**Java DateTime Format**

**Indrajeet Saswade**

**2023-M-30082001**

**MCA-Masters of Computer Application**

**Sairaj Gholap**

**2023-M-10072002A**

**MCA-Masters of Computer Application**

**Satyam Bhumbre**

**2023-M-21022001**

**MCA-Masters of Computer Application**

**Java DateTime Format**

Thesis submitted in partial fulfilment

of the requirements of the degree of

**Master of Computer Applications**

**(MCA)**

by

**Indrajeet Saswade**

**2023-M-30082001**

**Sairaj Gholap**

**2023-M-10072002A**

**Satyam Bhumbre**

**2023-M-21022001**

Under the Supervision of

**Deoyani B kamble**



**April 2024**

**School of Engineering**

**Ajeenkya DY Patil University, Pune**

****

**CERTIFICATE**

This is to certify that the dissertation entitled **“Java DateTime Format”** is a bonafide work of “**Indrajeet Saswade (2023-M-30082001), Sairaj Gholap(2023-M-10072002), Satyam Bhumbre (2023-M-21022001)”**submitted to the School of Engineering, Ajeenkya DY Patil University, Pune in partial fulfilment of the requirement for the award of the degree of **“Master of Computer Applications”**.

**Deoyani B kamble**

Supervisor

**Internal-Examiner/s**

****

**Deoyani B Kamble**

Assistant Professor

**April 29, 2024**

**Supervisor’s Certificate**

This is to certify that the dissertation entitled **“Java DateTime Format”** submitted by **Indrajeet Saswade (2023-M-30082001), Sairaj Gholap(2023-M-10072002), Satyam Bhumbre (2023-M-21022001),** is a record of original work carried out by him/her under my supervision and guidance in partial fulfillment of the requirements of the degree of **Master of Computer Applications** at **School of Engineering**, **Ajeenkya DY Patil University, Pune, Maharashtra-412105**. Neither this dissertation nor any part of it has been submitted earlier for any degree or diploma to any institute or university in India or abroad.

**Deoyani B Kamble**

Supervisor

****

**Declaration of Originality**

I, Indrajeet Saswade (2023-M-30082001), Sairaj Gholap(2023-M-10072002), Satyam Bhumbre (2023-M-21022001) hereby declare that this dissertation entitled “***Java DateTime Format”***presents my original work carried out as a master student of School of Engineering, Ajeenkya D Y Patil University, Pune, Maharashtra. To the best of my knowledge, this dissertation contains no material previously published or written by another person, nor any material presented by me for the award of any degree or diploma of Ajeenkya D Y Patil University, Pune or any other institution. Any contribution made to this project by others, with whom I have worked at Ajeenkya D Y Patil University, Pune or elsewhere, is explicitly acknowledged in the dissertation. Works of other authors cited in this dissertation have been duly acknowledged under the sections “Reference” or “Bibliography”. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission.

I am fully aware that in case of any non-compliance detected in future, the Academic Council of Ajeenkya D Y Patil University, Pune may withdraw the degree awarded to me on the basis of the present dissertation.

**Date:**

**Place:** Lohegaon, Pune

**Indrajeet Saswade**

**Sairaj Gholap**

**Satyam Bhumbre**

**Acknowledgement**

A few words of gratitude to be inserted with project “Java DateTime Format”It is our earnest duty to express our thanks to all those who contributed directly or indirectly to our project.

Firstly, we would like to thank Ajeenkya DY Patil University, Pune and Department of Computer Application for giving us an opportunity. Thanks to Dr. Uttam Deshmukh assistant professor and Head of Department for his encouragement and valuable guidance.

We would like to thank Ms. Deoyani B Kamble who initiated us to complete this project and guided us timely. It is our privilege to express our gratitude to all staff members & non-Teaching Staff members for their excellent suggestions and active co-ordination.

And finally, we would like to thank all our friends for their support and the timely help.

