Department of Computer Science and Engineering

Project Title : Atomic Dictionary

Team Member's Name and ID	
Name	**ID**
Lutfun Nahar Barsha	2233081334
Md. Riad Hasan	2233081332
Md. Kowser Mahmood	2233081349

Table 1: Team Members & Their ID

Supervisor:

Samia Yasmin
Lecturer
Department of Computer Science and Engineering

Object Oriented Programming Project Report
Fall 2024

November 24, 2024

Abstract

The **Atomic Dictionary** is a user-friendly application built in C# to help users explore chemical elements. It allows users to search for an atom by entering its name or symbol and provides detailed information, including Atomic Name, Symbol, Atomic Number, Mass, Group, Period, State, Melting Point, Boiling Point, Valency, and Radioactive Status. The app's simple design makes it accessible even for those unfamiliar with technology, while features like auto-complete improve the speed and accuracy of searches. Designed for reliability and efficiency, the **Atomic Dictionary** is a helpful resource for students and anyone wanting to understand the periodic table and the building blocks of matter.

Keywords: Atomic Dictionary, User-friendly, C# (C Sharp), Chemical Elements, Atomic Name, Symbol, Atomic Number, Atomic Mass, Melting Point, Boiling Point, Valency, Radioactive Status, Efficiency, Periodic Table, Accessibility.

Contents

List of Figures

List of Tables

List of Abbreviations

OOP	Object Oriented Programming
STEM	Science, Technology, Engineering, and Mathematics
HTML	Hypertext Markup Language
EXE file	Executable File
PCs	Personal Computers
ID	Identification

Chapter 1

Introduction

The **Atomic Dictionary** is a simple and intuitive application designed to make understanding chemical elements easy for everyone. This project allows users to search for and learn about elements from the periodic table by entering their name or symbol. The application provides detailed information about each element, such as its atomic number, atomic weight, and other important properties, all presented in a clear and organized format. The user-friendly interface ensures that people of all backgrounds, including students and educators, can navigate the app effortlessly.

This project not only fulfills the need for accessible chemical knowledge but also encourages a deeper appreciation for the elements that form the foundation of our world. By combining ease of use with educational value, the **Atomic Dictionary** aims to be a practical and effective tool for anyone looking to explore the fascinating science behind the periodic table.

1.1 Background

Chemical elements are key to understanding fields like chemistry, biology, and environmental science. Traditionally, learning about the periodic table relied on textbooks, which can be difficult to use and not always easily accessible. With the advancement of technology, tools like the **Atomic Dictionary** offer a simple and interactive way to quickly find specific information about elements. This project supports *STEM education* by making science more approachable and engaging for all ages. It provides a faster, clearer, and more accessible alternative to traditional methods, helping students and teachers explore the periodic table efficiently. The **Atomic Dictionary** makes studying chemistry easier and more enjoyable by offering a user-friendly digital platform that traditional resources cannot match.

1.2 Problem statement

During the development of the **Atomic Dictionary** project using Visual Studio Community 2022, several challenges arose:

1.2.1 Form Visibility Issue: Initially, when the user provided input on the first form and navigated to the second form for the output, the second form remained visible when returning to the first form. This caused the second form to appear in front of the first form, leading to an overlapping display of both forms. This issue was eventually resolved by properly managing the form transitions.

Code to Solve Form Visibility Issue :

```
Form2 temp9 = new Form2("PLEASE INSERT AN ATOM'S NAME OR SYMBOL.");
temp9.FormClosed += Form2_FormClosed;
this.Hide();
temp9.Show();
```

1.2.2 Input Box Persistence: After providing input on the first form and returning from the second form, the input box would retain the previously entered data. This was confusing for users as the input was not cleared when returning to the first form. The issue was resolved by ensuring that the input field is reset when the user navigates back to the first form.

Code to Clear the Input Box on Return

```
inputBox.Clear();
```

1.2.3 Missing Key Press Functions: The initial version of the project did not have key press functionalities for the **Enter** key (to trigger the action) and the **Esc** key (to return to the previous form). This was added later to improve the user experience, making the application more intuitive and allowing users to navigate the forms more easily and making the project more user-friendly.

