



Republic of the Philippines
Technological University of the Philippines



COLLEGE OF SCIENCE

Manila

COMPUTER STUDIES DEPARTMENT
BACHELOR OF SCIENCE IN INFORMATION TECHNOLOGY

Project 2: Page Replacement Algorithms

Victoria, Jerico M.

Matullano, Raymond T.

Panaligan, Francis Edian M.

Santos, Ramino Jake H.

TABLE OF CONTENTS

Background.....	3
Objectives.....	4
Scope and Delimitations.....	4
Results and Discussion.....	5
Technical Difficulties.....	13
Group Evaluation.....	13
Conclusion and Recommendations.....	15
Appendices.....	16

Background:

Page Replacement Algorithms

A page replacement algorithm is needed to decide which page needs to be replaced when a new page comes in. Page fault occurs when the current programs attempt to access the memory page for mapping into virtual address space, but it is unable to load into the physical memory. Least Recently Used (LRU) and Optimal (with FIFO) are the algorithms that we will discuss and present appropriately. **Least Recently Used (LRU)** in this algorithm, a page will be replaced which is least recently used. LRU checks backward reference in the page reference string. **Optimal Page Replacement Algorithm** (also known as OPT or Beladys optimal page replacement policy) in this algorithm, when a page needs to be swapped, that page is replaced which would not be used for the longest duration of time in the future.

Objectives:

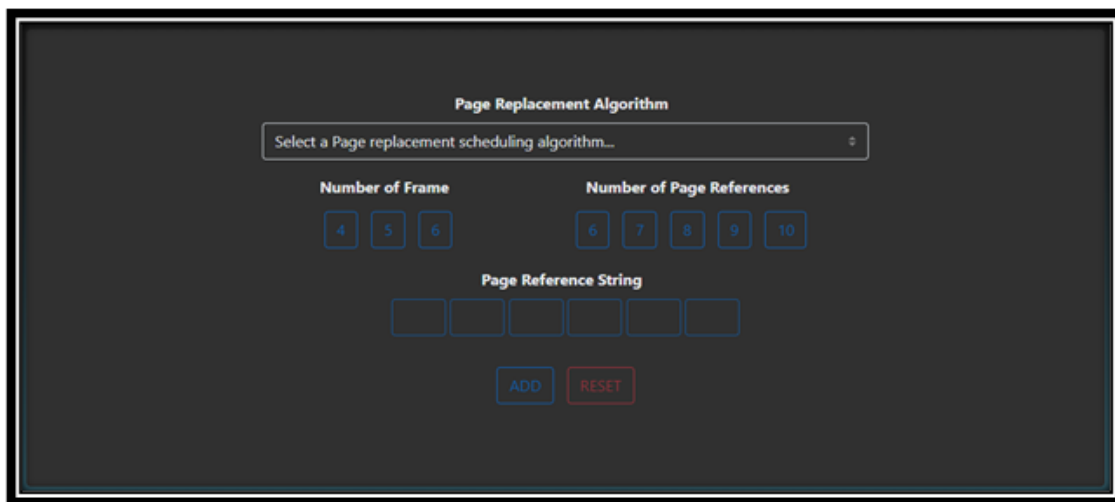
Our team's goal is to create a program that can calculate the exact number of page faults or the number of memory errors and decrease the maximum number of page faults. Our objective is to divide each process in the form of pages of fixed size. The fewer the page faults the better is the algorithm for that situation. Generates a table for visualization of LRU and Optimal. These programs of LRU and Optimal were implemented with the use of JavaScript programming language and it is very similar to C. We used CSS, HTML and Bootstrap for the design and architecture of this program.

Scope and Delimitations:

Our algorithms are LRU and Optimal. It can hold a minimum of 4 with a maximum of 6 frames. At least 6 characters in string or integer and at most 10. It will generate the table or chart that displays the data and information inputted by the user and compute the total number of misses, hits, and ratios.

Results and Discussions:

After testing our systems for several times, we can say that LRU and Optimal do their jobs and process appropriately. We do not see any errors and bugs after we finalize the program and according to the sample we provide, all of the data we've inputted in both LRU and Optimal has an exact result of page faults, hits and its ratio. Based on what type of input the user wants, whether it could be an integer or character, the user can choose the number of frames either 4, 5, or 6 and also the number of page references between 6 and 10. After the user input is done, the table will generate below where it will show the outcome and how these two algorithms work accordingly. Page faults, page hits and its ratio will appear at the bottom with its percentage.



