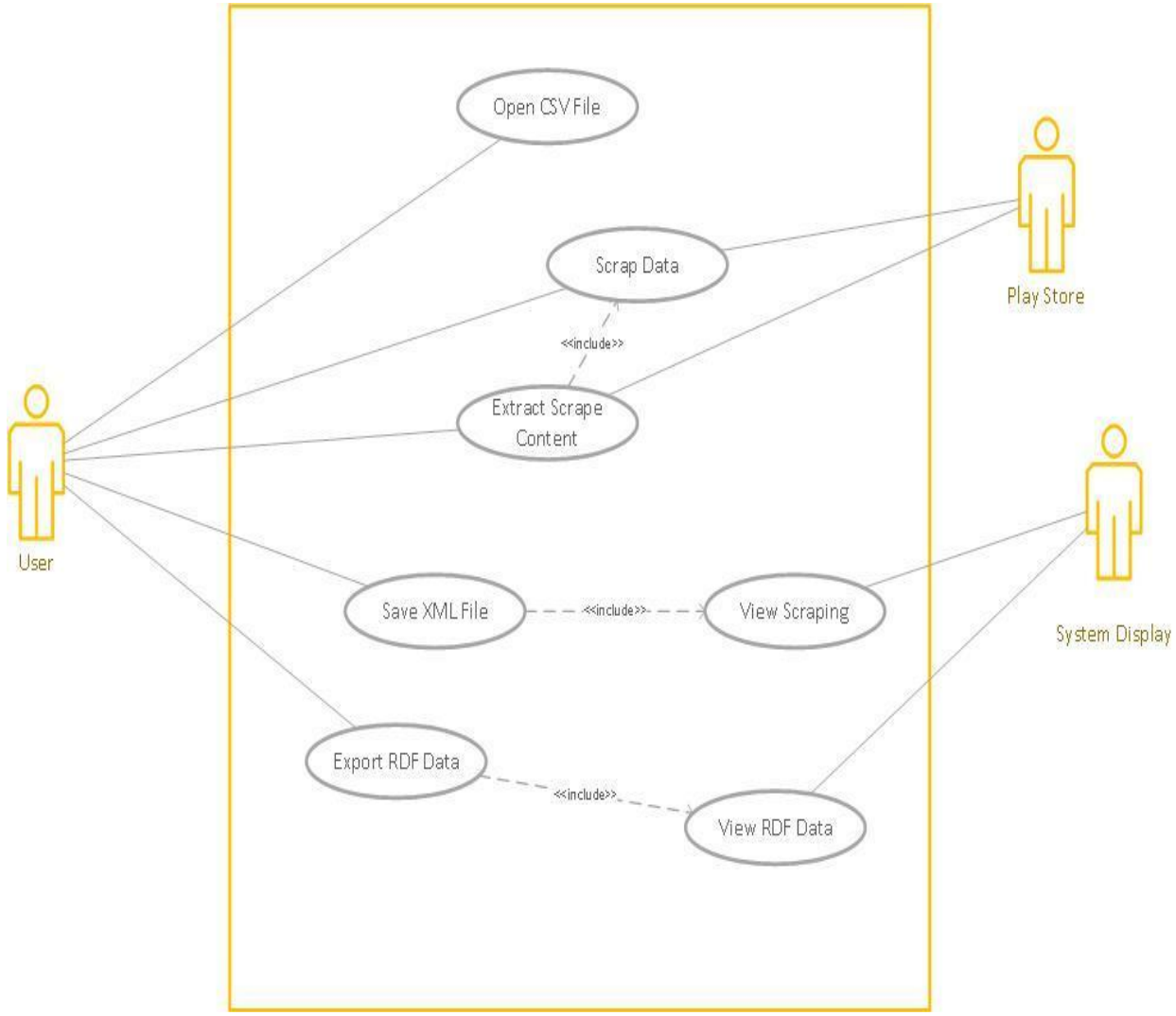
Sequence Diagram for UC4:



3.3 Analysis Model

1. Assumptions
    - Scraping can retrieve data faster than humans
    - Some websites have anti scraping mechanisms.
    - The software tool is independent
    - CSV file follows certain standards and allow the system to parse.

