PET PARTNER

An

Object-Oriented Programming through Java Course Project Report

in partial fulfilment of the degree

Bachelor of Technology

in

Computer Science & Engineering

By

M.Abhay Raj	**2103A52155**
Ch.Keerthana	**2103A52129**
K.Pavan Raj	**2103A52146**
K.Abhinay	**2103A52141**
K.Rajeev	**2103A52143**

Under the Guidance

of

Padala Sravan(Asst.Prof)

School of CS & AI

Submited to

SR UNIVERSITY

SR UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the Object Oriented Programming through Java-Course Project Report entitled **"PET PARTNER"** is a record of bonafide work carried out by the students M.Abhay Raj, Ch.Keerthana ,K.Pavan Raj, K.Abhinay ,K.Rajeev bearing RollNo(s) 2103A52155, 2103A52129,2103A52146,2103A52141, 2103A52143 during the academic year 2022-2023 in partial fulfillment of the award of the degree of ***Bachelor of Technology*** in **Computer Science & Engineering** by the SR University,Hasanparthy,Warangal.

Lab In-charge **Head of the Department**

ACKNOWLEDGEMENT

We owe an enormous debt of gratitude to our project guide MR .Padala Sravan ,as well as Dean of CSE AIML Dr.C.V.Guru Roa Head of the CSE Department Dr.M.Sheshikala, Associate Professor for guiding us from the beginning through the end of the course Project with their intellectual advices and insightful suggestions. We truly value their consistent feedback on our progress, which always constructive and encouraging and ultimately drove us to the right direction .We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved Principal, Dr.V.MAHESH, for his continuous support and guidance tocompletethis project. Finally, we express our thanks to all the teaching and non-teaching staff of the department for their suggestions and timely support.

TABLE OF CONTENTS:

ABSTRACT:

The "Pet Partner" platform is a web-based application that connects busy pet owners with registered caretakers for flexible and reliable pet care services. It features user-friendly registration, search, and booking systems for both pet owners and caretakers. The platform includes a secure payment gateway, a rating and review system, and notification features. An admin panel is in place for managing accounts and addressing issues. Data security measures are also implemented. Overall, "Pet Partner" aims to help pet owners find trustworthy caretakers, allowing them to balance their schedules while ensuring their pets' well-being, strengthening the bond between humans and their pets.

INTRODUCTION:

In today's fast-paced world, pet ownership is on the rise, and pet owners are increasingly seeking reliable and convenient pet care services. To address this growing demand, our team has developed an innovative solution: "Pet Partner," a comprehensive pet care management system built using Java programming language. Pet Partner aims to bridge the gap between pet owners and professional caretakers by providing a user-friendly platform for scheduling and managing pet care services efficiently.

Project Overview

Pet Partner is a feature-rich Java application designed to streamline the process of pet care service delivery. This project focuses on enhancing the pet ownership experience by connecting pet owners with trained caretakers, ensuring their beloved pets receive the best care possible. The system offers a range of functionalities catering to both pet owners and caretakers, allowing them to interact seamlessly and manage pet care services effortlessly.

Key Features

User Authentication: Pet Partner provides a secure login system for both pet owners and caretakers. Users can create accounts, log in securely, and manage their profiles within the system.User Profile Management: Pet owners can create and manage profiles for their pets, including essential details like pet names, species, and specific care requirements. This information helps caretakers understand the unique needs of each pet.Care Event Scheduling: Pet owners can schedule various pet care events, such as walking, feeding, playtime, and grooming, based on their preferences and pet's requirements.

Caretakers can view these schedules and confirm their availability for specific events.Real-time Communication: The system supports real-time communication between pet owners and caretakers. Users can exchange messages, discuss pet care instructions, and address any concerns directly through the platform.

Payment and Billing: Pet Partner includes a billing system to calculate charges for different care events. Caretakers can view their earnings, and pet owners can make payments securely through the application.

User-Friendly Interface: The graphical user interface (GUI) of Pet Partner is intuitive and easy to navigate. It provides a seamless experience for users, ensuring they can access the required information and functionalities effortlessly.

Project Benefits

Pet Partner offers several benefits to both pet owners and caretakers, including:

Convenience: Pet owners can schedule pet care services at their convenience, and caretakers can manage their availability, creating a hassle-free experience for both parties.

Reliability: By connecting pet owners with verified and professional caretakers, Pet Partner ensures reliable and trustworthy pet care services.

Efficiency: The automation of scheduling, communication, and payment processes enhances efficiency, allowing pet owners and caretakers to focus on the well-being of pets.

Personalized Care: Pet owners can provide specific instructions and preferences, ensuring that their pets receive personalized and tailored care from experienced caretakers.

OBJECTIVE:

The “Pet Partner” platform is a web-based application designed to address the needs of busy pet owners who require flexible and reliable pet care services. The platform connects pet owners with registered caretakers, providing them with hourly pet care services and ensuring the well-being of their pets when they are unable to personally attend to their needs. Caretakers, on the other hand, can sign up on the platform and offer their services, gaining the flexibility to work at their preferred times while earning income.

- Our application name is “petpartner”.
- We want to create a application where people can book caretakers for their pets. We also provide different services like Pet walking, pet grooming, pet boarding.
- The people who want to be apply for partner role should undergo a training and after becoming a certified trainer they can start their services.
- We ourselves give training to people who want to be a partner and make them a certified trainer.
- The partners can earn on hourly basis. It is a huge help for part-time job seekers.
- We are available all over India.

DEFINITIONS OF THE PROJECT:

1. **Graphical User Interface (GUI):** A GUI is a type of user interface that allows users to interact with a program through graphical elements like windows, buttons, and text fields. In this project, Swing components are used to create a GUI for the Pet Partner application.
2. **Event-Driven Programming:** Event-driven programming is a paradigm in which the flow of a program is determned by events such as user actions (e.g., button clicks). Action listeners are used to respond to events and trigger specific actions in the application.
3. **Swing Components:** Swing is a set of Java GUI components that provide a wide range of elements like frames, labels, buttons, and combo boxes. These components are used to build the user interface of the application.
4. **JFrame:** A `JFrame` is a top-level container for GUI components. It provides the main window of the application.
5. **Layout Manager:** Layout managers are used to control the positioning and sizing of components within a container. In this project, the `FlowLayout` layout manager is used to arrange components within frames.
6. **ActionListener:** An `ActionListener` is an interface in Java that defines the `actionPerformed` method. It's used to handle events triggered by user actions, such as button clicks.
7. **Custom Classes:** Custom classes (`Pet` and `CareEvent`) are defined to represent and store data about pets and pet care events. These classes are used to encapsulate data and logic related to pets and care events.
8. **Collections (ArrayList):** An `ArrayList` is a dynamic data structure that can store and manage collections of objects. In this project, `ArrayList` is used to store lists of pets and care events.
9. **Data Parsing:** Data parsing involves converting data from one format to another. In this project, parsing is used to extract time values from time range strings.
10. **Secure Data Entry:** The `JPasswordField` component is used to securely enter and store sensitive data, like passwords, where the input is usually masked for security.
11. **User Authentication**: Although not explicitly implemented in this project, user authentication is a concept where user credentials (e.g., username and password) are verified to ensure that only authorized users can access specific features of an application.
12. **Dialog Boxes:** Dialog boxes (e.g., `JOptionPane`) are used to display messages or prompts to the user, such as login success messages and details submission confirmation.

MODULES DESCRIPTION:

Pet Class:Represents the data structure for a pet, encapsulating properties and behaviors related to a pet.

CareEvent Class:Represents the data structure for a care event, encapsulating details like the type of event, time, and charge.

PetP Class (Main Class):Contains the main method as the entry point of the program.

Provides methods for creating the graphical user interface (GUI) and handling user interactions.

Contains methods for user and caretaker login (login(String role)), capturing pet details (openPetDetailsForm(String username, JFrame parentFrame)), and caretaker details (openCaretakerDetailsForm(String username, JFrame parentFrame)).Includes a method to calculate charges based on the selected event and time range (calculateCharges(String event, String timeRange)).

MODULE INPUT:

It allows users to log in as either a user or a caretaker, enter pet details (for users), and select pet care services and time ranges (for both users and caretakers). The code handles various events and interactions using Swing components.

Here is a brief overview of the functionality:

User Login: Users can log in by entering the username and password.

Caretaker Login: Caretakers can log in by entering their username and password.

User (Pet Owner) Details Entry:Users can enter details such as owner's name, pet name, species, selected pet care event, and available time range.Pet details are stored in Pet objects and added to the pets list.

Caretaker Details Entry:Caretakers can enter their name, select a pet care event, and choose an available time range.Caretaker details, along with calculated charges, are stored in CareEvent objects and added to the careEvents list.Please provide specific input or let me know if you need assistance with a particular aspect of this code.