**Indrajeet Saswade**

**Sairaj Gholap**

**Satyam Bhumbre**

|  |  |  |
| --- | --- | --- |
| **Sr. No.** | **Contents** | **Page No.** |
| **Chapter 1** | **INTRODUCTION** |  |
| 1.1 Existing System |  |
| 1.2 Problem Definition- Need of Computerization |  |
| **Chapter 2** | **PROPOSED SYSTEM** |  |
| 2.1 Proposed System |  |
| 2.2 Objectives of System |  |
| 2.3 Operating Environment – Hardware and Software |  |
| **Chapter 3** | **ANALYSIS AND DESIGN** |  |
| Module List |  |
| 3.1 Table Design |  |
| 3.2 Data Dictionary |  |
| 3.3 Screen Shots |  |
|  |  |
|  |  |
|  |  |
| **Chapter 4** | **USER MANUAL** |  |
| 4.1 User Manual |  |
| 4.2 Menu Explanation |  |
| **Chapter 5** | **CONCLUSION** |  |
| 5.1 Limitations & Drawbacks |  |
| 5.2 Future Enhancement |  |
| 5.3 Conclusion |  |
| 5.4 References & Bibliography |  |

**Chapter 1**

**Introduction**

**1.1 Existing System-**

**Title: Java DateTime Format Utility**

**Overview:** The Java DateTime Format Utility is a crucial component in Java development for managing date and time representations. It provides developers with functionalities to parse, format, and manipulate dates and times according to specific patterns and locales.

**Key Features:**

1. **Parsing:** The utility supports parsing date and time strings into Java **LocalDate**, **LocalTime**, **LocalDateTime**, or **ZonedDateTime** objects. It accommodates various input formats, including ISO 8601, custom patterns, and localized date/time representations.
2. **Formatting:** Developers can easily format date and time objects into strings according to specified patterns. This feature allows for precise control over the output format, including date order, delimiters, and time zone representation.
3. **Locale Support:** The utility seamlessly handles localization, enabling date and time formatting based on different locales and cultural conventions. It ensures that date and time representations are tailored to the user's preferred language and region.
4. **Time Zone Handling:** It provides robust support for managing time zones, allowing developers to convert between different time zones and adjust date/time values accordingly. This feature ensures accurate representation and calculation of dates and times across different geographical regions.
5. **Relative Time Formatting:** The utility offers functionality to format relative time expressions, such as "yesterday," "today," or "tomorrow," making it easier to display time differences in a human-readable format.
6. **Error Handling:** Comprehensive error handling mechanisms are in place to gracefully manage exceptions and edge cases during parsing and formatting operations. This ensures robustness and reliability in date and time processing tasks.

**1.2 Problem Statement-**

Develop a Java program that efficiently handles date and time formatting to meet diverse application requirements, ensuring accuracy, flexibility, and ease of use across different locales and time zones.

Top of Form

Java AWT are part of the Java Foundation Classes (JFC), which is used to create graphical applications. It requies basic understanding of java programming and Integrated Development Environment (IDE) or a text editor and a Java Development Kit (JDK).

**Chapter 2**

**Proposed System**

**2.1 Proposed system-**

**Proposed System:**

The proposed system aims to provide a comprehensive solution for managing date and time formatting in Java applications. Key points include:

1. **Enhanced Formatting Flexibility**: Introduce a robust set of formatting options to accommodate various date and time representations, including different calendars, time zones, and cultural conventions.

2. **Simplified API**: Design an intuitive and easy-to-use API that allows developers to perform common date and time formatting tasks with minimal complexity.

3. **Localization Support**: Implement support for localization to cater to diverse language and region-specific formatting requirements, ensuring compatibility with internationalization standards.

4. **Time Zone Management**: Incorporate features for seamless handling of time zones, including conversion between different time zones and accurate representation of daylight saving time transitions.

5. **Error Handling and Validation**: Implement robust error handling mechanisms to detect and gracefully handle invalid date and time inputs, ensuring reliability and data integrity.

6. **Performance Optimization**: Optimize performance to efficiently handle date and time formatting operations, minimizing computational overhead and memory footprint.

7. **Documentation and Examples**: Provide comprehensive documentation and code examples to guide developers in effectively utilizing the DateTime formatting functionalities in their Java projects.

By implementing these features, the proposed system aims to empower Java developers with a versatile and reliable tool for managing date and time formatting in their applications.