Code to handle Enter Key :

```
public Form1()
{
    InitializeComponent();
    this.AcceptButton = searchButton;
}
```

Code to handle Esc Key :

```
public Form2(string elementInfo)
{
    InitializeComponent();
    label3.Text = elementInfo;
    this.CancelButton = button1;
}
```

1.2.4 Accessor Property Issue: In the initial implementation of the `Element` class, all properties were private and read-only, making them inaccessible from other parts of the application. This prevented proper retrieval of element data, which was a critical functionality for the Atomic Dictionary. The issue was resolved by introducing public getter methods for each property while maintaining encapsulation of the private fields.

Fault Code:

```
public class Element
{
    private string Symbol { get; }
    private string Name { get; }
    private int AtomicNumber { get; }
    private double AtomicMass { get; }
    private string ElectronConfig { get; }
    private string Group { get; }
    private string Period { get; }
    private double MeltTemp { get; }
    private double BoilTemp { get; }
    private int Valency { get; }
    private string Radioactive { get; }
    private string State { get; }
}

```

Corrected Code:

```
public class Element
{
    private string Name;
    public string _name { get { return Name; } }
    private string Symbol;
    public string _symbol { get { return Symbol; } }
    private int AtomicNumber;
    public int _atomicNumber { get { return AtomicNumber; } }
    private double AtomicMass;
    public double _atomicMass { get { return AtomicMass; } }
    private string ElectronConfig;
    public string _electronConfig { get { return ElectronConfig; } }
    private string Grp;
    public string _grp { get { return Grp; } }
    private string Period;
    public string _period { get { return Period; } }
    private double MeltTemp;
    public double _meltTemp { get { return MeltTemp; } }
    private double BoilTemp;
    public double _boilTemp { get { return BoilTemp; } }
    private int Valency;
    public int _valency { get { return Valency; } }
    private string State;
    public string _state { get { return State; } }
    private string Radioactive_Status;
    public string _radioActive_status { get { return
    Radioactive_Status; } }
}

```

These challenges were addressed to ensure smoother user interactions and a better overall experience with the application.

1.3 Aims and objectives

1.3.1 Aims

The aim of the **Atomic Dictionary** project is to create a simple, interactive, and easy-to-use application that allows users to efficiently search for and learn about chemical elements. This application is designed to provide instant access to essential information about elements from the periodic table, such as atomic number, atomic weight, state, and various chemical properties. The goal is to make the process of learning about elements faster and more accessible to a wide range of users, including students, educators, and anyone interested in chemistry.

1.3.2 Objectives

The specific objectives of the project are as follows:

1. To design a user-friendly interface that simplifies the process of searching for elements, making it easy for anyone, regardless of their background in chemistry, to access important data by pressing **Enter** and **Esc** keys and also trigger buttons.

2. To enable users to search for elements by either their name or symbol, ensuring that information can be retrieved quickly and accurately.

3. To provide detailed information about each element, including atomic number, atomic mass, electron configuration, state of matter, group and period in the periodic table, melting and boiling points, and valency.

4. To ensure the application is responsive, providing quick and accurate results to users without unnecessary delays, allowing for smooth and efficient use.

5. To maintain an organized and clear display of information, ensuring that users can easily read and understand the data provided for each element.

6. To regularly update the application with the latest data and improvements, ensuring that the information stays relevant and accurate as new discoveries are made in the field of chemistry.

7. To offer a valuable educational tool for both learning and teaching, encouraging users to explore and deepen their understanding of the chemical elements that make up our world.

Chapter 2

Methodology

2.1 Software Environment Setup:

This project was built using **Visual Studio Community 2022** (v17.11.5) with the C# programming language and the .NET (v8.0.403) framework for Windows Forms applications. Visual Studio offers a comprehensive platform for developing applications on multiple platforms. It combines design, editing, debugging, and profiling tools into one powerful interface. With support for various programming languages like C#, Visual Basic, F#, C++, HTML, JavaScript, and Python, it provides a rich and flexible environment for developers.

