FR. CONCEICAO RODRIGUES COLLEGE OF ENGINEERING.
Fr. Agnel Ashram, Bandstand, Bandra (W) Mumbai 400 050.

Lab Manual

Software Engineering

B.E. (Computer, Sem - V)

2022-23

By-

Dr.B. S. Daga
(Computer Engg.)

Index

#	Experiment Name	Aim	Date
1	**Implementation of software requirements specification document**	To implement software requirements specification document for a Course Scheduling System	04/08/2022
2	**Course Scheduling System using Agile Methodology using JIRA (SCRUM)**	Course Scheduling System using SCRUM method on JIRA Tool.	11/08/2022
3	**Course scheduling system using agile methodology using JIRA (Kanban)**	To calculate function point for Course Scheduling System	25/08/2022
4	**Function point calculation**	Implement Course Scheduling System using KANBAN method on JIRA Tool.	01/09/2022
5	**Cost estimation using COCOMO model**	To estimate project cost using COCOMO Model for Course Scheduling System	08/09/2022
6	**Structured data flow analysis of CSS**	Develop diagrams for data flow analysis Course Scheduling System	15/09/2022
7	**Implementation of data flow design pattern**	Application & Analysis of data flow design patterns in the case study	22/09/2022
8	**Implementation of Object Oriented approach for understanding COHESION AND COUPLING**	Do design using OO approach and hence highlight Cohesion and Coupling in the design	29/09/2022
9.A	**Software testing**	To design test cases for performing black box testing & white box testing for Course Scheduling System	06/10/2022
9.B	**White box testing**	To design test cases for performing white box testing for Course Scheduling System	06/10/2022
10	**Version controlling & Risk Analysis of the project.**	1. Version controlling of the project using Git-Hub 2. Develop a risk table for a Course Scheduling System	13/10/2022

TECHNICAL DOCUMENT

SOFTWARE ENGINEERING
Lab Experiments
Group 21

9231	Sarah Abraham
9193	Ivan Dsilva
9211	Malaika Monteiro
9185	Kris Corriea

Department of Computer Engineering

Academic Term: First Term 2022-23

Class: T.E /Computer Sem – V / Software Engineering

Practical No:	1
Title:	**Software Requirement Specification**
Date of Performance:	**02/8/2022**
Roll No:	9185, 9193, 9211, 9231
Team Members:	Kris Correia, Ivan Dsilva, Malaika Monteiro, Sarah Abraham

Rubrics for Evaluation:

Sr. No	Performance Indicator	Excellent	Good	Below Average	Total Score
1	On time Completion & Submission (01)	01 (On Time)	NA	00 (Not on Time)	
2	Theory Understanding(02)	02(Correct)	NA	01 (Tried)	
3	Result Accuracy (03)	03(All used)	02 (Partial)	01 (rarely followed)	
4	Post Lab Questions (04)	04(done well)	3 (Partially Correct)	2(submitted)	

Signature of the Teacher:

EXPERIMENT NO. 1

TITLE : System Requirement Specification Document for Course Scheduling System

1. Introduction :

1.1 Purpose :

The purpose of this document is to present a detailed description of the course scheduling system. It will explain the purpose and features of the system, the interfaces of the system will do, the constraints under which it must operate and how the system will react to external stimuli. This document is intended for both stakeholders and developers of the system.

1.2 Scope :

It has many effective and practical uses across a wide range of institutes providing an effective course scheduling system.

1.3 Definitions and abbreviations :

SIS : Student Information System.
AIS : Academic Information System.
CSS : Course Scheduling System.

1.4 References :

INTERNET, TAS,IBM REQUISITEPRO,INSTRUCTOR.

1.5 Overview :

The next chapter, the Overall Description section, of this document gives an overview of the functionality of the product. It describes the informal requirements and is used to establish a context for the technical requirements specification in the next chapter.
The third chapter, Requirements Specification section, of this document is written primarily for the developers and describes in technical terms the details of the functionality of the product.
Both sections of the document describe the same software product in its entirety, but are intended for different audiences and thus use different Language.

2. Overall Description :

2.1 Product Perspective :

The system will operate within the university environment. This environment will interact with the course scheduling system so we need interfaces between these systems .

2.2 Product Functions :

2.2.1 The system shall be able to take capacity of rooms, availability of rooms, time slots during the days of the week, courses and number of students enrolled for each course

2.2.2 After taking the above input the Course Scheduling System will generate a time table that will be referred by the faculty and students

2.3 User Characteristics :

The student expected to be Internet literate Once he/she can log in the system and navigate between WebPages he/she can use basic functionality of the system.
Instructor expected to be internet literate and t be able use more complex functionality of the system.

2.4 Constraints :

2.4.1 The system must run in the Windows operating system environment.

2.4.2 The system shall use a database for all data management tasks.

2.4.3 The system shall work based on XYZ-standard to keep copyright.

2.5 Assumptions and Dependencies :

The Course Scheduling System assumes that changes such as change in capacity of students enrolled for a course, unavailability of rooms due to construction, insufficient/replacement faculty will be handled by the Management.

The Course Scheduling System prepares a timetable excluding the accommodation of seminars, workshops, examinations etc.

2.6 Apportioning of Requirements :

The course scheduling system requires a back-end, which includes the data-base containing data about the room capacity, available time slots, students enrolled in course etc and the operations performed on that data to solve student queries. First, the database is built, followed by the programming of the course scheduling system. Lastly, the frontend i.e the user interface is built and then the entire system undergoes testing to test accuracy and efficiency.

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

It must interfaces icons or wizard

3.1.2 Hardware Interfaces

Its must be pc computer to link to course scheduling system

3.1.3 Software Interfaces

We must have a browser to show course
Scheduling system

3.2 Functional Requirements

3.2.1 Creating Courses

3.2.1.1 Integration with registration system: The system shall periodically upload the latest registrar's classes list to determine courses that are offered in the current semester.

3.2.1.2 The system shall generate course for each class that registered and determine the current set of students that enrolled in that Class

3.2.1.3 The system shall allow faculty to update course schedule

3.3 Performance Requirements :

3.3.1 Response Time :

Average response time shall be less than 2 seconds

3.3.2 Throughput :

The system shall accommodate 1000 users per minute.

3.3.3 Recovery Time:

In case of a system failure, redundant system shall resume operations within 30 seconds.
Average repair time shall be less than 1 hour.

3.3.4 Start-up/Shutdown Time :

The system shall be operational within 1 minute of starting-up.

3.3.5 Capacity :

The system accommodates 4000 concurrent users.

3.4 Software System Attributes :

3.4.1 Security

1. Firewall Protection: The course management software system shall run inside a firewall.

2. Support different roles: The system shall support different roles for users, such as Instructors, Students, and administrative staff, the user logged in with given role should only be allowed access consistent with that role. For example a student shall only be allowed to see the schedule, not to modify it.

3.4.2 Reliability :

The system shall not be down more than 2 times in a year.

3.4.3 Scalability:

Scaling the system to large number of users: large courses will have hundreds of students

4. Change Management Process :

Change in user requirements such as user-interface requires students to log in with their university assigned email-ids to access the schedule instead of Roll. No will be first include, updating the SRS to highlight the changes, followed by implementation while ensuring reusability of pre-existing components and efficiency in development of the scheduling system.

5. Supporting Information :

Index

Department of Computer Engineering

Academic Term: First Term 2022-23

Class: T.E /Computer Sem – V / Software Engineering

Practical No:	**2**
Title:	**Implement Course Scheduling System using SCRUM method on JIRA Too**
Date of Performance:	**09/8/2022**
Roll No:	9185, 9193, 9211, 9231
Team Members:	Kris Correia, Ivan Dsilva, Malaika Monteiro, Sarah Abraham

Rubrics for Evaluation:

Sr. No	Performance Indicator	Excellent	Good	Below Average	Total Score
1	On time Completion & Submission (01)	01 (On Time)	NA	00 (Not on Time)	
2	Theory Understanding(02)	02(Correct)	NA	01 (Tried)	
3	Result Accuracy (03)	03(All used)	02 (Partial)	01 (rarely followed)	
4	Post Lab Questions (04)	04(done well)	3 (Partially Correct)	2(submitted)	

Signature of the Teacher:

EXPERIMENT NO. 2

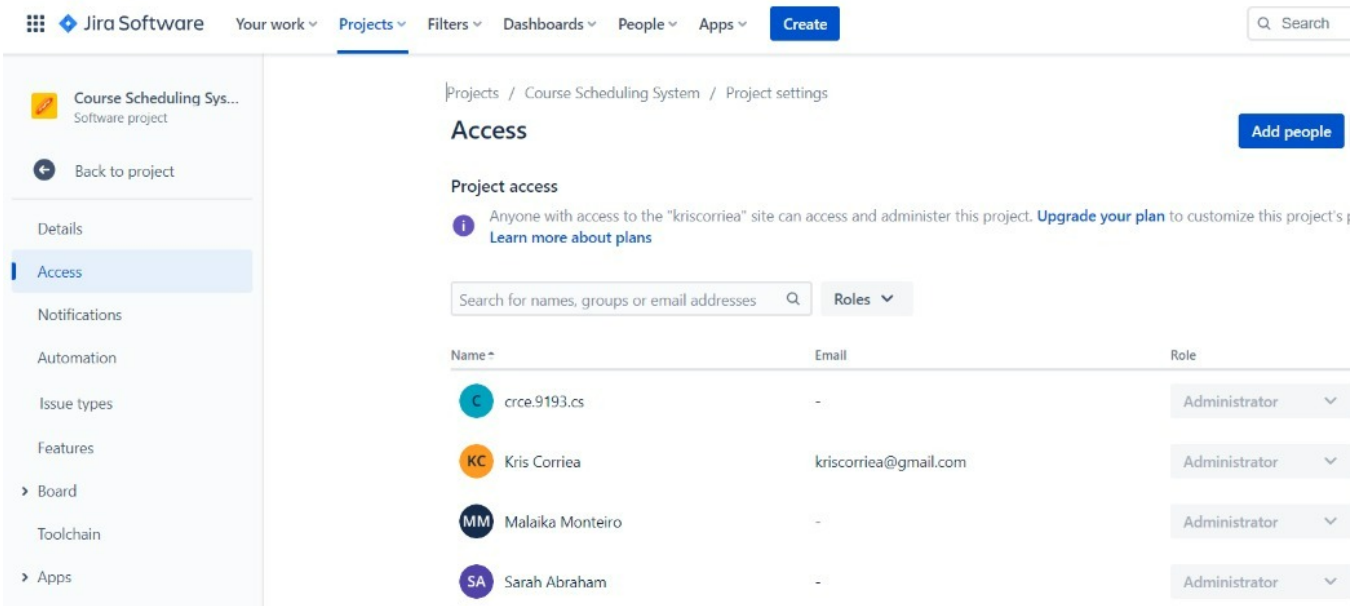
Course Scheduling System using Agile Methodology using JIRA

Aim

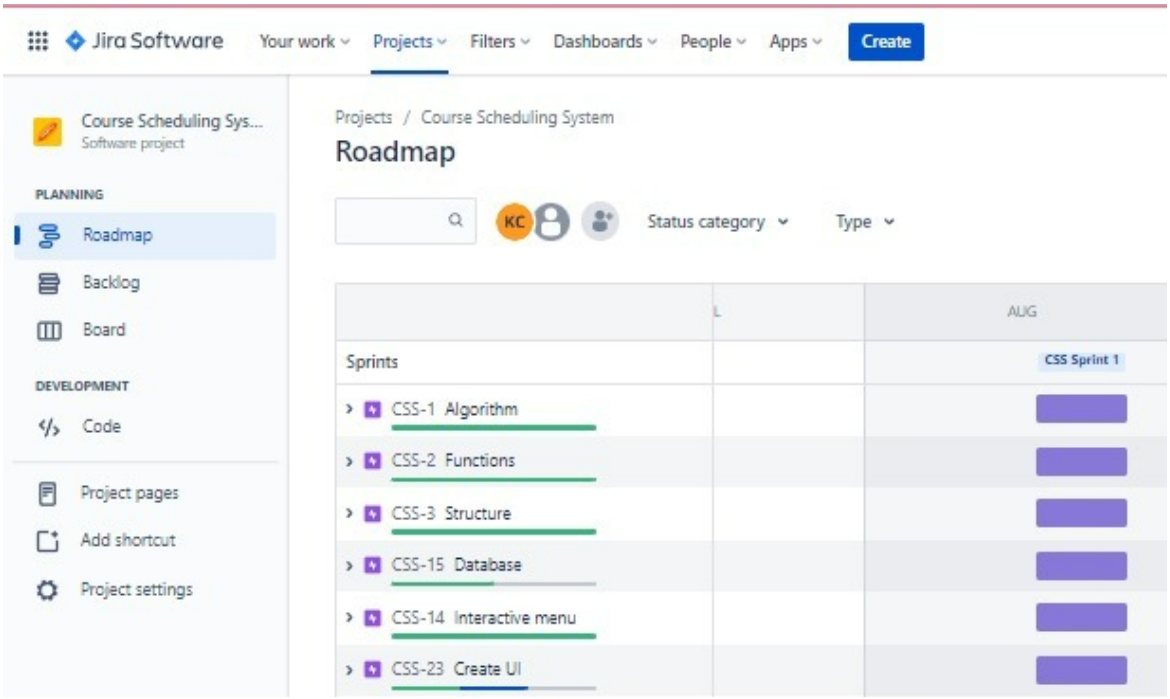
Implement Course Scheduling System using SCRUM method on JIRA Tool.