Top of Form

**2.2 Objectives of the system**

Objective of the System:

The primary objectives of the Java DateTime Format project are:

1. \*\***Accuracy and Precision**\*\*: Ensure precise and accurate representation of date and time values, adhering to standardized formatting rules and conventions.

2. \*\***Flexibility and Customization**\*\*: Provide developers with a flexible and customizable framework for formatting date and time according to application-specific requirements, including support for various date formats, time zones, and locales.

3. \*\***Internationalization and Localization**\*\*: Support internationalization and localization by enabling the formatting of date and time in different languages and cultural conventions, facilitating the development of globally accessible applications.

4. \*\***Ease of Use and Integration**\*\*: Design an intuitive and easy-to-use API that seamlessly integrates with existing Java applications, allowing developers to perform date and time formatting tasks with minimal effort and complexity.

5. \*\***Robust Error Handling**\*\*: Implement robust error handling mechanisms to detect and handle invalid date and time inputs gracefully, ensuring data integrity and preventing runtime errors.

6. \*\***Performance Optimization**\*\*: Optimize performance to efficiently handle date and time formatting operations, minimizing computational overhead and latency, and ensuring responsive application behavior.

7. **\*\*Documentation and Support\*\*:** Provide comprehensive documentation, tutorials, and support resources to assist developers in effectively utilizing the DateTime formatting functionalities, promoting rapid adoption and successful implementation in Java projects.

By achieving these objectives, the Java DateTime Format project aims to enhance the reliability, usability, and versatility of date and time formatting capabilities in Java applications, ultimately improving the overall development experience and end-user satisfaction.

**2.3 Operating Environment – Hardware and Software**

**Hardware Requirements:**

**1. Processor**: Intel Core i3 or equivalent

2**. RAM**: 4GB or higher

3. **Storage**: 100MB of available disk space for installation and additional space for development and testing purposes

4. **Display:** Minimum 1280x800 resolution monitor

**Software Requirements:**

**1. Operating System**:

- Windows 10 or later

- macOS Catalina (10.15) or later

- Ubuntu 20.04 LTS or later

2**. Java Development Kit (JDK):**

- JDK 8 or later for development

- Recommended: JDK 11 or later for optimal compatibility and features

3. **Integrated Development Environment (IDE):**

- Eclipse IDE, IntelliJ IDEA, or NetBeans IDE for Java development

- Alternatively, any text editor with Java syntax highlighting can be used.

4. **Build Automation Tool:**

- Apache Maven or Gradle for managing dependencies and building the project.

5. **Version Control:**

- Git for version control management, with hosting on platforms like GitHub, GitLab, or Bitbucket.

6. **Documentation:**

- Markdown Editor (e.g., Typora) for writing project documentation.

7. **Testing Framework (optional):**

- JUnit or TestNG for unit testing.

8**. Continuous Integration (CI) Tools (optional):**

- Jenkins, Travis CI, or CircleCI for automating build and testing processes.

9. **Dependency Management**:

- Apache Maven or Gradle for managing project dependencies.

10. **Code Quality Tools (optional):**

- SonarQube, Checkstyle, or PMD for code analysis and quality assurance.

These hardware and software requirements provide a foundation for developing, building, testing, and managing the Java DateTime Format project effectively. Adjustments may be necessary based on specific project needs and preferences.