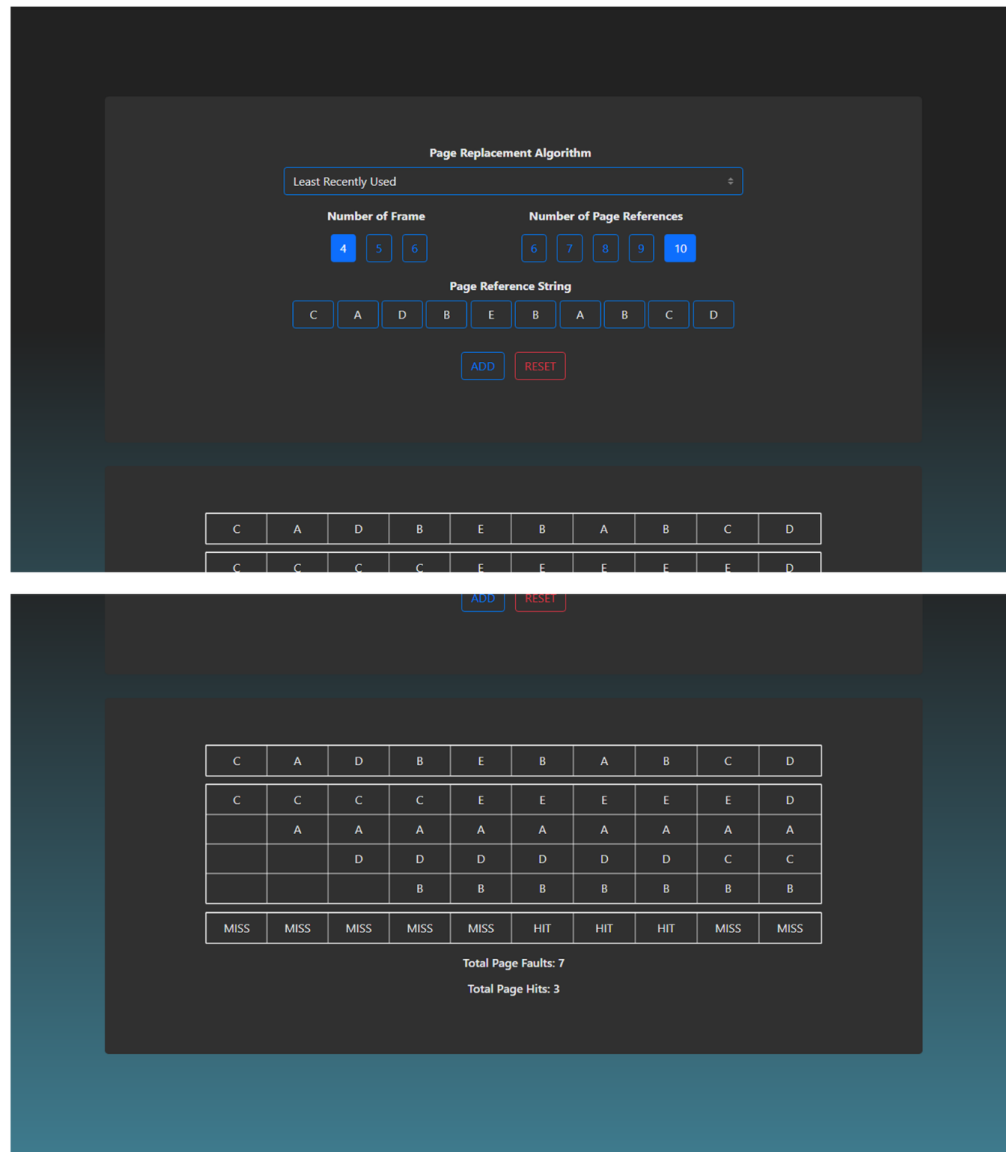
The screenshot shows a web-based interface for a Page Replacement Algorithm. At the top, there is a title "Page Replacement Algorithm" and a dropdown menu labeled "Select a Page replacement scheduling algorithm...". Below this, there are two sections: "Number of Frame" with buttons for 4, 5, and 6; and "Number of Page References" with buttons for 6, 7, 8, 9, and 10. Underneath these is a "Page Reference String" section with six empty input boxes. At the bottom, there are two buttons: "ADD" and "RESET".

Here's the program what it looks like at the beginning. The user needs to select a page replacement scheduling algorithm to proceed and start the program.

We have four data samples for each algorithm. First is for LRU and next is for Optimal scheduling algorithm.

