Report on Innovative Teaching Learning method adopted: Open Book Assignment on design modelling of problem statement & MiniProject on RTOS problem statement by Prof. Priya Deshpande

Sem VI, EXTC

Subject: Real-Time Embedded Systems (EC332),

Academic Year: 2023-24

In the SEM VI Real-Time Embedded Systems (RTES) Lab, an innovative teaching and learning approach was adopted through the implementation of Open Book Assignments (OBA). This method was introduced to enhance students' practical understanding and problem-solving skills in the design and modeling of real-time systems.

Open Book Assignment Structure:

- **Context and Objectives:** The OBA focused on the design and modeling of real-time embedded systems, aiming to bridge the gap between theoretical knowledge and practical application. Students were provided with complex problem statements relevant to real-world scenarios.
- **Resource Utilization:** Students were allowed to use textbooks, class notes, and online resources to research and develop their solutions.
- **Implementation Process:**
    - **Preparation:** Problem statements were carefully crafted to cover key topics such as task scheduling, real-time operating systems, and inter-task communication.
    - **Assignment Distribution:** Assignments were distributed with clear guidelines, including objectives, expected outcomes, and assessment criteria.
    - **Student Work:** Students individually worked on the assignments, utilizing available resources to design, model, and document their solutions.
    - **Submission and Evaluation:** Completed assignments were submitted electronically, followed by detailed evaluations based on predefined rubrics.

Problem Statement Example: Different Problem for each students were given

6) Draw a Activity diagram for Online shopping system

Following criteria has to be achieved while designing it

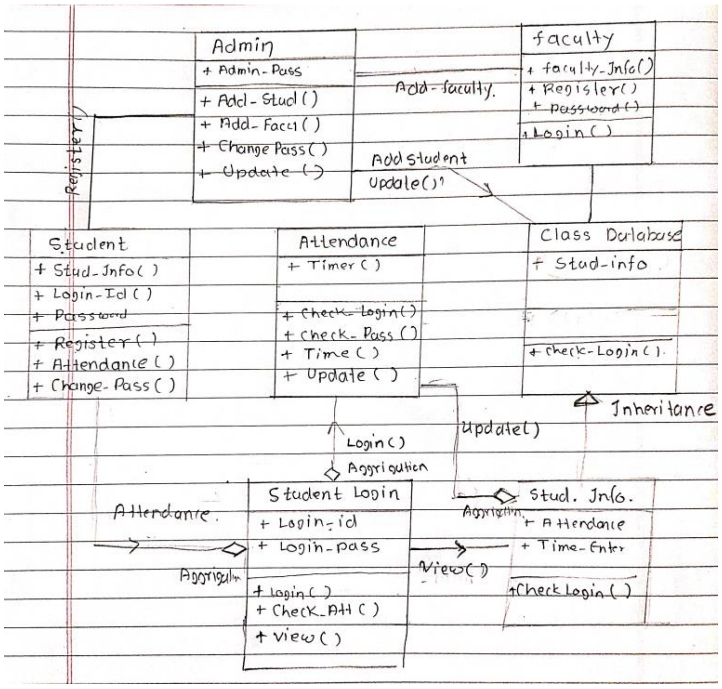
1. Registration of new user
2. Login of existing user
3. Managing cart: editing and deleting items from cart
4. Viewing items
5. Recommendation of items
6. Giving discounts
7. Check out of cart

7) Draw a Class diagram for Online shopping system

---

11) Draw UML Class diagram of student attendance system
Following criteria has to be achieved while designing it
1. Registration of student
2. Change password
3. Registration of time slot
4. Modification of student details
5. Take attendance if student
6. View attendance class wise
7. View attendance student wise
8. Add user or faculty

Sample Solution:



Introduction

In the context of the MiniProject for Real-Time Operating Systems (RTOS), an innovative teaching and learning approach was implemented. This method aimed to enhance students' practical skills and understanding of RTOS by engaging them in a hands-on mini-project focused on solving a real-world problem statement.

