**HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY**

**School of Information and Communication Technology**

­­­-----------------------------------

A picture containing text, sign

Description automatically generated

**FINAL PROJECT REPORT**

**Group 1**

IT3100E – Object Oriented Programming

Class: 139411

Lecturer: PhD. Trinh Tuan Dat

Semester 20221

Hanoi, June 2023

**TABLE OF CONTENTS**

[**CHAPTER 1.** **INTRODUCTION** 4](#_Toc139154249)

[**CHAPTER 2.** **PROBLEM DESCRIPTION** 5](#_Toc139154250)

[**I.** **Study Case** 5](#_Toc139154251)

[**II.** **Problem Solving Strategies** 5](#_Toc139154252)

[**III.** **Technologies, Frameworks, Libraries in use** 6](#_Toc139154253)

[**CHAPTER 3.** **REQUIREMENT ANALYSIS** 8](#_Toc139154254)

[**I.** **Base data Specification** 8](#_Toc139154255)

[**II.** **Business process analysis** 9](#_Toc139154256)

[**1.** **Data searching process activity diagram** 9](#_Toc139154257)

[**2.** **Web scraping process activity diagram** 10](#_Toc139154258)

[**III.** **Use Case diagram** 11](#_Toc139154259)

[**CHAPTER 4.** **DESIGN ANALYSIS** 13](#_Toc139154260)

[**I.** **System Design: Package diagram** 13](#_Toc139154261)

[**II.** **Program Design: Class diagram** 14](#_Toc139154262)

[**1.** **Package “model”** 14](#_Toc139154263)

[**2.** **Package “crawler”** 17](#_Toc139154264)

[**2.1.** **Package crawler.util** 17](#_Toc139154265)

[**2.2.** **Package crawler.thoiky** 17](#_Toc139154266)

[**2.3.** **Package crawler.nhanvatlichsu** 18](#_Toc139154267)

[**2.4.** **Package crawler.sukienlichsu** 19](#_Toc139154268)

[**2.5.** **Package crawler.ditichlichsu** 19](#_Toc139154269)

[**2.6.** **Package crawler.lehoi** 20](#_Toc139154270)

[**3.** **Package “application”** 21](#_Toc139154271)

[**4.** **Data storing design** 23](#_Toc139154272)

[**CHAPTER 5.** **RESULTS & PROGRAM TESTING** 24](#_Toc139154273)

[**I.** **Statistics of data** 24](#_Toc139154274)

[**II.** **Final program testing** 24](#_Toc139154275)

[**CHAPTER 6.** **OOP TECHNIQUES IN USE** 28](#_Toc139154276)

[**I.** **OOP Techniques** 28](#_Toc139154277)

[**1.** **Encapsulation:** 28](#_Toc139154278)

[**2.** **Inheritance:** 28](#_Toc139154279)

[**3.** **Polymorphism:** 28](#_Toc139154280)

[**4.** **Abstraction:** 28](#_Toc139154281)

[**5.** **Generic Programming:** 28](#_Toc139154282)

[**II.** **Algorithms:** 28](#_Toc139154283)

[**CHAPTER 7.** **CONCLUSION & FUTURE DEVELOPMENT** 29](#_Toc139154284)

[**APPENDIX A.** **INSTRUCTION & TEAM CONTRIBUTION** 30](#_Toc139154285)

[**REFERENCES** ………………………………………………………………………………………31](#_Toc139154286)

[**TABLE OF FIGURES** 32](#_Toc139154287)

1. **INTRODUCTION**

The course IT3100E – Object Oriented Programming provides students with object-oriented concepts, languages, principles, and techniques. Students are first introduced to object-oriented technology, the overview of Java programming language and Unified Modelling Language (UML). Four object-oriented programming principles, i.e., abstraction, encapsulation, inheritance, and polymorphism, are then delivered out also. Aggregation, association, abstract class, interface as well as generic programming, exception handling, and graphical user interface (GUI) programming with Java, during the course, are presented to students.

This hands-on project is assigned to groups of students along with the course to apply the knowledge related to the design of object-oriented programs to an object-oriented application solving real-world problems.

In this report, our group, under the guidance of Prof. Trinh Tuan Dat, will present the ways in which our group deal with the project assignment in details, which includes system design, application design, algorithms, GUI, technologies, etc. every component of our programme besides team contributions and project process management.

Our team members include:

|  |  |
| --- | --- |
| **Student’s name** | **Student’s ID** |
| Nguyen Trong Huy | 20210451 |
| Bui Phuong Nam | 20215227 |
| Tran Duc Nam | 20215228 |
| Trinh Giang Nam | 20215229 |
| Nguyen Chinh Minh | 20215224 |
| Hoang Minh Quan | 20215236 |

Figure 1 Group 1’s members

1. **PROBLEM DESCRIPTION**
2. **Study Case**

There are numerous websites that provide information about the history of Vietnam, such *as https://nguoikesu.com*, *Wikipedia*, *DBPedia*, and more. The task is to search and automatically gather data about the history of Vietnam from these websites and interlink the collected data. The entities to be collected include:

- Historical dynasties of Vietnam

- Historical figures of Vietnam

- Tourist destinations and historical landmarks

- Cultural festivals

- Historical events in Vietnam

Each entity needs to have an identifier and various attributes. Importantly, the entities should be interconnected.

Please note that:

- Data gathering process must be solved automatically.

- Consistency in naming attributes for each entity.

However, there are many criteria to evaluate your system:

- A diverse range of entities for each type from multiple sources.

- Ensure accuracy.

- Data integration from multiple sources is required.

For the final version of your application, the collected data needs to be stored in either JSON or CSV format. Subsequently, provide search functionality and display information for end users.

1. **Problem Solving Strategies**

In our very first stage to handle this problem, developing a strategy to develop our system serves as the bridge to the success of the project.