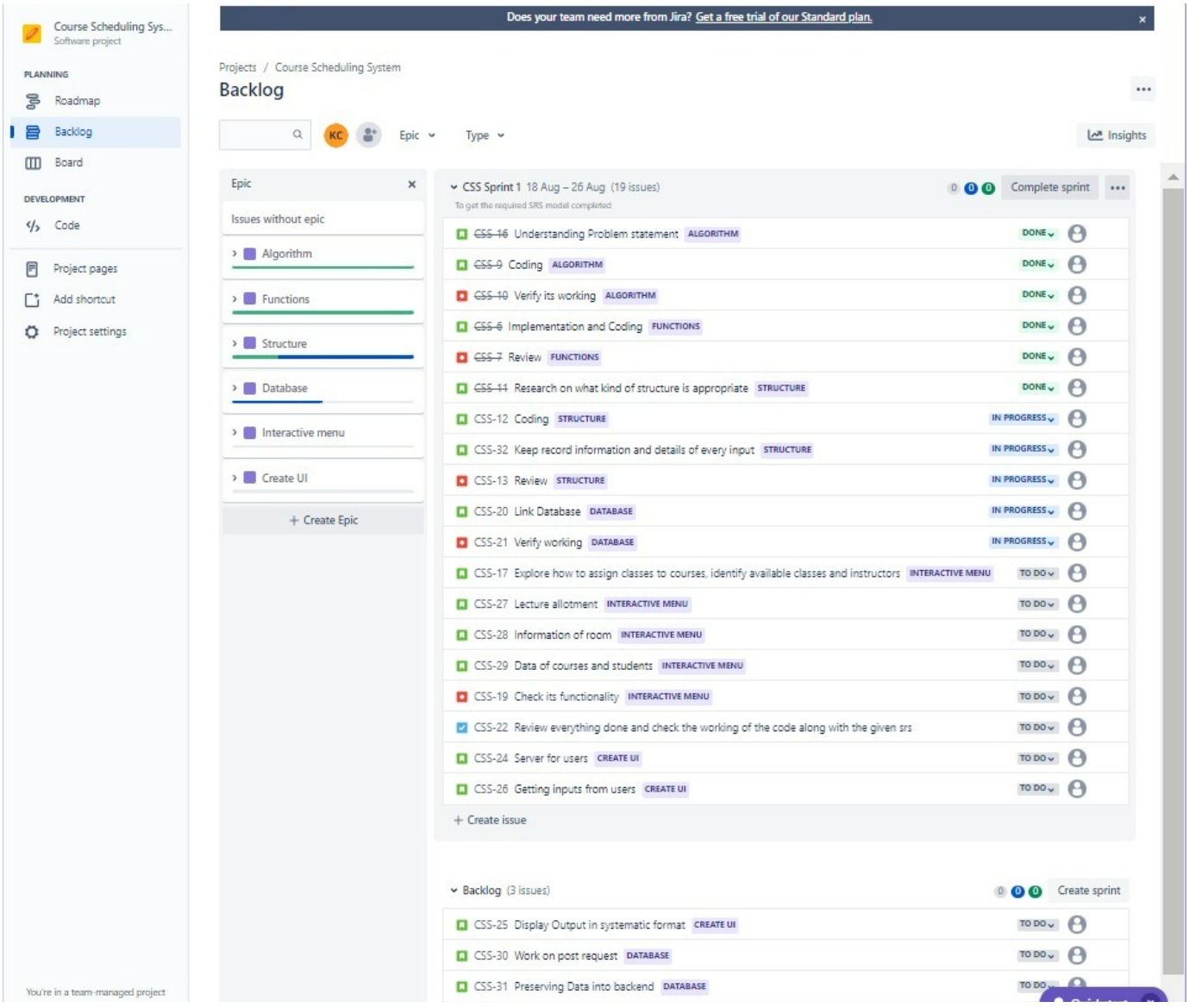
1. **Identify at least 5 Epics & 15 user stories from the specification document .Use your own research & interpretation . Link above stories to Epics.**
2. **Create Scrum project using free Jira account.**
3. **Enter backlog in Jira(Epics,stories & subtasks)**
4. **Start & complete one Sprint**
5. **Submit screen shots of Epics,stories, subtasks , backlog ,scrum board & release plan.**

Team:

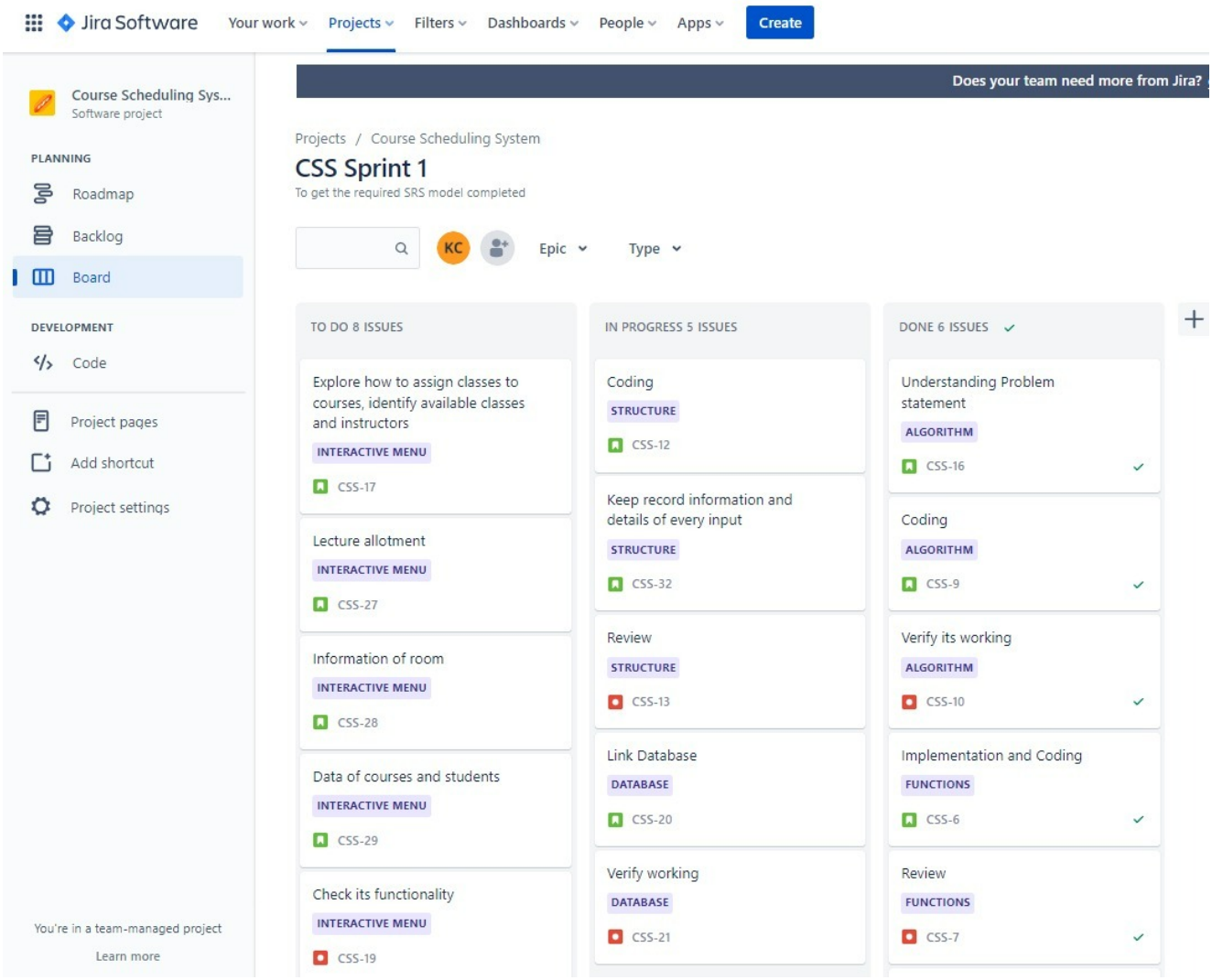


Epics and Stories -

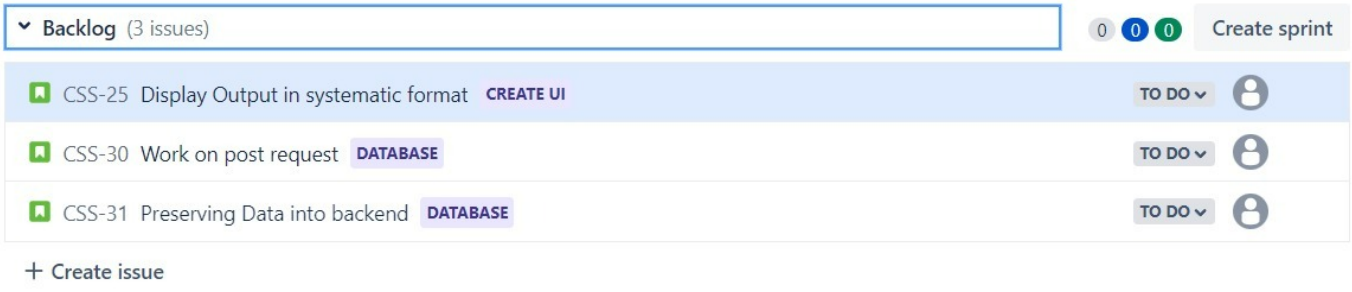




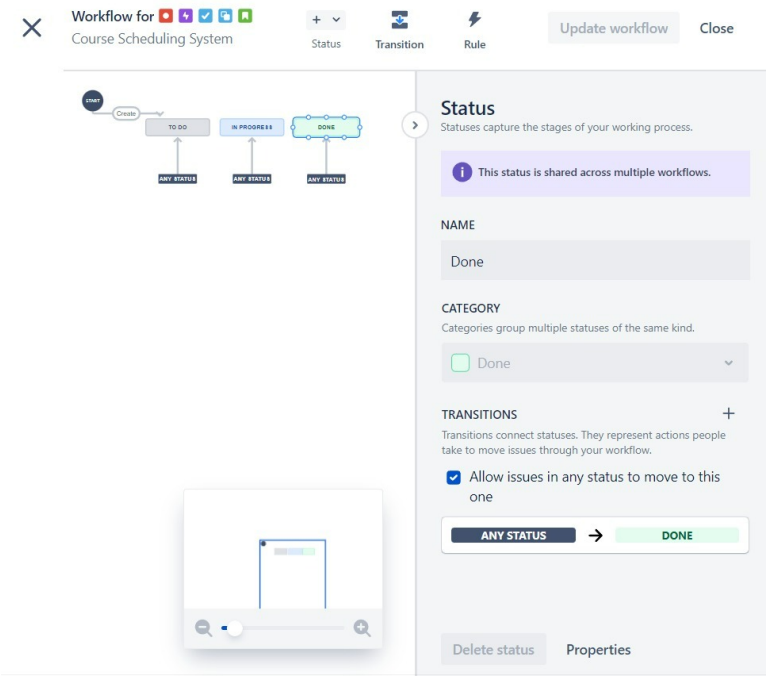
Scrum Board -



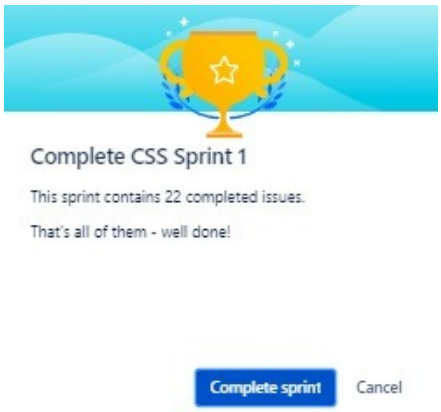
Backlog -



Workflow



Completion of Sprint



An SRS is basically an organization's understanding of a customer or potential client's system requirements and dependencies.

In SRS following aspects are addressed:
a) **Functionality**. What is the software supposed to do?

b) **External interfaces.** How does the software interact with people, the system's hardware, other hardware, and other software?

c) **Performance.** What is the speed, availability, response time, recovery time of various software functions, etc.?

d) **Attributes.** What are the portability, correctness, maintainability, security, etc. considerations?

e) **Design constraints** imposed on an implementation. Are there any required standards in effect, implementation language, policies for database integrity, resource limits, operating environment(s) etc.

Conclusion: Write the learning in details on Agile Process as compared to traditional development

Agile methodology is the most popular approach to project management.

Department of Computer Engineering

Academic Term: First Term 2022-23

Class: T.E /Computer Sem – V / Software Engineering

Practical No:	3
Title:	**Implement Course Scheduling System using KANBAN method on JIRA Too**
Date of Performance:	**30/8/2022**
Roll Nos:	9185, 9193, 9211, 9231
Team Members:	KRIS CORREIA, IVAN DSILVA, MALAIKA MONTEIRO, SARAH ABRAHAM

Rubrics for Evaluation:

Sr. No	Performance Indicator	Excellent	Good	Below Average	Total Score
1	On time Completion & Submission (01)	01 (On Time)	NA	00 (Not on Time)	
2	Theory Understanding(02)	02(Correct)	NA	01 (Tried)	
3	Result Accuracy (03)	03(All used)	02 (Partial)	01 (rarely followed)	
4	Post Lab Questions (04)	04(done well)	3 (Partially Correct)	2(submitted)	

Signature of the Teacher:

EXPERIMENT NO. 3

Course Scheduling System using Agile Methodology using JIRA

Aim

Implement Course Scheduling System using KANBAN method on JIRA Tool.