MODULE OUTPUT:

It includes user interfaces for user and caretaker login, entering pet details, and selecting pet care services and time ranges. When the user or caretaker submits the details, they are stored in Pet and CareEvent objects, respectively. Below is a summary of the code's functionality and the expected output flow

Start of the Program:The program starts by displaying the main window titled "Pet Partner."Users can click "User Login" or "Caretaker Login" buttons to proceed.

User Login:After clicking "User Login," a new window opens where the user can enter their username and password.Upon successful login, the user is shown a form to enter pet details.The user enters owner's name, pet name, species, selects a pet care event, and chooses an available time range.

After clicking "Submit," the entered pet details are stored in a Pet object and displayed in a confirmation dialog.

Caretaker Login:After clicking "Caretaker Login," a new window opens where the caretaker can enter their username and password.Upon successful login, the caretaker is shown a form to select a pet care event and available time range.After clicking "Submit," the selected pet care event, time range, and calculated charge are stored in a CareEvent object.A confirmation dialog displays the selected event, time range, and the corresponding charge.

Exiting the Application:At any point, the user or caretaker can click the "Exit" button to close the application.

Expected Output:If a user logs in successfully and submits pet details, the confirmation dialog displays the entered pet details.If a caretaker logs in successfully and submits pet care event and time range, the confirmation dialog displays the selected event, time range, and the calculated charge in rupees.

Please note that the actual output and user interactions depend on the specific inputs provided during runtime. You can run the code in a Java environment to see the graphical interface and interact with the application as described above.

LITERATURE SURVEY:

Pet care management systems have become increasingly popular due to the growing number of pet owners and their need for reliable pet care services. Several existing systems and research studies have explored various aspects of pet care management, including scheduling, communication, and payment processing. Here's a brief overview of some related literature in the field:

"A Survey of Pet Monitoring Systems"This survey explores different pet monitoring systems and their applications in pet care. It covers wearable devices, IoT-based solutions, and mobile applications designed to monitor pets' health, activities, and behavior. The study delves into the technologies employed and the impact of these systems on pet owners' peace of mind.

"Mobile Applications for Pet Care Services"Research in mobile applications for pet care services focuses on user experience, interface design, and functionality. Studies in this area investigate the usability of mobile apps for scheduling pet care services, real-time communication with caretakers, and payment processing. The literature provides insights into user preferences and the challenges faced in designing intuitive pet care apps.

"Automated Scheduling and Routing Algorithms in Pet Care Services"This research area focuses on the development of algorithms for automated scheduling and routing of pet care services. It explores optimization techniques to efficiently assign caretakers to different tasks based on their availability, location, and expertise. These algorithms aim to reduce costs, improve efficiency, and enhance the overall user experience for both pet owners and caretakers.

"Secure Payment Systems for Pet Care Services"

Studies related to payment systems in pet care services investigate secure and convenient methods for processing payments between pet owners and caretakers. Researchers explore encryption techniques, secure APIs, and mobile payment gateways to ensure safe and reliable financial transactions within pet care applications. The literature emphasizes the importance of data security and user privacy in payment processing.

"User Interaction and Experience in Pet Care Management Systems"

User-centered design and user experience (UX) studies focus on understanding user behavior and preferences in pet care management systems. Researchers conduct surveys, interviews, and usability testing to evaluate the effectiveness of user interfaces, features, and overall satisfaction of pet owners and caretakers. The findings guide the design and development of user-friendly pet care applications.

Gap Analysis:

While existing literature provides valuable insights into specific aspects of pet care management, there is a notable gap in integrating all essential features seamlessly into a comprehensive pet care application. The presented Java-based Pet Partner application aims to bridge this gap by combining user authentication, scheduling, real-time communication, and payment processing within a single platform. This integration addresses the holistic needs of pet owners and caretakers, offering a unified and convenient experience in the realm of pet care management.

EXISTING SYSTEM:

It is designed to connect pet owners with caretakers for various pet care services. The application provides different functionalities for pet owners and caretakers.

1. User Roles:

Pet Owners: Individuals seeking pet care services for their pets.

Caretakers: Individuals offering pet care services, such as walking, feeding, playtime, grooming, and hourly care.

2. Features and Functionalities:

User Authentication:

Users can log in as either pet owners or caretakers.

Pet Details Management: Pet owners can provide information about their pets, including owner's name, pet name, species, selected care event, and available time range.

Scheduling: Pet owners can schedule specific pet care events (e.g., walking, feeding) at preferred time ranges. Caretakers can view and confirm scheduled events.

Event Confirmation: Caretakers can confirm scheduled events and specify the services provided.

Payment Processing: Calculation of service charges based on selected events and time ranges.

Secure payment processing between pet owners and caretakers.

User Interface: The application uses Swing components to create a graphical user interface with login screens, input forms, and buttons for user interactions.

3. Classes and Data Structures:Pet Class: Represents a pet with attributes such as owner's name, pet name, species, selected event, and time range.

CareEvent Class: Represents a pet care event with attributes such as event type, time, and charge.

Main Class (PetP): Contains the main method and orchestrates the application's flow.Manages user authentication, input forms, event scheduling, and payment processing.

4. Payment Calculation:The application calculates service charges based on selected pet care events and time ranges.Hourly Care events are charged based on the duration of the service.

5. User Interaction:Users interact with the application through graphical forms and buttons.

Messages and notifications are displayed using JOptionPane for user feedback.

6. GUI Design:The application's graphical user interface includes login screens and forms for entering pet details, scheduling events, and confirming services.

7. Execution:The application starts by invoking the main method, initializing the Swing components, and displaying the main frame.

8. Areas for Improvement:The application can be enhanced by adding features like user feedback, rating systems, and notifications for scheduled events.

Error handling and validation of user inputs can be improved for a more robust user experience.

•“Petmojo” is an application which created a huge impact in pet industry.

•This application is developed in India and currently running very successfully.

•In this the customers can book caretakers for their dogs because they are busy and cant take care of them.

•Petmojo provides different services like pet walking, pet grooming, pet boarding.

•It also has certified trainers.

PROPOSED SYSTEM

It aims to enhance the existing Pet Partner application by implementing additional features, improving user experience, and ensuring robustness in functionality and security.

1. User Authentication:Implement secure authentication mechanisms, such as password hashing and salting, to enhance user login security.

2. User Roles and Privileges:Implement multiple user roles (Admin, Pet Owner, Caretaker) with specific privileges and access levels.Admin can manage user accounts, view transaction history, and resolve disputes.Pet Owners can schedule pet care services, view service history, and provide feedback.

Caretakers can accept/reject service requests, update service status, and receive payments.

3. Enhanced User Interface:Design an intuitive and responsive user interface with modern UI/UX

principles.Implement error handling and validation for user inputs to provide informative feedback to users.

4. Service Requests and Scheduling:

Pet Owners can send service requests to Caretakers, specifying the type of service, preferred time, and duration.Caretakers can accept or reject service requests based on their availability and preferences.

5. Real-time Notifications:Implement real-time notifications (e.g., push notifications, emails) to inform users about service requests, confirmations, and payment updates.

6. Payment Gateway Integration:Integrate a secure payment gateway to facilitate online payments between Pet Owners and Caretakers.Implement payment tracking and receipts generation for transparency.

7. Service History and Feedback:Maintain a detailed history of completed services for both Pet Owners and Caretakers.Allow Pet Owners to provide ratings and feedback for services, contributing to Caretaker reputation.

8. Reporting and Analytics:Implement reporting features for Admin to analyze business performance, user satisfaction, and service demand.Generate insights to improve service offerings and optimize operations.

9. Security Measures:Implement data encryption for sensitive user information.

Regularly update security protocols to protect against potential vulnerabilities.

10. Documentation and Support:Provide comprehensive documentation for users and developers explaining how to use the application and its features.

Offer user support through various channels (email, chat, phone) to assist users with issues and inquiries.

11. Testing and Quality Assurance:Perform thorough testing, including unit tests, integration tests, and user acceptance tests, to ensure the application's functionality, reliability, and performance.

12. Scalability:Design the application architecture to be scalable, allowing it to handle a growing user base and increased service demands.

DEFINITIONS OF THE PROJECT:

1. **Graphical User Interface (GUI):** A GUI is a type of user interface that allows users to interact with a program through graphical elements like windows, buttons, and text fields. In this project, Swing components are used to create a GUI for the Pet Partner application.

2. **Event-Driven Programming:** Event-driven programming is a paradigm in which the flow of a program is determined by events such as user actions (e.g., button clicks). Action listeners are used to respond to events and trigger specific actions in the application.

3. **Swing Components:** Swing is a set of Java GUI components that provide a wide range of elements like frames, labels, buttons, and combo boxes. These components are used to build the user interface of

the application.