To facilitate the software development process under such requirements, we have detailed plans for each stage of developing our product for the study case in a period of 6 weeks of project. The way we allocate individuals to different jobs will be shown later in the appendix of this report.

|  |  |  |
| --- | --- | --- |
| **Stage** | **Description** | **Period** |
| Requirement Analysis | - Analyse the requirement of the study case.  - Feature the specific functionality that the system should have.  - Examine the websites that are helpful for data crawling.  - Study the process to crawl data from a website | 3-4 days |
| Design | - System design  - Program design | 1 week |
| Implementation | - Coding  - GUI implementation | 3 weeks |
| Integration & Testing | - Integrate each component of system  - Mannual Testing | 4-5 days |
| Realease | - Deliver final report, source code, etc.  - Presentation & demo video | - |

Figure 2 Details of our development process

1. **Technologies, Frameworks, Libraries in use**

This table describes tools that we use to build up our program.

|  |  |
| --- | --- |
| **Tools** | **Objectives** |
| E(fx)clipse  Windows 64 bits | IDE |
| GitHub | Version Control |
| Asana | Project Management |
| SceneBuilder | GUI design |
| Java | Programming Language |
| CSS | Styling language |
| Maven Project | Java library management tool |
| Astah | UML design tool |

Figure 3 Technologies in use

Here are some Java libraries we use during the implementation process.

|  |  |  |
| --- | --- | --- |
| **Library** | **Version** | **Description** |
| JDK | 19.0.0 | Java core library |
| gson | 2.9.0 | Converting Java objects to JSON |
| json-simple | 1.1.1 | Parsing and generating JSON |
| jsoup | 1.15.3 | Web scraping and HTML parsing |
| json | 20220320 | A lightweight data interchange format |
| javaFX | 16 | GUI coding (event handling, …) |

Figure 4 Java libraries

1. **REQUIREMENT ANALYSIS**
2. **Base data Specification**

When attempting to do a survey related to the entities, we realise that these entities should have the following information for the final system.

|  |  |  |
| --- | --- | --- |
| **Entity** | **Attribute** | **Description** |
| Thời kỳ  (Dynasties) | Tên (Name) | This enity depicts all dynasties of Viet Nam History |
| Kinh đô (Capital) |
| Các đời vua (Kings) |
| Người sáng lập (Founder) |
| Thời gian tồn tại (Period) |
| Quốc hiệu (Nation’s name) |
| Nhân vật lịch sử  (Figures) | Tên (Name) | This entity lists all Vietnamese historical figures, which may also include Vua (King) |
| Năm sinh (Year of birth) |
| Năm mất (Year of death) |
| Quê quán (Hometown) |
| Thời kỳ (dynasty) |
| Sự kiện lịch sử  (Events) | Tên (Name) | This concept shows information of Vietnamese historical events. This also includes Chiến Tranh (Vietnamese War) |
| Thời gian (Period) |
| Thời kỳ (Dynasty) |
| Nhân vật liên quan (Figure) |
| Nội dung (Content) |
| Lễ hội  (Festivals) | Tên (Name) | Describe Vietnamese historical festivals |
| Thời gian (Period) |
| Địa điểm (Place) |
| Nhân vật liên quan (Figure) |
| Di tích  (Relics) | Tên (Name) | List all Vietnamese relics |
| Địa điểm (Place) |
| Loại di tích (Classification) |
| Năm (Year of recognition) |
| Nhân vật liên quan (Figure) |

Figure 5 Base data

1. **Business process analysis**

To get into the process of the system, we first need to estimate the business requirement. Based on that, we then can divide the business process into two main processes: data viewing process and web scraping process. The first term describes the way end users could search for data in the final program, from that we can concentrate on GUI design that meet the user experience. While the later one describes the way the system could get the data from the website automatically and then supply the data for the backend process.

For each process analysis, we need to make enquiries about what the clients want for the final version of the system besides how the data could be obtained from the website to deliver the activity diagrams, which can be considered the foundation to system design.

By doing this stage, we can easily approach the system design and progamming design regarding client-oriented requirements and minimal error.

1. **Data searching process activity diagram**

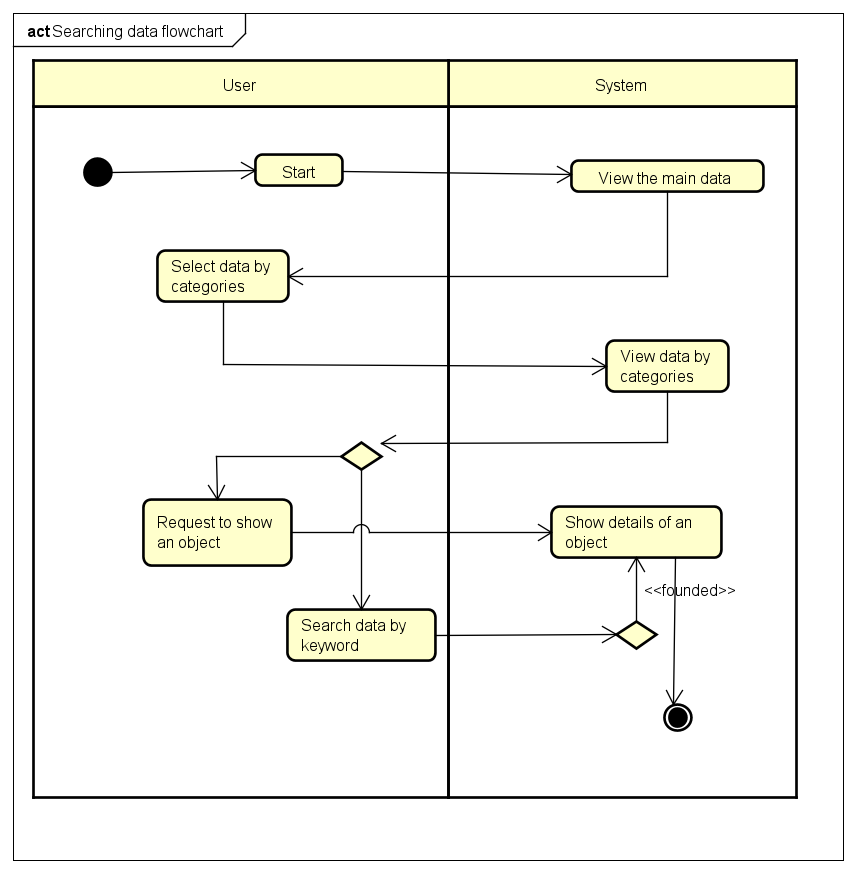
****

Figure 6 Data viewing process analysis

1. **Web scraping process activity diagram**

Based on what we discovered about the web scraping process, which shows below, we propose the web scraping process activity diagram.

