**CHAPTER - 1**

**INTRODUCTION**

**Introduction to the Project:**

In this project, a free, educational software is proposed to demonstrate and verify basic Operating System Concepts such as Process Creation, Process Scheduling, Process Synchronization, Deadlock and Page Replacement Algorithms along with designing a Website where the software and its documentation is available to be downloaded by the general public.

This software aims to transform the theory of above topics into programmable code so as to graphically represent the concepts. As well as test and compare the results for each simulation. The software provides graphical aid to the user to perform- scheduling of processes to view turnaround time, waiting time and Gantt chart (FCFS, SJF, RR, Priority), simulation of Producer and Consumer Problem to see use of buffer and synchronization, testing on a process sequence for deadlock (Banker’s Algorithm) and implementation of FIFO, LRU and Optimal page replacement algorithm on a page sequence. The website will be the medium of access to the software and provides information on how to use the software. Users can provide feedback and updates and bug fixes can be released on the website./

* 1. Background:

This project is designed to know about the content of Operating System concepts like Process Creation, Process Scheduling, Banker’s Algorithm and Page replacement algorithms. It is Desktop application that graphically represents the concept of Operating system.

* 1. Problem Statement:

To develop free educational software using java and java swing to demonstrate and verify basic OS concepts such as:

i) Process creation

ii) Process scheduling (FCFS, SJF, Priority, and Round Robin)

iii) Process Synchronization (producer consumer problem)

iv) Deadlock (banker’s algorithm)

v) Page replacement algorithm (FIFO, LRU, and Optimal)

**CHAPTER - 2**

**SOFTWARE REQUIREMENT SPECIFICATIONS**

**FOR**

**VINAYA (VERSION: 0.0.1)**

**A FREE EDUCATIONAL SOFTWARE.**

**SUB CHAPTER – 1**

**ABSTRACT**

**1.0 ABSTRACT**

In this project, a free, educational software is proposed to demonstrate and verify basic Operating System Concepts such as Process Scheduling, Process Synchronization, Deadlock and Page Replacement Algorithms along with designing a Website where the software and its documentation is available to be downloaded by the general public.

This software aims to transform the theory of above topics into programmable code so as to graphically represent the concepts. As well as test and compare the results for each simulation. The software provides graphical aid to the user to perform- scheduling of processes to view turnaround time, waiting time and Gantt chart (FCFS, SJF, RR, Priority), simulation of Producer and Consumer Problem to see use of buffer and synchronization, testing on a process sequence for deadlock (Banker’s Algorithm) and implementation of FIFO, LRU and Optimal page replacement algorithm on a page sequence. The website will be the medium of access to the software and also provides information on how to use the software. Users can provide feedback and updates and bug fixes can be released on the website.

The software is coded in Java and Java Swing while the website is coded in HTML, CSS, and JavaScript. The UI design is based on Microsoft’s Fluent and Metro UI Design languages. Web domain and hosting is done by help of AWS.

The targeted audiences for this software are students and teachers, who have basic understanding of Operating System concepts, for practical implementation of topics. The benefit of using this software is that it will provide understanding of the theory behind a topic, proper visualization, use of code for simulation and comparison of found results. Like a calculator provides correct answer for any calculation, this software may serve as digital truth for every simulation by the user.

The results for all simulations are viewed in separate java frames with interactive UI. Data provided during simulation may be stored as files in local system for future reference and comparison.

The software strives for educational visualization i.e. to help learners to practically implement the concepts as it is used inside the Operating System.

**SUB CHAPTER – 2**

**INTRODUCTION**

**2.0 INTRODUCTION**

This Software Requirements Specification specifies the requirements of Vinaya (Version 0.0.1), a free, GUI based, educational software that can be downloaded from its official deployment website using which users can simulate various Operating System concepts such as:-

* Process Creation
* Process Scheduling (FCFS, SJF, RR, Priority)
* Process Synchronization (Producer Consumer Problem )
* Deadlock Avoidance (Banker’s Algorithm)
* Page Replacement Algorithms (FIFO, LRU, Optimal)

While the deployment website shall include:-

* Launch / Home Page
* Software download Page
* Documentation and User Manual Page
* User Forum and Feedback Page
* Version Update and Release Page
* Credits / About Page

**2.1 PURPOSE**

The purpose of this Software Requirements Specification is to verify that all the specifications are verified. This document also serves to ensure that the software is traceable throughout its software development life cycle and that it adheres to the Scrum framework under Agile Methodology.

**2.2 INTENDED AUDIENCE**

This SRS would be used by the following people:-

* Product Owner:
* Scrum Master:
* Developers: The developers would use this document to implement the functionalities and to ensure traceability of the software.
* Users: The users would use this document to verify if the requirements specified satisfy their needs.

**2.3 SCOPE OF THE PROJECT**

The purpose of this proposed educational software is to allow users to demonstrate and verify earlier mentioned Operating System concepts through simulation. The GUI of this software shall allow ease of access to the user and help understand the working of algorithms behind these topics.

The software promotes:

* Education – via demonstration of concepts and verification of OS problems.
* Student Research – via comparing functionalities of various algorithms on a particular data set.

The deliverables of the software should be a document/text/image file of conducted simulation. This file may be used in another simulation or be saved for future reference.

This project has a lot of scope for future development. Features such as Pre-emptive Scheduling, Classical Synchronization Paradigms, RAG for Deadlock and many more such functionalities can be implemented in the later versions of this software.

**2.4 OVERVIEW OF THE DOCUMENT**

The first section of the document gave a brief description about Vinaya and the various benefits that it provides. In the following sections we will describe the requirements, assumptions, dependencies, constraints and other such concepts about the software.

**SUB CHAPTER – 3**

**OVERALL PRODUCT DESCRIPTION**

**3.1 PRODUCT PERSPECTIVE**

Vinaya can be downloaded from its official deployment website and can be used to demonstrate and verify earlier mentioned basic Operating System concepts. The simulation may be saved as document, text or image file for future reference. These files may later be used as comparison between different primitive algorithms as part of student research.

**3.2 PRODUCT FEATURES**

The project showcases simulation of mentioned OS concepts using features such as:

* An interactive, static, deployment website to allow users to:
  + download the software and user manual
  + check update and release log
  + give feedback and discuss common issues in a forum
* Accept user defined processes and store it for future use in a database.
* Schedule a set of processes to view Gantt chart or scheduling details.
* Simulate synchronization in between Producer and Consumer.
* Input Resource type, instances, processes along with Allocation, Available and Max matrix to implement Banker’s Algorithm and deem system safe or unsafe.
* Input a page sequence and implement FIFO, LRU, Optimal to simulate page replacement and show page faults, page hit, efficiency ratios.

**3.3 USER CLASSES AND CHARACTERISTICS**

There will be primarily two types of targeted audience for this software:

* Teachers: To demonstrate working of various basic concepts inside an OS and implementation of primitive algorithms behind topics.
* Students: To verify the results obtained in each simulation; to compare between simulation results and document findings.

**3.4.1 OPERATING ENVIRONMENT**

Hardware Specification:-

- Processor – Intel(R) Core(TM) i5-7200 CPU @ 2.50GHz

- RAM – 4.00 GB

- Hard Disk – 1 TB

- System Type – 64-bit Operating System, x64-based processor

Software Specification;-

- Operating System – Windows 10 Home Edition

- IDE – NetBeans IDE 8.2

- Java – version 8 1.8.0\_131; Java HotSpot(TM) 64 bit server

- Runtime – Java(TM) SE Runtime Environment

- JDK – JDK 1.8.0\_131

- Web Editor – Brackets

- Repository – GitHub

https://github.com/Ankita2101/MajorProj-OS-

**3.4.2 HARDWARE REQUIREMENTS**

- Processor – Intel Pentium & above

- RAM – 1.00 GB & above

- Hard Disk – 512mb & above

- System Type – 32 or 64-bit Operating System

-All platforms

**3.4.3 SOFTWARE REQUIREMENTS**

- Java – version 8

- Runtime – Java(TM) SE Runtime Environment

- JDK – JDK 1.8.0\_131

- Web Browser- Chrome, Firefox etc.

**3.5 DESIGN AND IMPLEMENTATION CONSTRAINTS**

* For ensuring platform independence of the software the implementation will be JAVA so the end users system must have a JAVA run time environment.
* The software may schedule at most 10 processes.
* FCFS, SJF and Priority shall be Non-Pre-emptive scheduling Algorithms.
* Priority and Round Robin scheduling will perform at same arrival time.
* For process synchronization, size of producer, consumer and buffer shall be at most 10 items. (recommended)
* In case of Banker’s Algorithm, storage of at most 10 resource types and 10 processes. (recommended)
* For page replacement, page sequence of 15 pages, frame size of 5 is recommended.

**3.6 USER DOCUMENTATION**

* A user document should be provided at the end of the development. It should have the following:
* A ReadMe file to help the user with the installation of the software.
* A well-documented User Manual.

**3.7 ASSUMPTIONS AND DEPENDENCIES**

ASSUMPTIONS:

* User already has basic idea about the concepts of OS being demonstrated in this software.
* User has experience of working with simulators.
* User knows how to use a software manual to run the software.

DEPENDENCIES:

* This software would need Java Runtime Environment for its execution. (Possible inclusion : Apache Derby and Glass Fish Server for Backend Support)
* It will need a web browser and an active Internet connection to download the software and user manual.
* It will require an approximate storage space of 50Mb.

**3.8 DATA REQUIREMENTS**

INPUT:

Process Creation: Location/path of a process

Process Scheduling: no. of processes, arrival time, burst time, priority, time quantum, scheduling algorithm, view.

Process Synchronization: No. of Items

Deadlock: No. of resources, processes, allocation matrix, max matrix, available matrix.

Page Replacement: no. of pages, frames, page sequence, replacement algorithm.

OUTPUT:

Process Creation: Execution of specified process

Process Scheduling: gantt chart, detailed view

Process Synchronization: synching between producer, consumer

Deadlock: need matrix, safe sequence

Page Replacement: pages in frame, page hit, page fault, page hit ratio, page fault ratio.