Figure 2.1: Visual Studio Community 2022 Software Logo

2.2 Input Handling:

The interface (form1.cs [Design]) includes a text input field (inputBox), where users can enter either the Symbol or Name of an atom. The input field is configured to respond to user actions, like search button **"FIND ATOM"** and also keypress events **"Enter"**, enhancing the user experience by allowing quick searches.

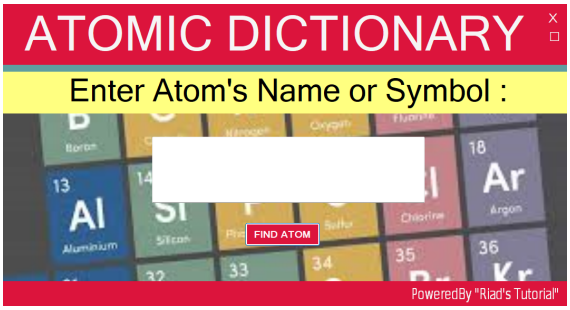

Figure 2.2: form1.cs [Design] | for Handling Input

2.3 Data Structure for Atom Details:

A structured dataset (such as a dictionary or list of objects) was created to store details for each atom. Each entry includes attributes like the Atomic Name, Symbol, State, Atomic Number, Atomic Mass, Group, Period, Electron Configuration, Melting Point, Boiling Point, Valency and Radioactive status. This dataset was embedded directly into the application for fast retrieval using Constructor.

Code Snippet of Constructor :

```
public Element(string symbol, string name, int atomicNumber, double
    atomicMass, string electronConfig, string grp,
                string period, double meltTemp, double boilTemp, int
    valency, string radioActive_status, string state)
{
    this.Name = name;
    this.Symbol = symbol;
    this.AtomicNumber = atomicNumber;
    this.AtomicMass = atomicMass;
    this.ElectronConfig = electronConfig;
    this.Grp = grp;
    this.Period = period;
    this.MeltTemp = meltTemp;
    this.BoilTemp = boilTemp;
    this.Valency = valency;
    this.Radioactive_Status = radioActive_status;
    this.State = state;
}
```

Listing 2.1: Constructor for the Element class

Example :

```
Element[] periodicTable = {
    new Element("H", "Hydrogen", 1, 1.008, "1s1", "1", "1", 14.01, 20.28,
    1, "Safe", "Gas"),
    new Element("He", "Helium", 2, 4.0026, "1s2", "18", "1", 0.95, 4.22,
    0, "Safe", "Gas"),
    new Element("Li", "Lithium", 3, 6.94, "[He] 2s1", "1", "2", 453.69,
    1560.15, 1, "Safe", "Solid"),
    new Element("Be", "Beryllium", 4, 9.0122, "[He] 2s2", "2", "2",
    1560.15, 2470, 2, "Safe", "Solid"),
    new Element("B", "Boron", 5, 10.81, "[He] 2s2 2p1", "13", "2", 2349,
    4200, 3, "Safe", "Solid"),
    new Element("C", "Carbon", 6, 12.011, "[He] 2s2 2p2", "14", "2", 3823,
     4098, 4, "Safe", "Solid"),
    new Element("N", "Nitrogen", 7, 14.007, "1s2 2s2 2p3", "15", "2",
    63.15, 77.36, 3, "Safe", "Gas"),
    new Element("O", "Oxygen", 8, 15.999, "1s2 2s2 2p4", "16", "2", 54.36,
     90.20, 2, "Safe", "Gas"),
    // ....................,
};
```

Listing 2.2: Periodic Table Elements Array

2.4 Search and Retrieval Logic:

Once receiving the input (Name or Symbol of an Atom), the application checks the entered text against the dataset. If a match is found based on either the atom's name or symbol, the program retrieves the corresponding details.