2. Analysis Model and Use case diagram for Scraping



3.4 Priority of Requirements

Table 1.1 presents the prioritisation of the functional and nonfunctional requirements of the Web Course Management System for the first iteration. These priorities are considered and reflected in the architectural and system design decisions relating to this product.

Use case ID	**Priority**	**Reason**
UC #1	High	User has to provide an input to instantiate system
UC #2	Low	User can view the content of CSV
UC #3	High	The Scraped data has to be saved.
UC #4	High	Provides user with the scraped information from CSV

4. System Design

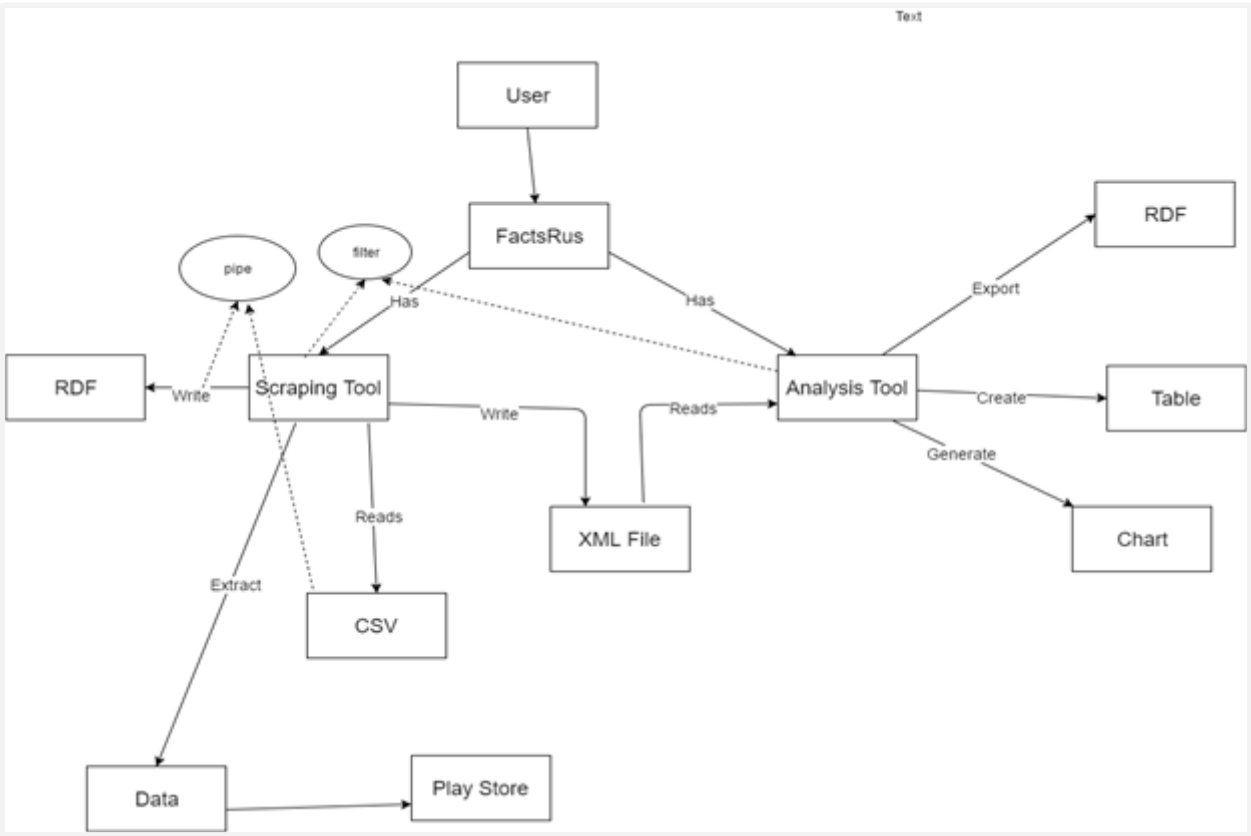
4.1 System Architecture

Pipe and filter
This architectures best suits when a lots of transformations are required and flexibility is required with great amount of robustness.
It is type of data flow architecture. It consist of a number of components which are connected to each other in order to share and transform data. This architectures follows a simple sequence. As a part of this architecture our main components are