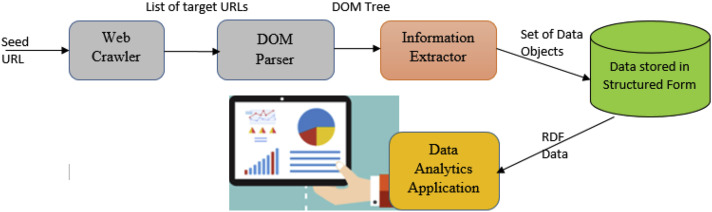


Figure 7 Actual web scraping process

This activity diagram might affect that way we develop class diagram for web scraping subsystem.

A screenshot of a computer screen

Description automatically generated with low confidence

Figure 8 Web scraping process activity diagram

1. **Use Case diagram**

A picture containing text, diagram, screenshot, line

Description automatically generated

Figure 9 Use case diagram.

Use case “View data” details:

|  |  |
| --- | --- |
| **UC** | View data |
| **Primary actor** | User |
| **Secondary actor** | No |
| **Pre-condition** | - The web data scraping process has been successfully completed.  - The scraped data has been stored in a structured format. |
| **Post-condition** | - The User is able to view the scraped data. |
| **Main flows** | 1. The data scraping process has been successfully completed. The scraped data has been stored in a structured format. 2. The application presents a list of available datasets or categories of scraped data to the users. 3. The user selects a specific data to view. 4. The user can navigate or search data in format. |
| **Regular variants** | - The user exits the program |
| **Exceptional flows** | No |

Figure 10 Use case "view data" details

Use case “Gather data” details:

|  |  |
| --- | --- |
| **UC** | Gather data |
| **Primary actor** | No |
| **Secondary actor** | Sources |
| **Pre-condition** | - The system requires to access the source to collect data |
| **Post-condition** | - Successful connection |
| **Main flows** | 1. After connecting successfully to the data source, it gathers the source code. 2. Extract source code which includes data. 3. Store data in structured format. |
| **Regular variants** | - No |
| **Exceptional flows** | - Failed connection  - Timeout request |

Figure 11 Use case "gather data" details.

Use case “Scrap Web” details:

|  |  |
| --- | --- |
| **UC** | Scrap Web |
| **Primary actor** | Web Scraper |
| **Secondary actor** | No |
| **Pre-condition** | - Web Scraper has access to the target websites.  - The Web Scraper is configured with the necessary scraping rules or algorithms. |
| **Post-condition** | - Relevant data from the target websites has been scraped and extracted. |
| **Main flows** | 1. The Web Scraper identifies the target websites. 2. The Web Scraper establishes a connection with the target websites. 3. For each web page, the web scraper applies the defined scraping rules or algorithms to extract the desired data. 4. The web scraper stored data in structured format. |
| **Regular variants** | - No |
| **Exceptional flows** | - The Web Scraper logs the error or notifies the system administrator.  - The Web Scraper may attempt to retry the scraping process or skip the problematic web page.  - The Web Scraper detects changes in the target websites' structure or conten |

Figure 12 Use case "scrap web" details

1. **DESIGN ANALYSIS**
2. **System Design: Package diagram**

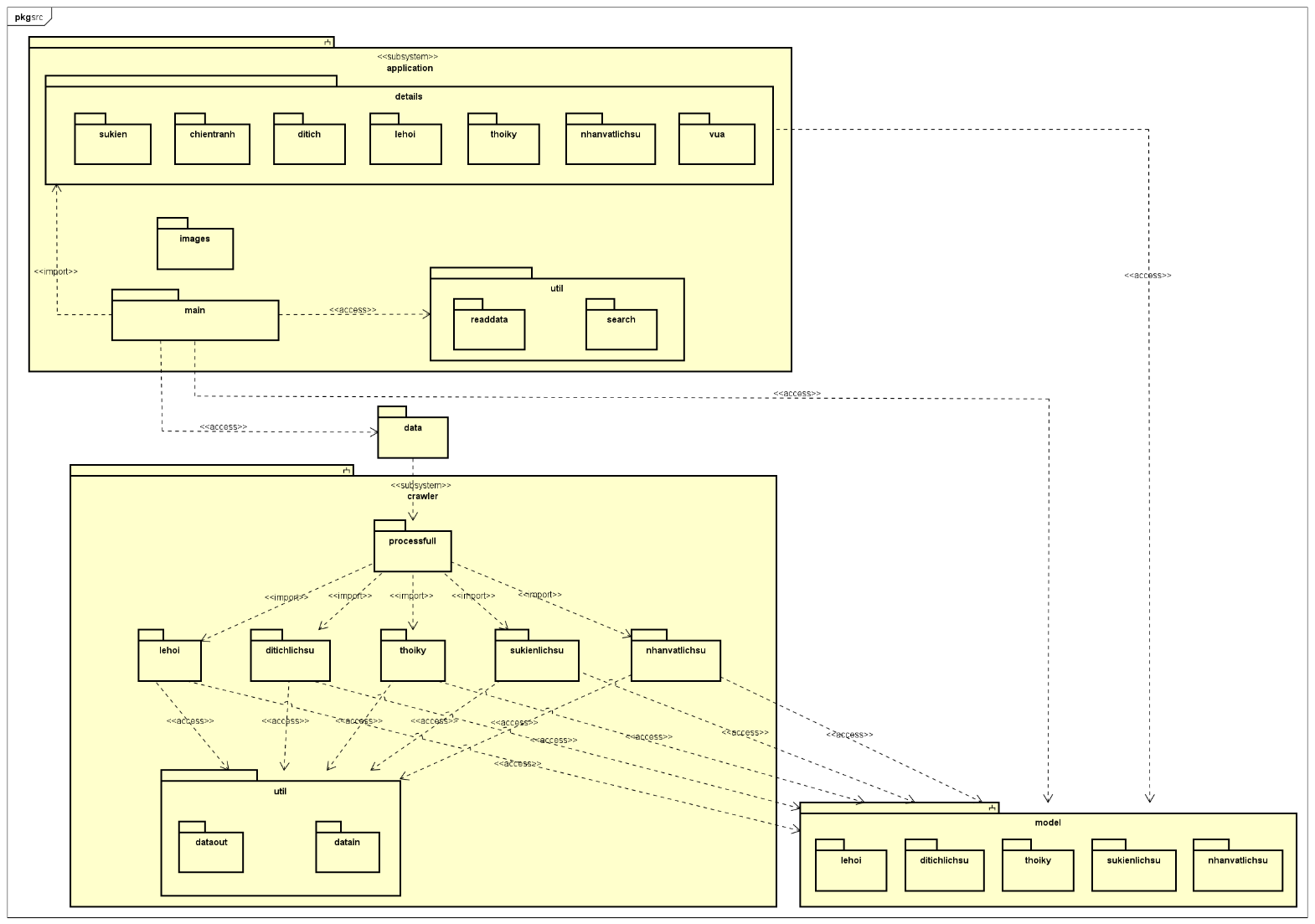


Figure 13 Package diagram of system design

The above figure shows the overall system design, which is described by package diagram.