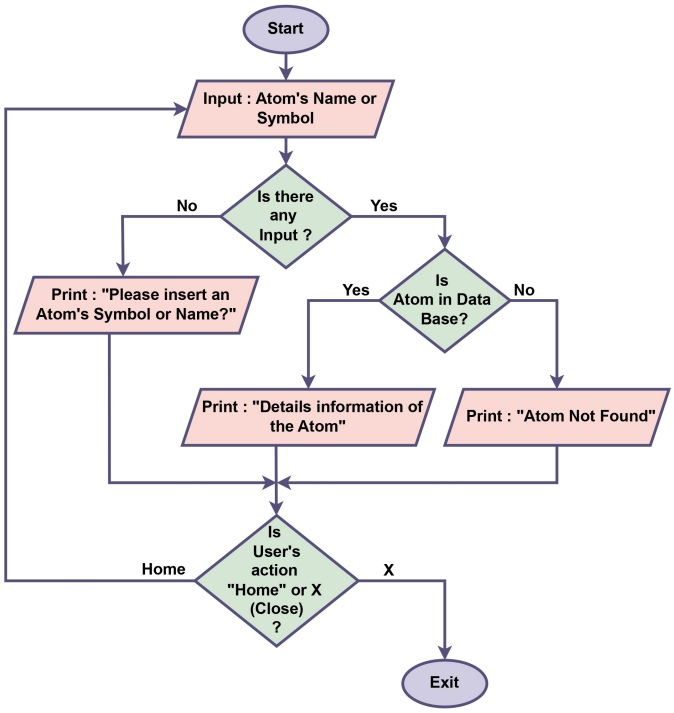

Figure 2.3: Flow Chart for the Application.

2.5 Display Mechanism:

The information about the atom is displayed on a new page (Form2.cs[Design]) in a ListBox control. This layout allows users to view structured details easily and keeps the interface organized and user-friendly. After checking one atom's details, user can return to the previous page by clicking on **Home** button or pressing **Esc** key on keyboard.

Figure 2.4: Input Interface (Form1.cs)

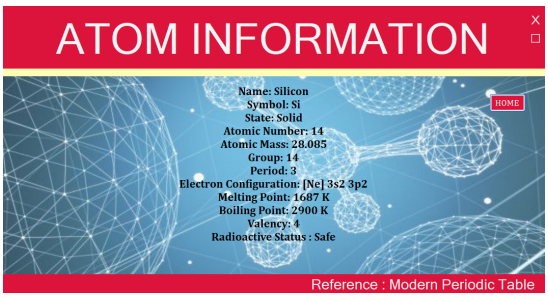

Figure 2.5: Output Interface (Form2.cs)

2.6 Error Handling:

Error handling was implemented to manage cases where the user input does not match any known atom or don't give any input in the inputBox. The program displays an error message, prompting the user to enter a valid atom name or symbol.

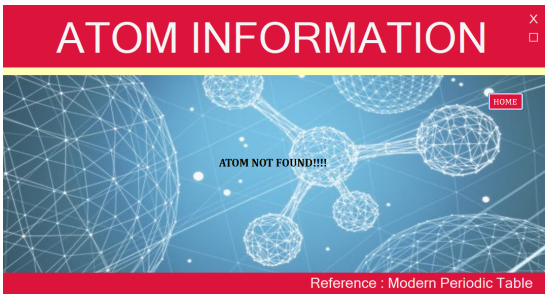

Figure 2.6: Error Output 1

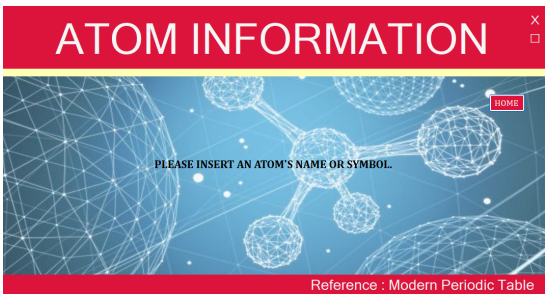

Figure 2.7: Error Output 2

2.7 Testing and Optimization:

The project was tested for accuracy and responsiveness. Adjustments were made to the data structure and search algorithm to ensure that the application quickly and correctly responds to user input, providing accurate atomic details.