Algorithm: Least Recently Used

Reference String: C A D B E B A B C D

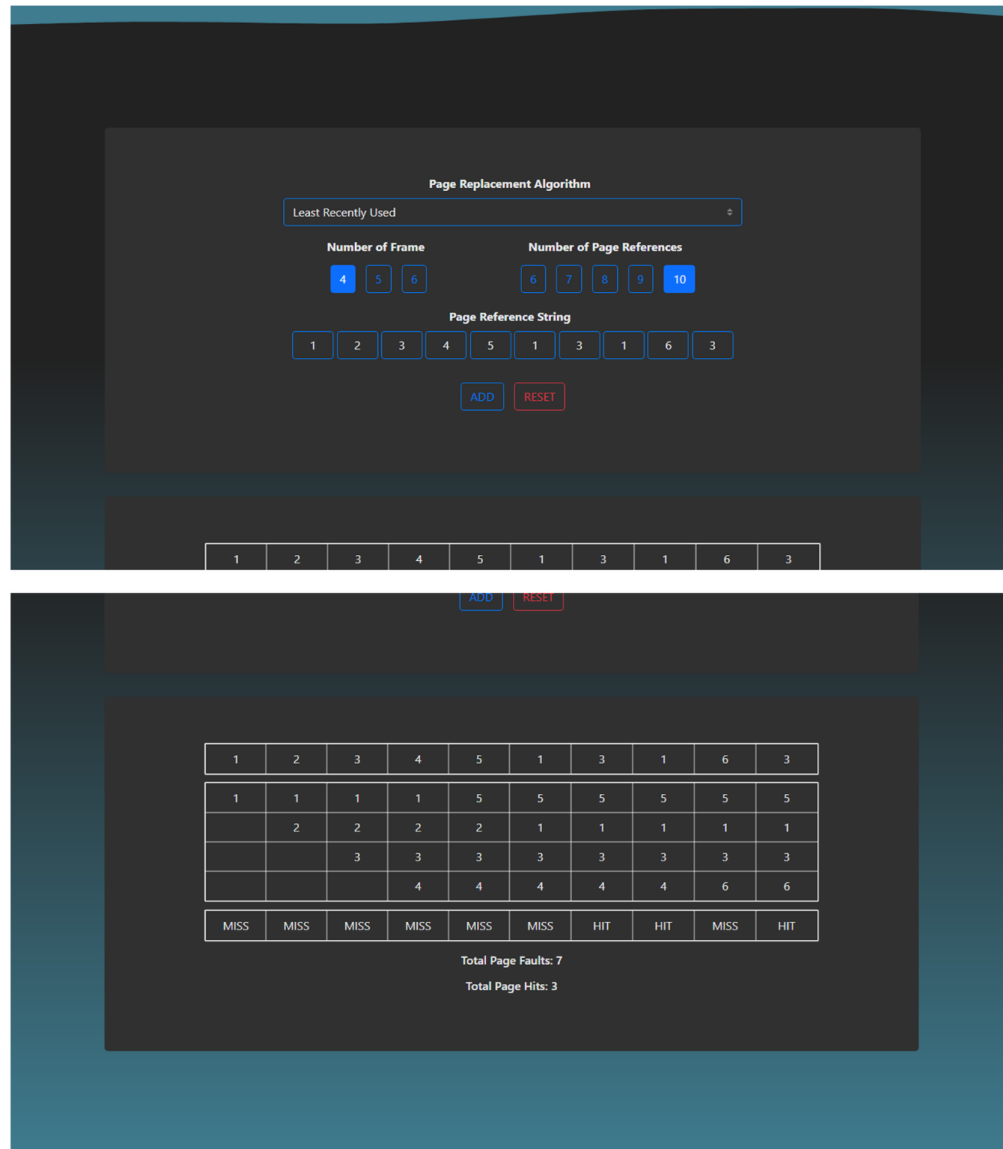


Total Page Faults; 7

Figure 1.1: LRU

Algorithm: Least Recently Used

Reference String: 1 2 3 4 5 1 3 1 6 3