**Chapter 3**

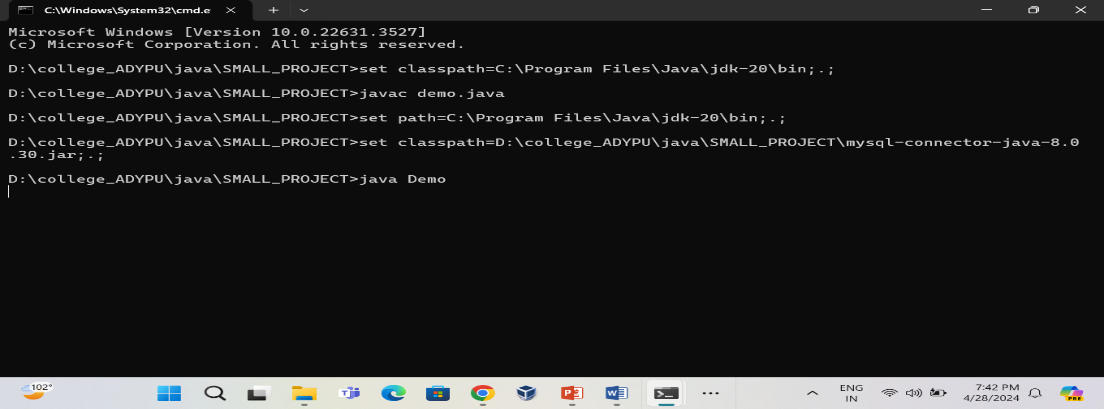
**Analysis and design**

* 1. **Table Design**
  2. **Data Dictionary**

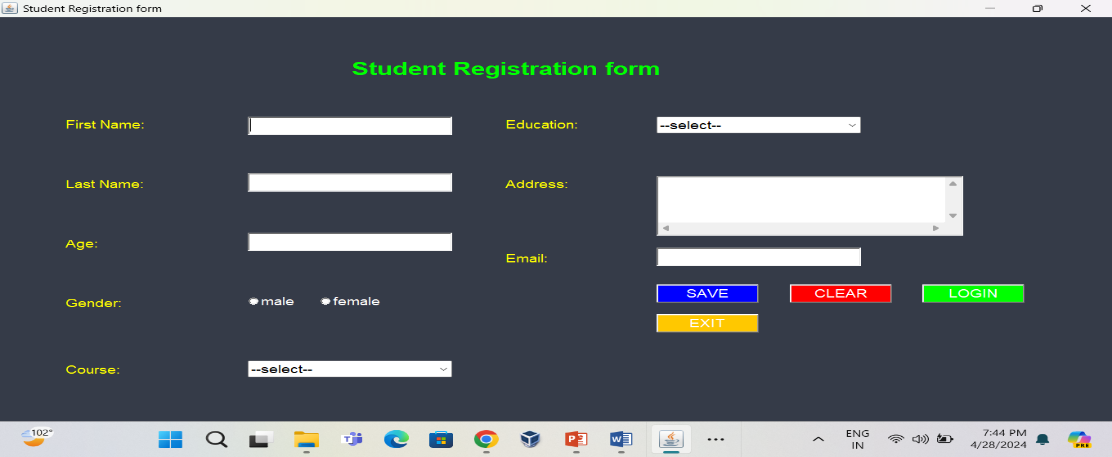
|  |  |  |  |
| --- | --- | --- | --- |
| Field | Type | Size | constraint |
| first\_name | varchar | 20 | not null |
| last\_name | varchar | 20 | not null |
| age | int |  | not null |
| gender | varchar | 10 | not null |
| course | varchar | 20 | not null |
| education | varchar | 20 | not null |
| address | varchar | 30 | not null |
| email | varchar | 40 | not null |