1. **Identify at least 5 Epics & 15 user stories from the specification document .Use your own research & interpretation . Link above stories to Epics.**
2. **Create Kanban project using free Jira account.**
3. **Enter backlog in Jira(Epics,stories & subtasks)**
4. **Submit screen shots of Epics,storie, subtasks , backlog ,kanban board & release plan.**

An SRS is basically an organization's understanding of a customer or potential client's system requirements and dependencies.

In SRS following aspects are addressed:
a) **Functionality**. What is the software supposed to do?

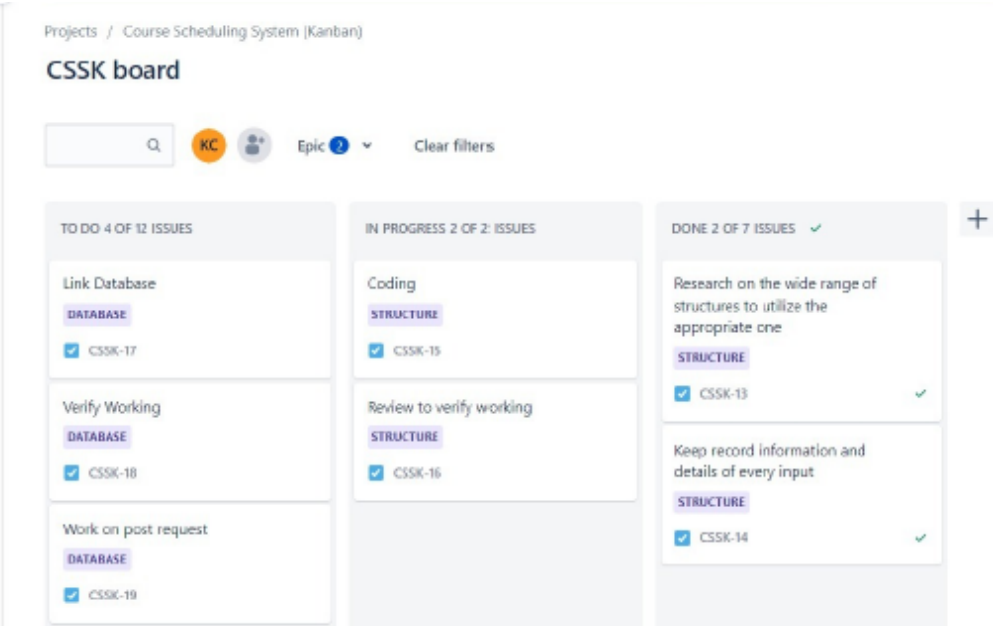
b) **External interfaces.** How does the software interact with people, the system's hardware, other hardware, and other software?

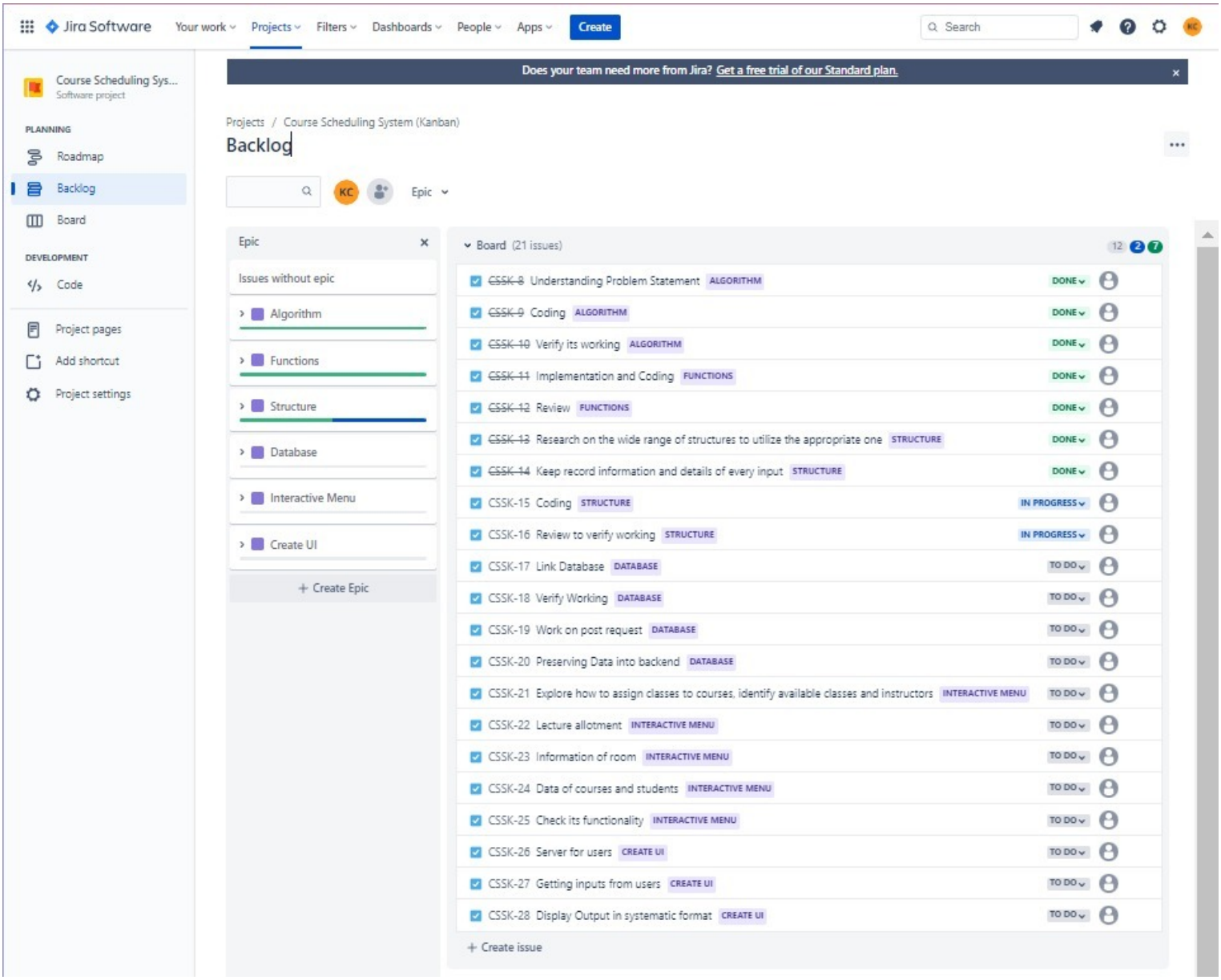
c) **Performance.** What is the speed, availability, response time, recovery time of various software functions, etc.?

d) **Attributes.** What are the portability, correctness, maintainability, security, etc. considerations?

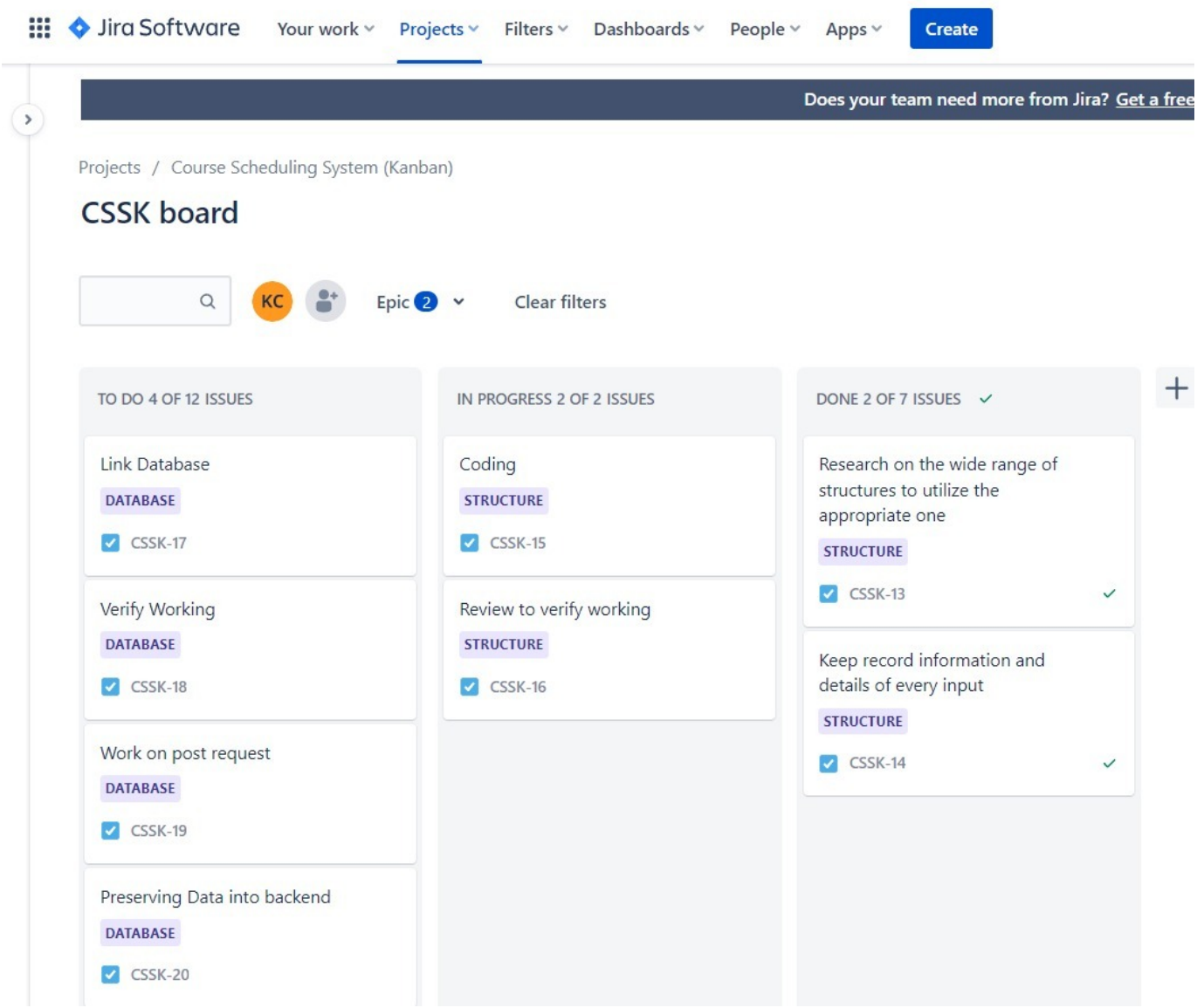
e) **Design constraints** imposed on an implementation. Are there any required standards in effect, implementation language, policies for database integrity, resource limits, operating environment(s) etc.

User Tasks -

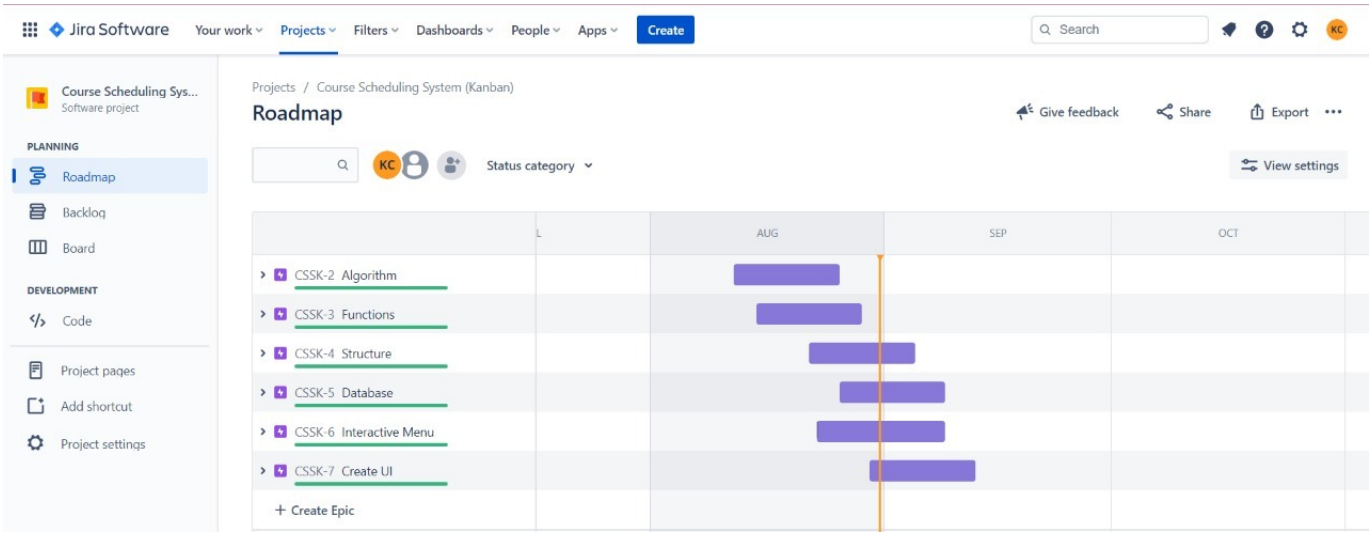



Jira Software
Your work
Projects
Filters
Dashboards
People
Apps
Create
Search
Course Scheduling Sys...
Software project
PLANNING
Roadmap
Backlog
Board
DEVELOPMENT
Code
Project pages
Add shortcut
Project settings
Does your team need more from Jira? Get a free trial of our Standard plan.
Projects / Course Scheduling System (Kanban)
Backlog
Epic
Issues without epic
Algorithm
Functions
Structure
Database
Interactive Menu
Create UI
Create Epic
Board (21 issues)
CSSK-8 Understanding Problem Statement ALGORITHM DONE
CSSK-9 Coding ALGORITHM DONE
CSSK-10 Verify its working ALGORITHM DONE
CSSK-11 Implementation and Coding FUNCTIONS DONE
CSSK-12 Review FUNCTIONS DONE
CSSK-13 Research on the wide range of structures to utilize the appropriate one STRUCTURE DONE
CSSK-14 Keep record information and details of every input STRUCTURE DONE
CSSK-15 Coding STRUCTURE IN PROGRESS
CSSK-16 Review to verify working STRUCTURE IN PROGRESS
CSSK-17 Link Database DATABASE TO DO
CSSK-18 Verify Working DATABASE TO DO
CSSK-19 Work on post request DATABASE TO DO
CSSK-20 Preserving Data into backend DATABASE TO DO
CSSK-21 Explore how to assign classes to courses, identify available classes and instructors INTERACTIVE MENU TO DO
CSSK-22 Lecture allotment INTERACTIVE MENU TO DO
CSSK-23 Information of room INTERACTIVE MENU TO DO
CSSK-24 Data of courses and students INTERACTIVE MENU TO DO
CSSK-25 Check its functionality INTERACTIVE MENU TO DO
CSSK-26 Server for users CREATE UI TO DO
CSSK-27 Getting inputs from users CREATE UI TO DO
CSSK-28 Display Output in systematic format CREATE UI TO DO
Create issue