**3.9 USER VIEW OF THE SOFTWARE**

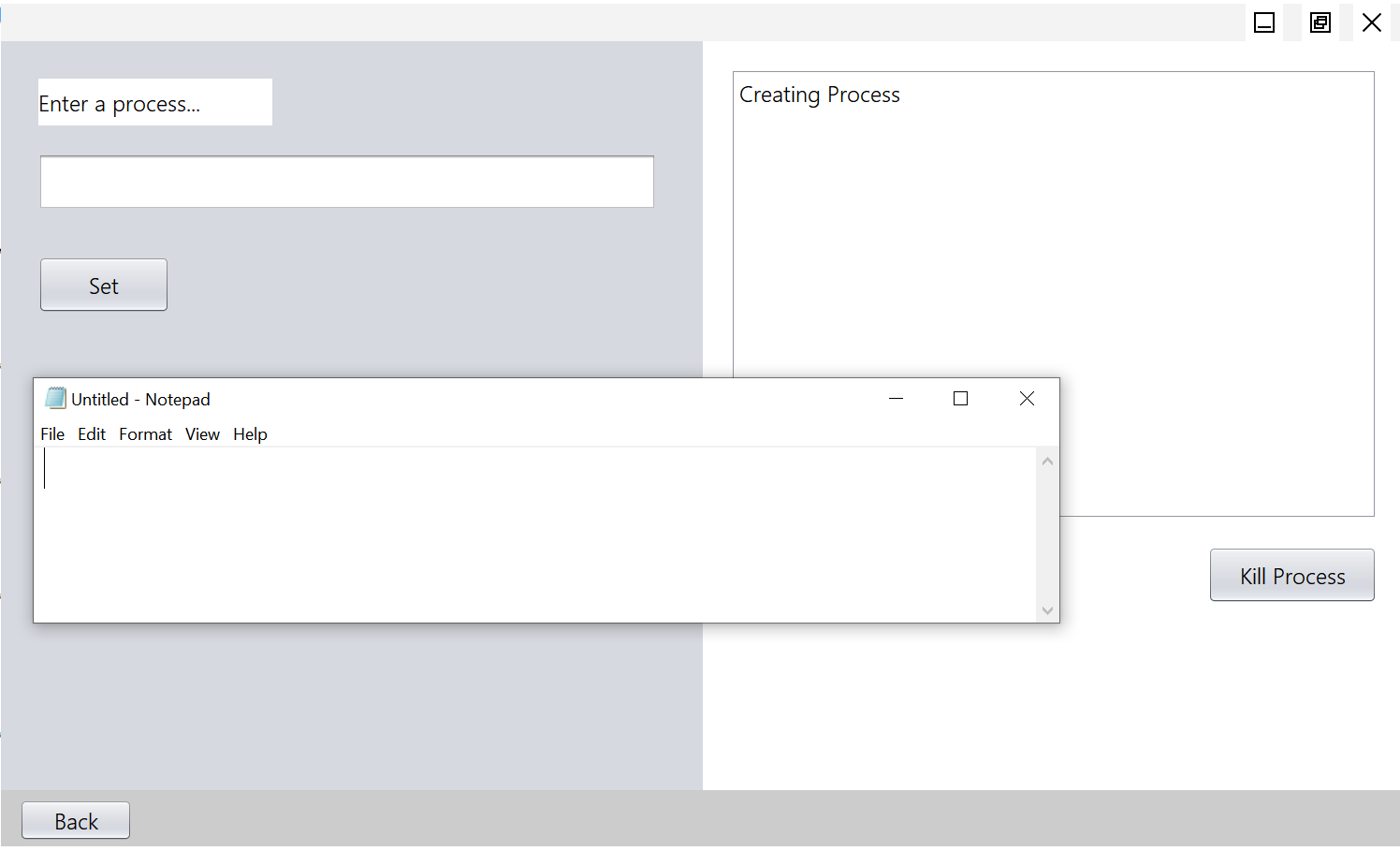
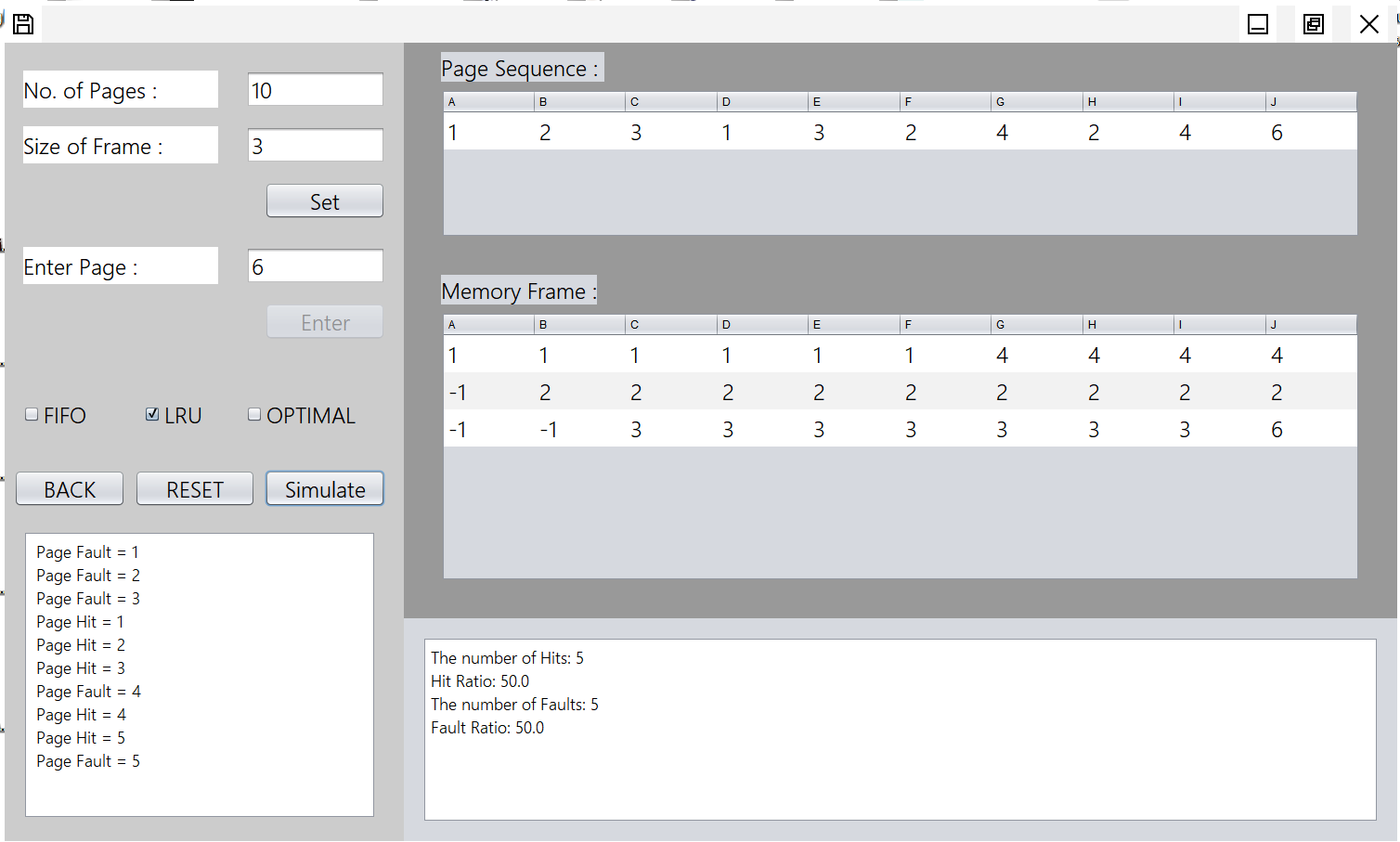
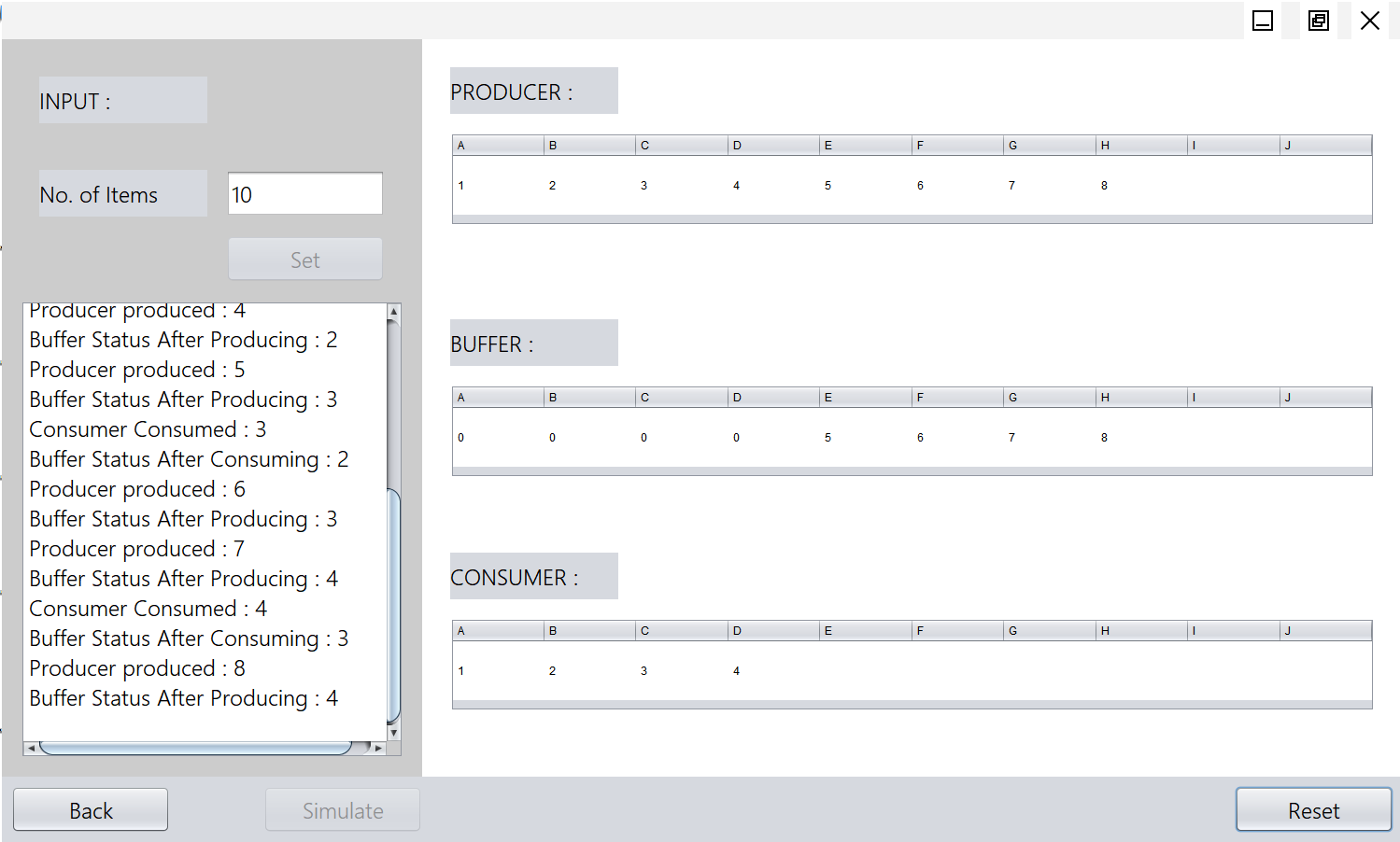
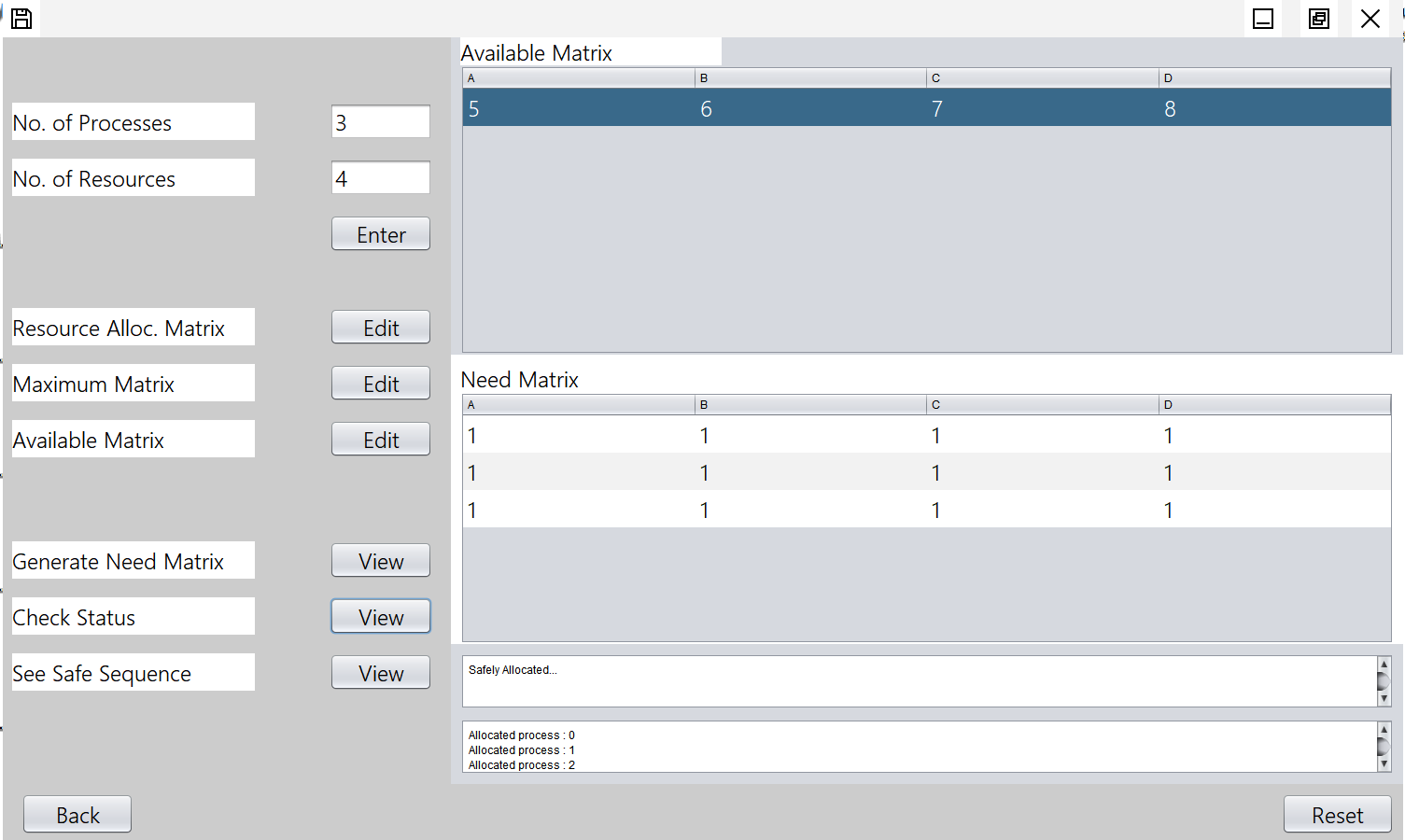
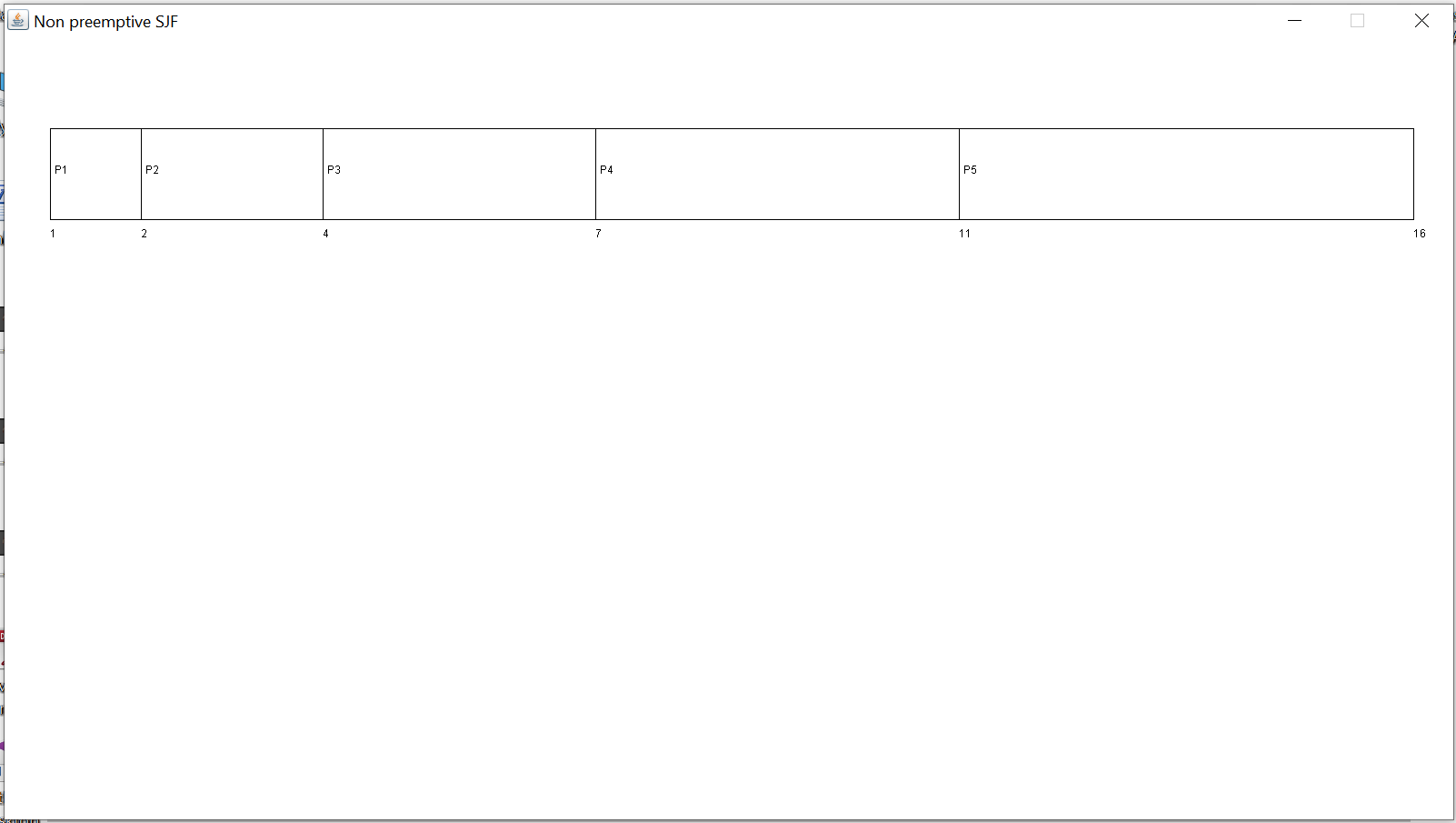
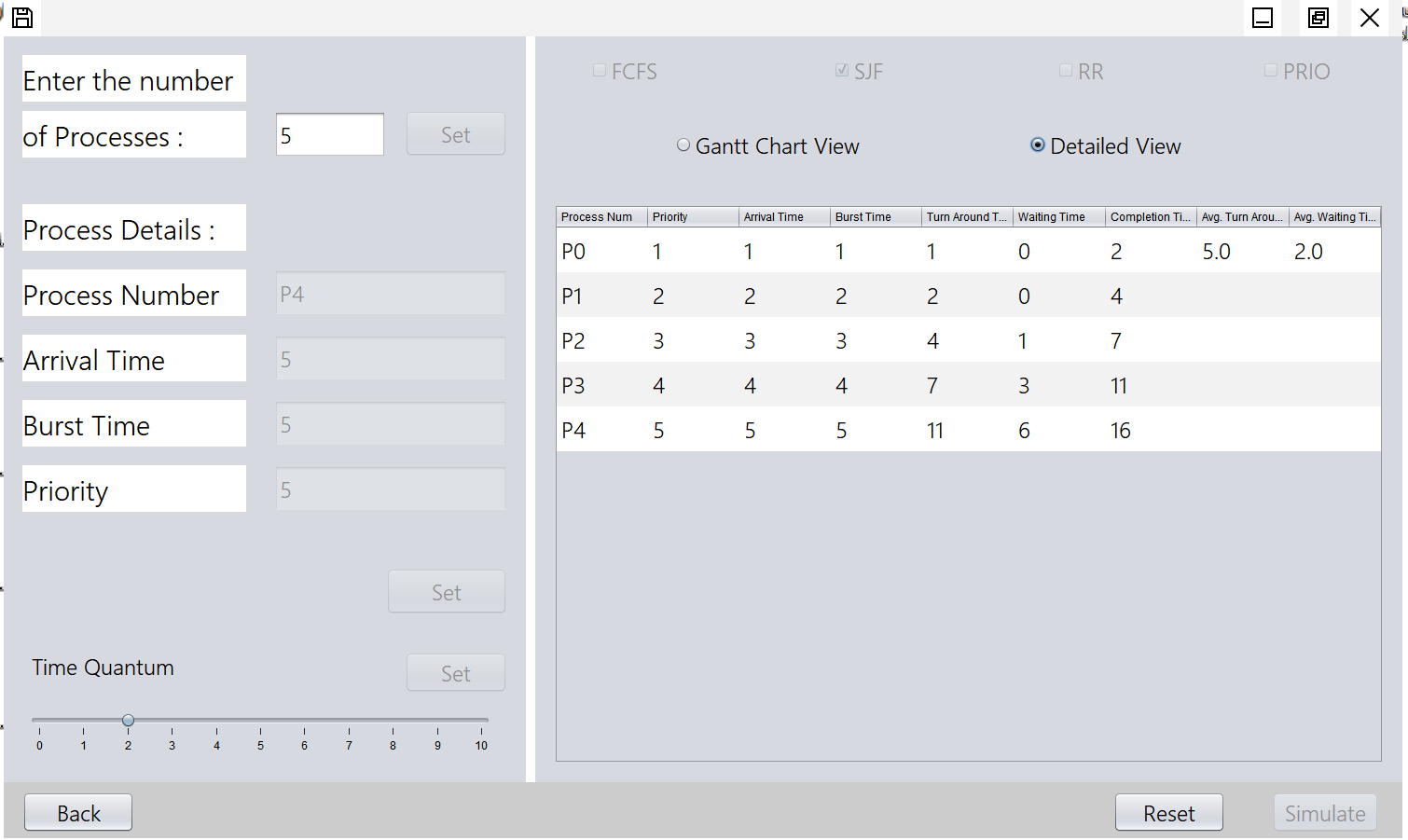