Based on how data should flow in our system, we derive the system design.

In our system we divide into 2 subsystems: crawler, application, and the base model.

Package “*model*” defines how data should be stored in our system, and then when data is scraped from the website, it could be processed into the way how this model defines.

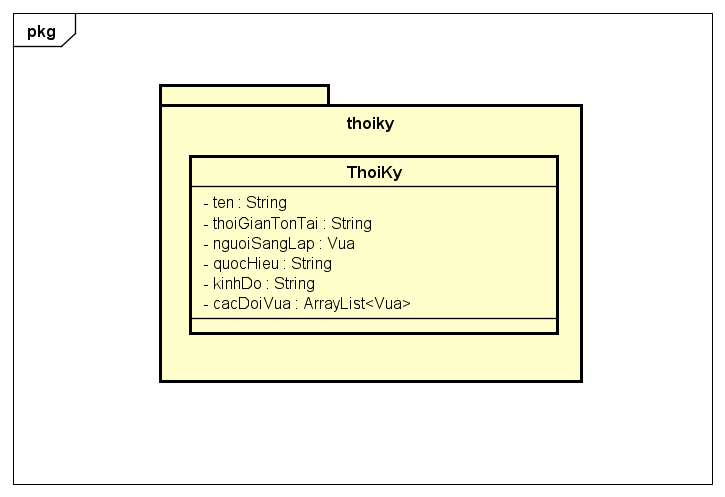
The subsystem “*crawler*” plays the role as a processing stage. It provides the stage and instruction to crawl data from the website based on the mould package “*model*” defines. In this sub system, there are many small packages: *lehoi*, *ditichlichsu*, *thoiky*, *sukienlichsu*, *nhanvatlichsu*, *processfull* and an *util* package. “*Util*” package owns the abstract classes, interfaces that the others should follow and crawl data. “*processfull*” package is the final stage where collects data from the others and stored it in Json file in the data package outside the subsystem.

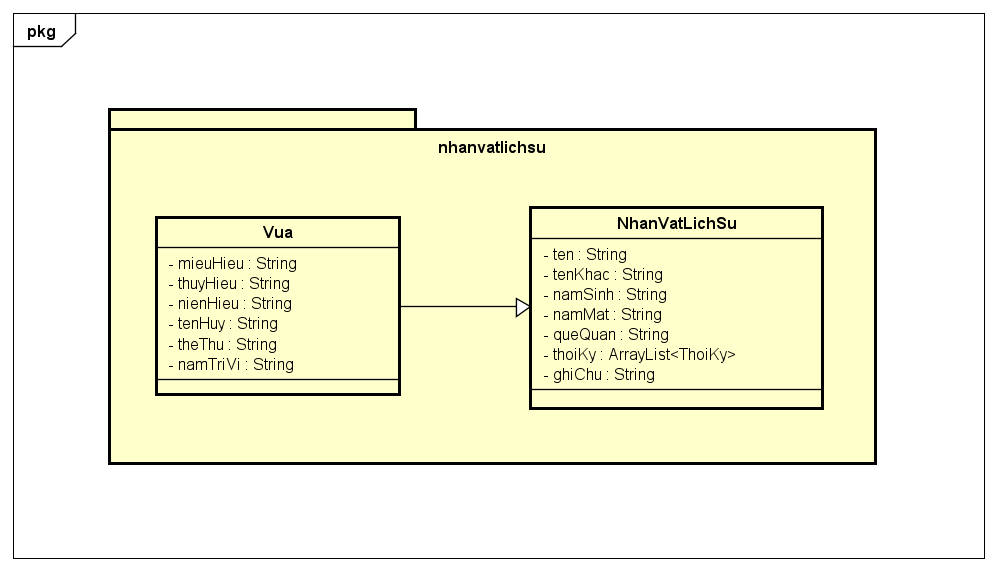
The subsystem “*application*” provides GUI for end user where collects data from data package which is supplied by subsystem “crawler” and follows the rules given by base package model. “*Util*” package gives the functionality that the application performs, and the end users can interact with the most outer layer of the system. The “*details*” package provides users with pop-ups when they want to view the details of an object. While “*images*” package provides media, “*main*” package provides the final application.