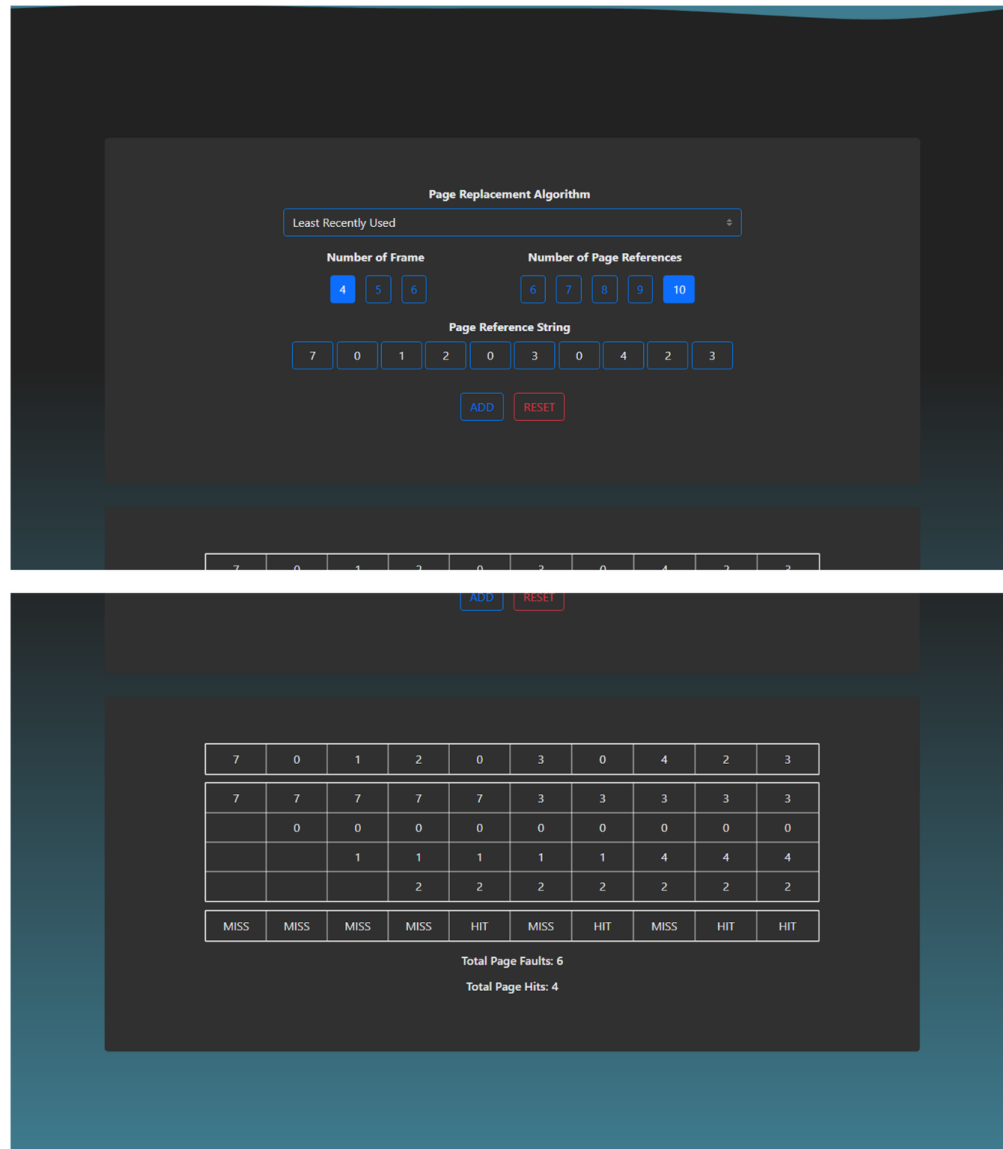


Total Page Faults; 7

Figure 1.2: LRU

Algorithm: Least Recently Used

Reference String: 7 0 1 2 0 3 0 4 2 3

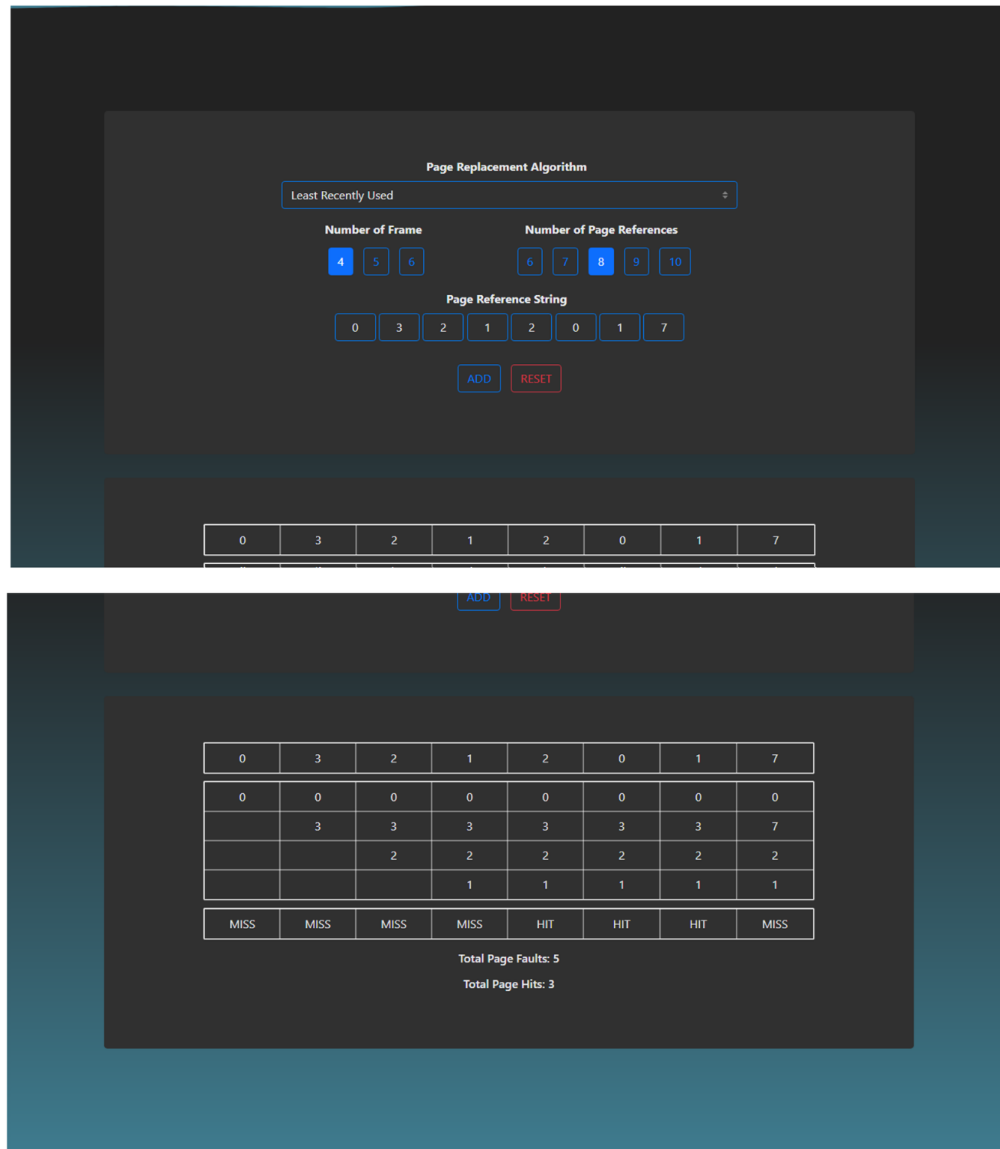


Total Page Faults: 6

Figure 1.3: LRU

Algorithm: Least Recently Used