* 1. **Screenshots:**

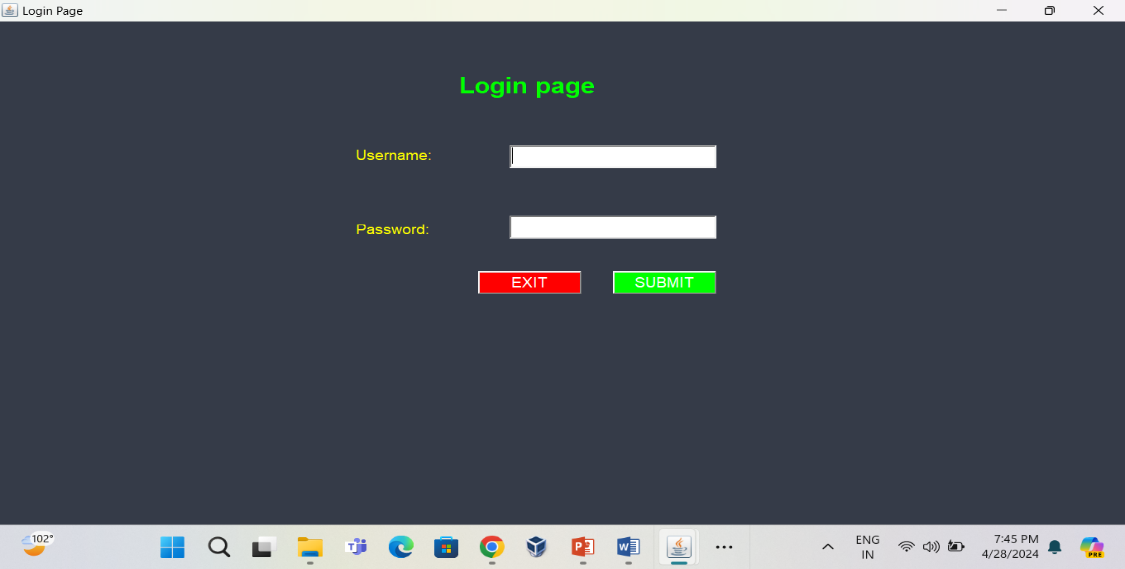
**3.3.1(set classpath (cmd)):**

****

**3.3.2(student Registration form using awt):**

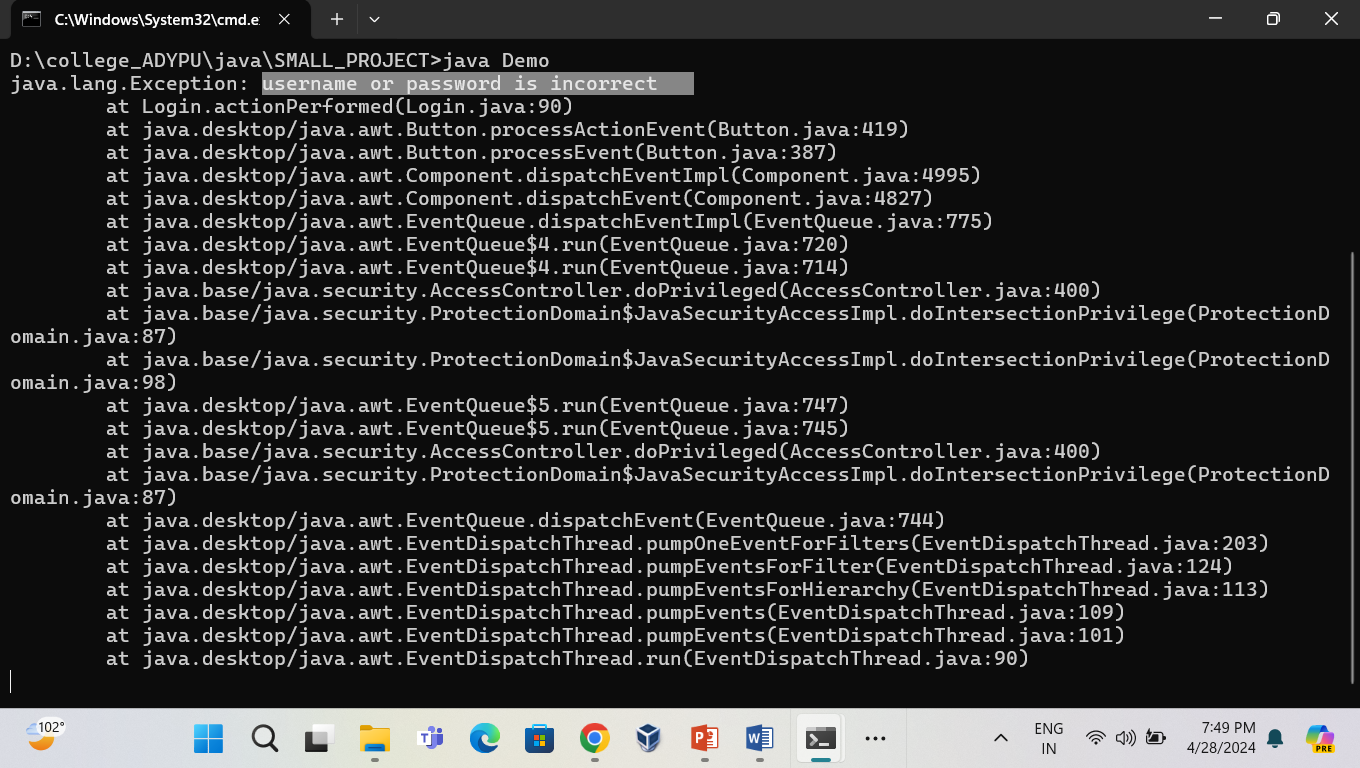
****

**3.3.3 Login page for the admin to see the database:**

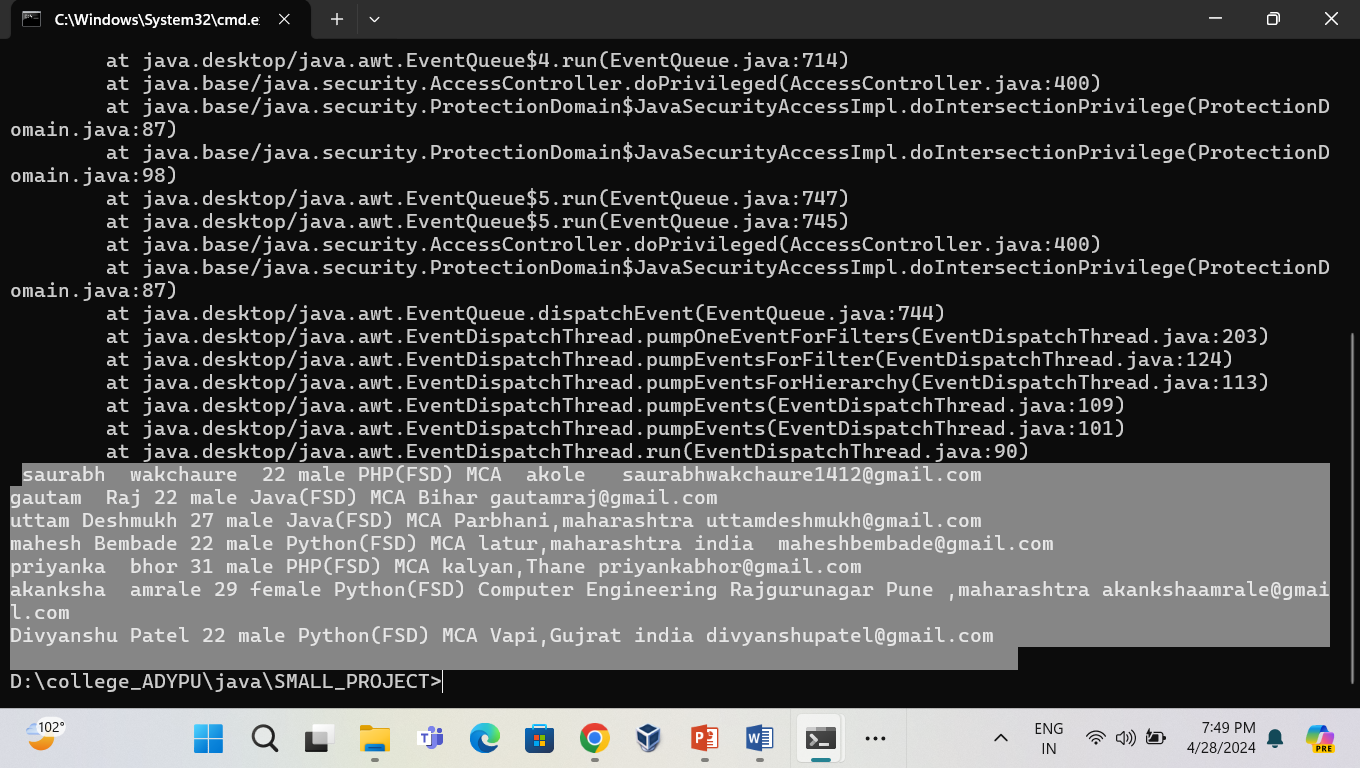
****

**3.3.4 Data display on console:**

**(for invalid username or password)**

****

**For valid username and password**

****

* 1. **Coding**

import java.util.Date;

import java.text.SimpleDateFormat;

class DemoDate

{

public static void main(String args[])

{

Date d=new Date();

//1:

SimpleDateFormat sdf=new SimpleDateFormat("dd/MM/yyyy");

String str=sdf.format(d);

System.out.println("date is:"+str);

//2:

sdf=new SimpleDateFormat("MM-dd-yyyy");

str=sdf.format(d);

System.out.println("date is:"+str);

//3:

sdf=new SimpleDateFormat("EEEEE MMMM dd yyyy");

str=sdf.format(d);

System.out.println("date is:"+str);

//4:

sdf=new SimpleDateFormat("EEE MMMM dd k:m:s yyyy");

str=sdf.format(d);

System.out.println("date and time :"+str);

//5:

sdf=new SimpleDateFormat("dd/MM/yyyy k:m:s a z");

str=sdf.format(d);

System.out.println("date and time:"+str);

//6:

sdf=new SimpleDateFormat("k:m:s");

str=sdf.format(d);

System.out.println("time:"+str);

//7:

sdf=new SimpleDateFormat("w");

str=sdf.format(d);

System.out.println("week of year:"+str);

//8:

sdf=new SimpleDateFormat("W");

str=sdf.format(d);

System.out.println("week of month:"+str);

//9:

sdf=new SimpleDateFormat("D");

str=sdf.format(d);

System.out.println("day of year is:"+str);

}

}

**Chapter 4**

**User manual**

**4.1 User Manual**

A user manual for a student registration form provides guidance for users on how to use the system effectively.it consists of following functionalities:

1. **Overview:-** The student registration form allows student to register for courses and academics. This manual will guide you through the process.
2. **Registration Process:** In that,we have to add our full name, age, courses, education, email, permanent address. After that when we click on the submit button then data stores into the database after that when you are a owner of the application then you have an absolutely login credentials of the this application the you can login by clicking on login button after that you have to enter username and password when it matches then it fetch data on the console when fails to login then it raise an exception.

This user manual covers the basic steps for registering and seeking support. It provides a concise guide for users to navigate the student registration form system.

**4.2 Menu Explanation**

**1. Form (student Registration):**

Here, we have to enter data in the textbox then click on the **SAVE** button then data store in the database. Also we have **CLEAR** button to clear the form and for exit click on **EXIT** button.

When you have a admin and you want to login then click on **LOGIN** button.

**2. Login page:**

Here, admin have to enter correct username and password when username or password is incorrect then it raise exception on console otherwise it fetch the data on the console.

When admin enter correct username and password then data fetch successfully and application gets stopped automatically.

**Chapter 5**

**CONCLUSION**

**5.1 Limitations and drawbacks**

Student registration forms can have limitations or drawbacks, which may affect their effectiveness, user experience, or data integrity.

Here are some common limitations or drawbacks:

1. **Limited Accessibility:** Our form having limited as this is not useful for the international students as well.
2. **Complex form:** Forms with too many fields or unclear instructions can confuse users, leading to errors or incomplete submissions.
3. **Insufficient security:** Forms that lack robust security features may expose sensitive information to unauthorized access or cyber threats.
4. **Data errors:** Forms without real-time validation may allow incorrect or invalid data to be submitted, requiring additional follow-up and corrections.
5. **Limited functionality:** Forms that can't interact with other educational platforms (like learning management systems or student information systems)
6. **Weak authentication:** Forms with weak authentication mechanisms might allow unauthorized access.

This is some of the limitations or drawbacks of our student registration form.

**5.2 Future Enhancement**

1. Improved user interface and experience

2. Enhanced security and privacy

3. Real time data validation

4. Integration with other systems

5. Improved communication and support

**5.3 Conclusion**

Student registration form is important for educational institutions, enabling efficient data collection and streamlined administrative processes. It should be user-friendly, with intuitive design. It is very useful for the student to register in any organization and the university in proper manner. Due to this student can be register from the remote distance which saves time and money for the student. This application also helpful to make registration of the student more automated and eco-friendly. It also saves the man power.

This is the conclusion of the **student registration form.**

**5.4** **References & Bibliography**

<https://www.javatpoint.com/java-awt>

<https://www.geeksforgeeks.org/java-awt-tutorial/>

<https://www.javatpoint.com/awt-program-in-java>

<https://chat.openai.com/>