1. Scraping tool

2. Analysing tool

These two are fetching and transforming data before passing it to other components via connectors (pipes). Both are connected to a number of input and output pipes.



Pipes are used as a connector to send data to different filters.

4.2 Subsystem Architecture

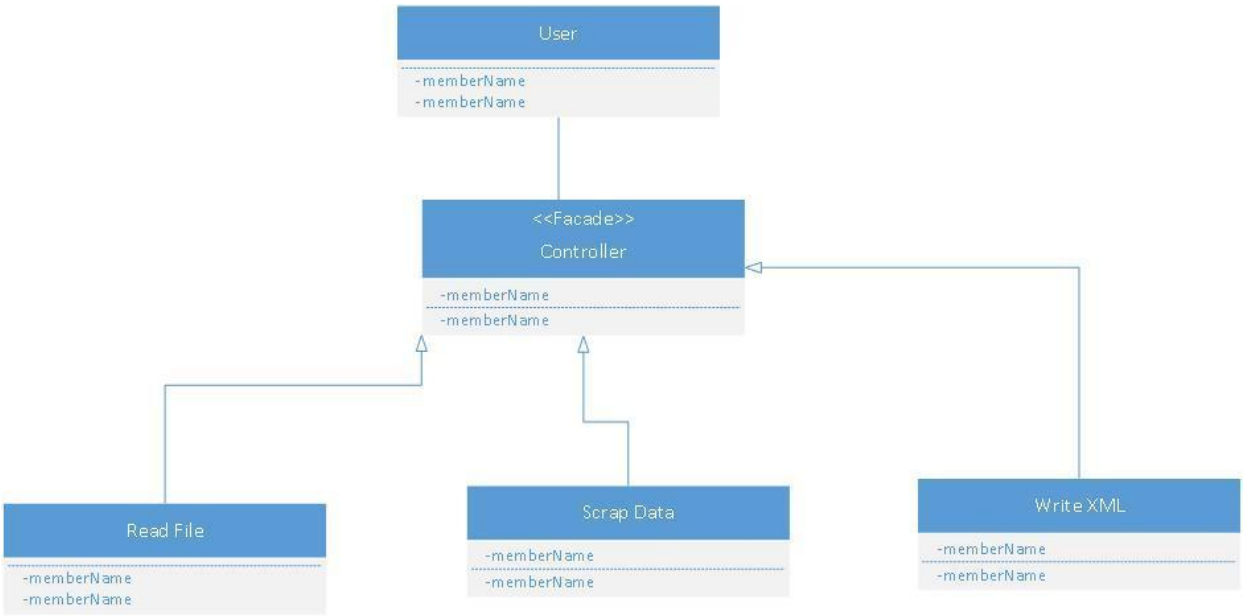
4.2.1 Purpose

Facade pattern hides the complexities of the system and provides an interface to the client using which the client can access the system. This type of design pattern comes under structural pattern as this pattern adds an interface to existing system to hide its complexities. This pattern involves a single class which provides simplified methods required by client and delegates calls to methods of existing system classes.

BENEFITS

1. Shields clients from subsystem classes
2. It can reduce the number of objects that clients deal with.
3. It promotes weak coupling between subsystem and its clients.
4. It helps in layering the system.
5. It helps in eliminating circular dependencies.