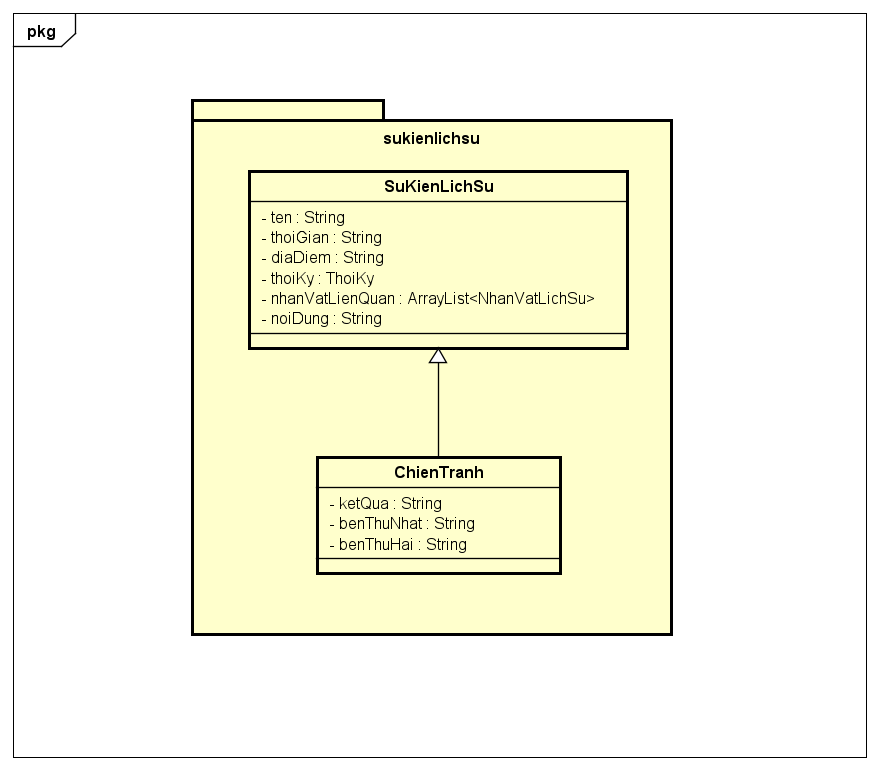
1. **Program Design: Class diagram**

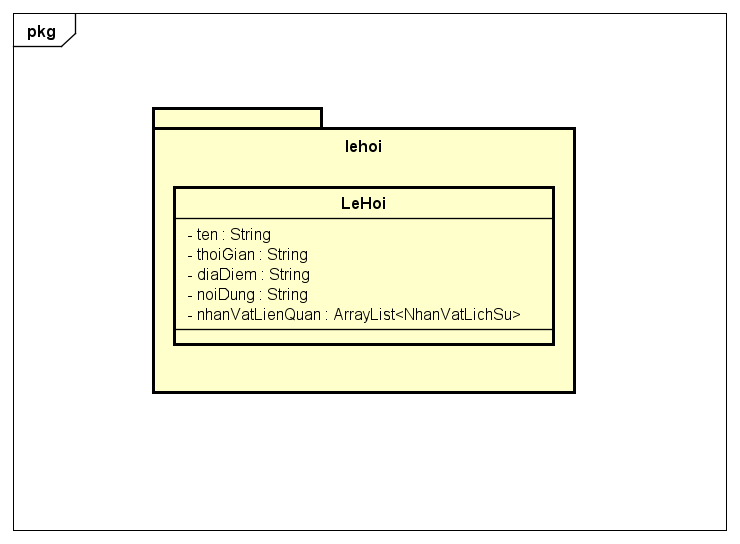
As we define above, the source code will be stored in that structure. For each package we come to the details by giving class diagram.