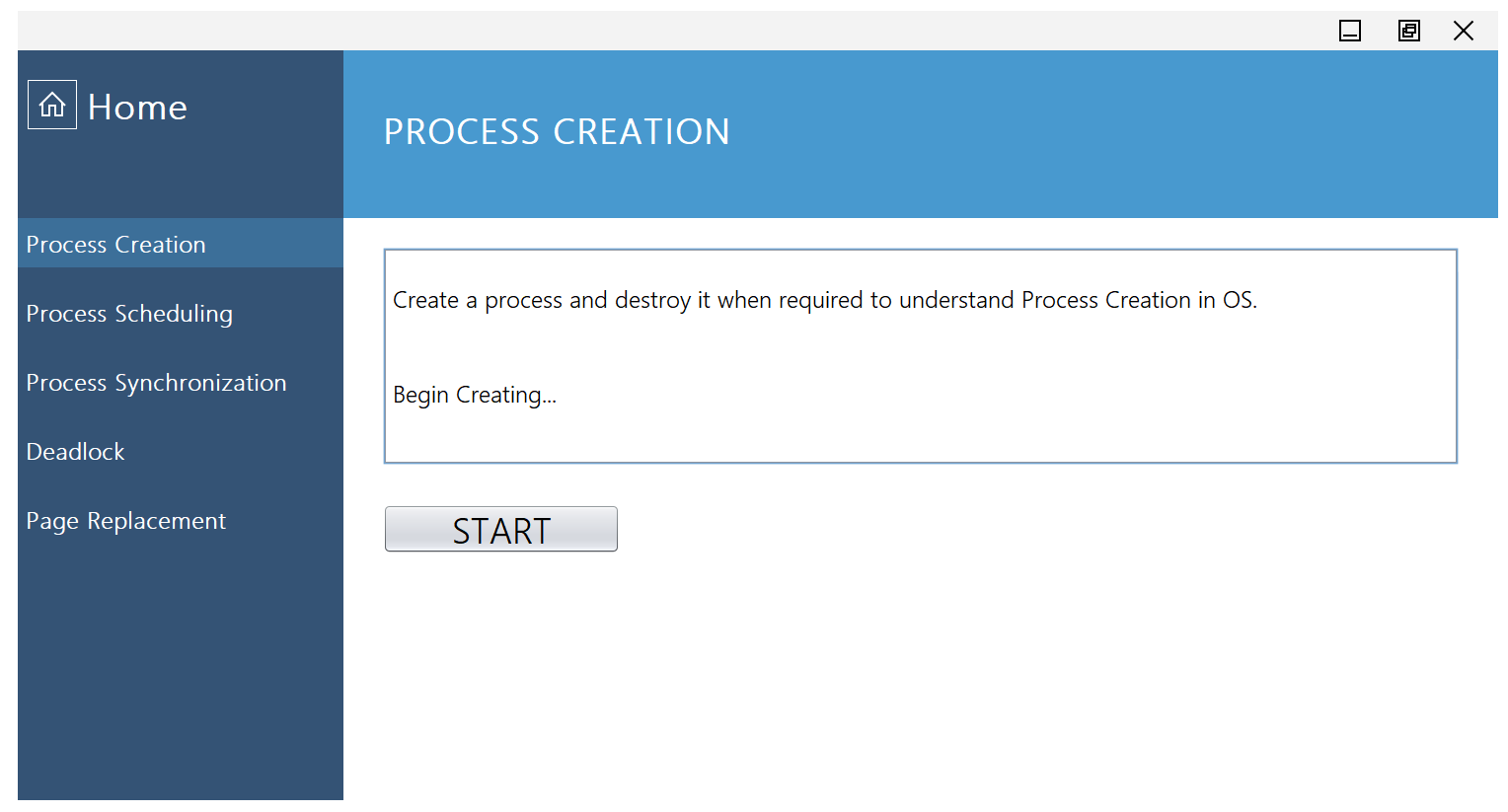
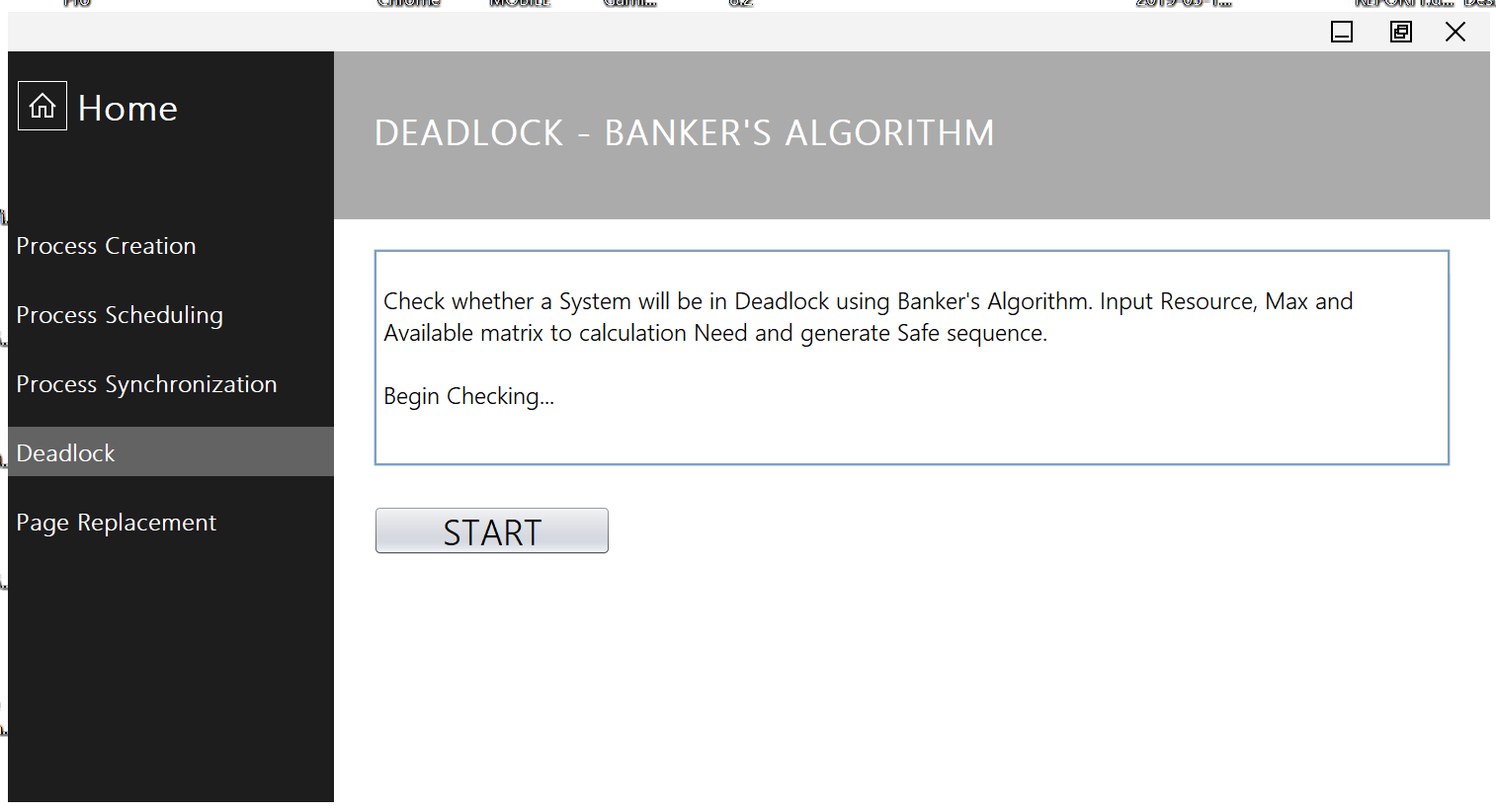
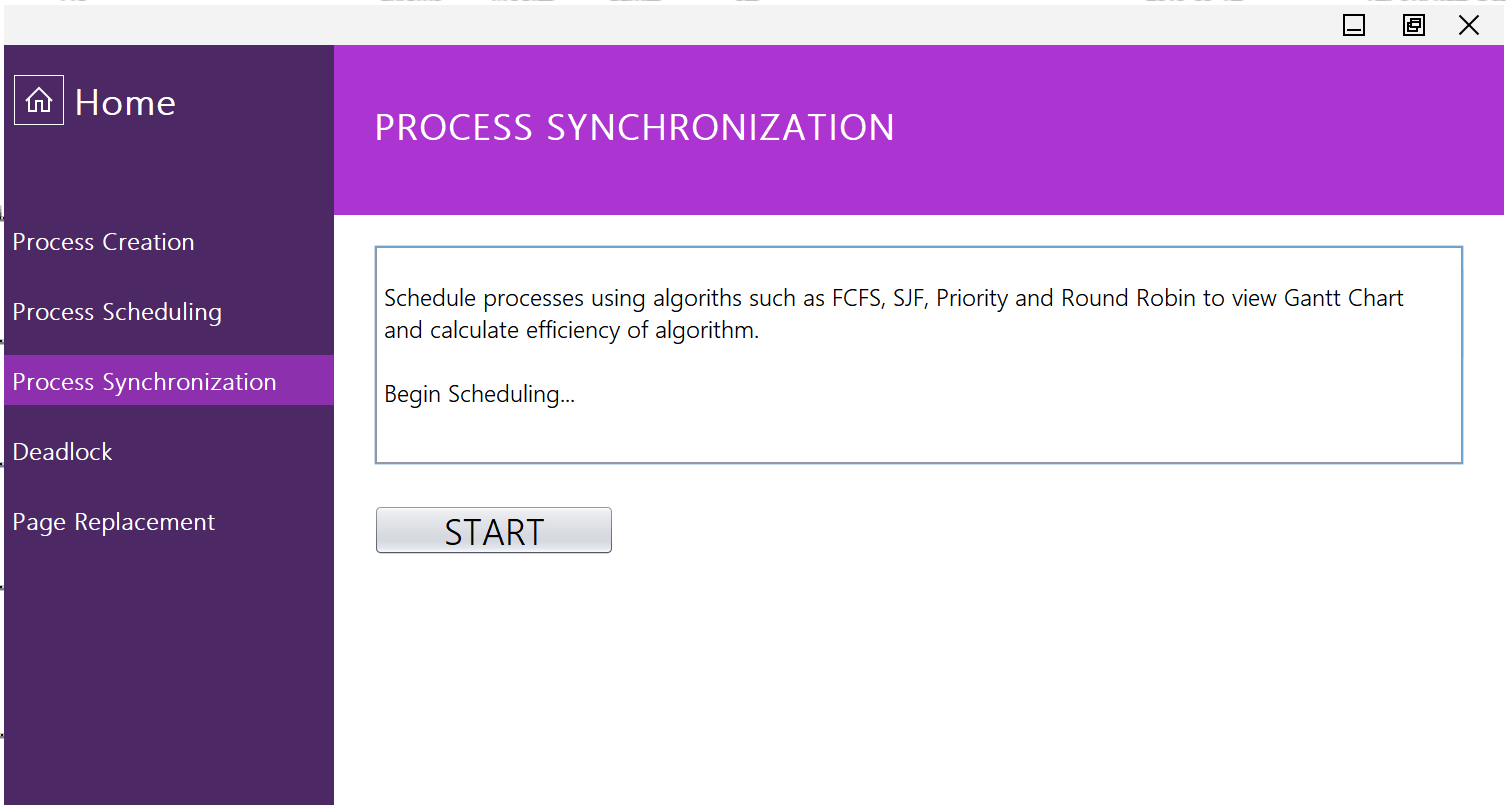
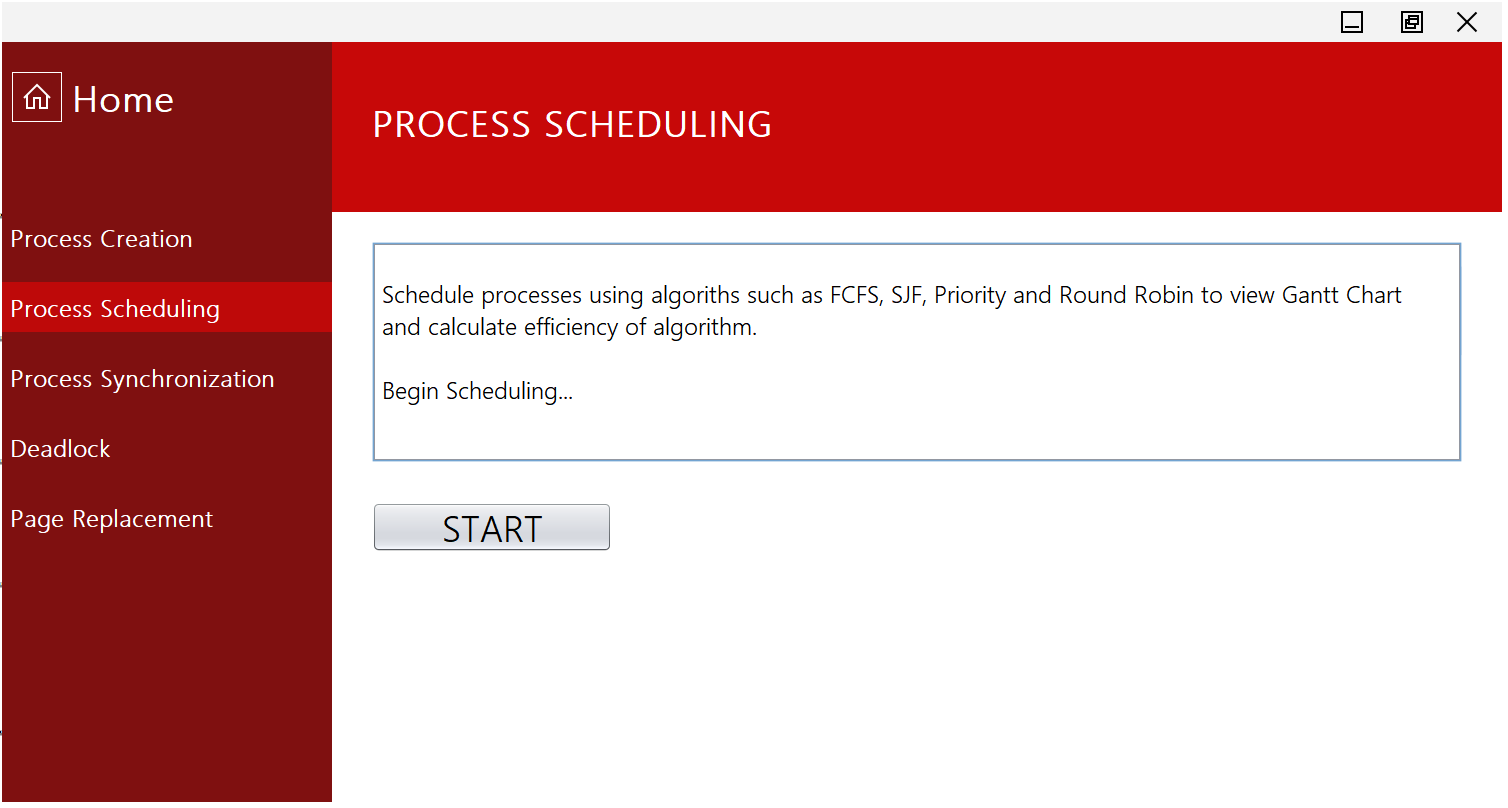
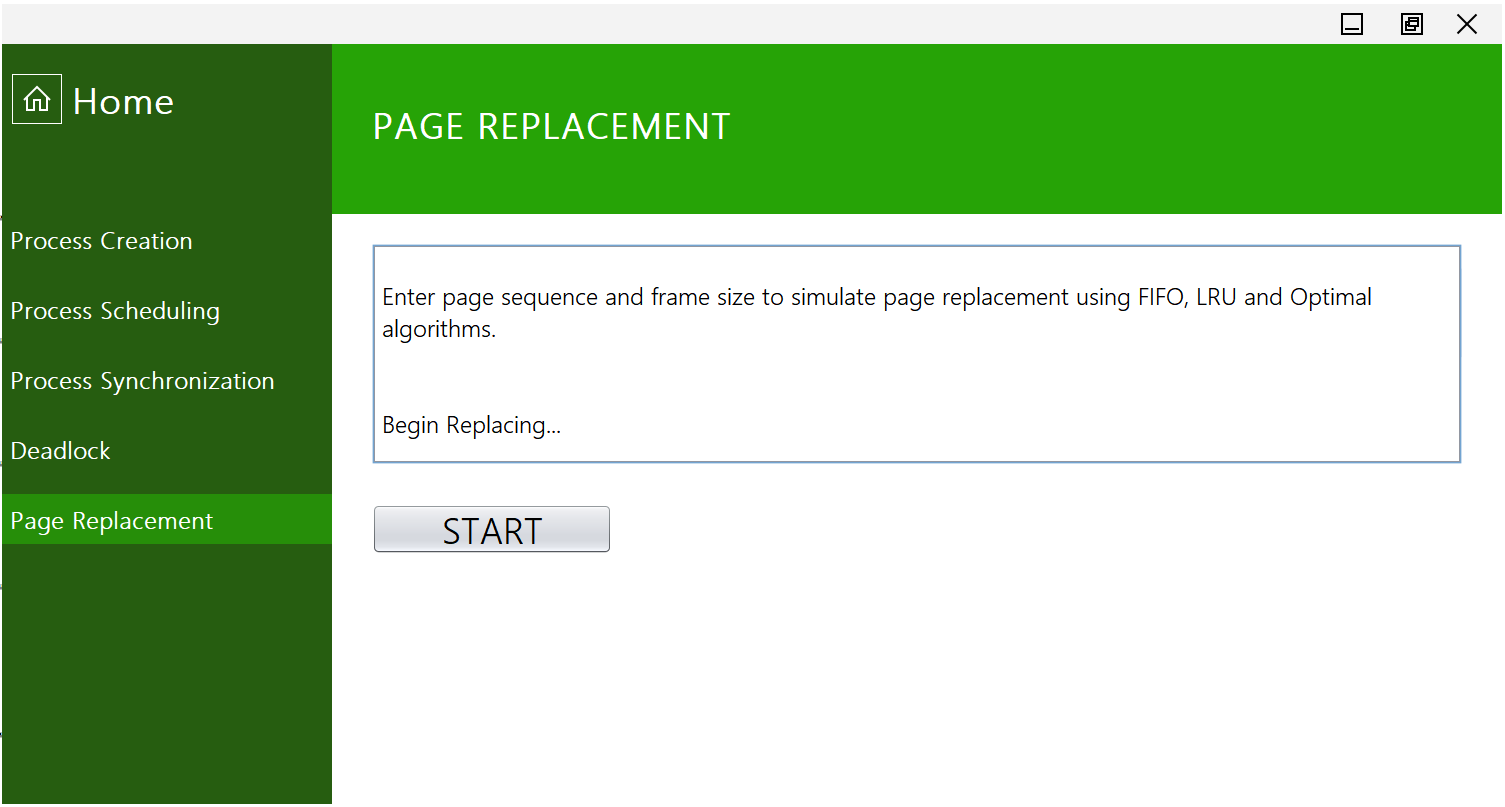
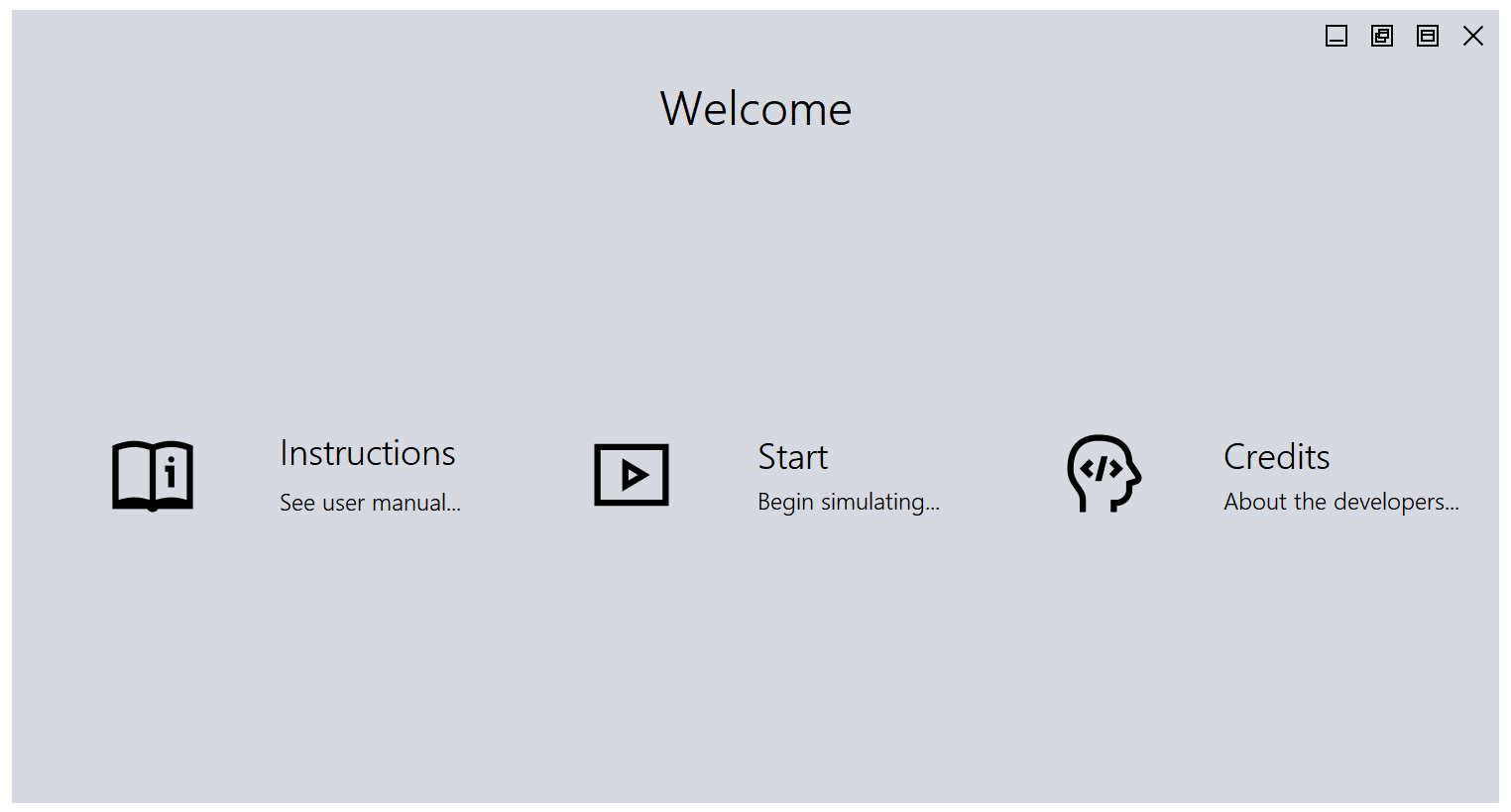
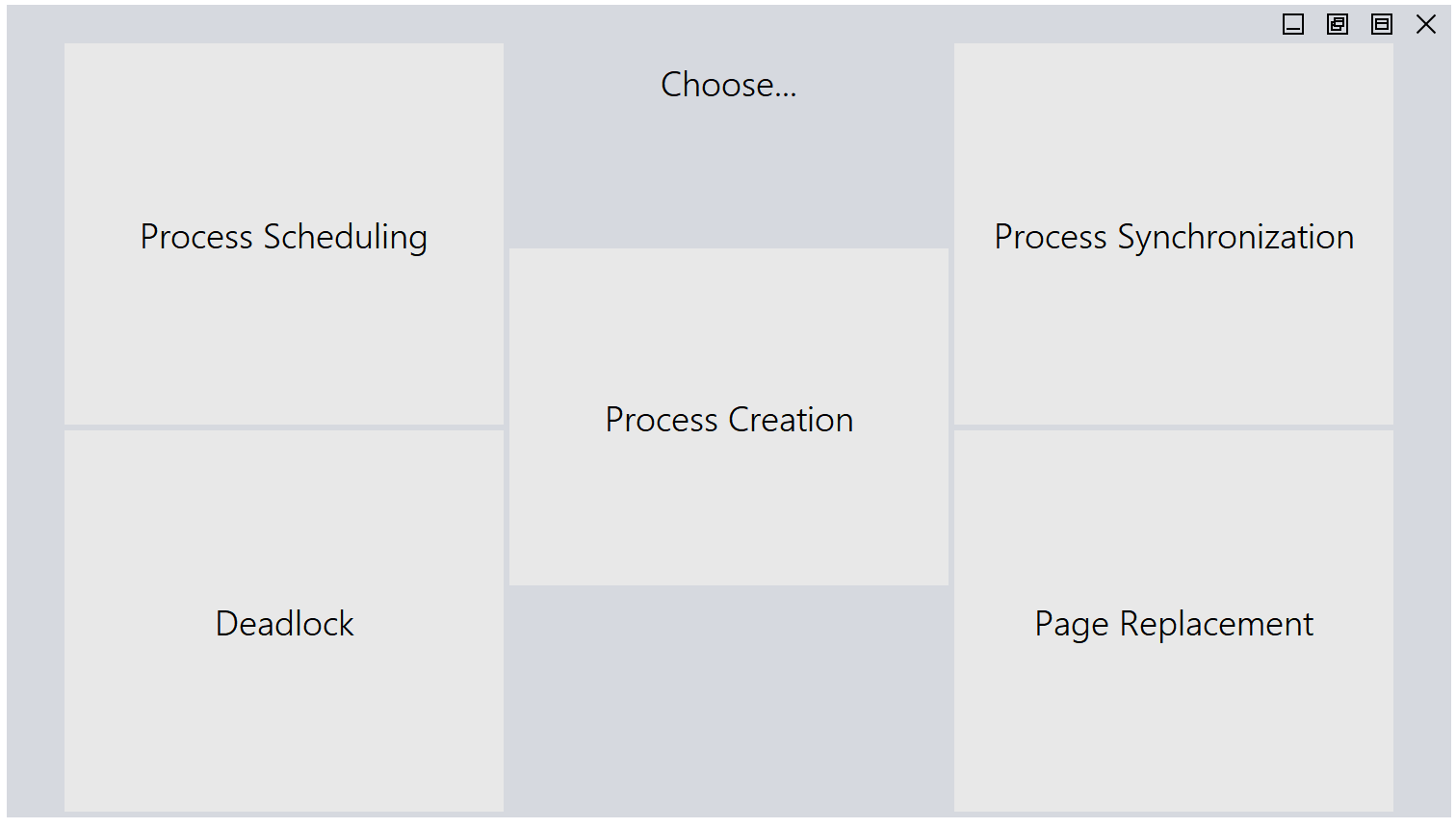
This software will have Graphical User Interface that will provide ease of access. The software will via frames/windows. The user can perform simulation by simply entering input into the text fields and clicking a label or button to begin the operation.