4.2.2 Subsystem Class Structure



5. Detailed Design

A detailed design allows us to evaluate design options prior to actually implementing them. It can potentially save a ton of needless work spent on an implementation that was deeply flawed due to a high level design choice and had to be extensively rewritten. This section contains detailed design of our Scraping Project.

5.1 Algorithm for Subsystem-Scraping

Detail Functionality of Scraping-1	
Method Name	scrapeFile(File f)
Class Name	ReadFile
Functionality	In this method scrape the CSV file and extract list of Url's
Pseudo Code	Start data.startWith("\" https"); ScrapeData.scrapeUrl(); End
Return Type	Return Array_Listurl

Detail Functionality of Scraping-2	
Method Name	scrapeUrl(ArrayList Url)
Class Name	ScrapeData
Functionality	In this method for each url it will retrieve key elements
Pseudo Code	Start doc=Jsoup.connect(url).get(); Element appName=doc.select("div.id-app-title").first(); Element rating=doc.select("div.right-info span").first(); End
Return Type	Return Array_ListElements

Detail Functionality of Scraping-3	
Method Name	writeXML(ArrayList applicationName,ArrayList applicationRating)
Class Name	WriteXML
Functionality	In this method generates the XML file

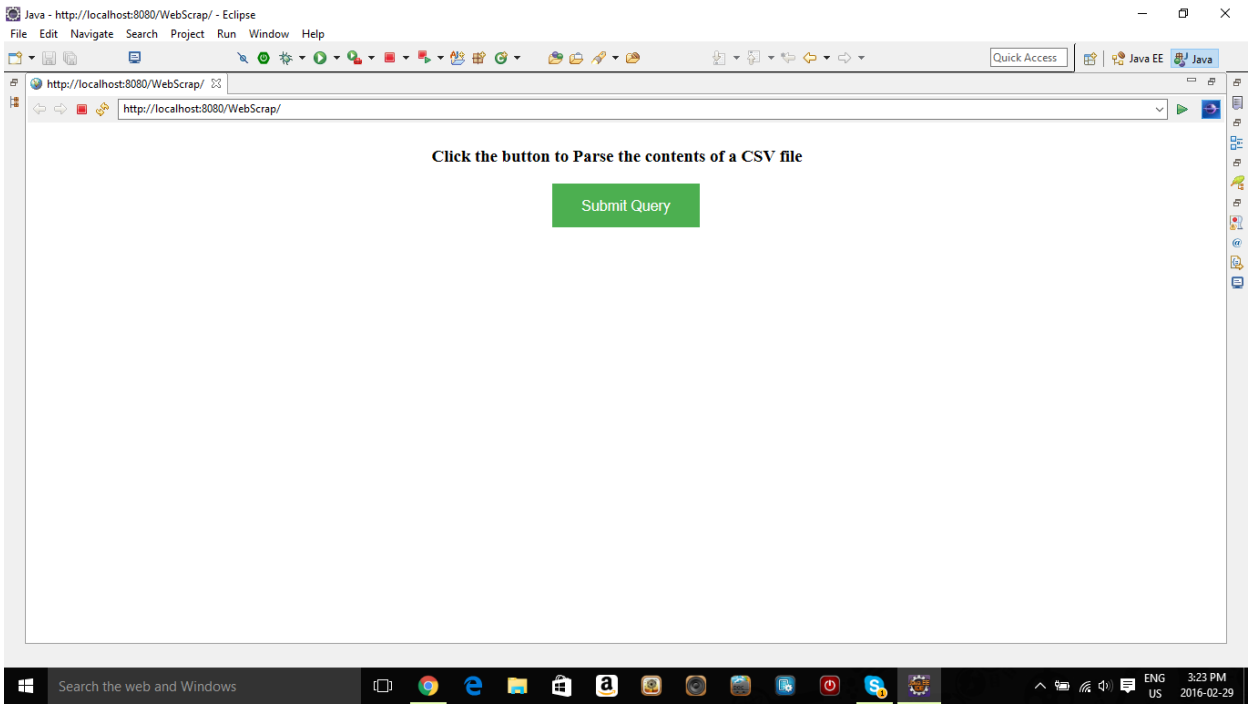
Pseudo Code	Start
	DocumentBuilderFactory docFactory=
	DocumentBuilderFactory.newInstance();
	DocumentBuilder
	docBuilder=docFactory.newDocumentBuilder();
	Document doc=docBuilder.newDocument();
	Element rootElement=doc.createElement("Apps");
	doc.appendChild(rootElement);
	Element appName=doc.createElement("appName");
	appName.setAttribute("name",applicationName.get(i));
	rootElement.appendChild(appName);
	Element dataPoint=doc.createElement("dataPoint");
	dataPoint.appendChild(doc.createTextNode(applicationRating.
	get(i)));
	appName.appendChild(dataPoint);
	End