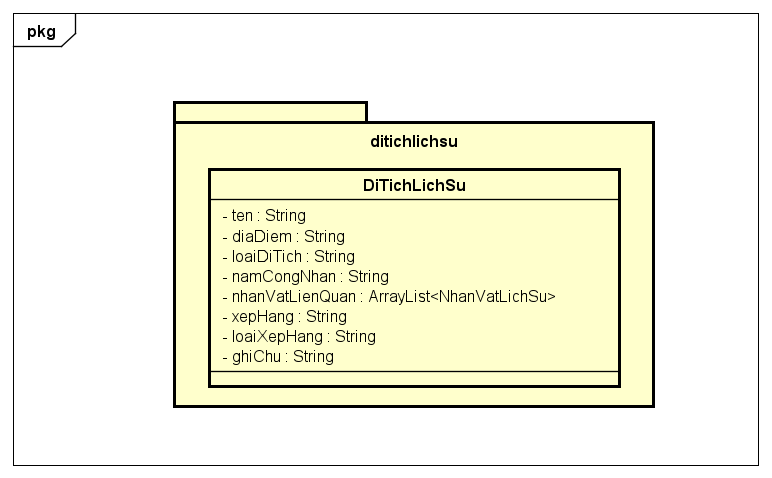
1. **Package “model”**











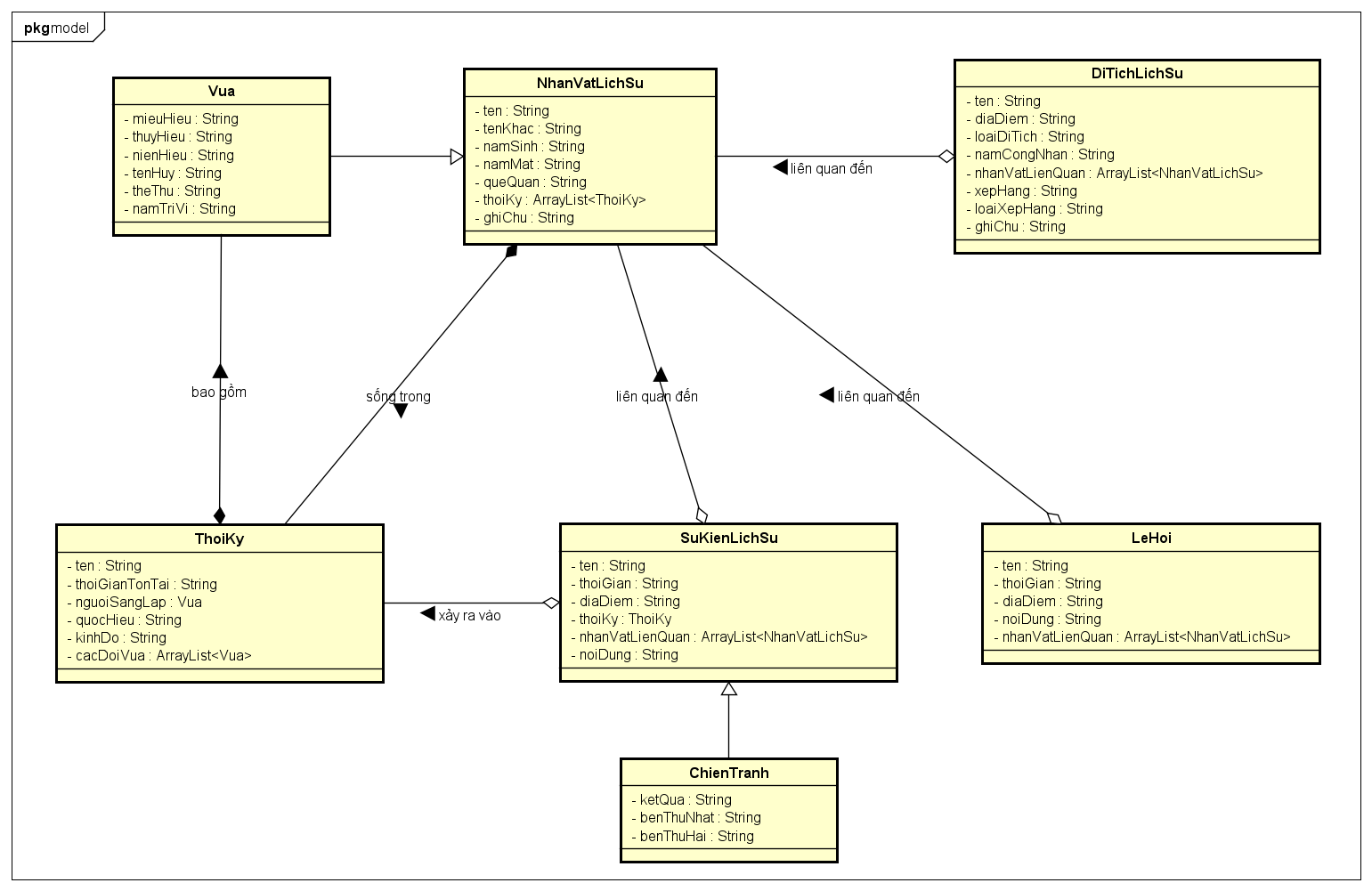


Figure 14 "model" class diagram

The following figure describes the class diagram in the package “*model*”. This indicates all attributes and getter/setter methods that our system should follow and collect data from websites. There are 8 relationships between each pair of classes: 2 generalization relationships, 2 composition relationships, and 4 aggregation relationships.

The number of attributes for each entity and the relationship between all entities are described in detail in the figure. The attribute and relationship are decided based on the requirement analysis and the fact what we can collect from the real website, which improves the quality of the data collected.

1. **Package “crawler”**
   1. **Package crawler.util**

**A screenshot of a computer

Description automatically generated with low confidence**

Figure 15 crawler.util package class diagram

In package crawler.util, which provides attributes and methods that the other classes. There are two smaller packages in this one: datain, and dataout.

Package crawler.util.datain helps to connect to web server and extract data. This was built based on the web scrap process. Firstly, we created an abstract class “AWebConnect”, which has two attributes: url (direct link to website to scrap) and doc (document contains the DOM element of web we need to crawl). Secondly, we created an interface “Iscraping”, which serves to scrap specific data from websites. Finally, an abstract class “AGetData” extends “AWebConnect” and implements Iscraping, form a general process to scrap data from a website.

Package crawler.util.dataout provides two interfaces: IWrtiteJson, which helps to convert data after scrap from web, then put in the model and write to Json file and ICombine provides combine method to gather data from multiple website, and filter them.

* 1. **Package crawler.thoiky**

In package crawler.thoiky, for each website, we create a class to scrap data from that website by interiting AGetData. After crawling data from all websites, “*ThoiKyFull*” class will combine all by implementing “*ICombine*” interface and export to Json file by implementing “*IWriteJson*” interface.

The package crawler.thoiky has a general class diagram as shown below. And the details of which class performs website is also shown below.

A picture containing text, diagram, line, plan

Description automatically generated

Figure 16 package crawler.thoiky class diagram

|  |  |
| --- | --- |
| **Class** | **Link to scrap data** |
| CacDoiVua | https://www.wikiwand.com/vi/B%E1%BA%A3n\_m%E1%BA%ABu:Danh\_s%C3%A1ch\_vua\_v%C3%A0\_ho%C3%A0ng\_%C4%91%E1%BA%BF\_Vi%E1%BB%87t\_Nam |
| KinhDo | https://vi.wikipedia.org/wiki/Th%E1%BB%A7\_%C4%91%C3%B4\_Vi%E1%BB%87t\_Nam |
| KinhDoFull |
| ThoiKy1 | https://nguoikesu.com/tu-lieu/bang-doi-chieu-cac-trieu-dai-viet-nam-va-cac-trieu-dai-trung-quoc |

Figure 17 Link to scrap data "thoiky"

* 1. **Package crawler.nhanvatlichsu**

A picture containing text, diagram, line, screenshot

Description automatically generated

Figure 18 package crawler.nhanvatlichsu class diagram

Similar to “crawler.thoiky” package, for each website, a class was created to scrap data from that link and NhanVatLichSuFull served to combine all data from these links and covert it to Json file.

|  |  |
| --- | --- |
| **Class** | **Link to scrap data** |
| NhanVatLichSu1 | https://nguoikesu.com/nhan-vat |
| NhanVatLichSu2 | https://vansu.vn/viet-nam/viet-nam-nhan-vat?page=0 |
| Vua1 | https://vi.wikipedia.org/wiki/Vua\_Vi%E1%BB%87t\_Nam |

Figure 19 Link to scrap data "nhanvatlichsu"

* 1. **Package crawler.sukienlichsu**

**A picture containing text, diagram, screenshot, plan

Description automatically generated**

Figure 20 package crawler.sukienlichsu class diagram

In this crawler.sukienlichsu package, “ChienTranhFull” class serves to collect data from “ChienTranh1” and “ChienTranh2” classes, each of which collects data from different links. “SuKienLichSuFull” then collect data from “ChienTranhFull” and “SuKien1”, “SuKien2” and “SuKien3” data.

Websites to scrap data is shown as below.

|  |  |
| --- | --- |
| **Class** | **Link to scrap data** |
| ChienTranh1 | https://vi.m.wikipedia.org/wiki/C%C3%A1c\_cu%E1%BB%99c\_chi%E1%BA%BFn\_tranh\_Vi%E1%BB%87t\_Nam\_tham\_gia |
| ChienTranh2 | https://nguoikesu.com/tu-lieu/quan-su?filter\_tag[0]= |
| SuKien1 | https://vi.wikipedia.org/wiki/Ni%C3%AAn\_bi%E1%BB%83u\_l%E1%BB%8Bch\_s%E1%BB%AD\_Vi%E1%BB%87t\_Nam |
| SuKien2 | https://thuvienlichsu.com/category/su-kien/ |
| SuKien3 | https://nguoikesu.com/tu-lieu/quan-su |

Figure 21 Links to scrap data "sukienlichsu"

* 1. **Package crawler.ditichlichsu**

For each website in two selected links, two classes “DiTich1” and “DiTich2” were created to scrap data from that link and DiTichFull served to combine all data from these links and covert it to Json file.