4. **JFrame:** A `JFrame` is a top-level container for GUI components. It provides the main window of the application.

5**. Layout Manager:** Layout managers are used to control the positioning and sizing of components within a container. In this project, the `FlowLayout` layout manager is used to arrange components within frames.

6. **ActionListener:** An `ActionListener` is an interface in Java that defines the `actionPerformed` method. It's used to handle events triggered by user actions, such as button clicks.

7. **Custom Classes:** Custom classes (`Pet` and `CareEvent`) are defined to represent and store data about pets and pet care events. These classes are used to encapsulate data and logic related to pets and care events.

8. **Collections (ArrayList):** An `ArrayList` is a dynamic data structure that can store and manage collections of objects. In this project, `ArrayList` is used to store lists of pets and care events.

9. **Data Parsing:** Data parsing involves converting data from one format to another. In this project, parsing is used to extract time values from time range strings.

10. **Secure Data Entry:** The `JPasswordField` component is used to securely enter and store sensitive data, like passwords, where the input is usually masked for security.

11. **User Authentication**: Although not explicitly implemented in this project, user authentication is a concept where user credentials (e.g., username and password) are verified to ensure that only authorized users can access specific features of an application.

12. **Dialog Boxes:** Dialog boxes (e.g., `JOptionPane`) are used to display messages or prompts to the user, such as login success messages and details submission confirmations.

FEASIBILITY

A feasibility study is an assessment of the practicality and viability of a proposed project, system, or endeavor. It aims to evaluate various aspects to determine if the project is feasible, achievable, and worth pursuing.

1. Technical Feasibility:

Programming Language: The code is written in Java, a widely used and well-supported programming language.

GUI Framework: Swing is used for the graphical user interface, which is a part of Java's standard library.

Data Management: Data is managed using basic data structures (lists) to store pet and care event information.

Calculation: The code involves calculations for determining charges based on selected care events and time ranges.

2. Operational Feasibility:

User Interface: The application provides a relatively simple and intuitive user interface, allowing users to log in, enter pet details, and select care events. The interface is easy to navigate.

User Experience: The application's basic functionality ensures a seamless experience for users. However, there might be room for improvements, such as error handling and validation of user inputs.

Scalability.

3. Economic Feasibility:

Cost of Development: The development cost is minimal since it uses standard Java libraries without the need for additional licenses or frameworks.

Maintenance: Maintenance costs are relatively low for small-scale applications. However, if the application grows and requires additional features, ongoing maintenance might become a factor to consider.

4. Legal and Compliance Feasibility:

Data Privacy: The code does not handle sensitive personal data in its current form. However, if the application were to expand and store sensitive user information, compliance with data privacy regulations (such as GDPR) would be essential.

Intellectual Property: The code does not involve any proprietary algorithms or technologies, ensuring there are no intellectual property concerns.

5. **Schedule Feasibility:**

Development Time: The provided code represents a basic version of the application. Developing additional features, enhancing the user interface, and adding error handling might extend the development timeline.

Testing: Proper testing, including unit tests and user acceptance tests, is crucial to ensure the application functions as intended and is free of bugs.In summary, the code demonstrates the core functionalities of a pet caretaker application and is technically feasible for its current scope. However, for a fully functional and production-ready application, further enhancements, testing, and considerations for scalability and user experience would be necessary.

SOFTWARE ENVIRONMENT:

Java Technology

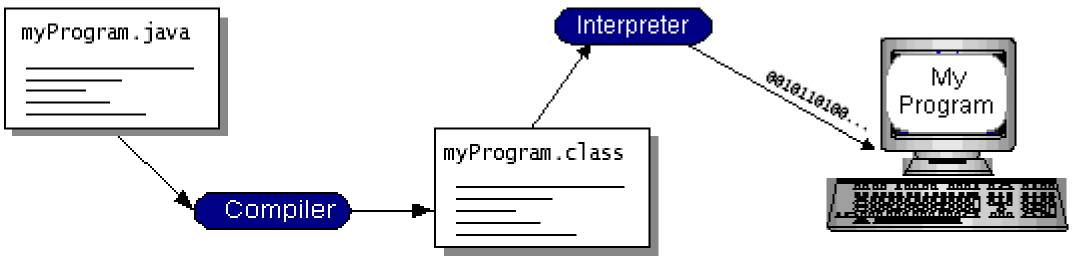
Java technology is both a programming language and a platform.

The Java Programming Language

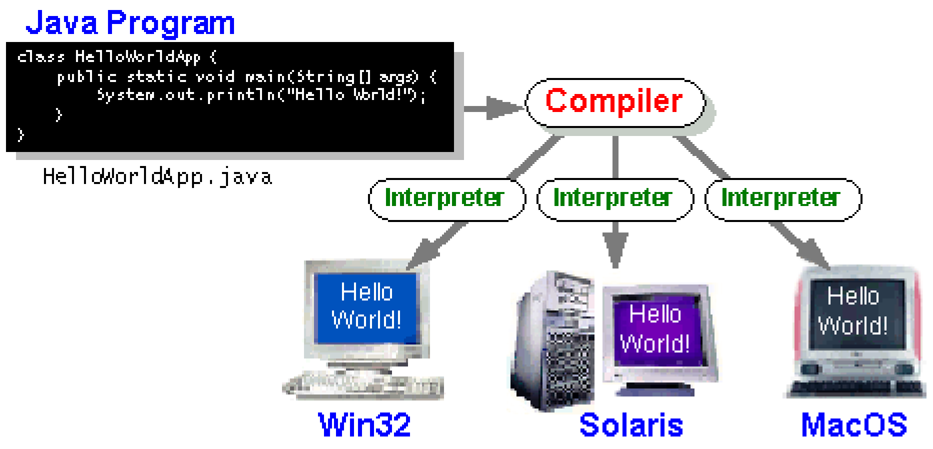
The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

- Simple
- Architecture neutral
- Object oriented
- Portable
- Distributed
- High performance
- Interpreted
- Multithreaded
- Robust
- Dynamic
- Secure

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called *Java byte codes* —the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.