Chapter 3

Results

The **Atomic Dictionary** application successfully allows users to retrieve detailed information about chemical elements by entering their name or symbol. The key features and outcomes are as follows:

1. **Input Form**: Users can easily input the name or symbol of an element. The interface is user-friendly and responsive.
2. **Output Form**: Displays comprehensive information about the selected element, such as atomic number, symbol, and other details from the periodic table.
3. **Keyboard Shortcuts**: The application supports the **Enter** key to proceed and the **Esc** key to return, improving navigation efficiency. It makes the project more user-friendly.

The project demonstrates the successful implementation of Object-Oriented Programming (OOP) principles and effective use of C# in creating a functional and interactive application.

Chapter 4

Discussion and Analysis

4.1 Discussion

The **Atomic Dictionary** project provides a simple yet efficient way for users to find information about chemical elements. By inputting the name or symbol of an element, the application quickly retrieves and displays relevant details, making it a helpful tool for students, educators, and science enthusiasts.

One of the key features of the project is its user-friendly interface, which allows users to interact with the system through two forms. The **first form** takes input, and the **second form** presents the output. This design ensures clarity and keeps the process organized.

The project also demonstrates the use of object-oriented programming (OOP) principles, such as classes and methods, to effectively manage and retrieve element data.These principles helped make the application simple to update, easy to use, and able to grow if needed.

4.2 Analysis

4.2.1 Challenges Faced

During development, two primary challenges were encountered:

1. The **second form remained visible** even after returning to the first form.
2. The **input box retained previous data**, which could confuse users.

These challenges were resolved by implementing proper form management techniques and resetting the input fields when switching back to the first form. This enhanced the usability and functionality of the application.

Additionally, the addition of a PressKey function allowed users to interact with the application using keyboard shortcuts, such as **Enter** to submit data and **Esc** to exit. This made the application more accessible and user-friendly.

4.2.2 Impact of the Project

The **Atomic Dictionary** project simplifies access to Periodic Table information, making it valuable for educational purposes. Furthermore, it allowed the developer to gain hands-on experience with OOP concepts and Windows Forms in C#. The project demonstrates how programming can solve real-world problems efficiently.

Chapter 5

Conclusions and Future Work

5.1 Conclusion

The **Atomic Dictionary** project has been successfully developed to provide detailed information about chemical elements based on user input. It features a simple and user-friendly interface, ensuring ease of use for students, educators, and science enthusiasts.

One of the major achievements of this project is that it has been published as an EXE (executable) file, allowing it to run on any PC without requiring **Visual Studio Community** software. This makes the application more accessible and convenient for users.

Overall, the project highlights the importance of combining object-oriented programming principles with practical applications. It has been a valuable learning experience in developing software that solves real-world problems efficiently.

5.2 Future Work

As a next step, the plan is to extend the project to create software that can run on both Android devices and PCs. This will increase the availability and usability of the application, allowing users to access the **Atomic Dictionary** on mobile devices as well.

Developing a cross-platform application will involve exploring technologies that support both Android and PC environments. This improvement will make the tool even more versatile and helpful for a wider audience.

Appendix A

An Appendix Chapter For Atomic Dictionary Project

A.1 Source Code

The following is a sample of the important code snippets used in the **Atomic Dictionary** project:

A.1.1 PressKey Function (C#)

This function handles keyboard actions for the Enter and Esc keys.

```
private void PressKey(object sender, KeyEventArgs e)
{
    if (e.Key == Key.Enter)
    {
        // Code to process the input when Enter key is pressed
    }
    else if (e.Key == Key.Escape)
    {
        // Code to reset the form or exit when Esc key is pressed
    }
}
```

A.1.2 Form Code (C#)

This is the code for handling input and displaying output in the first and second forms.

```
    private void SubmitButton_Click(object sender, EventArgs e)
    {
        string elementInput = inputTextBox.Text;
        // Code to find and display information based on element input
        outputLabel.Text = GetElementInfo(elementInput);
    }
```

A.2 Diagrams and Flowcharts

Below are diagrams that illustrate the structure and flow of the **Atomic Dictionary** application.