Outcomes

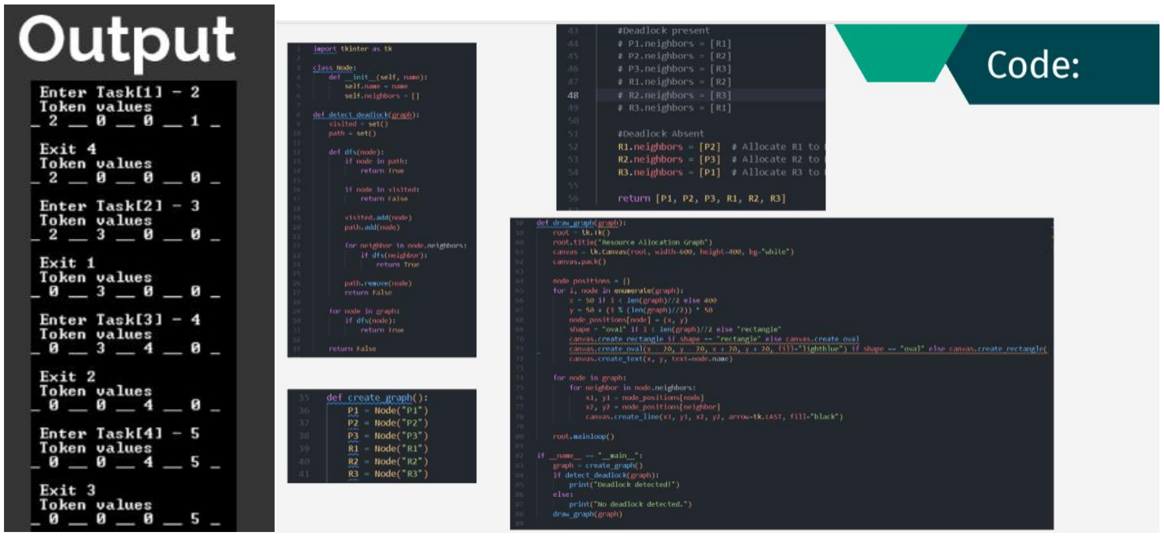
1. Enhanced Understanding of RTES Concepts:

- **Application of Theory:** Students applied theoretical knowledge to practical scenarios, deepening their understanding of RTOS principles.
- **Critical Thinking:** The OBA format encouraged critical thinking as students analyzed and solved complex problems.

2. Development of Research Skills:

- **Resource Utilization:** Students effectively used multiple resources to gather information and support their designs.
- **Information Synthesis:** The assignment required synthesizing information from various sources to develop comprehensive models.

Algorithms like Bakers Algorithm, Bankers Algorithm, Deadlock detection were developed on STM32 platform and python platform



```
Enter Task[1] - 2
Token values
_ 2 __ 0 __ 0 __ 1 _
Exit 4
Token values
_ 2 __ 0 __ 0 __ 0 _
Enter Task[2] - 3
Token values
_ 2 __ 3 __ 0 __ 0 _
Exit 1
Token values
_ 0 __ 3 __ 0 __ 0 _
Enter Task[3] - 4
Token values
_ 0 __ 3 __ 4 __ 0 _
Exit 2
Token values
_ 0 __ 0 __ 4 __ 0 _
Enter Task[4] - 5
Token values
_ 0 __ 0 __ 4 __ 5 _
Exit 3
Token values
_ 0 __ 0 __ 0 __ 5 _
```

```python
import tkinter as tk

class Node:
    def __init__(self, name):
        self.name = name
        self.neighbors = []

def detect_deadlock(graph):
    visited = set()
    path = set()

    def dfs(node):
        if node in path:
            return True

        if node in visited:
            return False

        visited.add(node)
        path.add(node)

        for neighbor in node.neighbors:
            if dfs(neighbor):
                return True

        path.remove(node)
        return False

    for node in graph:
        if dfs(node):
            return True

    return False
```

```python
def create_graph():
    P1 = Node("P1")
    P2 = Node("P2")
    P3 = Node("P3")
    R1 = Node("R1")
    R2 = Node("R2")
    R3 = Node("R3")
```

```python
    #Deadlock present
    # P1.neighbors = [R1]
    # P2.neighbors = [R2]
    # P3.neighbors = [R3]
    # R1.neighbors = [R2]
    # R2.neighbors = [R3]
    # R3.neighbors = [R1]

    #Deadlock Absent
    R1.neighbors = [P2]  # Allocate R1 to
    R2.neighbors = [P3]  # Allocate R2 to
    R3.neighbors = [P1]  # Allocate R3 to

    return [P1, P2, P3, R1, R2, R3]
```