You can think of Java byte codes as the machine code instructions for the *Java Virtual Machine* (Java VM). Every Java interpreter, whether it's a development tool or a Web browser that can run applets, is an implementation of the Java VM. Java byte codes help make "write once, run anywhere" possible. You can compile your program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java VM. That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.



```
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

The Java Platform

A *platform* is the hardware or software environment in which a program runs. We've already mentioned some of the most popular platforms like Windows 2000, Linux, Solaris, and MacOS. Most platforms can be described as a combination of the operating system and hardware. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other hardware-based platforms.

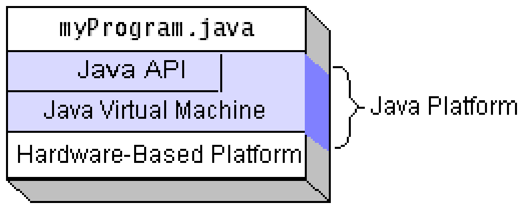
The Java platform has two components:

- The *Java Virtual Machine* (Java VM)
- The *Java Application Programming Interface* (Java API)

You've already been introduced to the Java VM. It's the base for the Java platform and is ported onto various hardware-based platforms.

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries of related classes and interfaces; these libraries are known as *packages*. The next section, What Can Java Technology Do? Highlights what functionality some of the packages in the Java API provide.

The following figure depicts a program that's running on the Java platform. As the figure shows, the Java API and the virtual machine insulate the program from the hardware.



Native code is code that after you compile it, the compiled code runs on a specific hardware platform. As a platform-independent environment, the Java platform can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring performance close to that of native code without threatening portability.

What Can Java Technology Do?

The most common types of programs written in the Java programming language are *applets* and *applications*. If you've surfed the Web, you're probably already familiar with applets. An applet is a program that adheres to certain conventions that allow it to run within a Java-enabled browser.

However, the Java programming language is not just for writing cute, entertaining applets for the Web. The general-purpose, high-level Java programming language is also a powerful software platform. Using the generous API, you can write many types of programs.

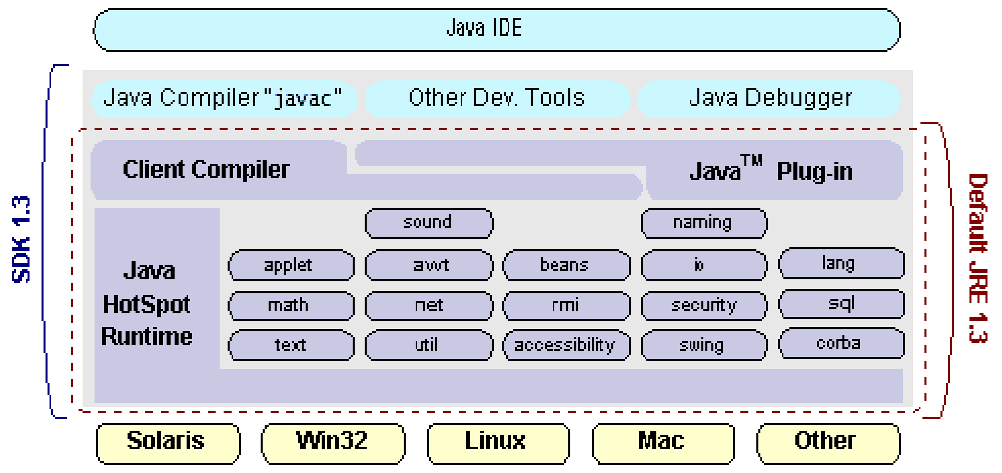
An application is a standalone program that runs directly on the Java platform. A special kind of application known as a *server* serves and supports clients on a network. Examples of servers are Web servers, proxy servers, mail servers, and print servers. Another specialized program is a *servlet*. A servlet can almost be thought of as an applet that runs on the server side. Java Servlets are a popular choice for building interactive web applications, replacing the use of CGI scripts. Servlets are similar to applets in that they are runtime extensions of applications. Instead of working in browsers, though, servlets run within Java Web servers, configuring or tailoring the server.

How does the API support all these kinds of programs? It does so with packages of software components that provides a wide range of functionality. Every full implementation of the Java platform gives you the following features:

- **The essentials**: Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.
- **Applets**: The set of conventions used by applets.

- **Networking**: URLs, TCP (Transmission Control Protocol), UDP (User Data gram Protocol) sockets, and IP (Internet Protocol) addresses.
- **Internationalization**: Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.
- **Security**: Both low level and high level, including electronic signatures, public and private key management, access control, and certificates.
- **Software components**: Known as JavaBeans^{TM}, can plug into existing component architectures.
- **Object serialization**: Allows lightweight persistence and communication via Remote Method Invocation (RMI).
- **Java Database Connectivity (JDBC^{TM})**: Provides uniform access to a wide range of relational databases.

The Java platform also has APIs for 2D and 3D graphics, accessibility, servers, collaboration, telephony, speech, animation, and more. The following figure depicts what is included in the Java 2 SDK.



How Will Java Technology Change My Life?

We can't promise you fame, fortune, or even a job if you learn the Java programming language. Still, it is likely to make your programs better and requires less effort than other languages. We believe that Java technology will help you do the following:

- **Get started quickly**: Although the Java programming language is a powerful object-oriented language, it's easy to learn, especially for programmers already familiar with C or C++.

- **Write less code**: Comparisons of program metrics (class counts, method counts, and so on) suggest that a program written in the Java programming language can be four times smaller than the same program in C++.
- **Write better code**: The Java programming language encourages good coding practices, and its garbage collection helps you avoid memory leaks. Its object orientation, its JavaBeans component architecture, and its wide-ranging, easily extendible API let you reuse other people's tested code and introduce fewer bugs.
- **Develop programs more quickly**: Your development time may be as much as twice as fast versus writing the same program in C++. Why? You write fewer lines of code and it is a simpler programming language than C++.
- **Avoid platform dependencies with 100% Pure Java**: You can keep your program portable by avoiding the use of libraries written in other languages. The 100% Pure Java^{TM} Product Certification Program has a repository of historical process manuals, white papers, brochures, and similar materials online.
- **Write once, run anywhere**: Because 100% Pure Java programs are compiled into machine-independent byte codes, they run consistently on any Java platform.
- **Distribute software more easily**: You can upgrade applets easily from a central server. Applets take advantage of the feature of allowing new classes to be loaded "on the fly," without recompiling the entire program.

TESTING:

Functional Testing:

Verify if the user and caretaker login functionalities work correctly.

Check if users can submit pet details, and caretakers can submit care event details.

Test the interaction between users and caretakers to ensure that pet details and care event details are properly linked.

Boundary Testing:

Test boundary cases for input fields such as owner's name, pet name, and species.

Verify the application's behavior when users enter incorrect or invalid data.

Event and Time Range Validation:

Validate the application's handling of different pet care events and available time ranges.

Check if the charges for each pet care event are calculated accurately.

Concurrency Testing:

Simulate multiple users and caretakers accessing the system simultaneously to check for any concurrency issues.

Ensure that the system can handle concurrent user interactions without data corruption or inconsistencies.

User Interface (UI) Testing:

Verify the layout and alignment of UI components to ensure a consistent and user-friendly interface.

Test user interactions with buttons, input fields, and dropdown menus.

Error Handling and Validation:

Check how the application handles errors, such as incorrect login credentials or invalid input data.

Verify that appropriate error messages are displayed to users in case of invalid input or system failures.

Integration Testing:

Test the integration between different modules, such as the user login module, pet details submission module, and caretaker details submission module.

Ensure that data flows correctly between these modules.

Performance Testing:

Assess the application's performance under various loads to ensure it can handle multiple simultaneous users without significant slowdowns or crashes.

Security Testing:

If the application handles sensitive user data, perform security testing to identify vulnerabilities related to data privacy and user authentication.

Database Integration (if applicable):

If the application integrates with a database, perform tests to ensure data integrity, consistency, and proper storage/retrieval of information.

Implementing these types of system checks will help you identify and address potential issues in the application, ensuring its reliability and user satisfaction.

System testing:

System testing is a level of software testing where the complete and integrated software is tested to evaluate its compliance with the specified requirements. In the provided code, there is no explicit implementation of system testing. System testing typically involves testing the entire application as a whole to ensure that it behaves according to the specified requirements and works seamlessly in different scenarios.

DESIGN:

First we have main menu frame. In that we have user login and caretaker login.

Both user and caretaker has there separate login frames where they enter username and password.

After logging in as user we have user details frame.

Logging in as caretaker we have caretaker details frame

We also have message frames in the middle.

INPUT DESIGN:

Owner's Name: JTextField for entering the owner's name.

Pet Name: JTextField for entering the pet's name.

Species: JTextField for entering the species of the pet.

Select Pet Care Event: JComboBox (dropdown) for selecting the type of pet care event (Walking, Feeding, Playtime, Grooming, Hourly Care).

Available Time Range: JComboBox (dropdown) for selecting the time range for the pet care event.

Caretaker Details Input Form:

Name: JTextField for entering the caretaker's name.

Select Pet Care Event: JComboBox (dropdown) for selecting the type of pet care event (Walking, Feeding, Playtime, Grooming, Hourly Care).

Available Time Range: JComboBox (dropdown) for selecting the time range for the pet care event.

These input forms allow users and caretakers to enter their details, including the type of pet care event and the time range during which the event will occur. Users can select the appropriate options from the dropdown menus and enter their names in the text fields. After entering the details and submitting the form, the information is stored and processed within the program logic.

OUTPUT DESIGN:

Main Interface:

The main interface of the application displays the title "Pet Partner" at the top.

Below the title, there are three buttons: "User Login", "Caretaker Login", and "Exit".

When the user clicks on "User Login" or "Caretaker Login", they are prompted to enter their username and password.

User Login Interface:

When the user clicks on "User Login", a new window titled "User Login" opens.

In this window, there are fields for entering "Username" and "Password".

The user can input their username and password and click the "Login" button.

Upon successful login, the user receives a message dialog indicating the successful login and can proceed to enter pet details.

Caretaker Login Interface:
When the user clicks on "Caretaker Login", a new window titled "Caretaker Login" opens.Similar to the user login interface, there are fields for entering "Username" and "Password".The caretaker can input their username and password and click the "Login" button.Upon successful login, the caretaker receives a message dialog indicating the successful login and can proceed to enter details about the pet care event they will handle.
Pet Details Input Interface:
After successful user login, a new window titled "Pet Details" opens.
In this window, there are fields for entering "Owner's Name", "Pet Name", and "Species".
There are dropdown menus for selecting the "Pet Care Event" (Walking, Feeding, Playtime, Grooming, Hourly Care) and "Available Time Range".
The user can input the details and select the pet care event and time range from the dropdown menus.
After clicking the "Submit" button, the entered pet details are stored, and a message dialog confirms the successful submission.
Caretaker Details Input Interface:
After successful caretaker login, a new window titled "Caretaker Details" opens.
In this window, there is a field for entering the caretaker's name.
There are dropdown menus for selecting the "Pet Care Event" (Walking, Feeding, Playtime, Grooming, Hourly Care) and "Available Time Range".
The caretaker can input their name and select the pet care event and time range from the dropdown menus.
After clicking the "Submit" button, the entered details are processed to calculate the event charge, and a message dialog displays the pet care event, time range, and the corresponding charge in rupees.
Another message dialog confirms the successful submission..

ADVANTAGES:

For Pet Caretakers:

Job Opportunities: Pet caretakers can find job opportunities easily, connecting them with pet owners in need of their services.
Flexible Schedule: Caretakers can set their availability, allowing them to work flexible hours and accommodate their personal schedules.
Earn Income: Caretakers can earn income by offering their services, making it a viable option for animal lovers seeking part-time or full-time employment.
Skill Utilization: Individuals with a passion for animals can utilize their skills and knowledge to provide

quality pet care services.

Build Reputation: Dedicated and reliable caretakers can build a positive reputation within the application, leading to more job opportunities and increased income.

Enhanced Skill Set: Caretakers can enhance their skills and knowledge by working with various pets, learning about different breeds, behaviors, and specific care requirements.

Job Security: A steady stream of pet care requests ensures job security for caretakers, especially if they establish a good rapport with pet owners.

Community Engagement: Caretakers can engage with a community of fellow animal lovers, sharing experiences and insights, and potentially collaborating on special services or events.

For Pet Owners:

Convenience: Pet owners can easily find and hire experienced caretakers for their pets through the application, saving them time and effort.

Professional Services: Pet owners can access professional pet care services, ensuring their pets receive proper attention, exercise, and grooming.

Customized Care: Pet owners can specify their pet's needs, allowing them to tailor the care services based on their pet's requirements, such as walking, feeding, or grooming.

Peace of Mind: Knowing that their pets are in capable hands, pet owners can have peace of mind while they are at work, traveling, or unable to attend to their pets.

Regular Updates: The application can facilitate communication between caretakers and pet owners, allowing caretakers to provide updates, photos, and feedback on their pet's well-being.

Emergency Services: In case of emergencies or sudden schedule changes, pet owners can quickly find replacement caretakers through the application.

FLOW CHART:

When the user clicks the login button with valid username and password, the user is directed to the dashboard, indicating a successful login.

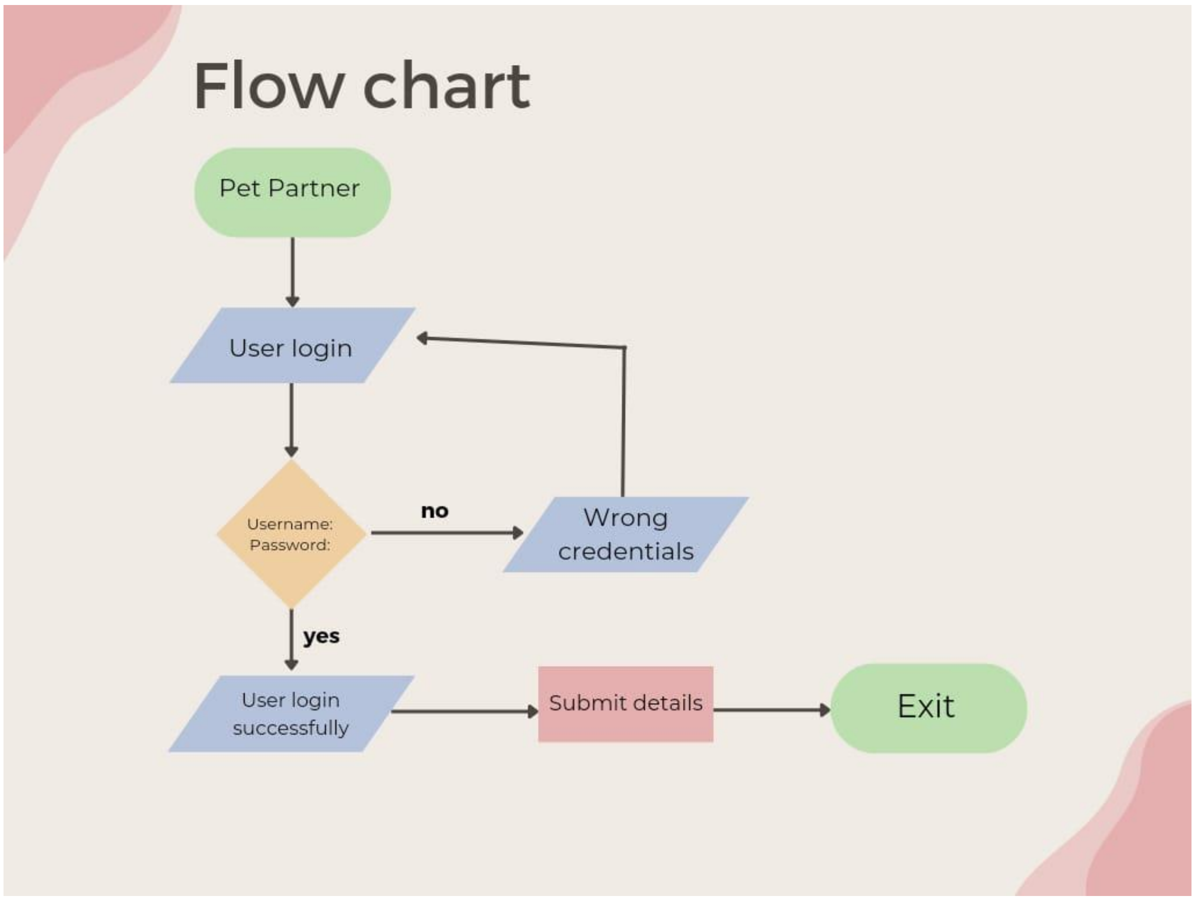
When the user clicks the login button with an incorrect username or password, the system displays an error message stating that the wrong credentials were provided. The user is not redirected to the dashboard and must re-enter their credentials to try again.

The "Submit details" option allows users to register for an account by providing necessary details. Upon successful submission, the user is redirected to the dashboard.

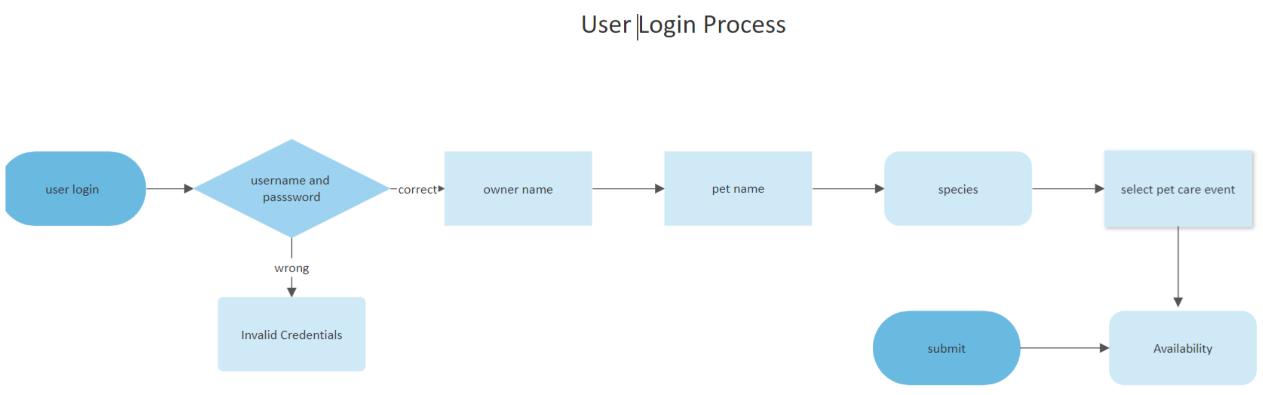
The "Exit" option provides a way for users to close the application or end their session. This action cancels any pending transactions or processes and closes all open windows and tabs associated with the application.

In conclusion, the Pet Partner User login flowchart is a useful tool for guiding users through the login process. It also helps them understand the necessary steps for registration and how to properly exit the application.

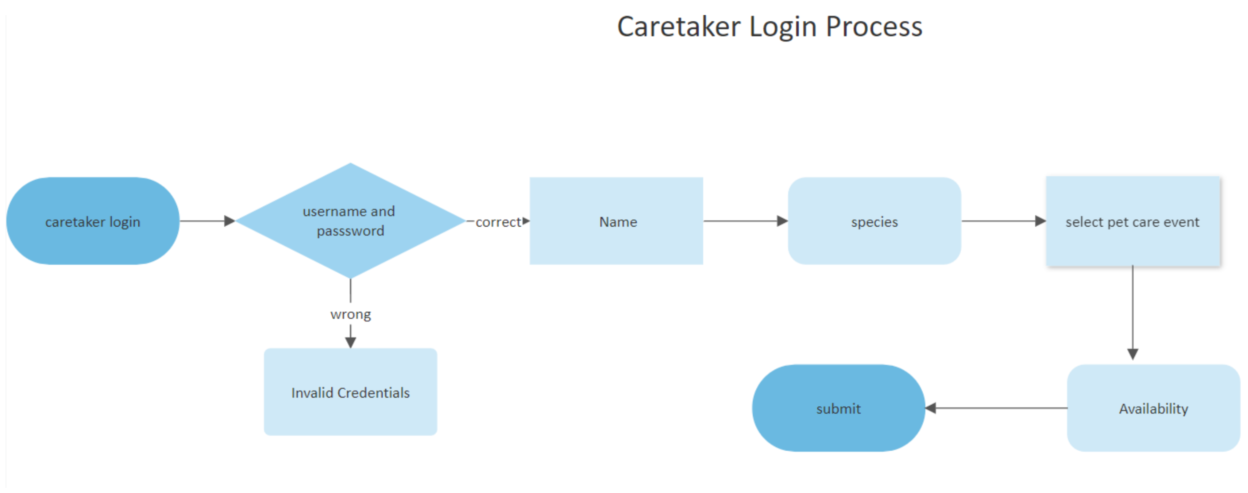
MAIN MENU FLOWCHART:



User login flowchart:



Caretaker login flowchart:



4.IMPLEMENTATION

4.1.CODE:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;

class Pet {
    private String ownerName;
    private String petName;
    private String species;
    private String selectedEvent;
    private String selectedTimeRange;

    public Pet(String ownerName, String petName, String species) {
        this.ownerName = ownerName;
        this.petName = petName;
        this.species = species;
        this.selectedEvent = null;
        this.selectedTimeRange = null;
    }

    public String getOwnerName() {
        return ownerName;
    }

    public String getPetName() {
        return petName;
    }

    public String getSpecies() {
        return species;
    }

    public String getSelectedEvent() {
        return selectedEvent;
    }
    public void setSelectedEvent(String event) {
        this.selectedEvent = event;
    }

    public String getSelectedTimeRange() {
        return selectedTimeRange;
    }

    public void setSelectedTimeRange(String timeRange) {
        this.selectedTimeRange = timeRange;
    }

    @Override
    public String toString() {
        return "Owner: " + ownerName + ", Pet Name: " + petName + ", Species: " + species +
                ", Event: " + selectedEvent + ", Time Range: " + selectedTimeRange;
    }
}

class CareEvent {
    private String event;
    private String time;
    private double charge;

    public CareEvent(String event, String time, double charge) {
        this.event = event;
        this.time = time;
        this.charge = charge;
    }

    public String getEvent() {
        return event;
    }
    public String getTime() {
        return time;
    }

    public double getCharge() {
        return charge;
    }

    @Override
    public String toString() {
        return "Event: " + event + ", Time: " + time + ", Charge: " + charge + " rupees";
    }
}

public class PetPartnerAppSwing {
    private static List<CareEvent> careEvents = new ArrayList<>();
    private static List<Pet> pets = new ArrayList<>();

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            createAndShowGUI();
        });
    }

    private static void createAndShowGUI() {
        JFrame frame = new JFrame("Pet Partner");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400, 150);
        frame.setLayout(new FlowLayout(FlowLayout.CENTER, 10, 10));

        JLabel label = new JLabel("Welcome to Pet Partner!");
        frame.add(label);

        JButton userButton = new JButton("User Login");
        userButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                login("User");
            }
        });
        frame.add(userButton);

        JButton caretakerButton = new JButton("Caretaker Login");
        caretakerButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                login("Caretaker");
            }
        });
        frame.add(caretakerButton);

        JButton exitButton = new JButton("Exit");
        exitButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                System.exit(0);
            }
        });
        frame.add(exitButton);

        frame.setVisible(true);
    }

    private static void login(String role) {
        JFrame loginFrame = new JFrame(role + " Login");
        loginFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        loginFrame.setSize(300, 150);
        loginFrame.setLayout(new FlowLayout(FlowLayout.CENTER, 10, 10));
```