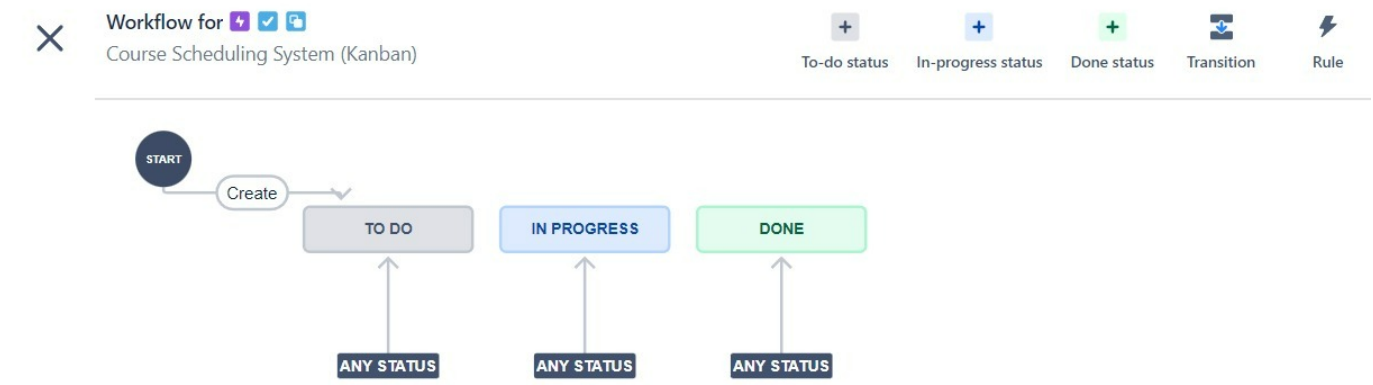
Kanban Board –



Roadmap –



Workflow-



Conclusion: Agile methodology is the most popular approach to project management which helps in managing tasks of projects according to their priorities.

Department of Computer Engineering

Academic Term: First Term 2022-23

Class: T.E /Computer Sem – V / Software Engineering

Practical No:	**4**
Title:	**To calculate function point for Course Scheduling System**
Date of Performance:	**13/9/2022**
Roll No:	9185, 9193, 9211, 9231
Team Members:	KRIS CORRIEA, IVAN DSILVA, MALAIKA MONTEIRO, SARAH ABRAHAM

Rubrics for Evaluation:

Sr. No	Performance Indicator	Excellent	Good	Below Average	Total Score
1	On time Completion & Submission (01)	01 (On Time)	NA	00 (Not on Time)	
2	Theory Understanding(02)	02(Correct)	NA	01 (Tried)	
3	Result Accuracy (03)	03(All used)	02 (Partial)	01 (rarely followed)	
4	Post Lab Questions (04)	04(done well)	3 (Partially Correct)	2(submitted)	

Signature of the Teacher:

EXPERIMENT NO. 4

FUNCTION POINT CALCULATION

Aim

To calculate function point for Course Scheduling System.

Description

The purpose of Function Points is to produce an estimate of software size from software requirements. Function Points are an indirect measure of software size based on external and internal application characteristics, as well as application performance. Function Points have a significant cost estimating relationship (CER) to software costs. Once determined, Function Points can be input into empirical statistical parametric software cost estimation equations and models in order to estimate software costs. Function Points are widely reported to be well suited for measuring the size of management information system (MIS), database intensive, and 4GL based application (e.g., software) system development.

Function Points are indirect quantitative measures of application software functionality and size that are based on objective counts of external application interfaces factored together with subjective counts of internal application complexity and overall performance characteristics. This procedure is composed of three logical divisions, determining the unadjusted function point count, value adjustment factor, and Function Points. Determining the unadjusted function point count consists of counting the number of external inputs, external outputs, external inquiries, internal logical files, and external interface files.

Determining the value adjustment factor consists of rating system, input and output, and application complexity. Determining Function Points consists of factoring unadjusted function points and value adjustment factor together. Function Points have two distinct purposes. The first purpose is to act as a basis for software measurement, comparison, and analysis (e.g., a software metrics approach). The second, and more important purpose, is to determine software size for input into software cost estimation models (e.g., equations) and tools that output effort (e.g., staff hours), which are based on empirical cost estimating relationships (CERs) between Function Points and effort.

1.1 Determine Unadjusted Function Point Count

No.	Measurement Parameter	Count		Low	Average	High		Total
1	External Inputs	2	x	3	**4**	6	=	**8**
2	External Outputs	1	x	4	**5**	7	=	**5**
3	External Inquires	0	x	3	**4**	6	=	**0**
4	Internal Logical Files	2	x	7	**10**	15	=	**20**
5	External Logical Files	0	x	5	**7**	10	=	**0**

Unadjusted Function Point Total ----------→ 33

1.2 Determine Value Adjustment Factor

Rate Each Factor: (0 – No Influence, 1 – Incidental, 2 – Moderate, 3 – Average, 4 – Significant, 5 - Essential)

1.How many data communication facilities are there? 5

2. How are distributed data and processing functions handled? 2

3. Was response time or throughput required by the user? 0

4. How heavily used is the current hardware platform? 0

5. How frequently are transactions executed? 0

6. What percentage of the information is entered online? 0

7. Was the application designed for end-user efficiency? 5

8. How many internal logical files are updated by on-line transaction? 0

9. Does the application have extensive logical or math processing? 5

10. Was the application developed to meet one or many user needs? 2

11. How difficult is conversion and installation? 0

12. How effective/automated are stamp, backup, and recovery? 4

13. Was the application designed for multiple sites/organizations? 1

14. Was the application designed to facilitate change? 5

Value Adjustment Factor ----------→ 29

1.3 Determine Function Points

Unadjusted Function Points x (0.65 + 0.01 x Value Adjustment Factor) ----------→ **31.02**

Conclusion

Our Course Scheduling System has 31.02 Function points.

Department of Computer Engineering

Academic Term: First Term 2022-23

Class: T.E /Computer Sem – V / Software Engineering

Practical No:	5
Title:	**To estimate project cost using COCOMO Model for Course Scheduling**
Date of Performance:	**20/9/2022**
Roll No:	9185, 9193, 9211, 9231
Team Members:	KRIS CORREIA, IVAN DSILVA, MALAIKA MONTEIRO, SARAH ABRAHAM

Rubrics for Evaluation:

Sr. No	Performance Indicator	Excellent	Good	Below Average	Total Score
1	On time Completion & Submission (01)	01 (On Time)	NA	00 (Not on Time)	
2	Theory Understanding(02)	02(Correct)	NA	01 (Tried)	
3	Result Accuracy (03)	03(All used)	02 (Partial)	01 (rarely followed)	
4	Post Lab Questions (04)	04(done well)	3 (Partially Correct)	2(submitted)	

Signature of the Teacher:

EXPERIMENT NO. 5

COST ESTIMATION USING COCOMO MODEL

Aim

To estimate project cost using COCOMO Model for <type your case study title here>.

Description

The **Constructive Cost Model** (**COCOMO**) is an algorithmic software cost estimation model developed by Barry Boehm. The model uses a basic regression formula, with parameters that are derived from historical project data and current project characteristics. COCOMO was first published in 1981 Barry W. Boehm's Book *Software engineering economics* as a model for estimating effort, cost, and schedule for software projects. It drew on a study of 63 projects at TRW Aerospace where Barry Boehm was Director of Software Research and Technology in 1981. The study examined projects ranging in size from 2,000 to 100,000 lines of code, and programming languages ranging from assembly to PL/I. These projects were based on the waterfall model of software development which was the prevalent software development process in 1981
COnstructive COst MOdel is static single-variable model. There is a hierarchy of these models.

Model 1:

Basic COCOMO model is static single-valued model that computes software development effort (and cost) as a function of program size expressed in estimated lines of code.

Model 2:

Intermediate COCOMO model computes software development effort as a function of program size and a set of "cost drivers" that include subjective assessments of product, hardware, personnel, and project attributes.

Model 3:

Advanced COCOMO model incorporates all characteristics of the intermediate version with an assessment of the cost driver's impact on each step, like analysis, design, etc.

COCOMO applies to three classes of software development mode:

- Organic - "small" teams with "good" experience working with "less than rigid" requirements
- Semi-detached - "medium" teams with mixed experience working with a mix of rigid and less than rigid requirements
- Embedded - developed within a set of "tight" constraints (hardware, software, operational, ...)

The basic COCOMO equations take the form

Effort Applied = $a_b(KLOC)^{b_b}$ **[person-months]**
Development Time = $c_b(\text{Effort Applied})^{d_b}$ **[months]**
People required = Effort Applied / Development Time **[count]**

Basic COCOMO is good for quick estimate of software costs. However it does not account for differences in hardware constraints, personnel quality and experience, use of modern tools and techniques, and so on.

Procedure:

step1 calculate the function point 74..92.
step2 use fp to calculate total number of lines of code for project
step3 calculating reation between loc and fp

loc =language factor*fp (language fatcor =30)
=30*31.02
=1706.1

For given project, set 'arbitrary size for every module and collectively form the LOC for the project.
For estimation we will go for **Basic COCOMO** model.
The table for constants for Basic COCOMO model is as follows:

Software Product	A_b	B_b	C_b	D_b
Organic	2.4	1.05	2.5	0.38
Semi- Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

For our given project phases/modules are there:

1) Screen Edit Size => 4k
2) Command Language interpreter => 2k
3) File input and output => 1k
4) Cursor movement. => 2k
5) Screen movement => 3k

Total size 12k

Now, we will go for overall estimation of project. If we analyzing our project, then we find that full Screen Editor is a Semi-detached project. So, for estimation purpose we will make use of those constants.

$E = A_b\ KLOC^{Bb}$

$D=C_bE^{Db}$

E = Effort applied in person months
D = Development time in chronological months

$E = 3.0 \text{ X } 1.7^{1.12}$
$= 5$ persons months

$D = 2.5 \text{ X } 5^{0.36}$
$= 4.46$ months

N =. E/D => Number of people
$= 5/4.46$
$= 1.12$
≈ 1
For completion of this project we will require 1 people.

Persons.Month in Basic Model:

Organic Mode	Semi-Detached Mode	Embedded Mode
4.19	5	6.8

Persons required in Intermediate:

Organic Mode	Semi-Detached Mode	Embedded Mode
1	1	1

Persons required in Advanced Model:

	RPD	DD	CUT	IT
Organic Mode	2	2	3	1
Semi-Detached Mode	2	3	4	1

EmbeddedMode	3	2	4	1

IMPLEMENTATION METHOD

1. *Define the classes, subsystems, interfaces of your system.*
2. *Decide the complexity of each of them while structuring the system.*
3. *Decide the technology that you are going to use for your system.*

4. *Decide the resource requirements & specify the job title of each of them.*
5. *If your project team members requires the training from outside then feed in the amount of efforts & cost for that.*

Conclusion

We required more people per month in the advanced model than the intermediate model and basic model

Department of Computer Engineering

Academic Term: First Term 2022-23

Class: T.E /Computer Sem – V / Software Engineering

Practical No:	**6**
Title:	**Develop diagrams for data flow analysis on Course Scheduling System**
Date of Performance:	**20/9/2022**
Roll No:	9185, 9193, 9211, 9231
Team Members:	KRIS CORREIA, IVAN DSILVA, MALAIKA MONTEIRO SARAH ABRAHAM

Rubrics for Evaluation:

Sr. No	Performance Indicator	Excellent	Good	Below Average	Total Score
1	On time Completion & Submission (01)	01 (On Time)	NA	00 (Not on Time)	
2	Theory Understanding(02)	02(Correct)	NA	01 (Tried)	
3	Result Accuracy (03)	03(All used)	02 (Partial)	01 (rarely followed)	
4	Post Lab Questions (04)	04(done well)	3 (Partially Correct)	2(submitted)	

Signature of the Teacher:

EXPERIMENT NO. 6

Structured data flow analysis of CSS

Aim

Develop diagrams for **data flow analysis** Course Scheduling System

Description

Data Flow Diagrams

A Data Flow Diagram (DFD) is a traditional visual representation of the information flows within a system. A neat and clear DFD can depict the right amount of the system requirement graphically. It can be manual, automated, or a combination of both.

