SDM COLLEGE OF ENGINEERING AND TECHNOLOGY

Dhavalagiri,Dharwad-580002, Karnataka State, India.

Email: cse.sdmcet@gmail.com

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

A Report

On

Minor work

COURSE CODE : 22UHUC500

COURSE TITLE :Software Engineering and Project Management

SEMISTER : 5th DIVISION : CSE 'B'

COURSE TEACHER : Dr. U.P Kulkarni

[Academic Year- 2024-25]

Date of Submission: 10-10-2024

Submitted

By

Ms. Sakshi S K **USN : 2SD22CS081**

Contents

Contents

Problem Statement 1: "Write a C program to show that C programming language support only call by value?"

```
#include <stdio.h>

int square(int num) {
   return num * num;
}
int main() {
   int x = 5;
   int result = square(x);

   printf("The square of %d is %d\n", x, result);
   return 0;
}
```

In this example:

Square function: Takes an integer num as input and returns its square.
Main function: Declares an integer x with the value 5.
Calls the square function with x as an argument.
The returned value from square is stored in the result variable.
The result is printed.
Key point: Even though the square function modifies the num parameter within its scope, it doesn't affect the original value of x in the main function because C uses call by value. The square function receives a copy of x, and any changes made to that copy are not reflected in the original variable.

Problem Statement 2: "Study the concept 'USABILITY. Prepare a report on Usability of atleast two UI's of major software products you have seen."

Definition:

Usability is a measure of how easy it is for a user to achieve their goals when interacting with a product. A usable product is intuitive, efficient, and satisfying to use. It should be easy to learn, remember, and use without significant effort.

Key Components of Usability:

1. Learnability: How easy it is for a new user to learn the basics of the product and start using it effectively.
2. Efficiency: How quickly and accurately users can perform tasks with the product once they have learned how to use it.
3. Memorability: How easy it is for users to remember how to use the product after a period of not using it.
4. Error Prevention: How well the product helps users avoid errors and recover from them when they do occur.
5. Satisfaction: How pleasant it is for users to interact with the product.

Software Product 1: Canva

Canva, a popular online graphic design platform, has gained widespread recognition for its user-friendly interface. This study will delve into the key aspects of Canva's UI and assess its usability.

Key Features of Canva's UI

Drag-and-Drop Interface: Canva's intuitive drag-and-drop interface makes it easy for users to create designs without extensive technical knowledge.
Template Library: A vast library of pre-designed templates for various graphic design needs, such as social media posts, presentations, and marketing materials.
Element Library: A collection of customizable elements, including shapes, icons, and illustrations, that can be easily added to designs.
Text Editor: A simple and effective text editor for adding and formatting text within designs.
Collaboration Features: Real-time collaboration allows multiple users to work on a design simultaneously.

Strengths:

- **User-Friendly Interface:** Canva's intuitive drag-and-drop interface makes it easy for users of all skill levels to create designs.
- **Extensive Template Library:** The vast collection of templates provides a starting point for users, saving time and effort.
- **Large Asset Library:** Canva's library of stock images, icons, and elements offers a wide range of options for enhancing designs.
- **Collaboration Features:** The ability to collaborate with others makes Canva suitable for team projects.
- **Integration with Other Tools:** Canva's integration with popular tools like Google Drive and Dropbox streamlines the workflow.
- **Affordable Pricing:** Canva offers a free plan with limited features and paid plans for businesses and teams.

Weaknesses:

- **Limited Customization for Advanced Users:** While Canva is great for basic designs, advanced users may find the customization options somewhat limited compared to professional design software.
- **Dependency on Internet Connection:** As an online tool, Canva requires a stable internet connection to function.
- **Potential for Overreliance on Templates:** While templates can be helpful, excessive reliance on them can lead to generic designs.
- **Limited Typography Options:** While Canva offers a variety of fonts, the options may be limited compared to dedicated typography software.
- **File Format Limitations:** There may be limitations in terms of file formats that can be exported or imported.

Software Product 2: Microsoft PowerPoint

Microsoft PowerPoint is a popular presentation software that has been used for decades to create and deliver effective presentations. It offers a wide range of features and templates to help users create visually appealing and informative presentations.

Key Features and Benefits:

- **Slide Design:** PowerPoint provides a variety of templates and themes to help users create

visually appealing slide designs.

- **Content Creation:** Users can add text, images, charts, graphs, and other elements to their slides.
- **Presentation Delivery:** PowerPoint can be used to deliver presentations in person, online, or as recorded presentations.
- **Collaboration:** Users can collaborate on presentations with others, making it suitable for team projects.
- **Integration with Other Microsoft Products:** PowerPoint integrates seamlessly with other Microsoft Office applications like Word and Excel, making it easy to share data and content between programs.

Strengths:

- **Versatility:** PowerPoint can be used to create a wide variety of presentations, from simple slideshows to complex presentations with multimedia elements.
- **Integration with Other Microsoft Products:** PowerPoint integrates seamlessly with other Microsoft Office applications, making it easy to share data and content between programs.
- **Robust Feature Set:** PowerPoint offers a robust set of features, including templates, themes, transitions, animations, and multimedia capabilities.
- **Customization Options:** Users can customize their presentations to meet their specific needs, adjusting layouts, fonts, and colors.