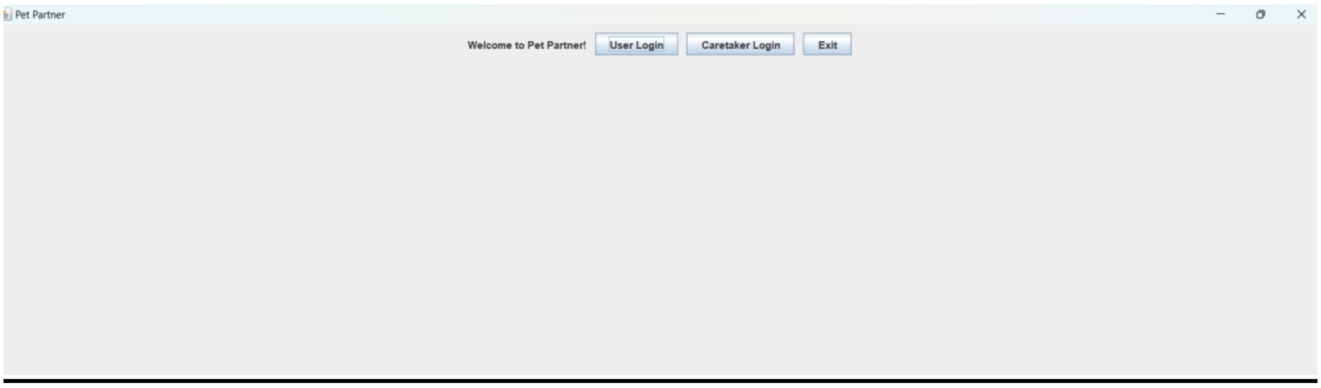
```
JLabel usernameLabel = new JLabel("Username:");
JTextField usernameField = new JTextField(20);
JLabel passwordLabel = new JLabel("Password:");
JPasswordField passwordField = new JPasswordField(20);
JButton loginButton = new JButton("Login");

loginButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        String username = usernameField.getText();
        char[] password = passwordField.getPassword();

        if (role.equals("User")) {
            JOptionPane.showMessageDialog(loginFrame, "User login successful!");
            openPetDetailsForm(username, loginFrame);
        } else {
            JOptionPane.showMessageDialog(loginFrame, "Caretaker login successful!");
            openCaretakerDetailsForm(username, loginFrame);
        }
    }
});

loginFrame.add(usernameLabel);
loginFrame.add(usernameField);
loginFrame.add(passwordLabel);
loginFrame.add(passwordField);
loginFrame.add(loginButton);

JButton exitButton = new JButton("Exit");
exitButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        loginFrame.dispose();
```

```
            }
        });
        loginFrame.add(exitButton);

        loginFrame.setVisible(true);
    }

    private static void openPetDetailsForm(String username, JFrame parentFrame) {
        JFrame petDetailsFrame = new JFrame("Pet Details");
        petDetailsFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        petDetailsFrame.setSize(400, 300);
        petDetailsFrame.setLayout(new FlowLayout(FlowLayout.CENTER, 10, 10));

        JLabel ownerNameLabel = new JLabel("Owner's Name:");
        JTextField ownerNameField = new JTextField(20);
        JLabel petNameLabel = new JLabel("Pet Name:");
        JTextField petNameField = new JTextField(20);
        JLabel speciesLabel = new JLabel("Species:");
        JTextField speciesField = new JTextField(20);

        JLabel eventLabel = new JLabel("Select Pet Care Event:");
        String[] events = {"Walking", "Feeding", "Playtime", "Grooming", "Hourly Care"};
        JComboBox<String> eventsComboBox = new JComboBox<>(events);

        JLabel timeLabel = new JLabel("Available Time Range:");
        String[] timeRanges = {"08:00 - 10:00", "12:00 - 14:00", "16:00 - 18:00", "20:00 - 22:00"};
        JComboBox<String> timeRangeComboBox = new JComboBox<>(timeRanges);

        JButton submitButton = new JButton("Submit");
        submitButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String ownerName = ownerNameField.getText();
                String petName = petNameField.getText();
        String species = speciesField.getText();
        String selectedEvent = eventsComboBox.getSelectedItem().toString();
        String selectedTimeRange = timeRangeComboBox.getSelectedItem().toString();

        Pet pet = new Pet(ownerName, petName, species);
        pet.setSelectedEvent(selectedEvent);
        pet.setSelectedTimeRange(selectedTimeRange);
        pets.add(pet);

        JOptionPane.showMessageDialog(petDetailsFrame, "Pet details submitted successfully!");
    }
});

petDetailsFrame.add(ownerNameLabel);
petDetailsFrame.add(ownerNameField);
petDetailsFrame.add(petNameLabel);
petDetailsFrame.add(petNameField);
petDetailsFrame.add(speciesLabel);
petDetailsFrame.add(speciesField);
petDetailsFrame.add(eventLabel);
petDetailsFrame.add(eventsComboBox);
petDetailsFrame.add(timeLabel);
petDetailsFrame.add(timeRangeComboBox);
petDetailsFrame.add(submitButton);

JButton exitButton = new JButton("Exit");
exitButton.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        petDetailsFrame.dispose();
    }
});
petDetailsFrame.add(exitButton);
```