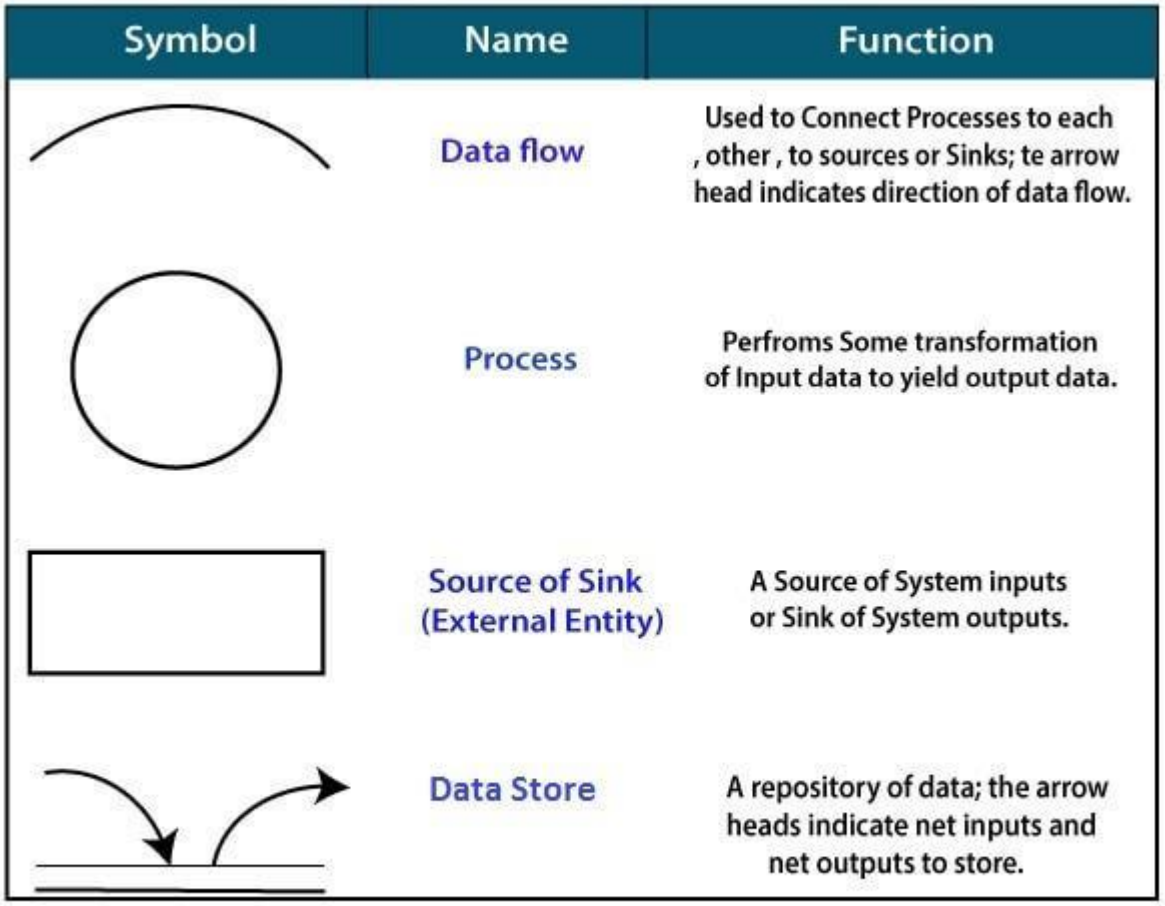
It shows how data enters and leaves the system, what changes the information, and where data is stored.

The objective of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communication tool between a system analyst and any person who plays a part in the order that acts as a starting point for redesigning a system. The DFD is also called as a data flow graph or bubble chart.

The following observations about DFDs are essential:

1. All names should be unique. This makes it easier to refer to elements in the DFD.
2. Remember that DFD is not a flow chart. Arrows is a flow chart that represents the order of events; arrows in DFD represents flowing data. A DFD does not involve any order of events.
3. Suppress logical decisions. If we ever have the urge to draw a diamond-shaped box in a DFD, suppress that urge! A diamond-shaped box is used in flow charts to represents decision points with multiple exists paths of which the only one is taken. This implies an ordering of events, which makes no sense in a DFD.
4. Do not become bogged down with details. Defer error conditions and error handling until the end of the analysis.

Standard symbols for DFDs are derived from the electric circuit diagram analysis and are shown in fig:



Symbols for Data Flow Diagrams

0-level DFDM

It is also known as fundamental system model, or **context diagram** represents the entire software requirement as a single bubble with input and output data denoted by incoming and outgoing arrows. Then the system is decomposed and described as a DFD with multiple bubbles. Parts of the system represented by each of these bubbles are then decomposed and documented as more and more detailed DFDs. This process may be repeated at as many levels as necessary until the program at hand is well understood. It is essential to preserve the number of inputs and outputs between levels, this concept is called leveling by DeMacro. Thus, if bubble "A" has two inputs x_1 and x_2 and one output y, then the expanded DFD, that represents "A" should have exactly two external inputs and one external output as shown in fig

The Level-0 DFD, also called context diagram of the result management system is shown in fig.

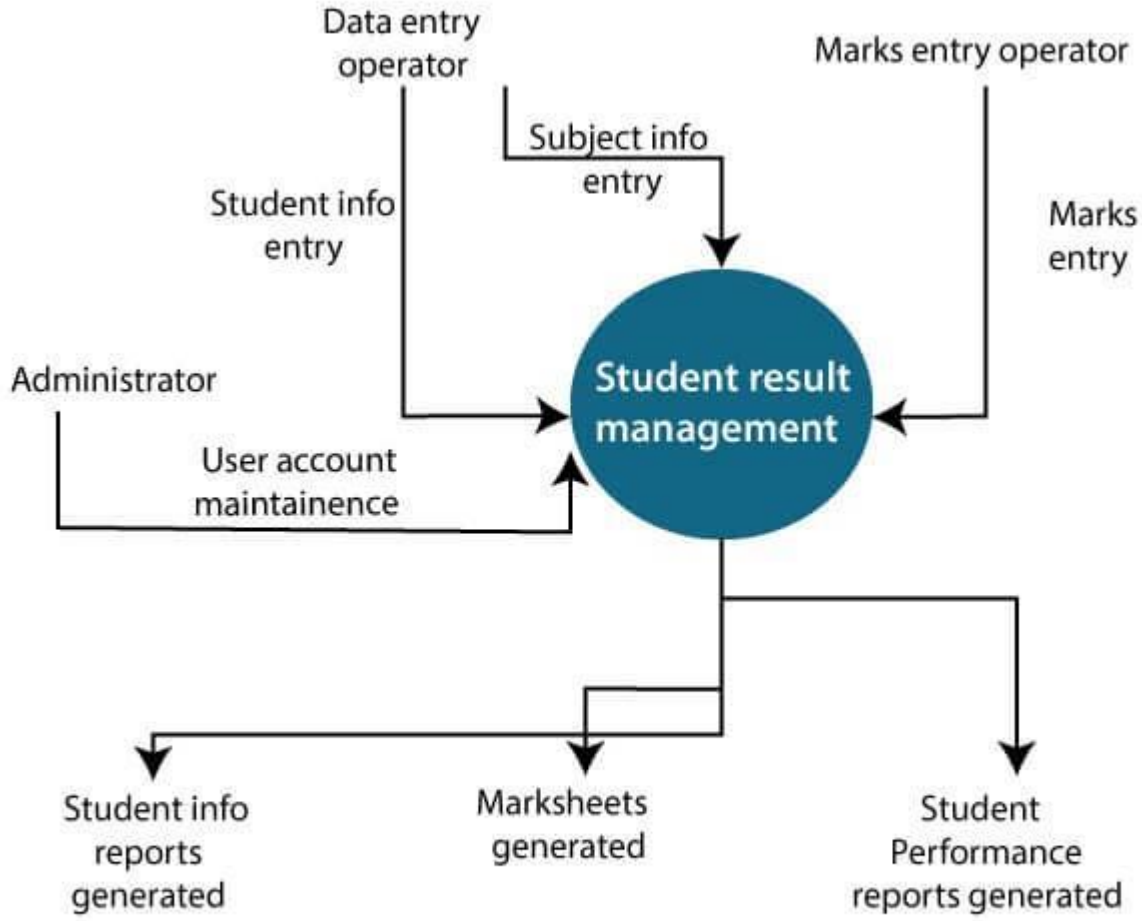
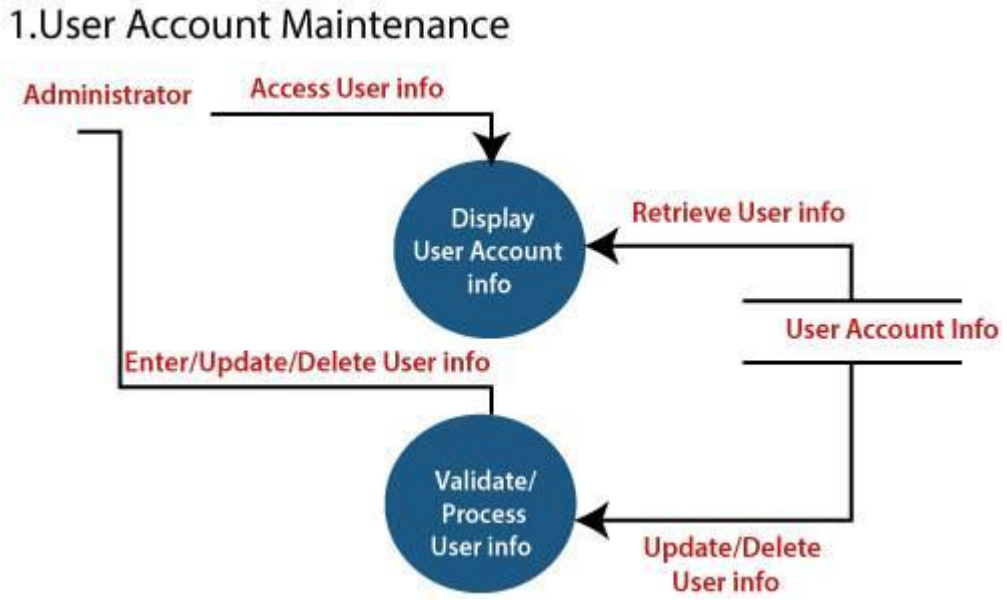


Fig: Level-0 DFD of result management system

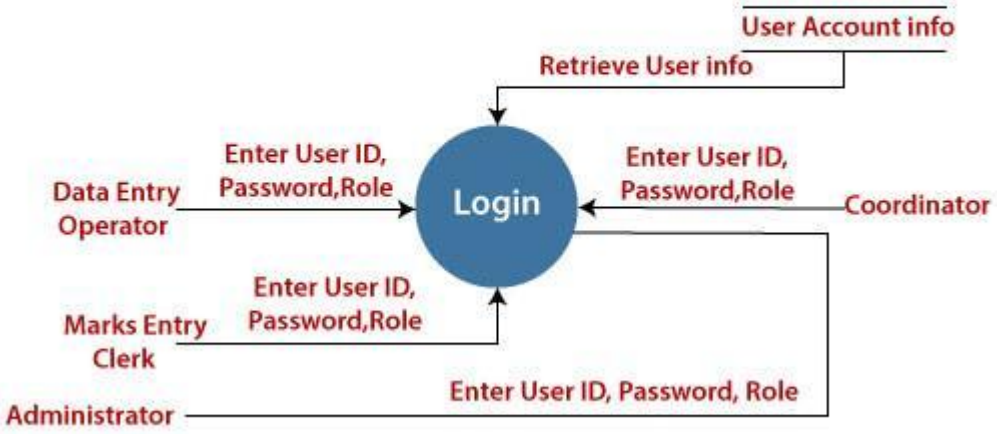
2-Level DFD

2-level DFD goes one process deeper into parts of 1-level DFD. It can be used to project or record the specific/necessary detail about the system's functioning.