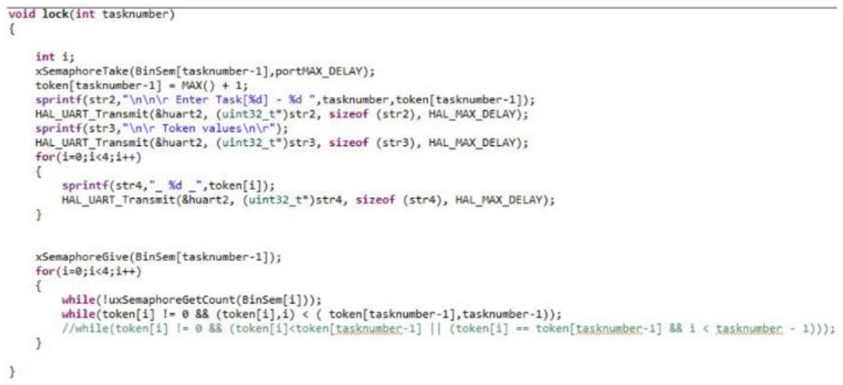
```python
def draw_graph(graph):
    root = tk.Tk()
    root.title("Resource Allocation Graph")
    canvas = tk.Canvas(root, width=600, height=400, bg="white")
    canvas.pack()

    node_positions = {}
    for i, node in enumerate(graph):
        x = 50 if i < len(graph)//2 else 400
        y = 50 + (i % (len(graph)//2)) * 50
        node_positions[node] = (x, y)
        shape = "oval" if i < len(graph)//2 else "rectangle"
        canvas.create_rectangle if shape == "rectangle" else canvas.create_oval
        canvas.create_oval(x - 20, y - 20, x + 20, y + 20, fill="lightblue") if shape == "oval" else canvas.create_rectangle(
        canvas.create_text(x, y, text=node.name)

    for node in graph:
        for neighbor in node.neighbors:
            x1, y1 = node_positions[node]
            x2, y2 = node_positions[neighbor]
            canvas.create_line(x1, y1, x2, y2, arrow=tk.LAST, fill="black")

    root.mainloop()

if __name__ == "__main__":
    graph = create_graph()
    if detect_deadlock(graph):
        print("Deadlock detected!")
    else:
        print("No deadlock detected.")
    draw_graph(graph)
```



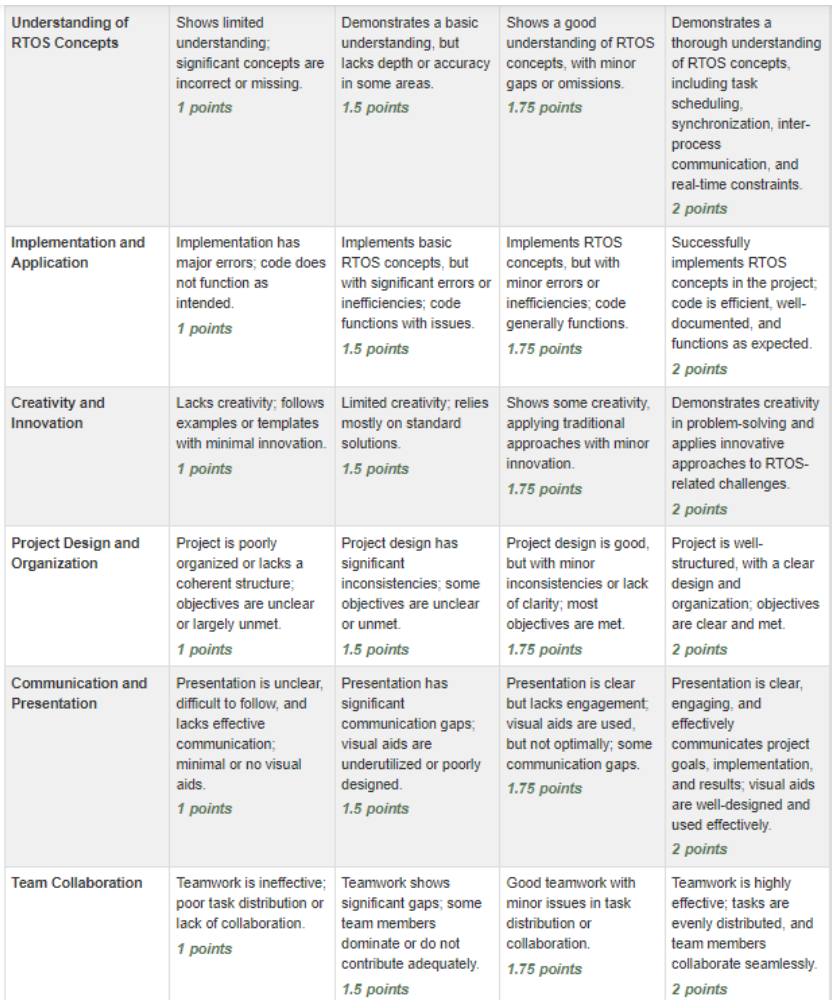
```c
void lock(int tasknumber)
{

    int i;
    xSemaphoreTake(BinSem[tasknumber-1],portMAX_DELAY);
    token[tasknumber-1] = MAX() + 1;
    sprintf(str2,"\n\n\r Enter Task[%d] - %d ",tasknumber,token[tasknumber-1]);
    HAL_UART_Transmit(&huart2, (uint32_t*)str2, sizeof (str2), HAL_MAX_DELAY);
    sprintf(str3,"\n\r Token values\n\r");
    HAL_UART_Transmit(&huart2, (uint32_t*)str3, sizeof (str3), HAL_MAX_DELAY);
    for(i=0;i<4;i++)
    {
        sprintf(str4,"_ %d _",token[i]);
        HAL_UART_Transmit(&huart2, (uint32_t*)str4, sizeof (str4), HAL_MAX_DELAY);
    }


    xSemaphoreGive(BinSem[tasknumber-1]);
    for(i=0;i<4;i++)
    {
        while(!uxSemaphoreGetCount(BinSem[i]));
        while(token[i] != 0 && (token[i],i) < ( token[tasknumber-1],tasknumber-1));
        //while(token[i] != 0 && (token[i]<token[tasknumber-1] || (token[i] == token[tasknumber-1] && i < tasknumber - 1)));
    }

}
```