A screenshot of a computer

Description automatically generated

Figure 22 package "crawler.ditichlichsu" class diagram

|  |  |
| --- | --- |
| **Class** | **Link to scrap data** |
| DiTich1 | https://vi.m.wikipedia.org/wiki/Danh\_s%C3%A1ch\_Di\_t%C3%ADch\_qu%E1%BB%91c\_gia\_Vi%E1%BB%87t\_Nam |
| DiTich2 | http://ditich.vn/FrontEnd/DiTich/Form?do=&ItemId=6144 |

Figure 23 Link to scrap data "ditichlichsu"

* 1. **Package crawler.lehoi**

A diagram of a folder

Description automatically generated with low confidence

Figure 24 crawler.lehoi package class diagram

“crawler.lehoi” package diagram depicts “LeHoiFull” class and 6 others. In this package, “LeHoiFull” combines data from the other classes and converts it to Json file in folder “data”.

|  |  |
| --- | --- |
| **Class** | **Link to scrap data** |
| LeHoi1 | https://vi.wikipedia.org/wiki/L%E1%BB%85\_h%E1%BB%99i\_Vi%E1%BB%87t\_Nam |
| LeHoi2 | https://angiangtourist.vn/thoi-gian-va-dia-diem-to-chuc-cac-le-hoi-lon-o-an-giang/#Tr |
| LeHoi3 | https://dulichkhampha24.com/le-hoi-o-da-nang.html |
| LeHoi4 | https://alltours.vn/tuyen-quang/cac-le-hoi-o-tuyen-quang.html |
| LeHoi5 | https://vinpearl.com/vi/le-hoi-o-ha-nam-top-10-le-hoi-dac-sac-lon-nhat-trong-nam |
| LeHoi6 | https://blog.mytour.vn/danh-muc/le-hoi-su-kien?page=1 |

Figure 25 links to scrap data "lehoi"

1. **Package “application”**

A screenshot of a computer

Description automatically generated with medium confidence

Figure 26 'application.util' package class diagram

The figure demonstrates details of ‘application. util’ package diagram. From above, util contains two smaller packages, each of which includes a class corresponding to one functionality that the application needs to run on or serve the end user.

‘readdata’ package contains ‘ReadData’ class. This class serves convert Json data from ‘data’ package to class in Java and convert it to ObservableList type then show it in the GUI.

While ‘search’ package includes ‘Search’ class. This helps ‘searching’ functionality. When a user types a keyword to find the data in the main board, this class finds the data in that table (case insensitive).

In general, these classes were called in ‘Controller’ class in ‘application.main’ package. In ‘Controller.java’ these classes are executing when an event is called, then return the results in the table or pop up the new panel which shows detail data of an object.



Figure 27 'application.details' class diagram

Those classes, which are in ‘application.details’ package, describe the detailed information about a specific object when the user double clicked to the name of that object in the final system. When executing, the ‘curSelect’ takes in the String then converts it to ‘Object’ type, then if it is matching, it will return the details of that.

A screenshot of a computer program

Description automatically generated with medium confidence

Figure 28 'application.main' class diagram

‘application.main’ package supplies all classes that we need to create the GUI and functionalities for the end users. ‘Main.java’, meanwhile, runs the final program, ‘Controller.java’ was used to handle the events and control the program. Class ‘TableViewCustom.java’ is setting the grapical interface of table view in the final program.

A picture containing text, line, screenshot, diagram

Description automatically generated

Figure 29 'application' class diagram

The above figure depicts the relationships and interations between all classes in package ‘application’.

1. **Data storing design**

The JSON data files storing the collected data are in the ‘data’ package:

|  |  |  |
| --- | --- | --- |
| **Json files** | **Objectives** | **Folder** |
| thoiKy.json | stores dynasty data | data.thoiky |
| ChienTranh.json | stores war data | data.sukienlichsu |
| SuKien.json | stores historical event data |
| nhanVatLichSu.json | stores historical figure data | data.nhanvatlichsu |
| vua.json | stores kings data |
| DiTich.json | stores historical relic data | data.ditichlichsu |
| leHoi.json | stores festival data | data.lehoi |

1. **RESULTS & PROGRAM TESTING**

1. **Statistics of data**

|  |  |
| --- | --- |
| **Entites** | **Quantity** |
| Dynasty (Thời kỳ) | 22 |
| Historical figure (Nhân vật lịch sử) | 2368 |
| King (Vua) | 164 |
| Historical event (Sự kiện lịch sử) | 345 |
| War (Chiến tranh) | 283 |
| Relic (Di tích lịch sử) | 5684 |
| Festival (Lễ hội) | 318 |

Figure 30 Statistics of data

1. **Final program testing**

**A screenshot of a game

Description automatically generated with low confidence**

Figure 31 Main screen (Trang chủ)

A screenshot of a computer

Description automatically generated with medium confidence

Figure 32 King screen (Vua)



Figure 33 Historical figure screen (Nhân vật lịch sử)



Figure 34 Dynasty screen (Thời kỳ lịch sử)



Figure 35 Event screen (Sự kiện lịch sử)

A screenshot of a computer

Description automatically generated with medium confidence

Figure 36 War screen (Các cuộc chiến tranh)

A screenshot of a computer

Description automatically generated with medium confidence

Figure 37 Relic Screen (Di tích lịch sử)

A screenshot of a computer

Description automatically generated with medium confidence

Figure 38 Festival Screen (Các lễ hội)

A screenshot of a computer

Description automatically generated with low confidence

Figure 39 Search Test

A screenshot of a computer

Description automatically generated with low confidence

Figure 40 Details of Kings

1. **OOP TECHNIQUES IN USE**
2. **OOP Techniques**
3. **Encapsulation:**

Properties are hidden using the private access modifier, and public methods are used to access and modify the data.

- Benefits: Ensures data security.

1. **Inheritance:**

In the data collection section, the program uses abstract classes and interfaces. The websites are created as separate classes and inherit from the abstract classes and interfaces.

In the program, single-level inheritance is used in the packages of entities, and multiple-level inheritance is used in the application.details package.

In many classes, the use of inheritance helps make the code more concise and readable. For example, when scraping data about dynasties, there is a ThoiKyFull class that calls different components such as name scraping and time scraping.

- Benefits: Inheritance facilitates rapid and concise programming.

1. **Polymorphism:**

Polymorphism is demonstrated in the program through the following points:

Method overloading is implemented in the classes of entities, allowing the creation of objects with different types of input parameters.

When classes inherit an interface, each class needs to override the methods defined in the interface.

- Benefits: Polymorphism allows writing a single method with multiple parameter types. Additionally, it enables customization of inherited methods to suit specific objects, such as collecting data from different websites in different ways.

1. **Abstraction:**

Abstraction is implemented through abstract classes and interfaces, with subsequent classes inheriting from them.