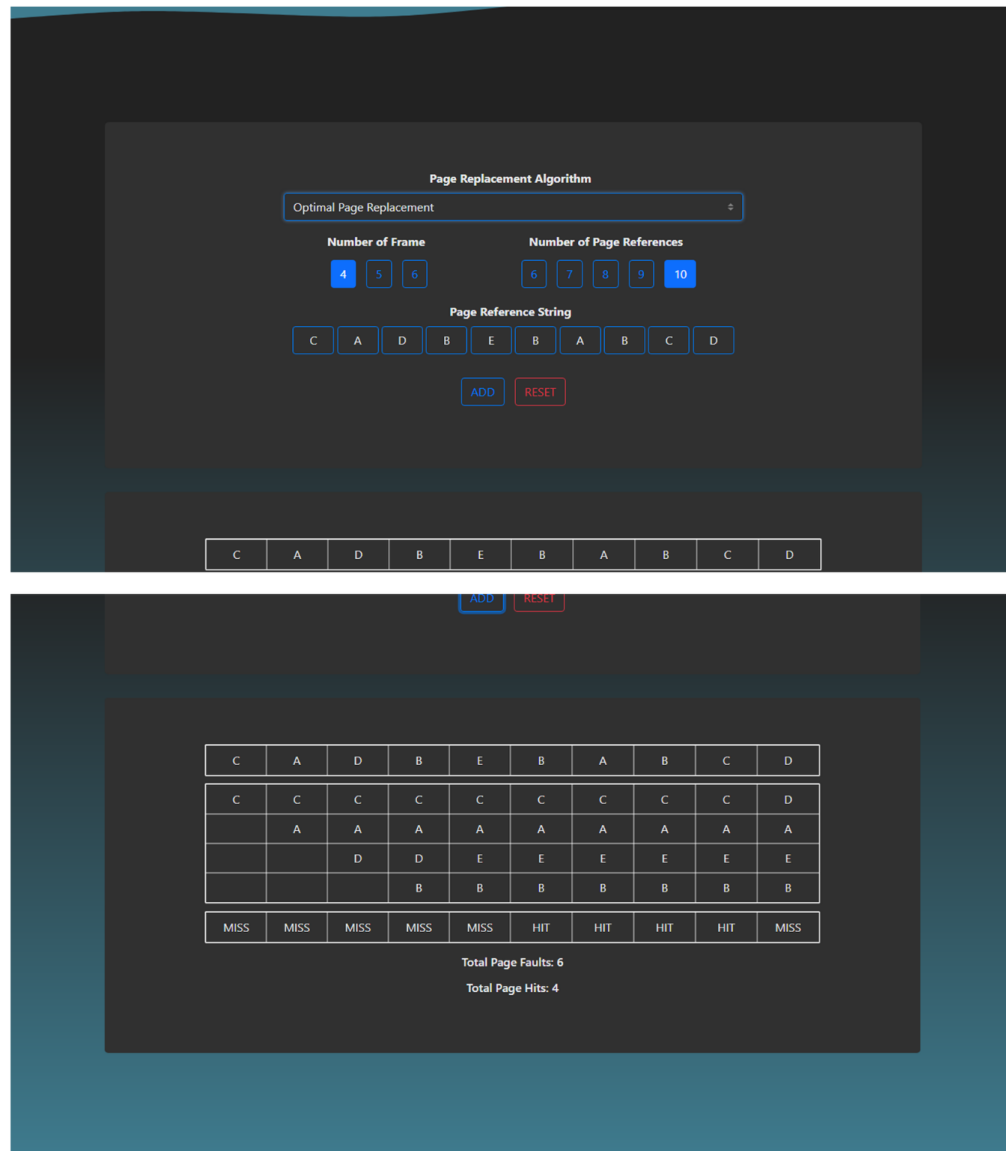
Reference String: 0 3 2 1 2 0 1 7



Total Page Faults: 5

Figure 1.4: LRU

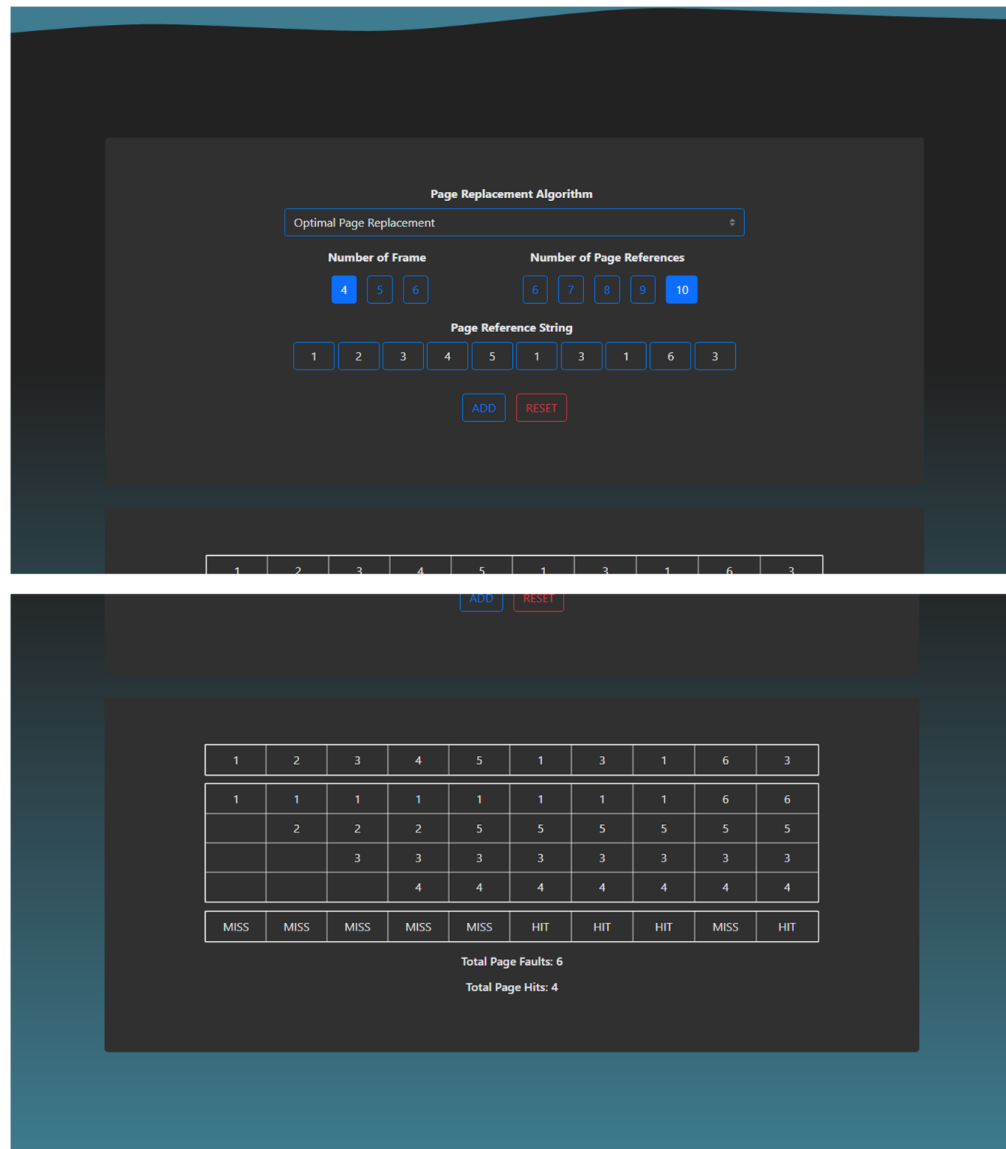
Algorithm: Optimal Page Replacement
Reference String: C A D B E B A B C D