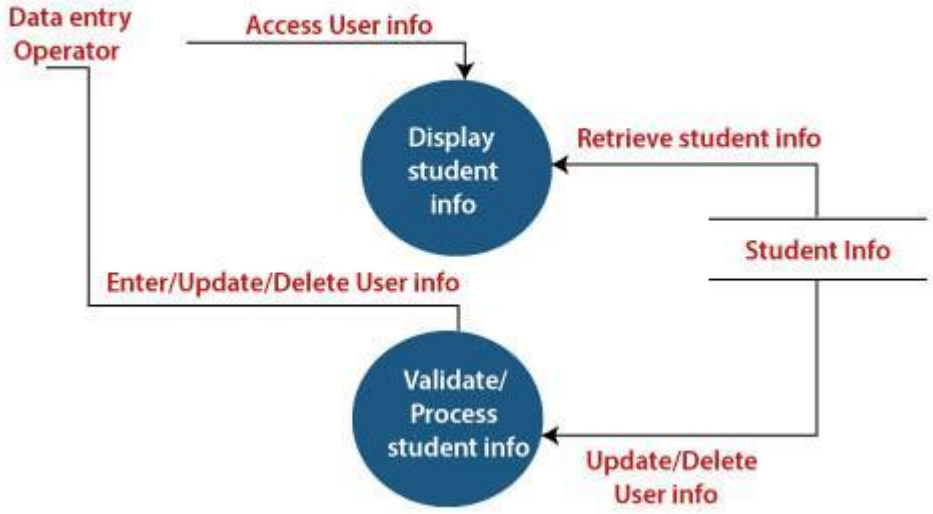


2. Login

The level 2 DFD of this process is given below:

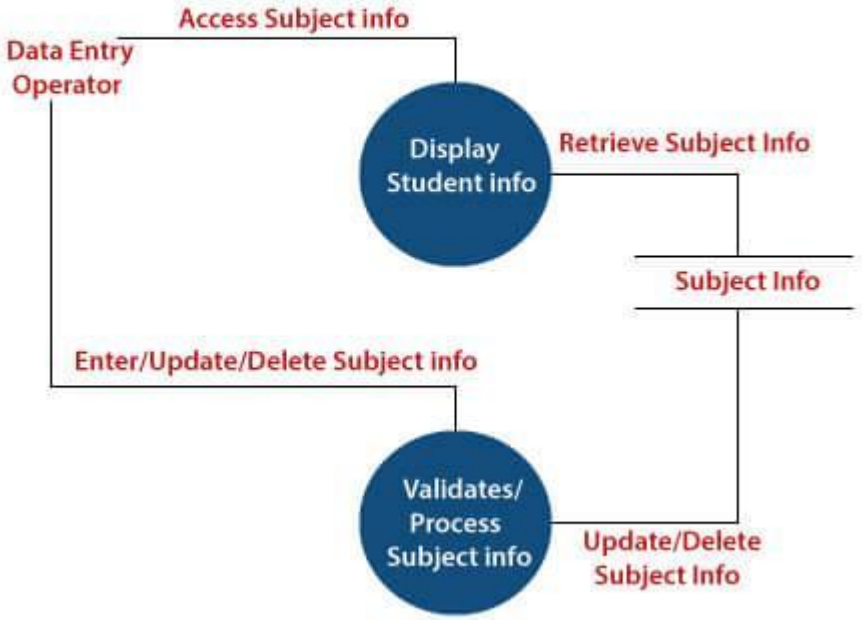


3. Student Information Management



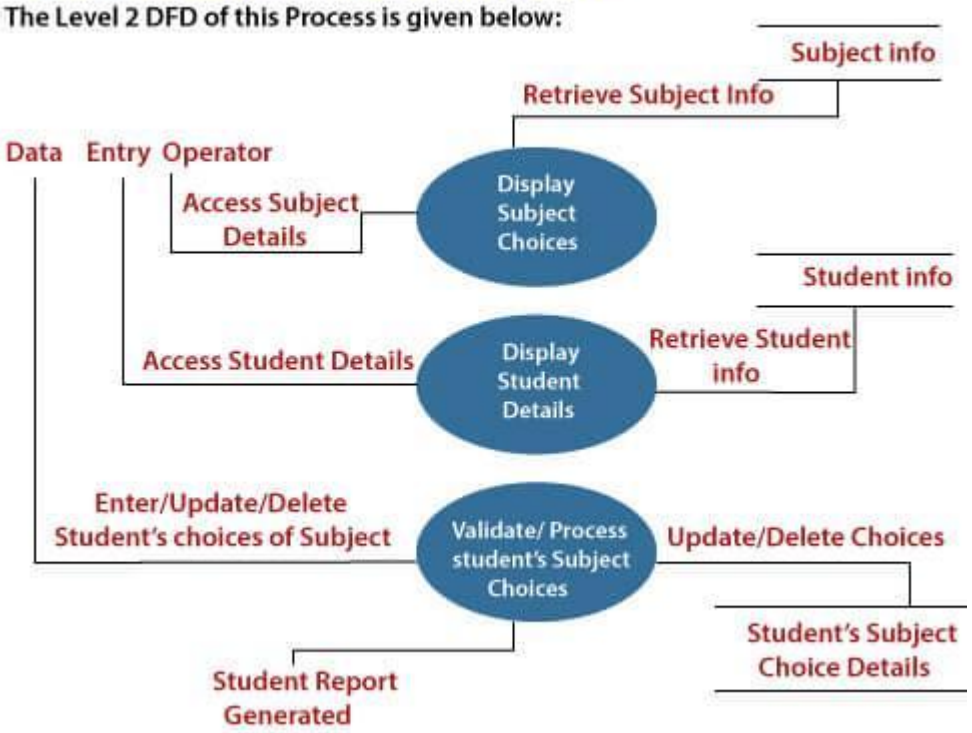
4. Subject Information Management

The level 2 DFD of this process is given below:



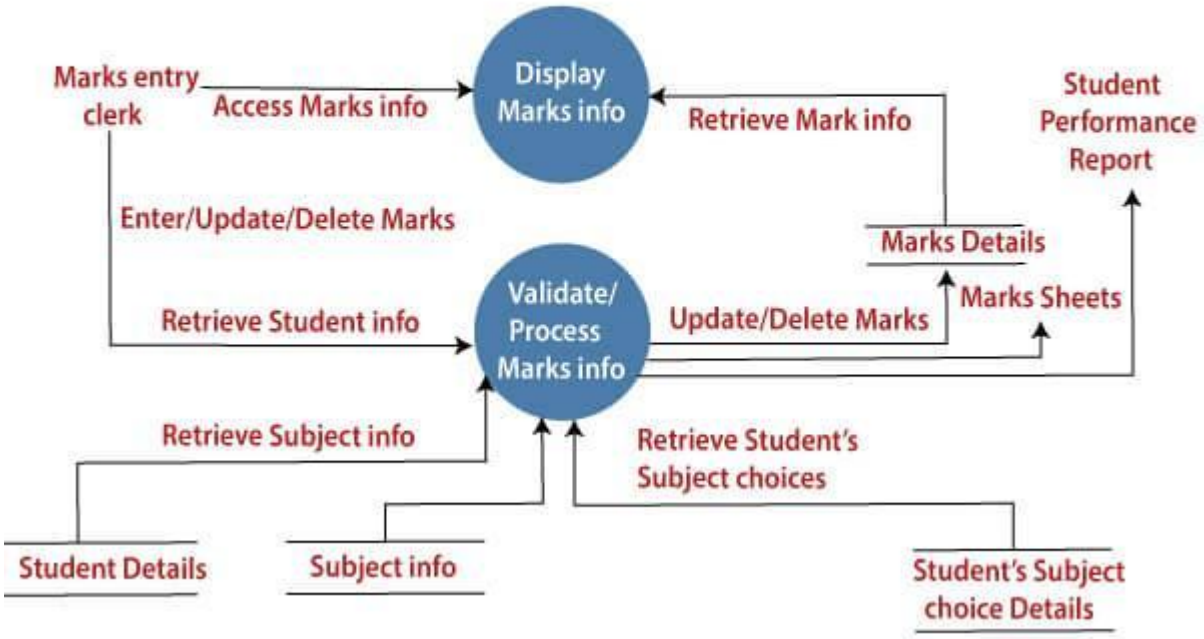
5. Student's Subject Choice Management

The Level 2 DFD of this Process is given below:

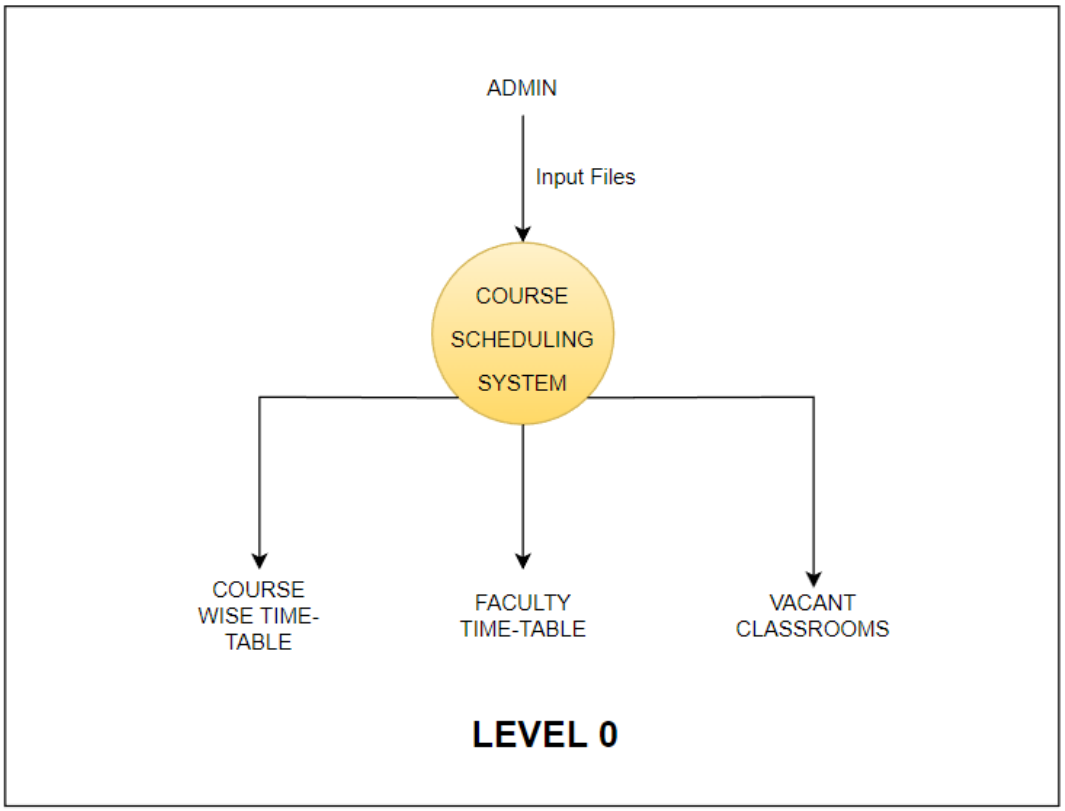


6. Marks Information Managment

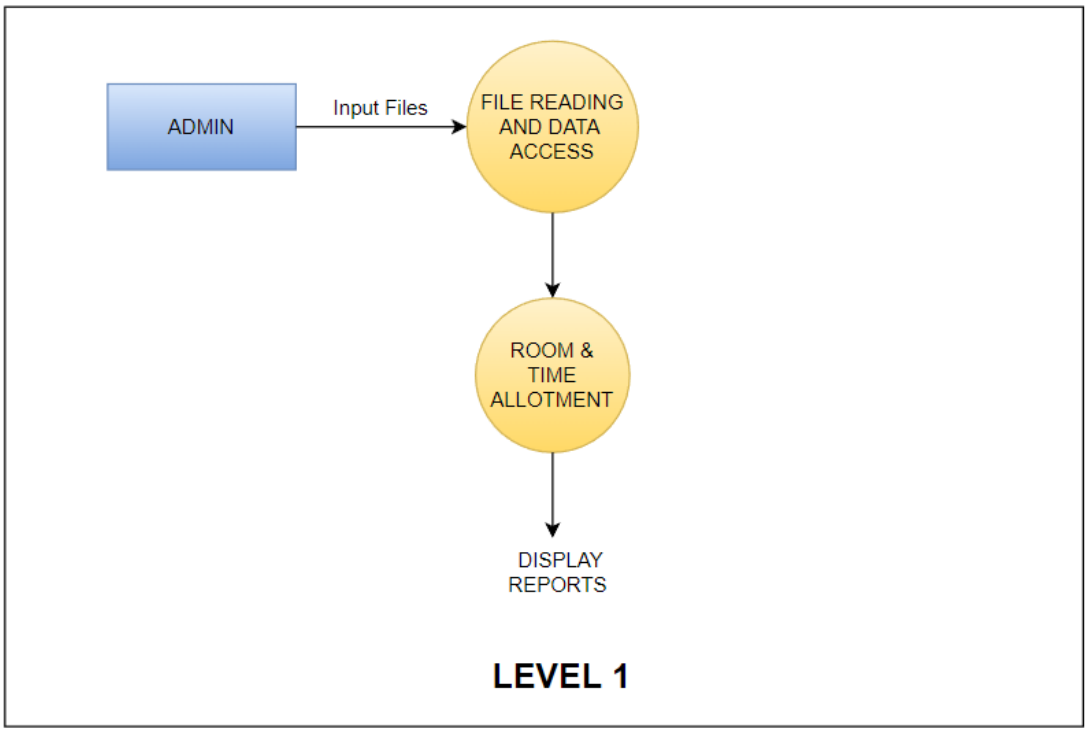
The Level 2 DFD of this Process is given below:



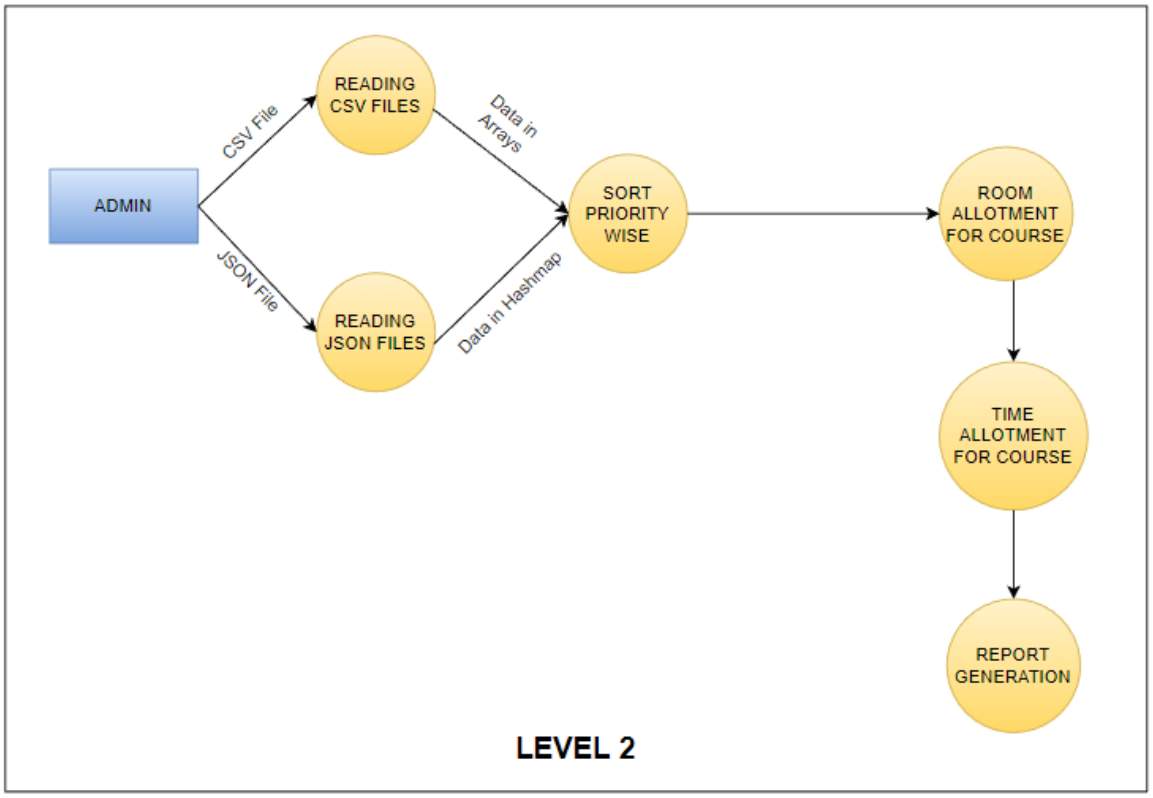
Level 0:



Level 1:



Level 2:



Conclusion : A diagrammatic representation of the Course Scheduling System which defines the scope and objectives clearly has been developed using Data Flow Diagrams.

Department of Computer Engineering

Academic Term: First Term 2022-23

Class: T.E /Computer Sem – V / Software Engineering

Practical No:	7
Title:	**Implementation of data flow design pattern**
Date of Performance:	**22/9/2022**
Roll No:	9185, 9193, 9211, 9231
Team Members:	KRIS CORRIEA, IVAN DSILVA, MALAIKA MONTEIRO, SARAH ABRAHAM

Rubrics for Evaluation:

Sr. No	Performance Indicator	Excellent	Good	Below Average	Total Score
1	On time Completion & Submission (01)	01 (On Time)	NA	00 (Not on Time)	
2	Theory Understanding(02)	02(Correct)	NA	01 (Tried)	
3	Result Accuracy (03)	03(All used)	02 (Partial)	01 (rarely followed)	
4	Post Lab Questions (04)	04(done well)	3 (Partially Correct)	2(submitted)	

Signature of the Teacher:

EXPERIMENT NO. 07

Implementation of data flow design pattern

Aim: Application & Analysis of data flow design patterns in the case study

Description : The aim of performing this experiment is to implement a set of particular design patterns in your project and show how your project adapts to that particular design pattern and show the changes that have been achieved by applying that particular design pattern to your project.

Design patterns are well-proved solution for solving the specific problem/task.

Data Flow Style

- Has the goal of modifiability
- Characterized by viewing the system as a series of transformations on successive pieces of input data
- Data enters the system and then flows through the components one at a time until they are assigned to output or a data store
- Batch sequential style
  - The processing steps are independent components
  - Each step runs to completion before the next step begins

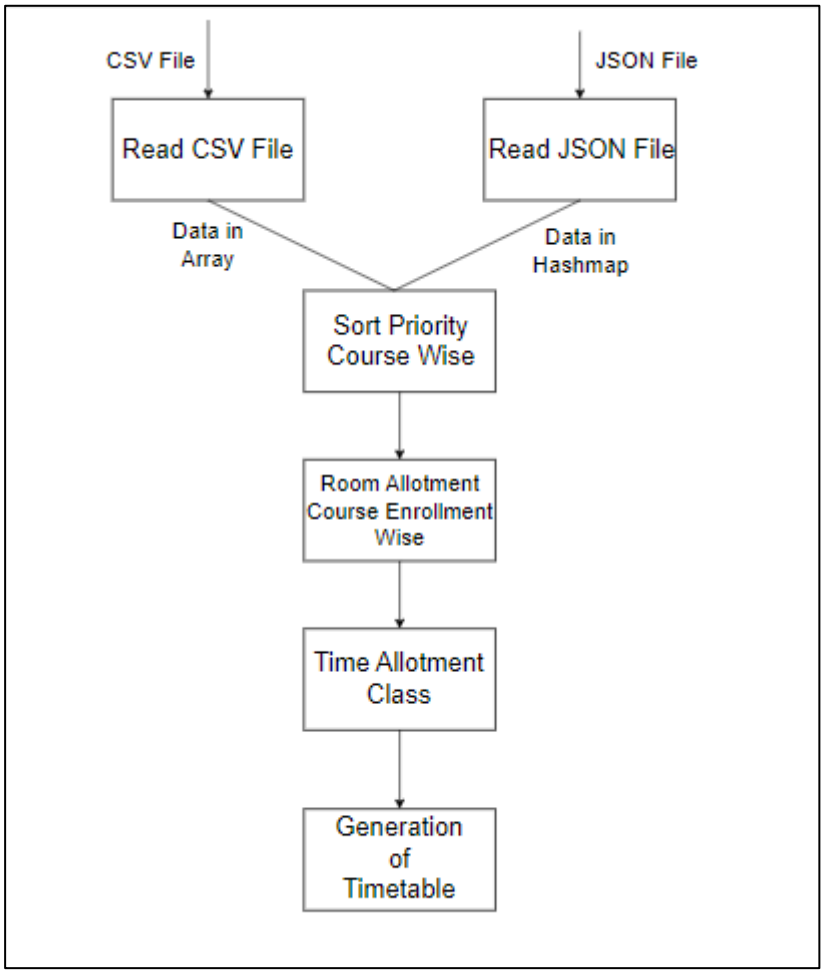
Pipe-and-filter style

  - Emphasizes the incremental transformation of data by successive components
  - The filters incrementally transform the data (entering and exiting via streams)
  - The filters use little contextual information and retain no state between instantiations
  - The pipes are stateless and simply exist to move data between filters
- Advantages
  - Has a simplistic design in which the components interact with the environment
  - Consists of no more and no less than the construction of its parts
  - Simplifies reuse and maintenance
  - Is easily made into a parallel or distributed execution in order to enhance system performance
- Disadvantages

- Implicitly encourages a batch mentality so interactive applications are difficult to create in this style
- Ordering of filters can be difficult to maintain so the filters cannot cooperatively interact to solve a problem
- Exhibits poor performance
    - Filters typically force the least common denominator of data representation (usually ASCII stream)
    - Filter may need unlimited buffers if they cannot start producing output until they receive all of the input

Each filter operates as a separate process or procedure call, thus incurring overhead in set-up and take-down time

Architecture Diagram:



Conclusion:

The architecture diagram represents the patterns implemented in the Course Scheduling System.

Department of Computer Engineering

Academic Term: First Term 2022-23

Class: T.E /Computer Sem – V / Software Engineering

Practical No:	**8**
Title:	**Do design using Object Oriented approach and hence highlight Cohesion and Coupling in the design**
Date of Performance:	**27/9/2022**
Roll No:	9185, 9193, 9211, 9231
Team Members:	KRIS CORREIA, IVAN DSILVA, MALAIKA MONTEIRO, SARAH ABRAHAM

Rubrics for Evaluation:

Sr. No	Performance Indicator	Excellent	Good	Below Average	Total Score
1	On time Completion & Submission (01)	01 (On Time)	NA	00 (Not on Time)	
2	Theory Understanding(02)	02(Correct)	NA	01 (Tried)	
3	Result Accuracy (03)	03(All used)	02 (Partial)	01 (rarely followed)	
4	Post Lab Questions (04)	04(done well)	3 (Partially Correct)	2(submitted)	

Signature of the Teacher:

EXPERIMENT NO. 8

Implementation of Object Oriented approach for understanding COHESION AND COUPLING

Aim

Do design using OO approach and hence highlight Cohesion and Coupling in the design.

Description

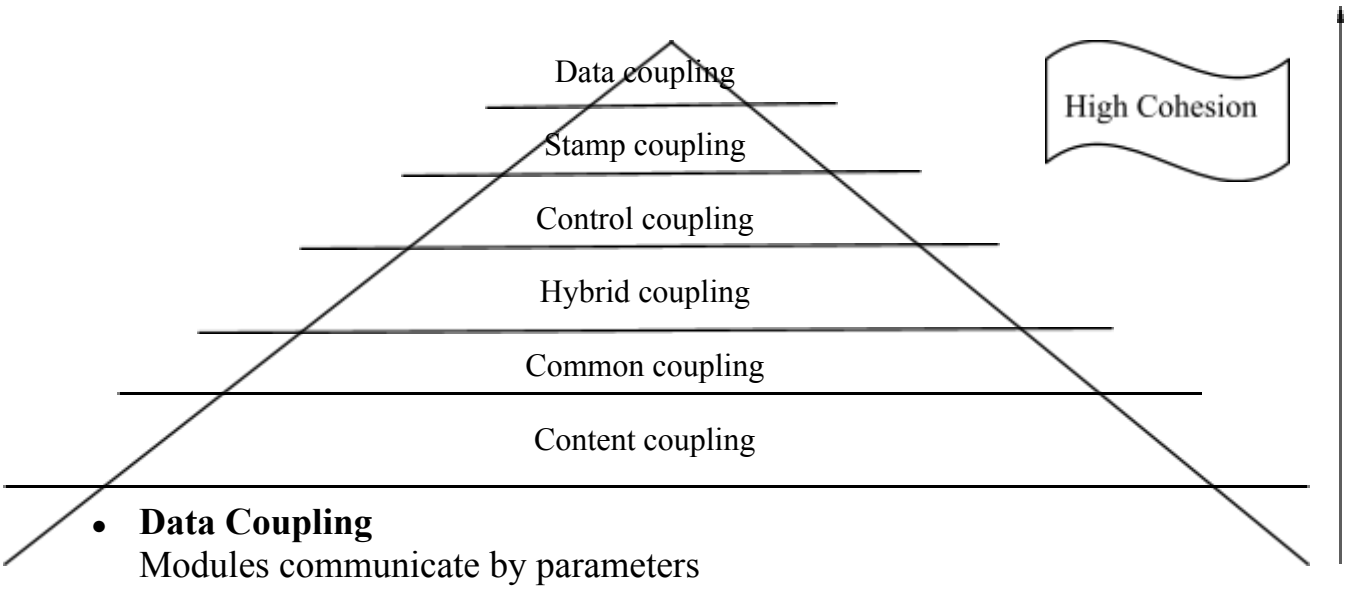
The aim of performing this experiment is to implementdata flow architecturein your project and show type of cohesion between operations and coupling between components in your project.

For Good project design, Cohesion should be high and coupling should be s lowas possible.