Rubrics for Evaluation:



Understanding of RTOS Concepts	Shows limited understanding; significant concepts are incorrect or missing. *1 points*	Demonstrates a basic understanding, but lacks depth or accuracy in some areas. *1.5 points*	Shows a good understanding of RTOS concepts, with minor gaps or omissions. *1.75 points*	Demonstrates a thorough understanding of RTOS concepts, including task scheduling, synchronization, inter-process communication, and real-time constraints. *2 points*
Implementation and Application	Implementation has major errors; code does not function as intended. *1 points*	Implements basic RTOS concepts, but with significant errors or inefficiencies; code functions with issues. *1.5 points*	Implements RTOS concepts, but with minor errors or inefficiencies; code generally functions. *1.75 points*	Successfully implements RTOS concepts in the project; code is efficient, well-documented, and functions as expected. *2 points*
Creativity and Innovation	Lacks creativity; follows examples or templates with minimal innovation. *1 points*	Limited creativity; relies mostly on standard solutions. *1.5 points*	Shows some creativity, applying traditional approaches with minor innovation. *1.75 points*	Demonstrates creativity in problem-solving and applies innovative approaches to RTOS-related challenges. *2 points*
Project Design and Organization	Project is poorly organized or lacks a coherent structure; objectives are unclear or largely unmet. *1 points*	Project design has significant inconsistencies; some objectives are unclear or unmet. *1.5 points*	Project design is good, but with minor inconsistencies or lack of clarity; most objectives are met. *1.75 points*	Project is well-structured, with a clear design and organization; objectives are clear and met. *2 points*
Communication and Presentation	Presentation is unclear, difficult to follow, and lacks effective communication; minimal or no visual aids. *1 points*	Presentation has significant communication gaps; visual aids are underutilized or poorly designed. *1.5 points*	Presentation is clear but lacks engagement; visual aids are used, but not optimally; some communication gaps. *1.75 points*	Presentation is clear, engaging, and effectively communicates project goals, implementation, and results; visual aids are well-designed and used effectively. *2 points*
Team Collaboration	Teamwork is ineffective; poor task distribution or lack of collaboration. *1 points*	Teamwork shows significant gaps; some team members dominate or do not contribute adequately. *1.5 points*	Good teamwork with minor issues in task distribution or collaboration. *1.75 points*	Teamwork is highly effective; tasks are evenly distributed, and team members collaborate seamlessly. *2 points*