Total Page Faults; 6

Figure 2.1: Optimal

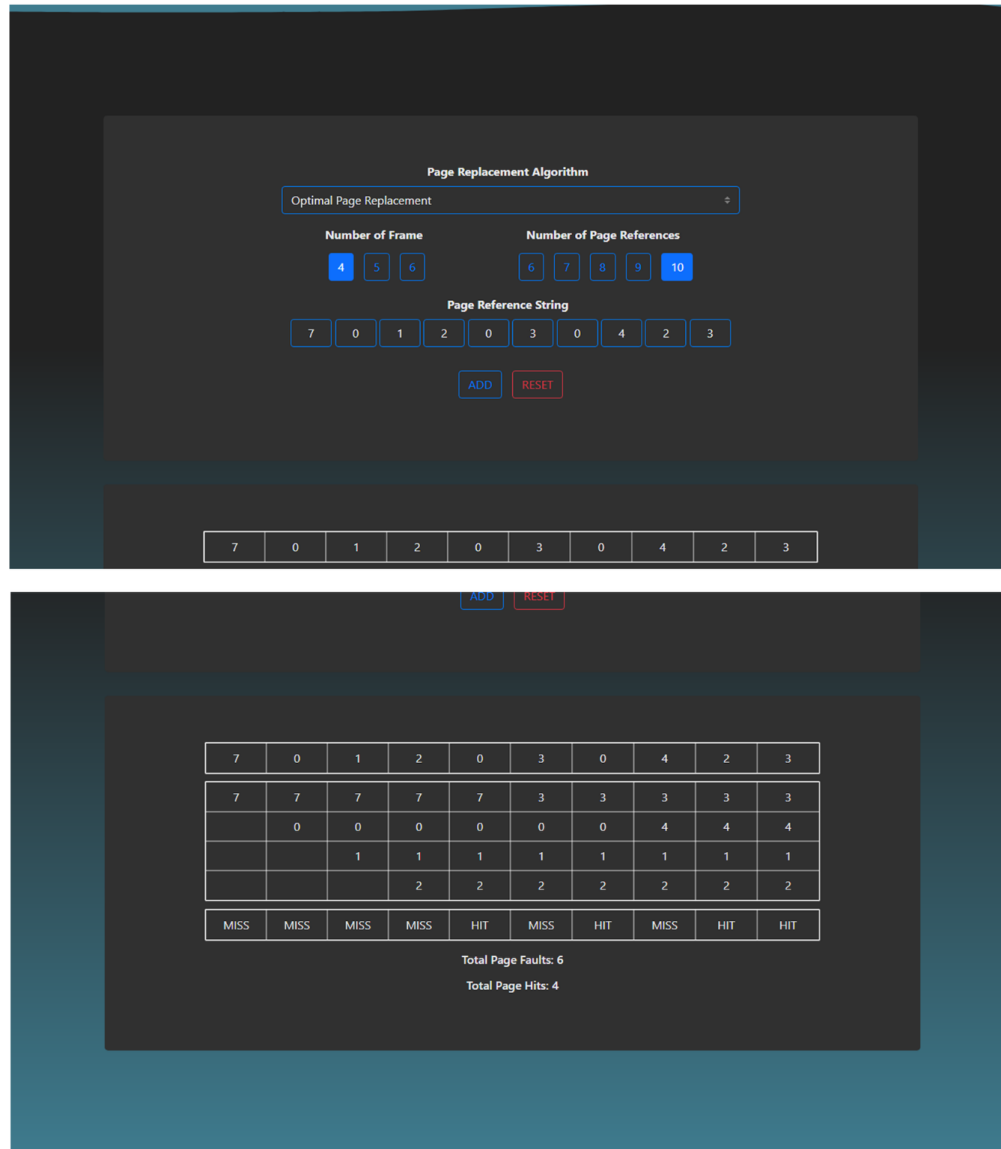
Algorithm: Optimal Page Replacement
Reference String: 1 2 3 4 5 1 3 1 6 3



Total Page Faults; 6

Figure 2.2: Optimal

Algorithm: Optimal Page Replacement
Reference String: 7 0 1 2 0 3 0 4 2 3



Total Page Faults: 6

Figure 2.3: Optimal

Technical Difficulties:

Here is a list of the technical difficulties that we have encountered:

1. The issue we ran into was figuring out how to generate the table; we solved it by using two loops that create rows and columns. A loop will also iterate through each cell in each column and row in order to print the values.
2. The next issue we encountered was determining how to create the two algorithms; we solved this by replicating the online codes and optimizing them to be JavaScript compatible.

Group Evaluation:

- ❑ **Functionality(25pts):** We are relieved because our system passed all of the tests we ran to see if there was a problem with the program. The program runs smoothly and produces an accurate table as well as a total for hit, miss, and ratio. As a result, we will award 25 points for functionality.
- ❑ **Reliability(25pts):** Our program is simple to use; users can easily understand the UI because it is straightforward and easy to operate. To avoid errors, we added some input validations. If there is an error at the inputs, a toast message will appear to notify the user and provide them

with necessary instructions; otherwise, a toast message will appear stating that it is successful. We will give 25 points for our Reliability.

❑ **Usability(25pts):** When it comes to the presentation of the system, we can say that it is easy to use. Users can immediately recognize the purpose of the input fields or option buttons. The user will also learn because he or she will be able to examine the table generated by our program. As a result, we will give 25 points for our Usability.

❑ **Efficiency(23pts):** In terms of efficiency, our program code was vastly long and if carefully examined, it takes up a lot of JavaScript memory. Despite the fact that the execution time is short, we believe that our algorithm can be improved with the use of jQuery and other ES6 scripts, allowing us to save more resources. We will give 23 points for Efficiency.

FUNCTIONALITY	25 pts.
RELIABILITY	25 pts.
USABILITY	25 pts.
EFFICIENCY	23 pts.
TOTAL	98 pts

Conclusion and Recommendation:

Our team can assure that the implementation of our program is well-organized and after we try several times with no encounter of any errors and bugs, we can say that this program we manage does its job. Once the user starts to operate our program the first thing the user needs to do is to select a page replacement scheduling algorithm then choose the number of frames and page reference. It's easy to operate, when a user input is done the user may click the add button and it generates the table that shows the page replacement of strings, the number of page faults, hits and also its ratio. It's simple to use because it is web-based and very informative. We are surely 100 percent assured that our program is understandable, reliable, and functional.

As of now, we need to stick to our program but we know someday we can improve and upgrade the program code we established including the algorithms and designs of our system. As we grow, the knowledge we have today still improves and becomes better in the future for our program.