Possible types of Screens:

* Launch/Landing Frame
* Home Frame
* Instructions Frame
* Credits Frame
* Concept Window (For every Module)
* Simulation Window
* Dialog Boxes (Confirmation/ Warning)
* Exit Page



Splash Screen:



**3.10 EXTERNAL INTERFACE REQUIREMENT**

Hardware Peripherals: Computer System

Keyboard, Mouse (Preferred)

**3.10.1 HARDWARE INTERFACE**

* Vinaya depends on User Input to perform simulations. Hence a keyboard and mouse is preferred for proper functioning.
* Screen resolution of at least 800 X 600 or above will be preferable for viewing the frames / windows.

**3.10.2 SOFTWARE INTERFACE**

* It should be possible for Vinaya to be implemented across all platforms and Operating System Environments.
* The entire software is to be coded in Java and the GUI is to be designed using Java Swing. It should embed certain useful hyperlinks to its official website.
* The software and user manual will need a web browser for downloading it.

**SUB CHAPTER – 4**

**SYSTEM FUNCTIONALITY**

**4.1 PROVISION OF BASIC INFORMATION REGARDING CONCEPTS**

Basic working of the software and instructions shall be provided in the software manual.

**4.2 PROCESS CREATION**

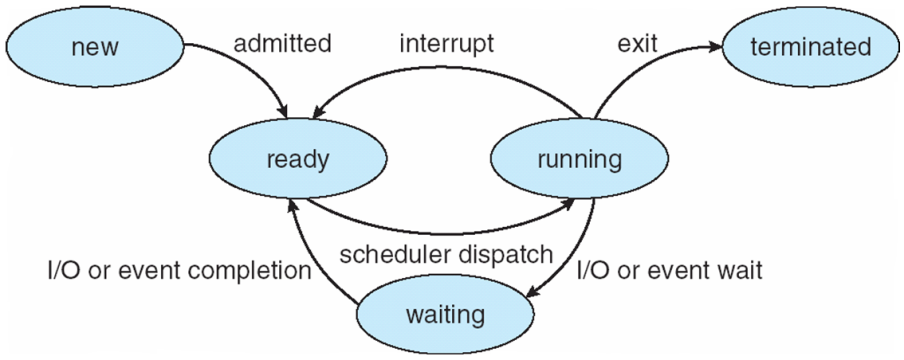
**PROCESS:-**

A process can be thought of as a program in execution. A process needs certain resources – such as CPU time, memory, files, and I/O devices to accomplish its task. These resources are allocated to the process either when it is created or while it is executing.

**PROCESS STATE :-**

As a process executes, it changes **State.** The state of a process is defined in part by the current activity of that process. A process may be in one of the following states:-

* New – The process is being created.
* Running – Instructions are being executed.
* Waiting – The process is waiting for some event to occur.
* Ready – The process is waiting to be assigned to a processor.
* Terminated – The process has finished execution.