Return Type	Return file_XML

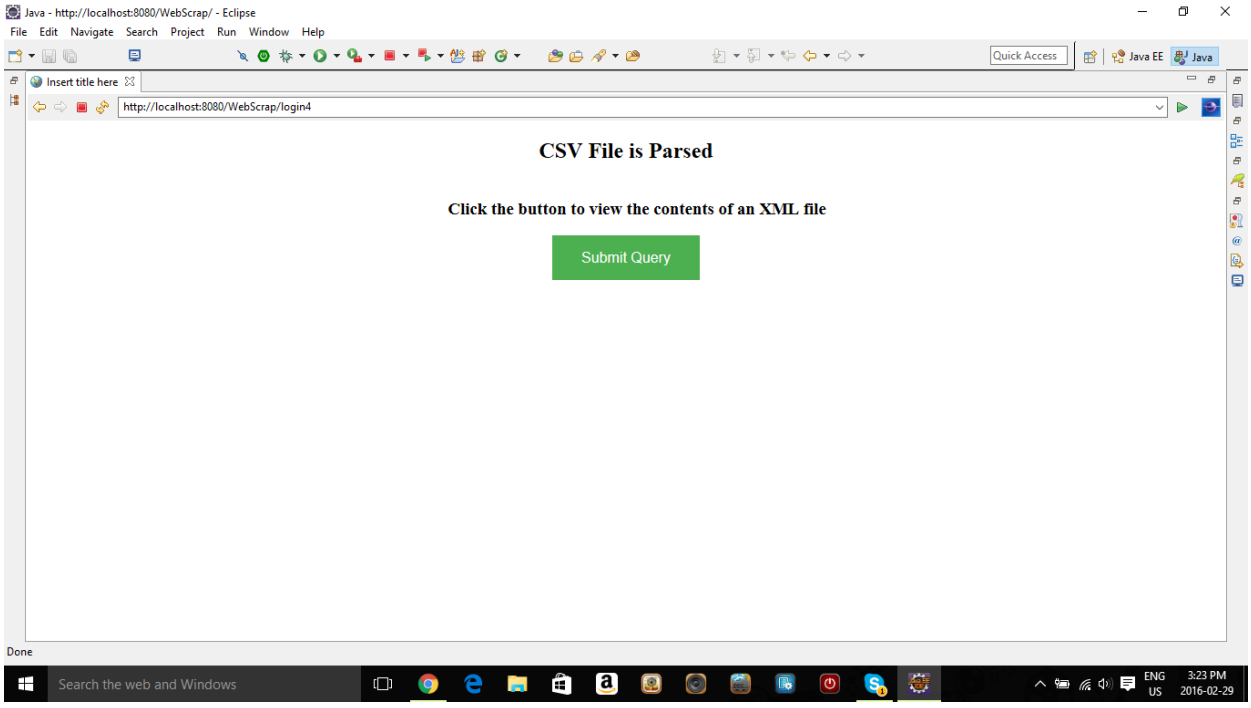
6. Prototype

6.1 Scraping CSV

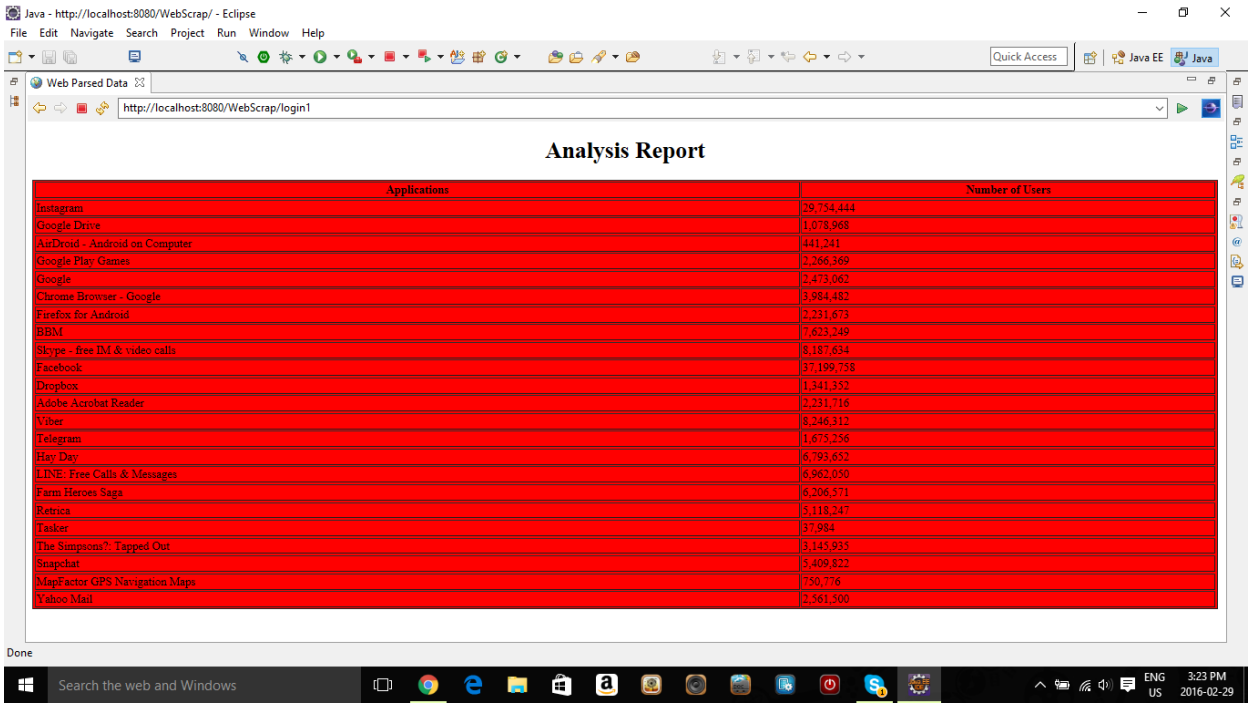
Once users have been clicked on submit query, the application will scrap the data from csv file and retrieve all the scrapped data from each URL



6.2 Saved Parsed data as XML



6.3 View Scraping Information



Applications	Number of Users
Instagram	29,754,444
Google Drive	1,078,968
AirDroid - Android on Computer	441,241
Google Play Games	2,266,369
Google	2,473,062
Chrome Browser - Google	3,984,482
Firefox for Android	2,231,673
BBM	7,623,249
Skype - free IM & video calls	8,187,634
Facebook	37,199,758
Dropbox	1,341,352
Adobe Acrobat Reader	2,231,716
Viber	8,246,312
Telegram	1,675,256
Hay Day	6,793,652
LINE: Free Calls & Messages	6,962,050
Farm Heroes Saga	6,206,571
Retrica	5,118,247
Tasker	37,984
The Simpsons?: Tapped Out	3,145,935
Snapchat	5,409,822
MapFactor GPS Navigation Maps	750,776
Yahoo Mail	2,561,500