Appendices:

Job Distribution:

- ❖ Victoria, Jerico M.
 - Lead Programmer
- ❖ Panaligan, Francis Edian M.
 - Programmer
 - Front End Designer
- ❖ Santos, Ramino Jake H.
 - Programmer
 - Quality Assurance
- ❖ Matullano, Raymond T.
 - Programmer
 - Documentation

Source Code:

page.js

```
var page_count;
var resetButton, addButton;
var pagesInputField = [];
var selectedOpts;
var number_of_frames;
var validation = true;
var algoName;

var toastElList = [].slice.call(document.querySelectorAll('.toast'))
var toastList = toastElList.map(function(toastEl) {
  return new bootstrap.Toast(toastEl, toastOption)
})

var toastOption = {
  animation: true,
  delay: 2000
}

function see() {
  for (let i = 0; i < toastList.length; i++) {
    toastList[i].hide();
  }
  if (validation == false)
    toastList[0].show();
  else
    toastList[1].show();
}

function checkIfArrayIsUnique(myArray) {
  return myArray.length === new Set(myArray).size;
}

function addData() {
  removeData(); //Reset everything
  let pagesInputField = document.getElementsByName('page-input');
  let pageValues = [];
  let pageValuesContainer = [];
  validation = true;

  for (let i = 0; i < page_count; i++) {
    if (pagesInputField[i].value == "") {
      validation = false;
      see();
      break;
    }
    pageValues.push(pagesInputField[i].value);
  }
  if (number_of_frames == undefined) {
    validation = false;
    see();
  }
}
```

```

}

if (validation == true) {
    createTable();

    var table = document.getElementById("main-table");
    for (let i = 0; i < number_of_frames; i++) {
        var row = table.insertRow(0);
        for (let j = 0; j < page_count; j++) {
            var cell1 = row.insertCell(0);
        }
    }
    var tableString = document.getElementById("table-string");
    var row = tableString.insertRow(0);
    for (let j = 0; j < page_count; j++) {
        var cell1 = row.insertCell(0);
    }
    var tableFault = document.getElementById("table-fault");
    var row = tableFault.insertRow(0);
    for (let j = 0; j < page_count; j++) {
        var cell1 = row.insertCell(0);
    }
    let temp = [];
    let memory = [];
    let time = [];
    let faults = new Array(page_count);
    let page_found = 0;
    let flag1;
    let flag2;
    let flag3;
    let pos = 0;
    let max = 0;
    let min = 0;
    let fault_count = 0;
    let hit_count = 0;
    let counter = 0;

    if (selectedOpts == "lru") {
        //main algorithm
        algoName = "Least Recently Used";
        for (let i = 0; i < page_count; ++i) {
            flag1 = flag2 = false;

            for (let j = 0; j < number_of_frames; ++j) {
                if (memory[j] == pageValues[i]) {
                    faults[i] = "HIT";
                    hit_count++;
                    counter++;
                    time[j] = counter;
                    flag1 = flag2 = true;
                    break;
                }
            }

            if (!flag1) {
                for (let j = 0; j < number_of_frames; ++j) {

```

```

        if (memory[j] == undefined) {
            counter++;
            faults[i] = "MISS";
            fault_count++;
            memory[j] = pageValues[i];
            time[j] = counter;
            flag2 = true;
            break;
        }
    }
}

if (!flag2) {
    min = time[0];
    pos = 0;
    for (let k = 1; k < number_of_frames; ++k) {
        if (time[k] < min) {
            min = time[k];
            pos = k;
        }
    }
    counter++;
    faults[i] = "MISS";
    fault_count++;
    memory[pos] = pageValues[i];
    time[pos] = counter;
}
// Print the current Memory snap
for (let j = 0; j < memory.length; j++) {
    document.getElementById("main-table").rows[j].cells[i].innerHTML =
memory[j];
}
}
} else if (selectedOpts == "optimal") {
    algoName = "Optimal Page Replacement";
    for (i = 0; i < pageValues.length; i++) {
        flag1 = flag2 = false;
        let page_found = 0;
        for (j = 0; j < number_of_frames; j++) {
            if (memory[j] == pageValues[i]) {
                flag1 = flag2 = true;
                faults[i] = "HIT";
                hit_count++;
                break;
            }
        }
    }
    if (!flag1) {
        for (j = 0; j < number_of_frames; j++) {
            if (memory[j] == undefined) {
                faults[i] = "MISS";
                fault_count++;
                memory[j] = pageValues[i];
                flag2 = 1;
                break;
            }
        }
    }
}
}

```

```

    }
    if (!flag2) {
        flag3 = false;
        for (j = 0; j < number_of_frames; j++) {
            temp[j] = undefined;

            for (k = i + 1; k < pageValues.length; k++) {
                if (memory[j] == pageValues[k]) {
                    temp[j] = k;
                    break;
                }
            }
        }
        for (j = 0; j < number_of_frames; j++) {
            if (temp[j] == undefined) {
                pos = j;
                flag3 = true;
                break;
            }
        }

        if (!flag3) {
            max = temp[0];
            pos = 0;

            for (j = 1; j < number_of_frames; j++) {
                if (temp[j] > max) {
                    max = temp[j];
                    pos = j;
                }
            }
        }
        memory[pos] = pageValues[i];
        faults[i] = "MISS";
        fault_count++;
    }
    for (let j = 0; j < memory.length; j++) {
        document.getElementById("main-table").rows[j].cells[i].innerHTML =
memory[j];
    }
}

document.getElementById("algo-label").innerHTML = "Algorithm: " +
algoName;
document.getElementById("fault-label").innerHTML = "Total Page Faults: " +
fault_count;
document.getElementById("hit-label").innerHTML = "Total Page Hits: " +
hit_count;
let h_percentage = (hit_count / page_count) * 100;
let f_percentage = (fault_count / page_count) * 100;
document.getElementById("fault-ratio-label").innerHTML = "Page Fault
Ratio: " + fault_count + ":" + page_count + " (" + f_percentage.toFixed(2) +
"%)";
document.getElementById("hit-ratio-label").innerHTML = "Page Hit Ratio: "
+ hit_count + ":" + page_count + " (" + h_percentage.toFixed(2) + "%)";