**4.3 PROCESS SCHEDULING**

**CPU** **SCHEDULING**

It is a process which allows one process to use the CPU while the execution of another process is on hold due to unavailability of any resources like I/O etc , thereby making full use of CPU.

**CPU SCHEDULING CRITERIA**

There are several criteria to consider when try to select the “best” scheduling algorithm for a particular situation and environment, including:

* CPU utilization – Keep the CPU as busy as possible.
* Throughput – if the CPU is busy executing processes, then work is being done. One measure of work is the number of processes that are being completed per unit of time, called throughput.
* Turnaround time – it is the sum of the periods spent waiting, executing on the CPU and doing I/O.
* Waiting time – it is the sum of the periods spent waiting in the ready queue.
* Response time – it is the time from submission of request until the first response to be produced.

**SCHEDULING ALGORITHM**

FCFS: It is the simplest scheduling algorithm. It simply queues processes in the order that they arrive in the ready queue. In this, the process that comes first will be executed first and next process starts only after the previous gets fully executed.

SJF: It is the best approach to minimize waiting time. In this, the process that has shortest job will be executed first and next process starts only after the previous gets fully executed.

ROUND ROBIN: It is an arrangement of choosing all elements in a group equally in some rational order, usually from the top to the bottom of a list and then starting again at the top of the list and so on.

PRIORITY: In this each process is assigned a priority. Process with the highest priority is to be executed first and so on. Processes with the same priority are executed on first come first serve basis. Priority can be decided based on memory requirements, time requirements or any other resource requirement.

**4.4 PROCESS SYNCHRONIZATION**

 Producer: Who produces the data items when buffer is empty is called Producer

 Consumer: Who consumes the data items when buffer has at least one element is called Consumer.

 Producer Consumer Problem

The problem describes two processes, the producer and the consumer, who share a common, fixed-size buffer used as a queue. The producer's job is to generate data, put it into the buffer, and start again. At the same time, the consumer is consuming the data (i.e., removing it from the buffer), one piece at a time. The problem is to make sure that the producer won't try to add data into the buffer if it's full and that the consumer won't try to remove data from an empty buffer.

 Solution

The solution for the producer is to either go to sleep or discard data if the buffer is full. The next time the consumer removes an item from the buffer, it notifies the producer, who starts to fill the buffer again. In the same way, the consumer can go to sleep if it finds the buffer empty. The next time the producer puts data into the buffer, it wakes up the sleeping consumer.

The solution can be reached by means of bounded buffer solution.

**4.5 DEADLOCK AVOIDANCE**

In concurrent computing, a deadlock is a state in which each member of a group is waiting for another member, including itself, to take action, such as sending a message or more commonly releasing a lock. Deadlock is a common problem in multiprocessing systems, parallel computing, and distributed systems, where software and hardware locks are used to arbitrate shared resources and implement process synchronization.

Requirements of a deadlock:

1. Mutual exclusion

2. Hold and wait

3. No pre-emption

4. Circular wait

Handling a deadlock:

1. Deadlock prevention and avoidance

2. Deadlock detection and recovery

3. Ignore the problem all together

Banker’s Algorithm:

It is a resource allocation and deadlock avoidance algorithm that tests for the safety by simulating the allocation for predetermined maximum possible amounts of all resources, then makes a ‘s-state’ check to test for allocation should be allowed to continue or not.

This algorithm works by taking the number of processes and the number of resources in a matrix. It also needs a matrix which represents the maximum resource and also a matrix of size 1Xn (where 1 = number of row and n = number of columns) which represents the available matrix.

**4.6 PAGE REPLACEMENT**

The page replacement algorithm decides which memory page is to be replaced. The process of replacement is sometimes called swap out or write to disk. Page replacement is done when the requested page is not found in the main memory (page fault).

There are two main aspects of virtual memory, Frame allocation and Page Replacement. It is very important to have the optimal frame allocation and page replacement algorithm. Frame allocation is all about how many frames are to be allocated to the process while the page replacement is all about determining the page number which needs to be replaced in order to make space for the requested page.

What If the algorithm is not optimal?

1. if the number of frames which are allocated to a process is not sufficient or accurate then there can be a problem of thrashing. Due to the lack of frames, most of the pages will be residing in the main memory and therefore more page faults will occur.

However, if OS allocates more frames to the process then there can be internal fragmentation.

2. If the page replacement algorithm is not optimal then there will also be the problem of thrashing. If the number of pages that are replaced by the requested pages will be referred in the near future then there will be more number of swap-in and swap-out and therefore the OS has to perform more replacements then usual which causes performance deficiency.

Therefore, the task of an optimal page replacement algorithm is to choose the page which can limit the thrashing.

**Types of Page Replacement Algorithms**

There are various page replacement algorithms. Each algorithm has a different method by which the pages can be replaced.