- Benefits: Abstracting common attributes of entities helps simplify and expedite programming.

1. **Generic Programming:**

Generics are used in the ReadData class, where the type of entity is passed as a parameter to the ReadData object to read information about that entity from a JSON file.

1. **Algorithms:**

The keys of the objects are stored in an ArrayList<String> to save memory space and to comfort the searching function. When information needs to be retrieved, a search is performed based on the key to that model.

1. **CONCLUSION & FUTURE DEVELOPMENT**

During the process of completing our major project, we have relatively fulfilled the required functionalities to a satisfactory extent using object-oriented programming (OOP) techniques. We had detailed ideas and analyzed the project using object-oriented methods to build an overall model of the product. Based on that model, we implemented the programming according to the completed design. The system is forwarding two principles: high cohension, low coupling.

However, due to limitations in our work process and knowledge, our team made efforts to refine the product, but there are still some errors and incomplete features in the project. We have recognized these limitations and have a plan to address them.

In the near future, we plan to add new features to the product and improve the interface to provide a better user experience. We understand that a visually appealing and user-friendly interface will facilitate easy and convenient interaction with the product.

1. **INSTRUCTION & TEAM CONTRIBUTION**

To run the application, simply execute the main file within the Main.java class in the application.main package. After running the app, you can proceed to perform the predefined functions available in the app.

Here is the team contribution and evaluation thoroughout the project.

|  |  |  |
| --- | --- | --- |
| **Member** | **Work** | **Score** |
| Nguyen Trong Huy | - Project manager/owner - System design - Program design (crawler, model) - Implementation (crawler.thoiky, crawler.processfull, crawler.util) - Report writer | 17% |
| Tran Duc Nam | - Prepare web sources  - Program design (model, application) - GUI designer - Implementation (application)  - Demo video preparation | 17% |
| Trinh Giang Nam | - Do requirement analysis  - Prepare web sources  - Implementation (crawler.nhanvatlichsu, model)  - Testing phase | 17% |
| Bui Phuong Nam | - Do requirement analysis  - Prepare web sources  - Program design (crawler)  - Implementation (crawler.ditichlichsu)  - Testing phase | 17% |
| Nguyen Chinh Minh | - Do requirement analysis  - Prepare web sources  - Implementation (crawler.sukienlichsu, model)  - Testing phase | 16% |
| Hoang Minh Quan | - Do requirement analysis  - Prepare web sources  - Implementation (crawler.lehoi)  - Testing phase | 16% |

Figure 41 Team contribution

# **REFERENCES**

|  |  |
| --- | --- |
| [1] | SOICT, Slides of OOP (IT3100E), Ha Noi: SOICT, HUST, 2023. |
| [2] | ZenRows, "Web Scraping in Java in 2023: The Complete Guide," 19 December 2022. [Online]. Available: https://www.zenrows.com/blog/web-scraping-java#can-you-web-scrape-with-java. [Accessed 2023]. |
| [3] | Oracle, "Getting Started with JavaFX," September 2013. [Online]. Available: https://docs.oracle.com/javafx/2/get\_started/jfxpub-get\_started.pdf. [Accessed June 2023]. |

**TABLE OF FIGURES**

[Figure 1 Group 1’s members 4](#_Toc139376443)

[Figure 2 Details of our development process 6](#_Toc139376444)

[Figure 3 Technologies in use 6](#_Toc139376445)

[Figure 4 Java libraries 7](#_Toc139376446)

[Figure 5 Base data 8](#_Toc139376447)

[Figure 6 Data viewing process analysis 9](#_Toc139376448)

[Figure 7 Actual web scraping process 10](#_Toc139376449)

[Figure 8 Web scraping process activity diagram 10](#_Toc139376450)

[Figure 9 Use case diagram. 11](#_Toc139376451)

[Figure 10 Use case "view data" details 11](#_Toc139376452)

[Figure 11 Use case "gather data" details. 11](#_Toc139376453)

[Figure 12 Use case "scrap web" details 12](#_Toc139376454)

[Figure 13 Package diagram of system design 13](#_Toc139376455)

[Figure 14 "model" class diagram 16](#_Toc139376456)

[Figure 15 crawler.util package class diagram 17](#_Toc139376457)

[Figure 16 package crawler.thoiky class diagram 18](#_Toc139376458)

[Figure 17 Link to scrap data "thoiky" 18](#_Toc139376459)

[Figure 18 package crawler.nhanvatlichsu class diagram 18](#_Toc139376460)

[Figure 19 Link to scrap data "nhanvatlichsu" 19](#_Toc139376461)

[Figure 20 package crawler.sukienlichsu class diagram 19](#_Toc139376462)

[Figure 21 Links to scrap data "sukienlichsu" 19](#_Toc139376463)

[Figure 22 package "crawler.ditichlichsu" class diagram 20](#_Toc139376464)

[Figure 23 Link to scrap data "ditichlichsu" 20](#_Toc139376465)

[Figure 24 crawler.lehoi package class diagram 20](#_Toc139376466)

[Figure 25 links to scrap data "lehoi" 21](#_Toc139376467)

[Figure 26 'application.util' package class diagram 21](#_Toc139376468)

[Figure 27 'application.details' class diagram 22](#_Toc139376469)

[Figure 28 'application.main' class diagram 22](#_Toc139376470)

[Figure 29 'application' class diagram 23](#_Toc139376471)

[Figure 30 Statistics of data 24](#_Toc139376472)

[Figure 31 Main screen (Trang chủ) 24](#_Toc139376473)

[Figure 32 King screen (Vua) 24](#_Toc139376474)

[Figure 33 Historical figure screen (Nhân vật lịch sử) 25](#_Toc139376475)

[Figure 34 Dynasty screen (Thời kỳ lịch sử) 25](#_Toc139376476)

[Figure 35 Event screen (Sự kiện lịch sử) 25](#_Toc139376477)

[Figure 36 War screen (Các cuộc chiến tranh) 26](#_Toc139376478)

[Figure 37 Relic Screen (Di tích lịch sử) 26](#_Toc139376479)

[Figure 38 Festival Screen (Các lễ hội) 26](#_Toc139376480)

[Figure 39 Search Test 27](#_Toc139376481)

[Figure 40 Details of Kings 27](#_Toc139376482)

[Figure 41 Team contribution 30](#_Toc139376483)