Weaknesses:

- **Steep Learning Curve:** For beginners, learning to use all of PowerPoint's features can be time-consuming.
- **Design Limitations:** While PowerPoint offers a variety of design options, some users may find the design capabilities limited compared to dedicated design software.
- **Multimedia Limitations:** While PowerPoint can handle multimedia elements, there may be limitations in terms of file formats and playback quality.
- **Complexity:** For simple presentations, PowerPoint can be overly complex, with many features that may not be necessary

Problem Statement 3: “List all the features of Programming language and write programs to show they help to write Robust code.”

Java is a popular programming language known for its robustness, platform independence, and object-oriented features. Here are some key features and code examples demonstrating their benefits:

1. Object-Oriented Programming (OOP)

Encapsulation: Bundles data and methods within objects, promoting data security and modularity.

Inheritance: Allows classes to inherit properties and methods from parent classes, promoting code reuse and extensibility.

Polymorphism: Enables objects of different classes to be treated as if they were of the same type, providing flexibility and dynamic behavior.

Example:

```
class Animal {
    void makeSound() {
        System.out.println("Generic animal sound");
    }
}
class Dog extends Animal {
    void makeSound() {
        System.out.println("Woof!");
}
}
class Cat extends Animal {
    void makeSound() {
        System.out.println("Meow!");
}
```

```
}


  public class   Main {
      public static void main(String[] args) {
          Animal[] animals = {new Dog(), new Cat()};
          for (Animal animal : animals) {
              animal.makeSound();
   }
  }
}
```

This code demonstrates polymorphism, where different animal objects can be treated as the same type and their specific sounds are called using the makeSound() method.

2. Automatic Memory Management (Garbage Collection)

Handles memory allocation and deallocation automatically, reducing the risk of memory leaks and errors.

Example:

```
public class Main {
    public static void main(String[] args) {
        String message = "Hello, world!";
        // No need to manually deallocate memory
    }
}
```

The Java garbage collector will automatically reclaim the memory used by the message object when it is no longer needed.

3. Exception Handling

Provides mechanisms to handle errors and unexpected situations gracefully, preventing program crashes.

Example:

```
public class Main {
    public static void main(String[] args) {
        try {
            int result = 10 / 0;
            System.out.println(result);
        } catch (ArithmeticException e) {
            System.out.println("Error:
 Division by zero");
        }
    }
}
```

This code catches the ArithmeticException thrown when dividing by zero and provides a meaningful error message.

4. Strong Typing

Enforces strict rules about data types, preventing unintended type conversions and errors.

Example:

```
public class Main {
    public static void main(String[] args) {
        int num = 5;
        double decimal = num / 2; // Implicit type conversion
        System.out.println(decimal);
    }
}
```

In this example, the integer num is implicitly converted to a double before division, ensuring correct type compatibility.

Problem Statement 4: Study the “ASSERTIONS” in C language and its importance in writing RELIABLE CODE. Study POSIX standard and write a C program under Unix to show use of POSIX standard in writing portable code.

Assertions in C and Their Importance

Assertions in C are a powerful tool for debugging and ensuring the correctness of code. They are essentially conditions that are expected to be true at a particular point in the program. If an assertion fails, the program terminates with an error message.

Importance of Assertions:

Early Detection of Errors: Assertions can help identify and fix bugs early in the development process, preventing them from propagating to later stages.

Documentation: Assertions can serve as documentation, clarifying the expected behavior of the code and making it easier to understand.

Testing: Assertions can be used as part of unit testing to verify that code is functioning as expected.

Defensive Programming: Assertions can help make code more defensive by checking for invalid inputs and unexpected conditions.

Using assert.h: To use assertions in C, you need to include the <assert.h> header. The assert macro takes a boolean expression as an argument. If the expression evaluates to false, the program terminates with an error message.

Example:

```
#include <stdio.h>

#include <assert.h>

int divide(int numerator, int denominator) {

    assert(denominator != 0);

    return numerator / denominator;

}

int main() {

    int result = divide(10,  0);
```

```
    printf("Result: %d\n", result);
    return 0;
}
```

In this example, the assert macro checks if the denominator is not zero. If it is, the program will terminate with an assertion failure.

POSIX Standard and Portable Code

POSIX (Portable Operating System Interface) is a family of standards for operating systems that define a common API for various functions, including file I/O, process management, and networking. By adhering to POSIX standards, you can write C code that is more portable and can be compiled and run on different Unix-like systems without significant modifications.

Business Scenario: Cross-Platform File I/O

Imagine a business that needs to develop a utility to write data to the file on different operating systems. Using POSIX functions, we can write a portable C program to achieve this:

```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>

int main() {
    // POSIX file creation using open()
    int fd = open("example.txt", O_WRONLY | O_CREAT, 0644);
    if (fd == -1) {
        perror("Failed to open file");
        return 1;
    }
  // Data to write to the file
    const char *data = "Hello, POSIX!\n";
```

```
 // POSIX write system call
    if (write(fd, data, strlen(data)) == -1) {
        perror("Failed to write to file");
        close(fd);
        return 1;
    }
// POSIX file close
    if (close(fd) == -1) {
        perror("Failed to close file");
        return 1;
    }
printf("Data written to file successfully!\n");
    return 0;
}
```