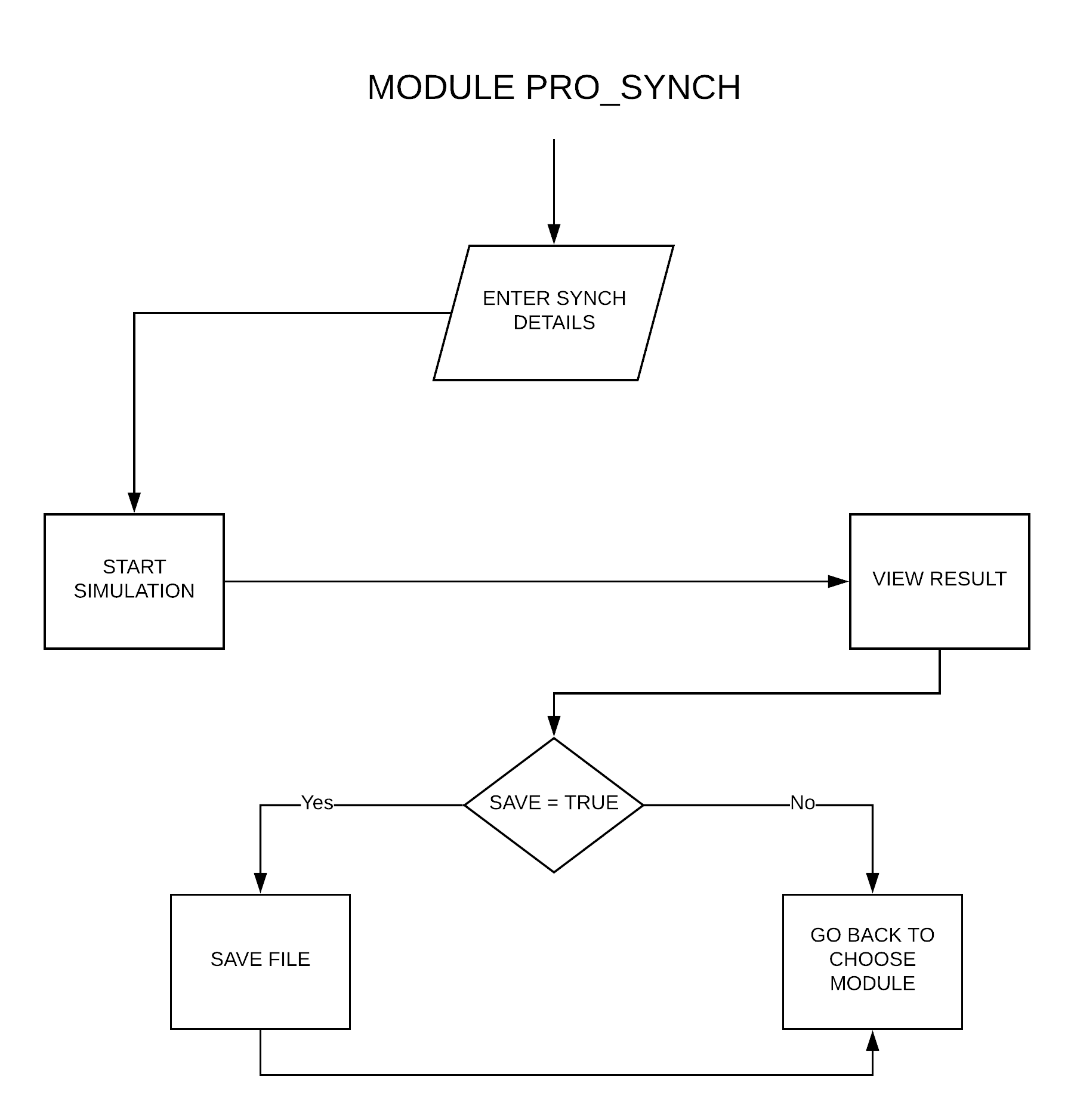
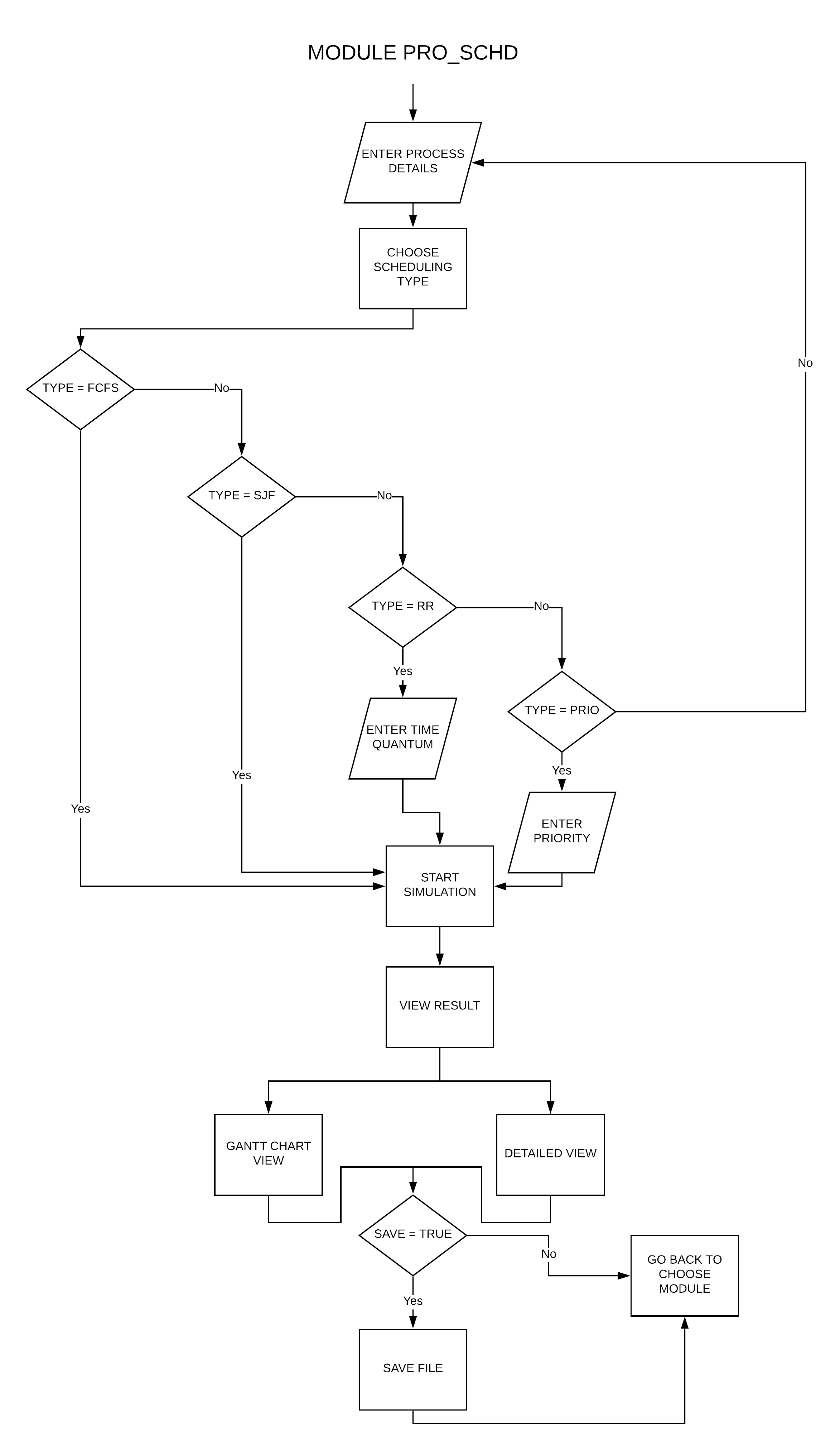
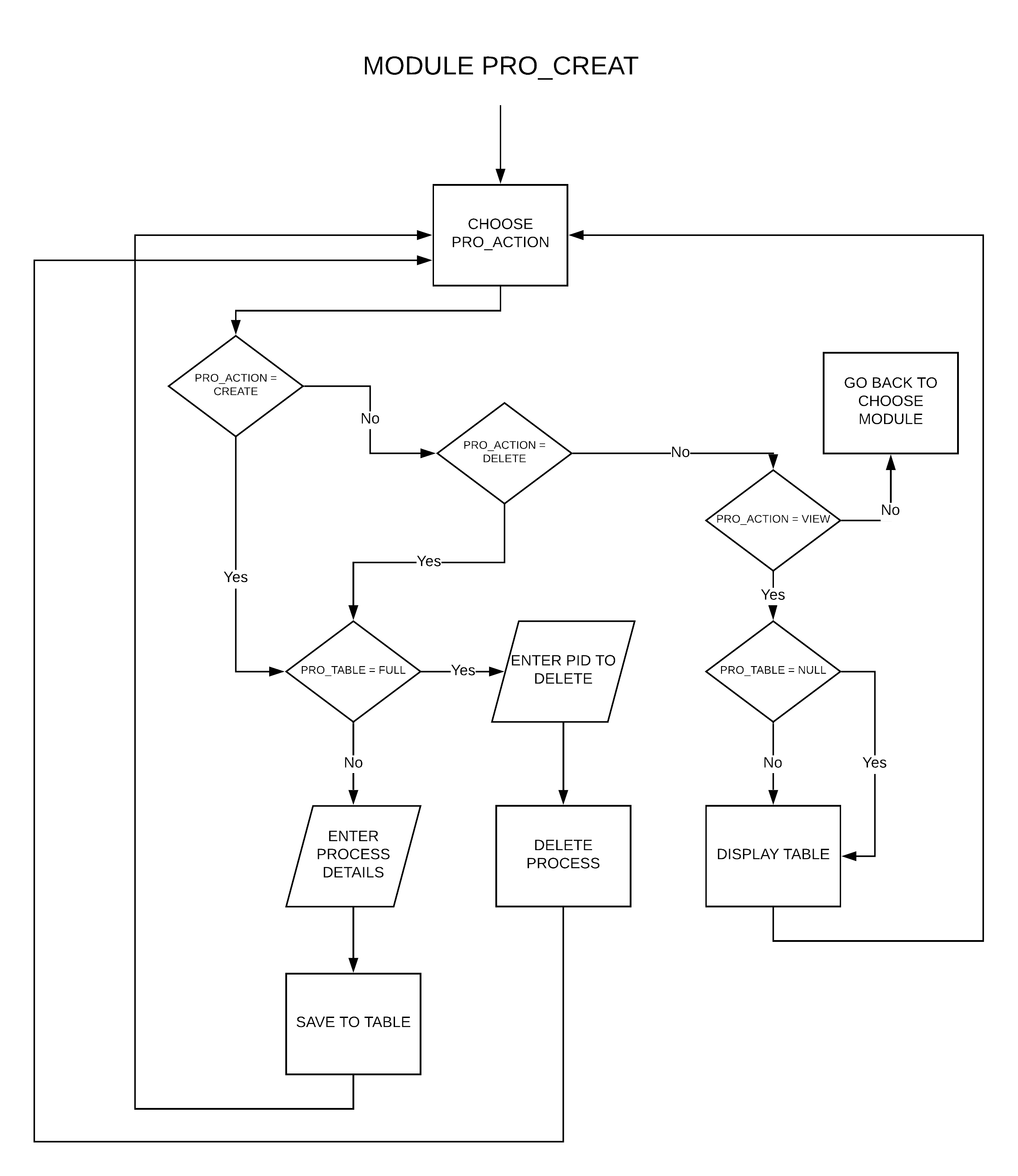
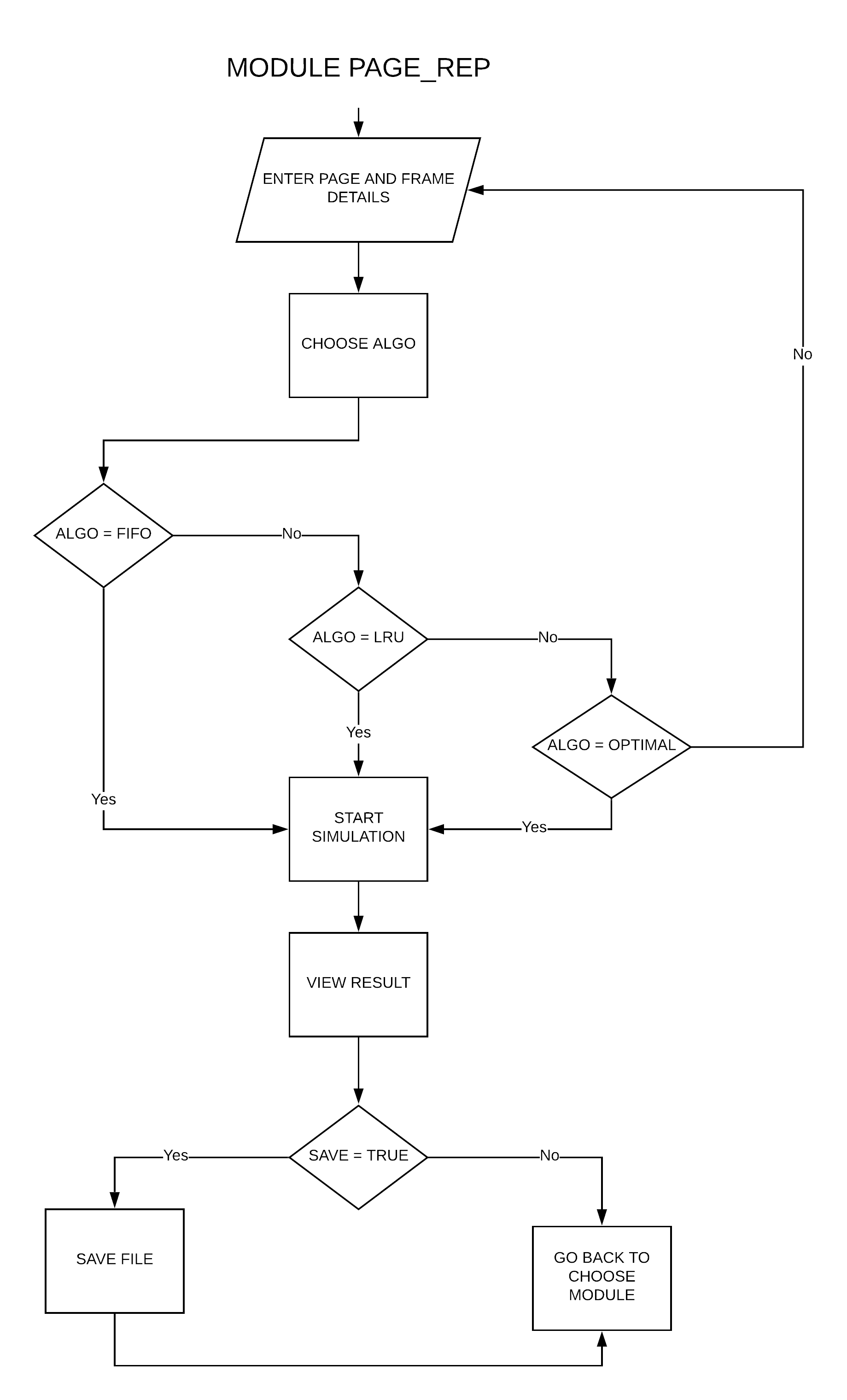
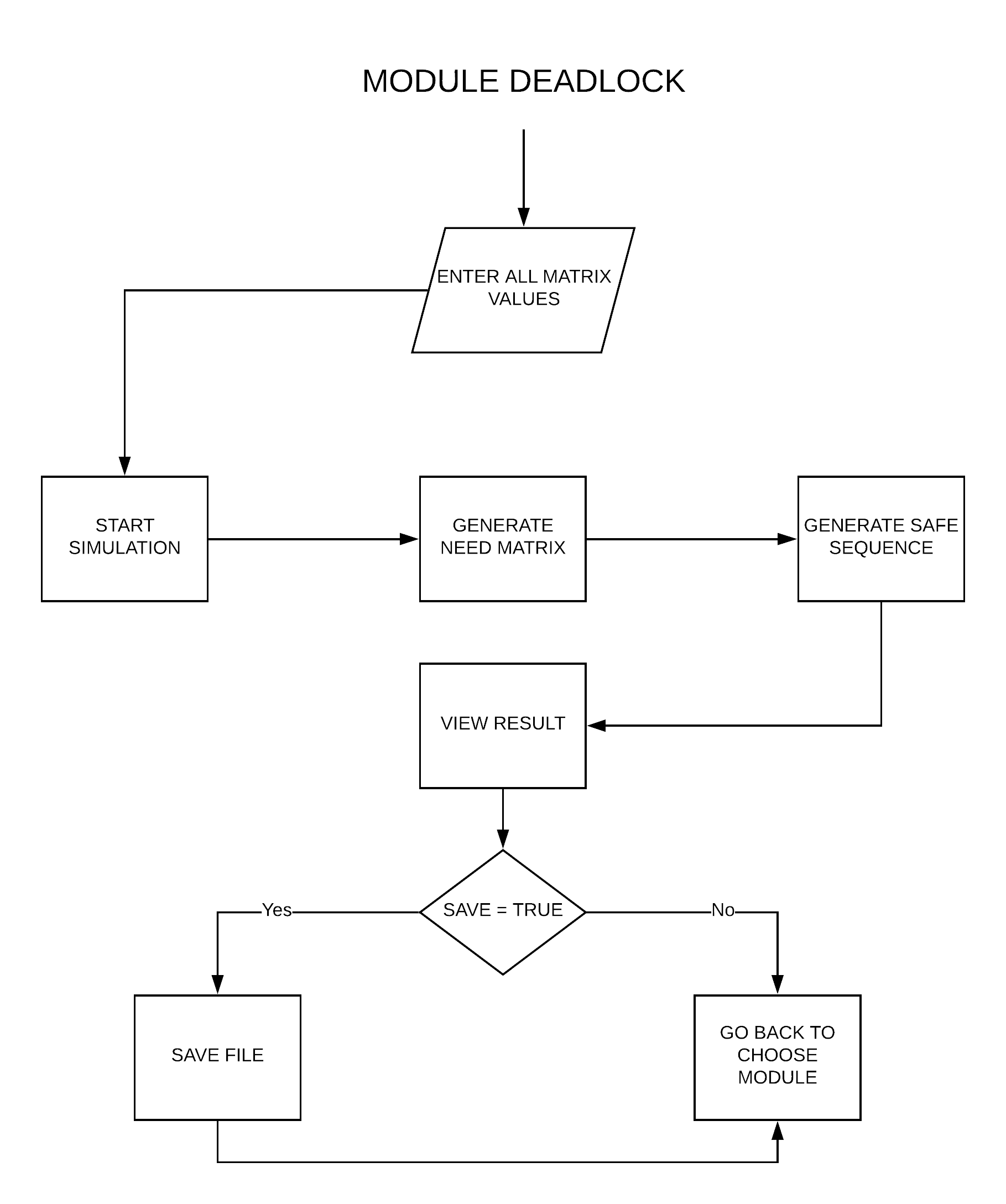
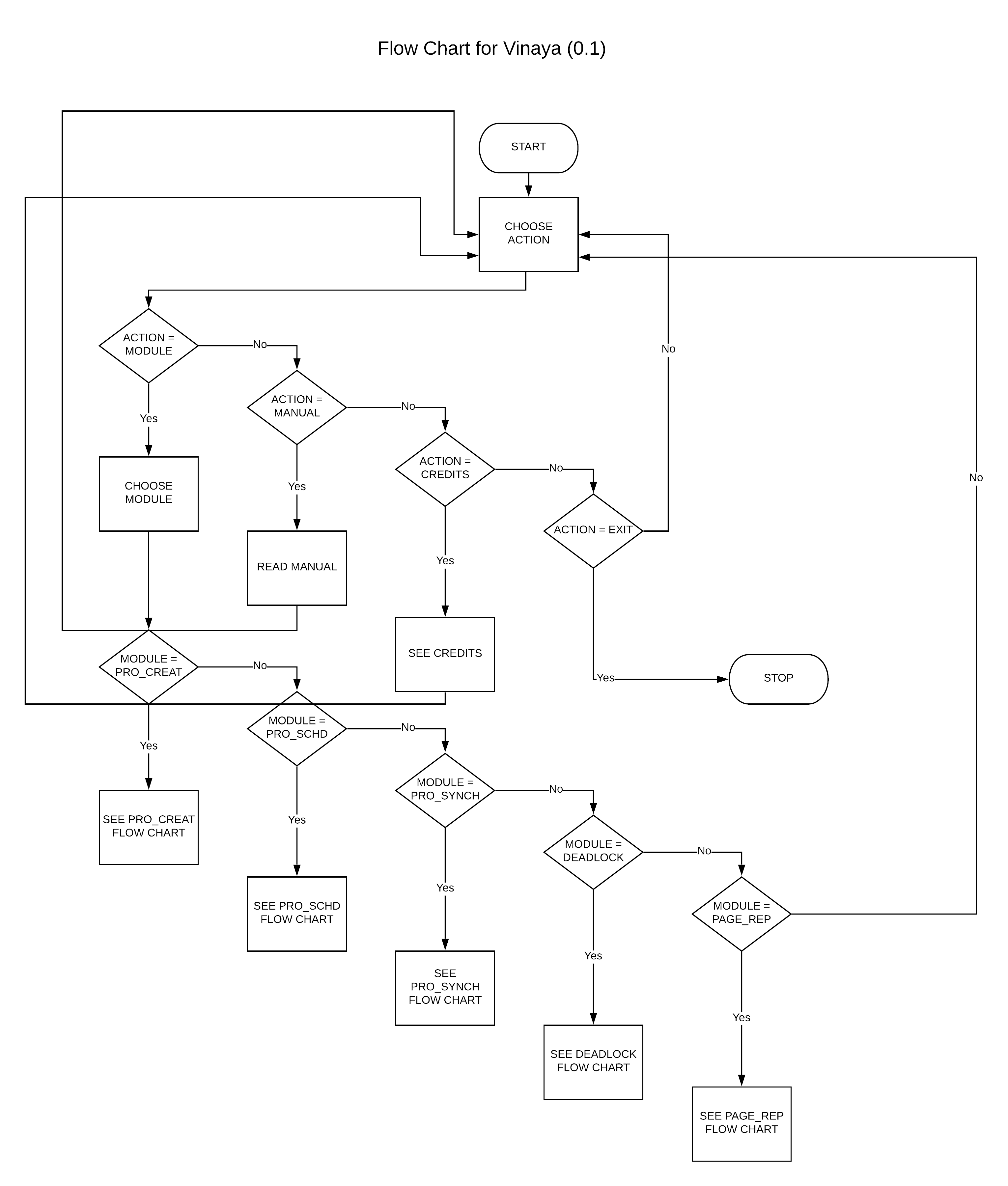
1. Optimal Page Replacement algorithm → this algorithms replaces the page which will not be referred for so long in future. Although it cannot be practically implemented, it can be used as a benchmark. Other algorithms are compared to this in terms of optimality.