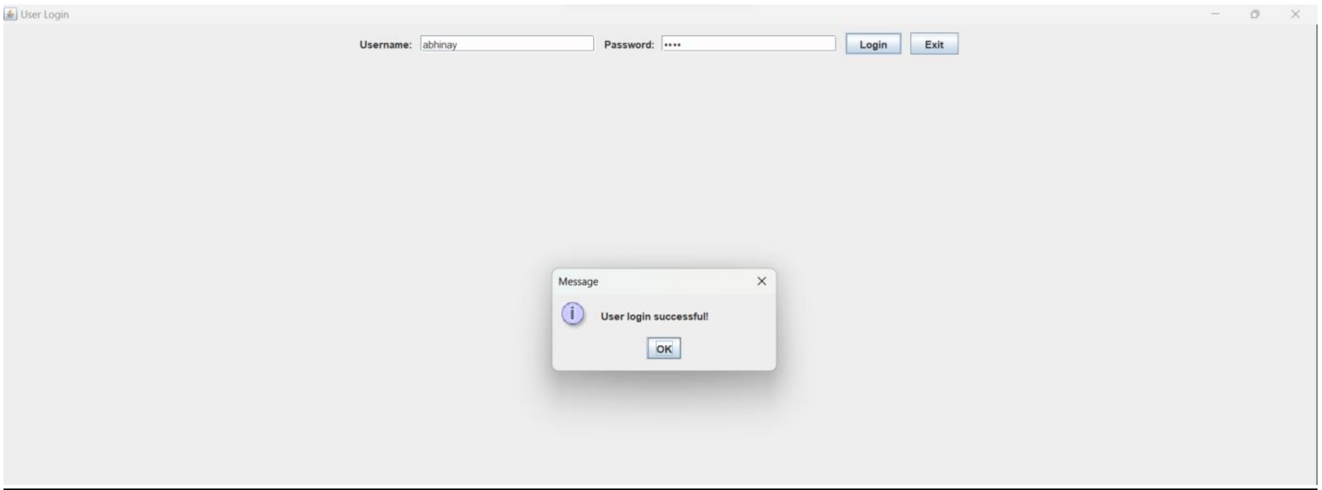
```
        petDetailsFrame.setVisible(true);
    }

    private static void openCaretakerDetailsForm(String username, JFrame parentFrame) {
        JFrame detailsFrame = new JFrame("Caretaker Details");
        detailsFrame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
        detailsFrame.setSize(400, 300);
        detailsFrame.setLayout(new FlowLayout(FlowLayout.CENTER, 10, 10));

        JLabel nameLabel = new JLabel("Name:");
        JTextField nameField = new JTextField(20);

        JLabel eventLabel = new JLabel("Select Pet Care Event:");
        String[] events = {"Walking", "Feeding", "Playtime", "Grooming", "Hourly Care"};
        JComboBox<String> eventsComboBox = new JComboBox<>(events);

        JLabel timeLabel = new JLabel("Available Time Range:");
        String[] timeRanges = {"08:00 - 10:00", "12:00 - 14:00", "16:00 - 18:00", "20:00 - 22:00"};
        JComboBox<String> timeRangeComboBox = new JComboBox<>(timeRanges);

        JButton submitButton = new JButton("Submit");
        submitButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                String name = nameField.getText();
                String selectedEvent = eventsComboBox.getSelectedItem().toString();
                String selectedTimeRange = timeRangeComboBox.getSelectedItem().toString();
                double eventCharge = calculateCharges(selectedEvent, selectedTimeRange);

                CareEvent careEvent = new CareEvent(selectedEvent, selectedTimeRange, eventCharge);
                careEvents.add(careEvent);

                JOptionPane.showMessageDialog(detailsFrame, "You will " + selectedEvent + " from " +
selectedTimeRange + ". Charge: " + eventCharge + " rupees");
```

```
            JOptionPane.showMessageDialog(detailsFrame, "Details submitted successfully!");
        }
    });

    detailsFrame.add(nameLabel);
    detailsFrame.add(nameField);
    detailsFrame.add(eventLabel);
    detailsFrame.add(eventsComboBox);
    detailsFrame.add(timeLabel);
    detailsFrame.add(timeRangeComboBox);
    detailsFrame.add(submitButton);

    JButton exitButton = new JButton("Exit");
    exitButton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            detailsFrame.dispose();
        }
    });
    detailsFrame.add(exitButton);

    detailsFrame.setVisible(true);
}

private static double calculateCharges(String event, String timeRange) {
    String[] events = {"Walking", "Feeding", "Playtime", "Grooming", "Hourly Care"};
    double[] charges = {75.0, 50.0, 60.0, 100.0, 150.0};
    double eventCharge = 0.0;

    for (int i = 0; i < events.length; i++) {
        if (event.equals(events[i])) {
            eventCharge = charges[i];
            break;
        }
        }

        if (event.equals("Hourly Care")) {
            String[] timeParts = timeRange.split(" - ");
            String startTime = timeParts[0];
            String endTime = timeParts[1];

            SimpleDateFormat timeFormat = new SimpleDateFormat("HH:mm");
            try {
                Date start = timeFormat.parse(startTime);
                Date end = timeFormat.parse(endTime);
                long durationMinutes = (end.getTime() - start.getTime()) / (60 * 1000);
                eventCharge = durationMinutes * charges[4] / 60.0;
            } catch (ParseException ex) {
                ex.printStackTrace();
            }
        }
        return eventCharge;
    }
}
```