A.2.1 Application Flowchart

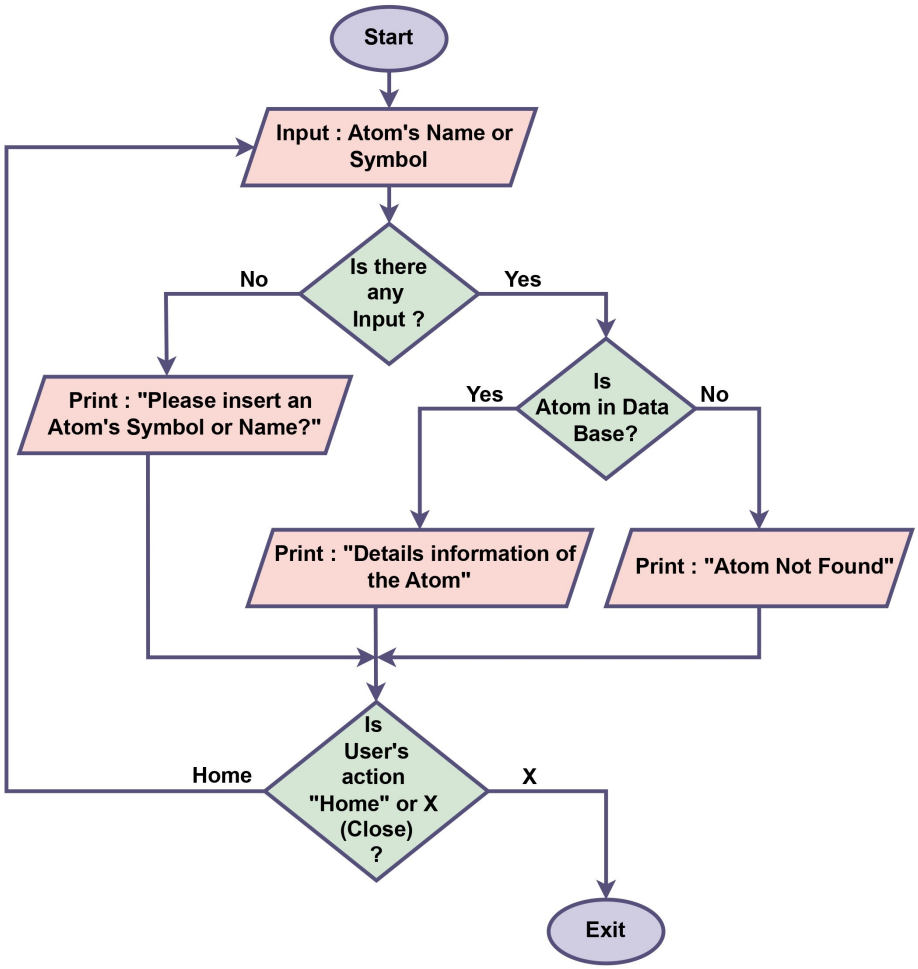

Figure A.1: Flowchart showing user interaction with the application

A.3 Test Results

The following are test results showing the input and output behavior of the **Atomic Dictionary** application.

A.3.1 Test Case 1: Valid Element Input

- **Input:** "U" (Uranium)
- **Expected Output:**
  - Name: Uranium
  - Symbol: U
  - State: Solid
  - Atomic Number: 92

  - Atomic Mass: 238.03
  - Group: 6
  - Period: 7
  - Electron Configuration: [Rn] 5f3 6d1 7s2
  - Melting Point: 1135
  - Boiling Point: 4131
  - Valency: 6
  - Radioactive Status: Radioactive

- **Actual Output:**
  - Name: Uranium
  - Symbol: U
  - State: Solid
  - Atomic Number: 92
  - Atomic Mass: 238.03
  - Group: 6
  - Period: 7
  - Electron Configuration: [Rn] 5f3 6d1 7s2
  - Melting Point: 1135
  - Boiling Point: 4131
  - Valency: 6
  - Radioactive Status: Radioactive