2. Least recent used (LRU) page replacement algorithm → this algorithm replaces the page which has not been referred for a long time. This algorithm is just opposite to the optimal page replacement algorithm. In this, we look at the past instead of staring at future.

3. FIFO → in this algorithm, a queue is maintained. The page which is assigned the frame first will be replaced first. In other words, the page which resides at the rare end of the queue will be replaced on the every page fault.

**SUB CHAPTER – 5**

**FLOW CHART**



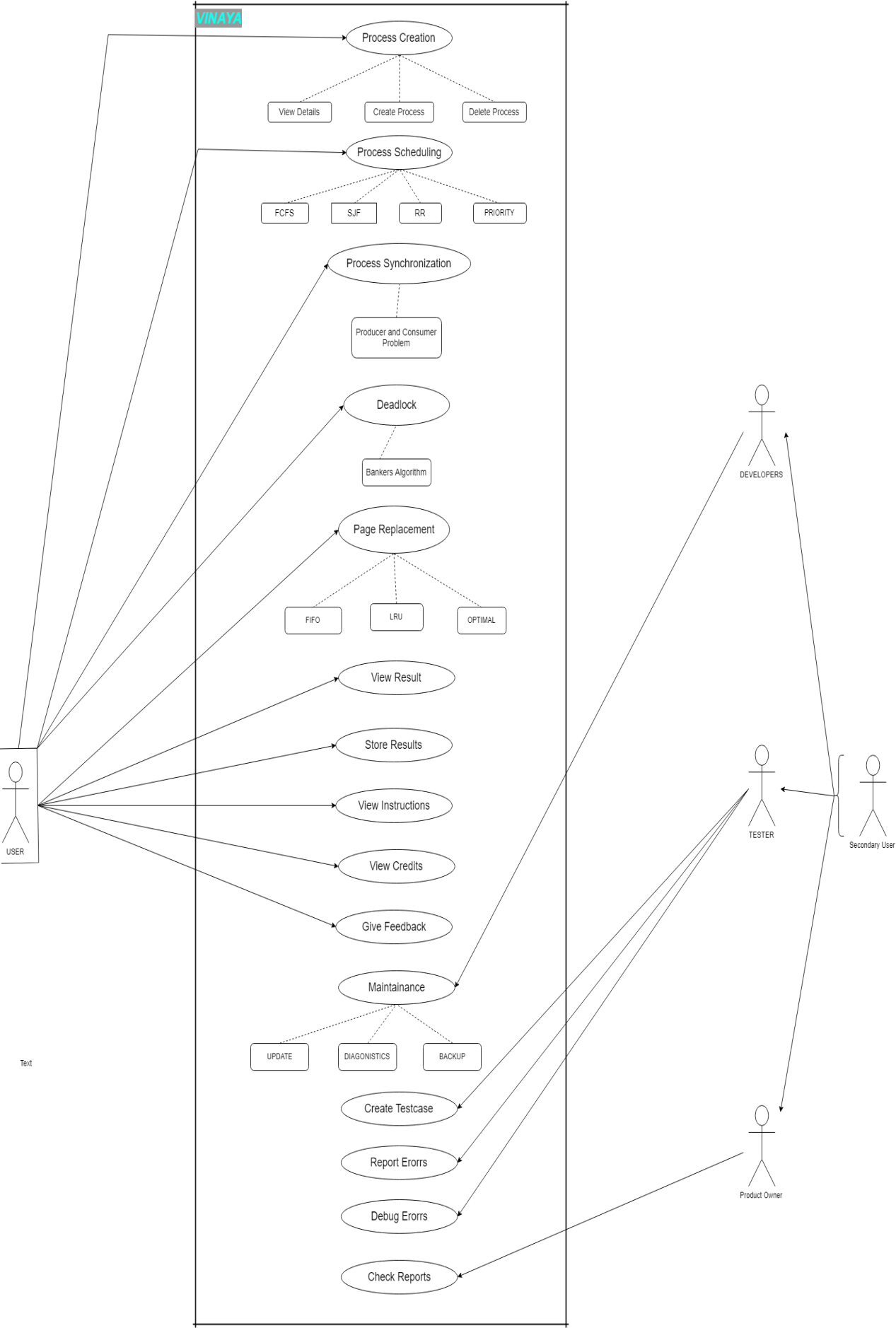
**DATA FLOW DIAGRAM**

**C:\Users\Ankita Singh\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Level 0 DFD.PNG**

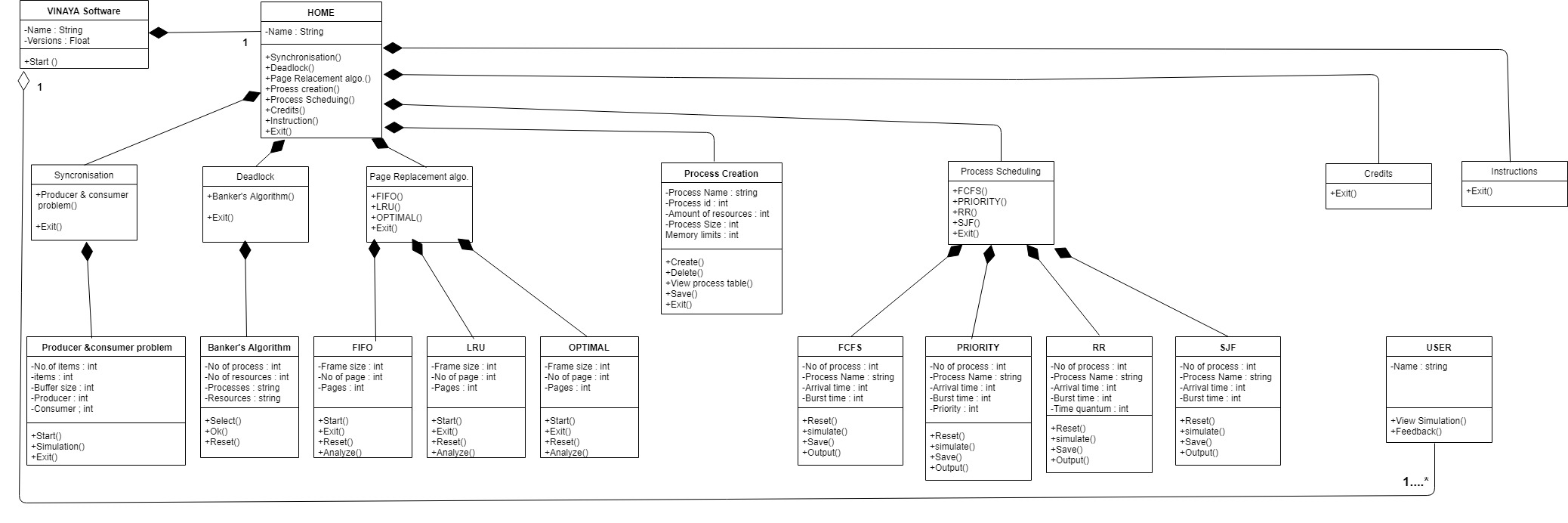
**C:\Users\Ankita Singh\AppData\Local\Microsoft\Windows\INetCache\Content.Word\Level 1 DFD.PNG**



**USE CASE DIAGRAM**

****

**CLASS DIAGRAM**

****

**SUB CHAPTER – 6**

**NON – FUNCTIONAL REQUIREMENTS**

**6.0 NON FUNCTIONAL REQUIREMENTS**

**6.1 PERFORMANCE RQUIREMENTS**

This software should perform ­the same way irrespective to its Operating System environments. Time taken for importing previously saved files and performing simulation should be negligible. The software should suffer from minimum to no crashes. The software must provide accurate calculation of results and proper visualization.

**6.2 SAFETY REQUIREMENTS**

The software available online for downloading should be free from any malware and should pose no threat to the system.

**6.3 SECURITY REQUIREMENTS**

As all the operations are to be done within a single system with no handling of personal information, security is not an issue for this software.

**6.4 QUALITY REQUIREMENTS**

Quality has a number of attributes. Some of the important attributes for this software are:

Portability: As this software is to work on multiple platforms, portability is an essential attribute and we ensure this by using JAVA as our programming language.

User Training: We assume that the user already has some previous experience in working with simulators. So the users will not need any specific training for using this software.

Testability: As a basic characteristic the software needs to be testable to ensure correctness. Manual testing with certain test cases and automated JUnit Testing shall be used.

**CHAPTER – 3**

**LIMITATIONS**

1. Process Scheduling consists mostly of non – pre-emptive algorithms.
2. It may schedule up to 10 processes.
3. Importing from Excel file is yet to be implemented
4. Producer Consumer Synchronization may only be shown by bounded buffer solution graphically..

**CHAPTER – 4**

**CONCLUSION**

After 3 months of hard work during this semester when we started the project we have gained a lot of knowledge and experience about many things like how to work in a team, How to work under pressure, etc. We also learn about many new things like how to use Netbeans, Brackets, how to make reports for the projects etc.

**CHAPTER – 5**

**BIBLIOGRAPHY**

**BIBLIOGRAPHY**

**BOOKS REFERRED:**

1. Abraham Silberschatz, Peter B. Galvin, Gerg Gagne (9th edition)

*Operating System Concepts*

Wiley Publication

1. Rajib Mall (5th Edition)

Fundamentals of Software Engineering

Eastern Economy Edition

**SITES REFERRED:**

<https://codingloverlavi.blogspot.com/search?q=gantt+chart+fcfs+java>

<http://codebyshaukat.blogspot.com/2014/12/bankers-algorithm-in-java.html>

<https://www.geeksforgeeks.org/java-lang-process-class-java/>

<https://www.lucidchart.com/pages/?a=1>

<https://logojoy.com/logo-maker/?company_name=Vinaya>

<https://www.draw.io>