5.RESULT SCREENS:



MAIN MENU


User Login

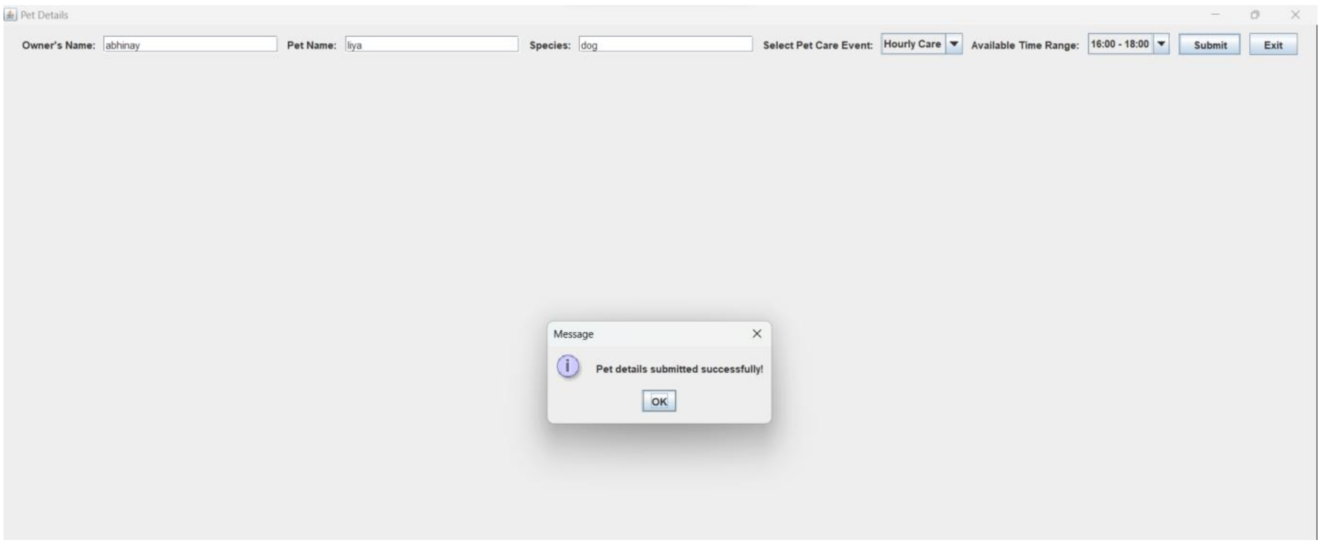
Username: abhinay Password: •••• Login Exit

Message

User login successful!