Figure A.2: Valid Input

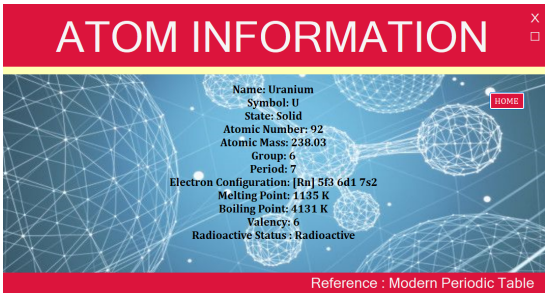

Figure A.3: Valid Output

A.3.2 Test Case 2: Invalid Element Input

- **Input:** "xyz" (Non-existent element)
- **Expected Output:** "Atom not found."
- **Actual Output:** "Atom not found."

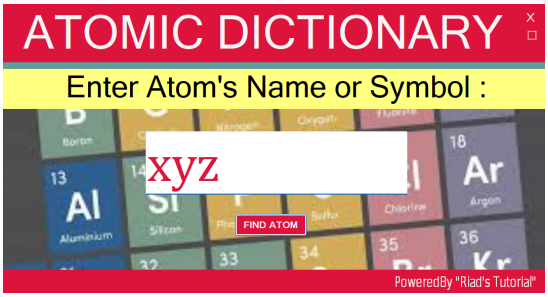

Figure A.4: Invalid Input

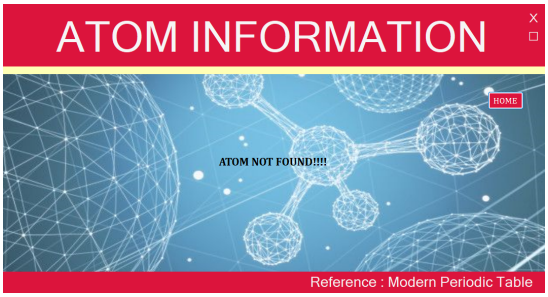

Figure A.5: Invalid Output

A.3.3 Test Case 3: When Input Box is Null

- **Input:** " "
- **Expected Output:** "Please Insert an Atom's Name or Symbol."
- **Actual Output:** "Please Insert an Atom's Name or Symbol."

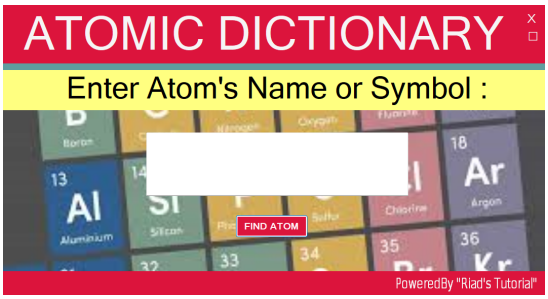

Figure A.6: Null Input

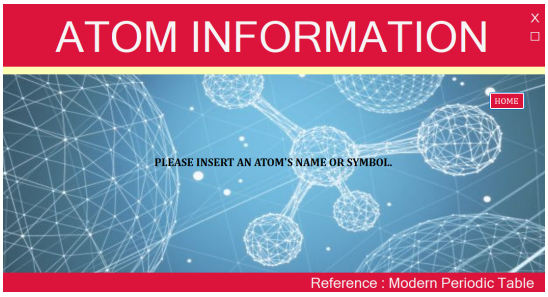

Figure A.7: Error Output

A.4 User Manual

This section provides instructions on how to use the **Atomic Dictionary** application.

A.4.1 Step 1: Input an Element

Enter the name or symbol of the chemical element in the text box on the first form.

A.4.2 Step 2: View the Output

Click the **FIND ATOM** button to view the detailed information about the element, which will be displayed in the second form.

A.4.3 Step 3: Use Keyboard Shortcuts

- Press **Enter** to submit the input. - Press **Esc** to reset the input box and clear the output.

A.5 Additional References

The following resources were used in the development of the **Atomic Dictionary** project:

- **Periodic Table Data:** Some of the data for the atoms were sourced from https://ptable.com/?lang=en#Properties.