```

```

        for (let j = 0; j < pageValues.length; j++) {
            document.getElementById("table-string").rows[0].cells[j].innerHTML =
pageValues[j];
        }
        for (let j = 0; j < pageValues.length; j++) {
            document.getElementById("table-fault").rows[0].cells[j].innerHTML =
faults[j];
        }
        document.getElementById("table-section").scrollIntoView();
    }
}

function createTable() {
    var tableSection = document.createElement("section");
    tableSection.setAttribute("id", "table-section");
    tableSection.setAttribute("class", "container");
    document.body.appendChild(tableSection);

    let tableSec = document.getElementById("table-section");

    var tableDiv = document.createElement("div");
    tableDiv.setAttribute("id", "table-container");
    tableDiv.setAttribute("class", "jumbotron");
    tableSec.appendChild(tableDiv);

    let tableContainer = document.getElementById("table-container");

    var labelContainers = document.createElement("div");
    labelContainers.setAttribute("id", "lbl-container");
    tableContainer.appendChild(labelContainers);

    var tableString = document.createElement("TABLE");
    tableString.setAttribute("id", "table-string");
    tableString.setAttribute("align", "center");
    tableString.setAttribute("width", "80%");
    tableContainer.appendChild(tableString);

    var table = document.createElement("TABLE");
    table.setAttribute("id", "main-table");
    table.setAttribute("align", "center");
    table.setAttribute("width", "80%");
    tableContainer.appendChild(table);

    var tableFault = document.createElement("TABLE");
    tableFault.setAttribute("id", "table-fault");
    tableFault.setAttribute("align", "center");
    tableFault.setAttribute("width", "80%");
    tableContainer.appendChild(tableFault);

    var div_f = document.createElement("div");
    div_f.setAttribute("id", "cont1");
    labelContainers.appendChild(div_f);

    var div_s = document.createElement("div");
    div_s.setAttribute("id", "cont2");
    tableContainer.appendChild(div_s);

```

```

var label = document.createElement("h6");
label.setAttribute("id", "algo-label");
div_f.appendChild(label);

var faultsContainer = document.createElement("div");
faultsContainer.setAttribute("id", "faults-container");
div_s.appendChild(faultsContainer);

var hitContainer = document.createElement("div");
hitContainer.setAttribute("id", "hit-container");
div_s.appendChild(hitContainer);

var label = document.createElement("h6");
label.setAttribute("id", "fault-label");
faultsContainer.appendChild(label);

var label = document.createElement("h6");
label.setAttribute("id", "hit-label");
hitContainer.appendChild(label);

var label = document.createElement("h6");
label.setAttribute("id", "fault-ratio-label");
faultsContainer.appendChild(label);

var label = document.createElement("h6");
label.setAttribute("id", "hit-ratio-label");
hitContainer.appendChild(label);
}

function removeData() {
    let table = document.getElementById('main-table');
    if (table != null) { //tanggalin kung present sa website yung table output.
        $("#table-section").remove();
    }
}

function resetData() {
    var pagesInputField = document.getElementsByName('page-input');
    for (let i = 0; i < pagesInputField.length; i++) {
        pagesInputField[i].value = "";
    }
    removeData();
    $('#exampleModal').modal('hide');
}

function clearInput(pagesInputField) {
    pagesInputField.value = "";
}

function lengthInput(pageInputField) {
    if (pageInputField.value.length > pageInputField.maxLength)
        pageInputField.value = pageInputField.value.slice(0,
pageInputField.maxLength);
}

```

```

function numPages(opts) {
    selectedOpts = opts.value;
    var numPagesCheck = document.getElementsByName('pages');
    var framesCheck = document.getElementsByName('frames');
    var pagesInputField = document.getElementsByName('page-input');
    addButton.disabled = false;
    resetButton.disabled = false;

    if (numPagesCheck[0].disabled == true) {
        numPagesCheck[0].checked = true;
    }

    if (framesCheck[0].disabled == true) {
        framesCheck[0].checked = true;
    }

    switch (selectedOpts) {
        case "lru":
        case "optimal":
            for (let i = 0; i < pagesInputField.length; i++) {
                pagesInputField[i].disabled = false;
            }
            for (i = 0; i < numPagesCheck.length; i++) {
                numPagesCheck[i].disabled = false;
            }
            for (i = 0; i < framesCheck.length; i++) {
                framesCheck[i].disabled = false;
            }
            break;
    }
}

function page_controller(page_handler) {
    page_count = parseInt(page_handler.value);
    var textFields = document.getElementsByName('page-input');
    switch (page_count) {
        case 6:
            textFields[6].style.display = "none";
            textFields[7].style.display = "none";
            textFields[8].style.display = "none";
            textFields[9].style.display = "none";
            break;
        case 7:
            textFields[6].style.display = "inline";
            textFields[7].style.display = "none";
            textFields[8].style.display = "none";
            textFields[9].style.display = "none";
            break;
        case 8:
            textFields[6].style.display = "inline";
            textFields[7].style.display = "inline";
            textFields[8].style.display = "none";
            textFields[9].style.display = "none";
            break;
        case 9:
            textFields[6].style.display = "inline";
    }
}

```

```

        textFields[7].style.display = "inline";
        textFields[8].style.display = "inline";
        textFields[9].style.display = "none";
        break;

    default:
        textFields[6].style.display = "inline";
        textFields[7].style.display = "inline";
        textFields[8].style.display = "inline";
        textFields[9].style.display = "inline";
    }
}

function frame_controller(opts) {
    number_of_frames = parseInt(opts.value);
}

function start_body() {
    resetButton = document.getElementById('resetButton');
    addButton = document.getElementById('addButton');
    addButton.disabled = true;
    resetButton.disabled = true;
    var textFields = document.getElementsByName('page-input');
    page_count = 6;
    number_of_frames = 4;
    textFields[6].style.display = "none";
    textFields[7].style.display = "none";
    textFields[8].style.display = "none";
    textFields[9].style.display = "none";
}

function getValue() {
    for (i = 0; i < pages; i++) {
        for (j = 0; j < frames; j++) {
            tmpPages =
document.getElementById("main-table").rows[j].getElementsByTagName('frames').c
ells[i].getElementsByTagName('page-input')[0].value;
            if (j == frames) {
                tmpPageFault =
document.getElementById("main-table").rows[j].getElementByTageName('pageFault'
).value;
            }
        }
    }
}

```

"All our dreams can come true, if we have the courage to pursue them."

– Walt Disney.