OK

USER LOGIN


Pet Details

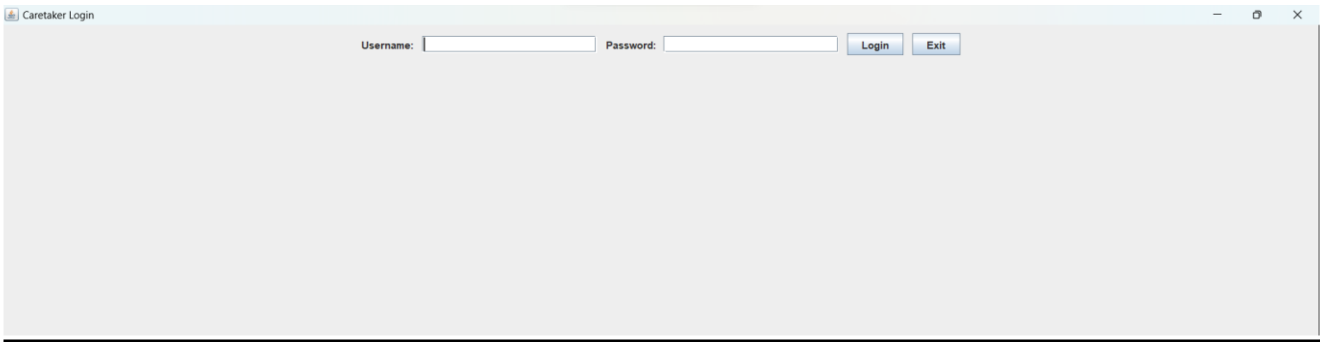
Owner's Name: abhinay Pet Name: liya Species: dog Select Pet Care Event: Hourly Care Available Time Range: 16:00 - 18:00 Submit Exit

Message

Pet details submitted successfully!

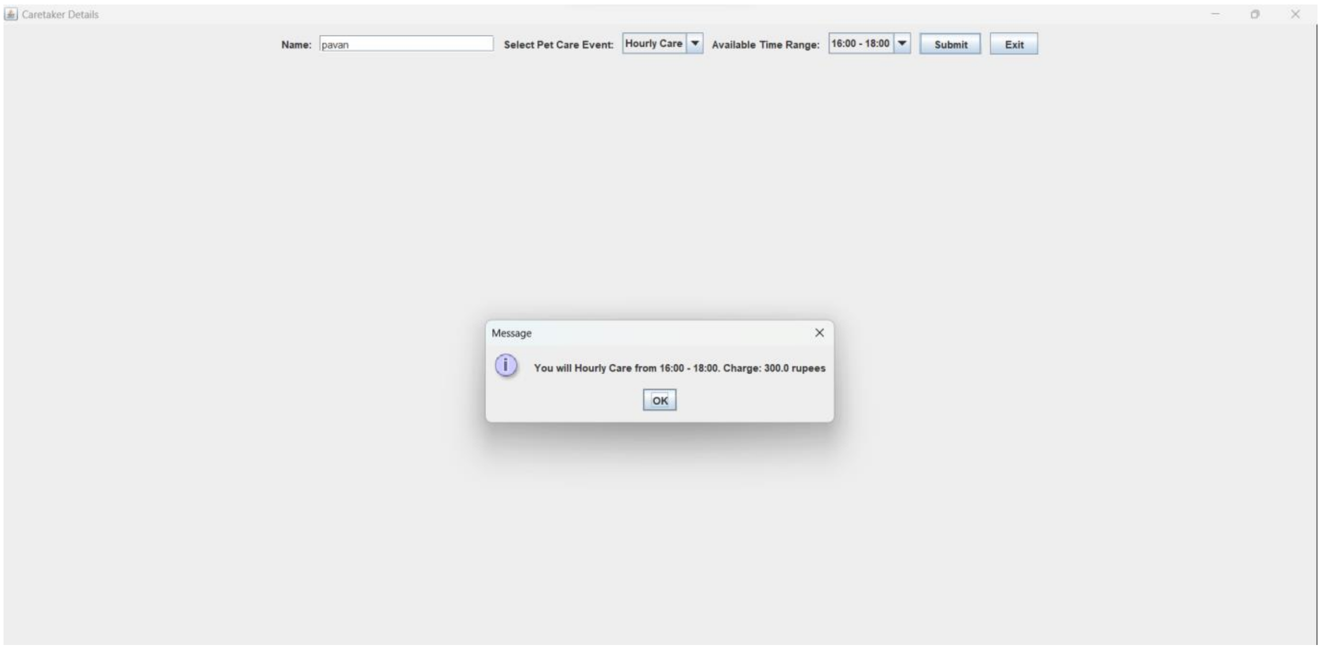
OK

USER AND PET DETAILS


Caretaker Login

Username: Password: Login Exit

CARETAKER LOGIN



Name: pavan

Select Pet Care Event: Hourly Care

Available Time Range: 16:00 - 18:00

You will Hourly Care from 16:00 - 18:00. Charge: 300.0 rupees

CARETAKER DETAILS

6.CONCLUSION

In conclusion, the "Pet Partner" platform serves as a valuable solution for both pet owners and caretakers. It bridges the gap between pet owners in need of flexible and reliable pet care services and caretakers looking for opportunities to provide these services. By offering a user-friendly web-based interface, "Pet Partner" makes it convenient for pet owners to find experienced caretakers for their beloved pets, ensuring their well-being even when the owners are occupied.

For pet owners, this platform provides peace of mind, knowing that their pets are in capable hands during their absence. It offers the convenience of selecting from a range of hourly pet care services, tailored to their specific needs and schedules.

Caretakers, in turn, benefit from the platform by gaining access to potential job opportunities that match their availability and skills. They can build a flexible work schedule while contributing to the welfare of animals and earning income.

In essence, the "Pet Partner" platform brings together pet owners and caretakers, fostering a community that prioritizes the well-being of pets. It streamlines the process of booking pet care services, making it a win-win for both parties involved. The platform plays a crucial role in enhancing the quality of life for pets and their owners.

REFERENCES:

1] Consumers Are Increasingly Researching Purchases Online. PYMNTS.com. (2018, January 9). https://www.pymnts.com/news/retail/2018/omichannel-ecommerce-consumer-habits/.

[2] Kaplan, K. (2020, February 3). Council Post: Why Every Business Needs A Website. Forbes.
https://www.forbes.com/sites/theyec/2020/02/03/why-every-business-needs-awebsite/?sh=218aefad6e75.

[3] Karin Brulliard, S. C. (2019, January 31). How many Americans have pets? An investigation of fuzzy statistics. The Washington Post.
https://www.washingtonpost.com/science/2019/01/31/howmany-americans-have-pets-an-investigation-into-fuzzy-statistics/.

[4] HTML Tutorial. (n.d.). https://www.w3schools.com/html/.

[5] JavaScript. MDN. (n.d.). https://developer.mozilla.org/en-US/docs/Web/JavaScript.

[6] Angular (Web Framework). (n.d.). Wikipedia.
https://en.wikipedia.org/wiki/Angular_(web_framework).

[7] AngularJS. (n.d.). https://docs.angularjs.org/guide/introduction.

[8] Introduction to MySQL. MySQL Introduction. (n.d.).
https://www.w3schools.com/MySQL/mysql_intro.asp.

[9] The Importance of a Website for Your Business Success. Digital Marketing Blog. (2020, June 29).
https://www.lyfemarketing.com/blog/importance-of-a-website/.

[10] HTML Tag. HTML head tag. (n.d.). https://www.w3schools.com/tags/tag_head.asp.

[11] 15 Alignment, font styles, and horizontal rules. Alignment, font styles, and horizontal rules in
HTML documents. (n.d.). https://www.w3.org/TR/REC-html40-971218/present/graphics.html.

[12] Wikimedia Foundation. (2020, July 10). Angular (web framework). Wikipedia.
https://en.wikipedia.org/wiki/Angular_(web_framework)#:~:text=Angular%20does%20not%20ha
ve%20a,as%20its%20primary%20architectural%20characteristic.

[13] John Hannah April 9, Hannah, J., & John Hannah John Hannah is a Senior JavaScript Developer.
(n.d.). Choosing the Right JavaScript Framework for the Job. Lullabot.
https://www.lullabot.com/articles/choosing-the-right-javascript-framework-for-the-job.

[14] Flexbox. (n.d.). https://css-tricks.com/snippets/css/a-guide-to-flexbox/.

15] Programing in Java . Week4.pptx - Programing in Java Week 4 Database Management System What is Database \u2022 The database is a collection of inter-related data which is used | Course Hero. (n.d.). https://www.coursehero.com/file/66023043/Week4pptx/.

[16] Database Relationship. (n.d.). https://github.com/susanBuck/dwa15-

fall2018/blob/master/laravel/db-one-to-many.md.
[17] Diebner, R., Silliman, E., Ungerman, K., & Vancauwenberghe, M. (2020, December 12). Adapting
customer experience in the time of coronavirus. McKinsey & Company.
https://www.mckinsey.com/business-functions/marketing-and-sales/our-insights/adaptingcustomer-experience-in-the-time-of-coronavirus.
[18] says:, S. K., says:, S., says:, E. S., says:, P., says:, R., says:, S., says:, P. S., says:, A. S., says:, K.
S. A. Y. E. D., says:, B. C., Says:, L. B., says:, S. L., says:, P. J., & says:, N. C. (2020, November 25). Angular Tutorial for Beginners: Getting Started with Angular 8. Edureka.
https://www.edureka.co/blog/angular-tutorial/.