Coupling

- The degree of interdependence between two modules"
- We aim to minimize coupling - to make modules as independent as possible

Types of Coupling



- **Data Coupling**
  Modules communicate by parameters

- **Data coupling problems**

Too many parameters - makes the interface difficult to understand and possible error to occur
A composite data is passed between modules

- **Control coupling**
  A module controls the logic of another module through the parameter

- **Hybrid coupling**
  A subset of data used as control

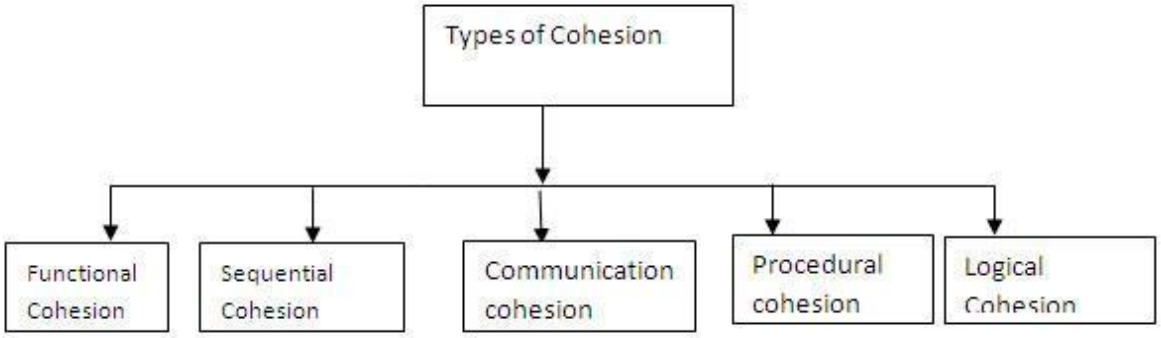
- **Common coupling**
  Use of global data as communication between modules

- **Content coupling**
  A module refers to the inside of another module

Cohesion

- "The measure of the strength of functional relatedness of elements within a module"
- Elements: instructions, groups of instructions, data definition, call of another module
- Strong cohesion will reduce relations between modules - minimize coupling

Types of Cohesion



Functional cohesion (Most Required)

- All elements contribute to the execution of one and only one problem-related task

Sequential cohesion

- Elements are involved in activities such that output data from one activity becomes input data to the next

Communicational Cohesion

- Elements contribute to activities that use the same input or output data

Procedural cohesion

- Elements are related only by sequence, otherwise the activities are unrelated

Temporal cohesion

- Elements are involved in activities that are related in time

Logical cohesion

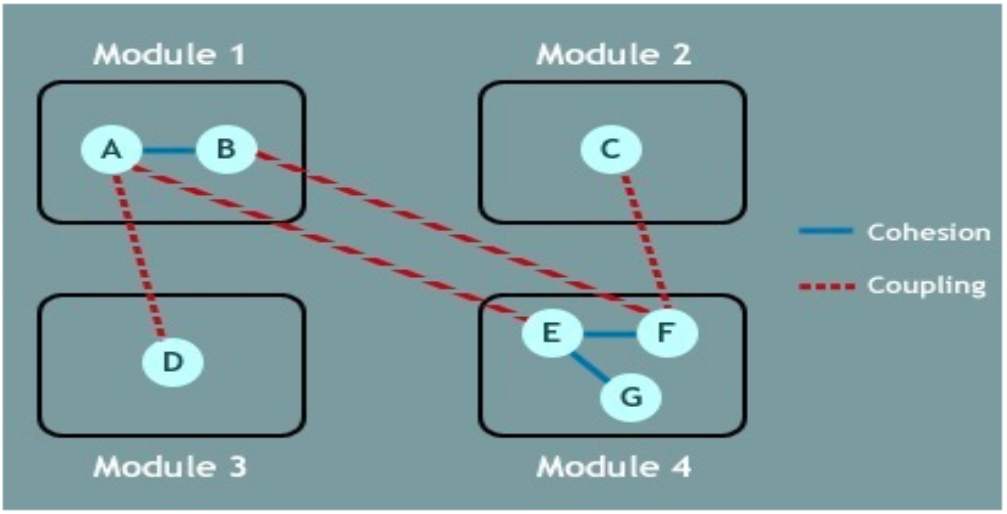
- Elements contribute to activities of the same general category

Coincidental cohesion(Least Required)

- Elements contribute to activities with no meaningful relationship to one another

1. Cohesion

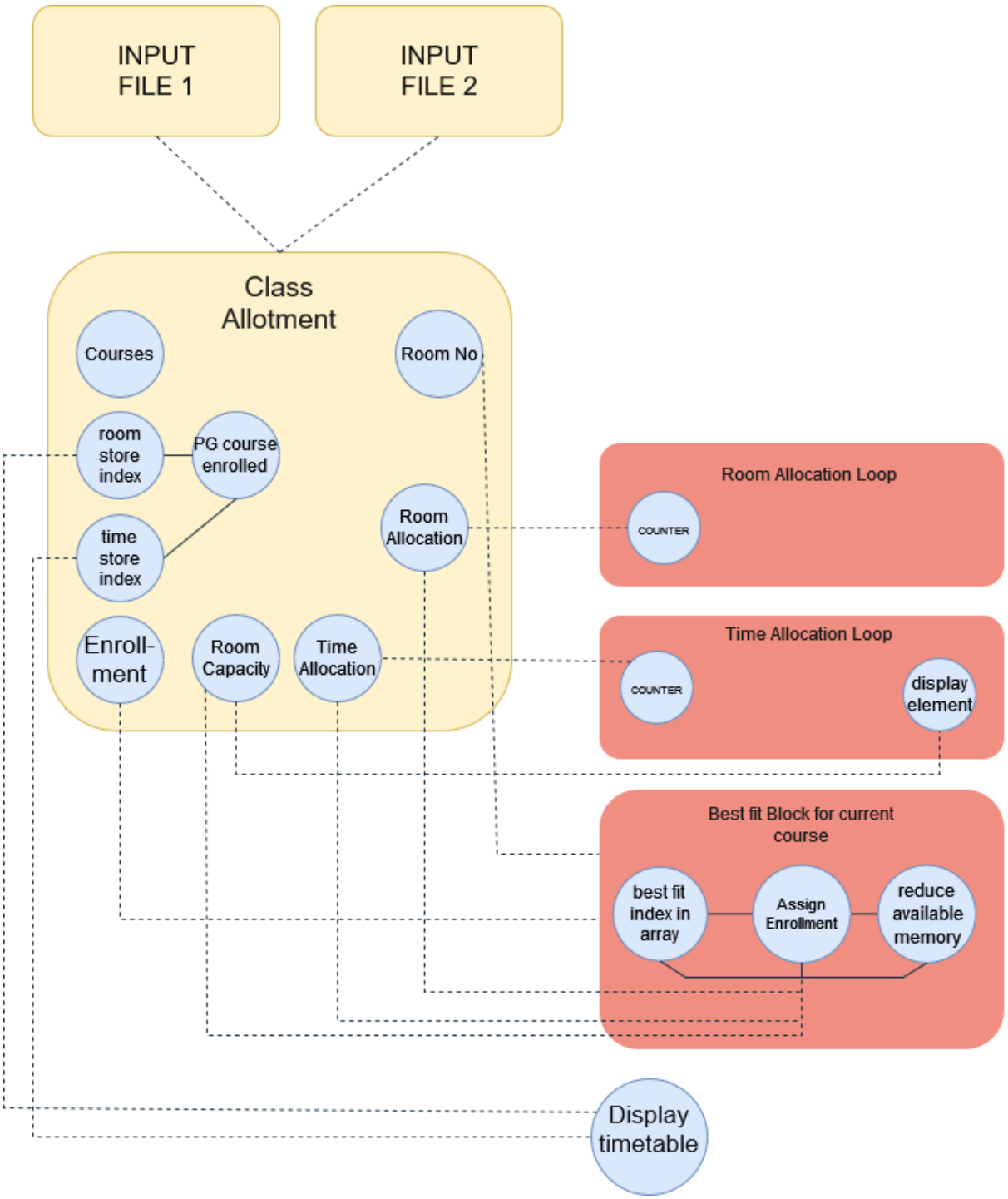
2. Coupling



Conclusion :

Implementation of data flow architecture in a course scheduling system that depicts cohesion and coupling.

Dotted Line - Coupling
Continuous Line - Cohesion



Three Modules : InputFile 1, InputFile 2 and Class Allotment Module
Sub-modules of Class Allotment Module :

- Room Allocation Loop
- Time Allocation Loop
- Best fit Block For Current Course

Department of Computer Engineering

Academic Term: First Term 2022-23

Class: T.E /Computer Sem – V / Software Engineering

Practical No:	**9-A**
Title:	**To design test cases for performing black box testing for Course Scheduling**
Date of Performance:	**27/9/2022**
Roll No:	9185, 9193, 9211, 9231
Team Members:	KRIS CORREIA, IVAN DSILVA, MALAIKA MONTEIRO, SARAH ABRAHAM

Rubrics for Evaluation:

Sr. No	Performance Indicator	Excellent	Good	Below Average	Total Score
1	On time Completion & Submission (01)	01 (On Time)	NA	00 (Not on Time)	
2	Theory Understanding(02)	02(Correct)	NA	01 (Tried)	
3	Result Accuracy (03)	03(All used)	02 (Partial)	01 (rarely followed)	
4	Post Lab Questions (04)	04(done well)	3 (Partially Correct)	2(submitted)	

Signature of the Teacher:

EXPERIMENT NO. 9-A

SOFTWARE TESTING

Aim

To design test cases for performing black box testing & white box testing for Course Scheduling system

Description

Black box testing (also called behavioral/functional testing), focuses on the functional requirements of the software. The test designer selects valid and invalid inputs and determines the correct output. There is no knowledge of the test object's internal structure. It is not an alternative to white box testing, and a complementary approach to white box testing. It uncovers different classes of errors than white box testing.

Black-box testing attempts to find errors in the following categories

- Incorrect/missing function.
- Interface errors.
- Errors in database/external database access.
- Behavior/performance errors.

Black Box testing techniques are:

1. Equivalence Partitioning

It is the black-box technique that divides the input domain into classes of data from which test cases can be derived. Equivalence partitioning defines test cases that uncover classes of errors thereby reducing the no. of test cases that must be developed. If the input condition specifies a range, one valid and two invalid equivalence classes are defined. If an input condition requires a specific value, one valid and two invalid equivalence classes are defined. If an input condition specifies a member of a set, one valid and one invalid equivalence class is defined. If an input condition is boolean, one valid and one invalid equivalence class is defined. By applying these guidelines, test cases for each input domain can be developed.

E.g. A program reads an input number in the range 1 and 5000 and computes the square of the input number. 3 Equivalence classes are :

1. Set of numbers less than 1.
2. Set of numbers between 1 and 5000.
3. Set of numbers greater than 5000.

A possible test case is {-10,100,7000}

2. Boundary Value Analysis

It focuses on the boundaries of the input domain rather than its center. The following guidelines can be used for performing boundary value analysis:

1. If the input condition specifies a range bounded by values a and b, test cases should include a and b, values just above and just below a and b
2. If an input condition specifies and number of values, test cases should exercise the minimum and maximum numbers, as well as values just above and just below the minimum and maximum values
3. Apply guidelines 1 and 2 to output conditions, test cases should be designed to produce the minimum and maximum output reports
4. If internal program data structures have boundaries (e.g. size limitations), be certain to test the boundaries

Test Cases:

Sr. No	Test Case	Test Steps	Expected Result	Actual Result	Pass/Fail
1.	Check whether wrong input of courses are scheduled	Add courses like CE1, CP09	Informs user that the input courses do not exist	Displays a message regarding wrong input	Pass
2	Check if non csv or json files can be given as input	Provide a non csv or json file as input	Gives exception and stops running	The process works as expected	Pass
3	Check whether user can mention an enrollment in a course more than the actual capacity of the course	Provide enrollment as 10000 for a particular course	Inform user that the number of enrollments are exceeding the number of enrollments that can actually be handled	The results are in sync with the expected result	Pass

4	Check if the program goes in an infinite loop on any type of input.	Use file where number of inputs is more than the capacity of array	Throw an error stating the number of inputs expected	Works as expected	Pass
4.	Check if data is being fetched correctly from database	Input data from user and store in database.	Correct data is passed to functions	Works as expected	Pass
5	Check the performance on large number of courses	Provide input with 30 courses	Work the same and schedule the 30 courses	Schedules all the courses	Pass
6	Check if PG courses are allotted before UG courses	Provide PG course without preference	Allot a classroom for PG course before UG course	Gives preference for PG course over UG course	Pass

Conclusion

Performed Black box testing of our Course Scheduling System to understand its functioning and how it behaves on different inputs.

Department of Computer Engineering

Academic Term: First Term 2022-23

Class: T.E /Computer Sem – V / Software Engineering

Practical No:	**9-B**
Title:	**To design test cases for performing white box testing for Course Scheduling**
Date of Performance:	**27/9/2022**
Roll No:	
Team Members:	

Rubrics for Evaluation:

Sr. No	Performance Indicator	Excellent	Good	Below Average	Total Score
1	On time Completion & Submission (01)	01 (On Time)	NA	00 (Not on Time)	
2	Theory Understanding(02)	02(Correct)	NA	01 (Tried)	
3	Result Accuracy (03)	03(All used)	02 (Partial)	01 (rarely followed)	
4	Post Lab Questions (04)	04(done well)	3 (Partially Correct)	2(submitted)	

Signature of the Teacher:

EXPERIMENT NO. 9-B

WHITE BOX TESTING

Aim

To design test cases for performing white box testing forCourse Scheduling system.

Description

White box testing, also called glass box testing, is a testing technique which exercises the internal logic of software components. Using white box testing, the software engineer can derive test cases that

1. Guarantee that all independent paths within a module have been exercised at least once.
2. Exercise all logical decisions on their true and false sides.
3. Execute all loops at their boundaries
4. Exercise internal data structures to ensure their validity.

Different types of White Box Testing techniques are:

1. Basis Path Testing

Basis path testing is a white-box testing technique first proposed by Tom McCabe. Basis path method enables to derive a logical complexity measure of a procedural design and use this measure as a guide for defining a basis set of execution paths. Test Cases derived to exercise the basis set are guaranteed to execute every statement in the program at least one time during testing. The flow graph depicts logical control flow within a program. The following steps constitute basis path testing.

Step1: Map flowchart into a corresponding flow graph

- Each circle, called a flow graph node, represents one or more procedural statements.
- A sequence of process boxes and a decision diamond can map into a single node.
- The arrows on the flow graph, called *edges* or *links,* represent flow of control and are analogous to flowchart arrows.
- Areas bounded by edges and nodes are called regions.

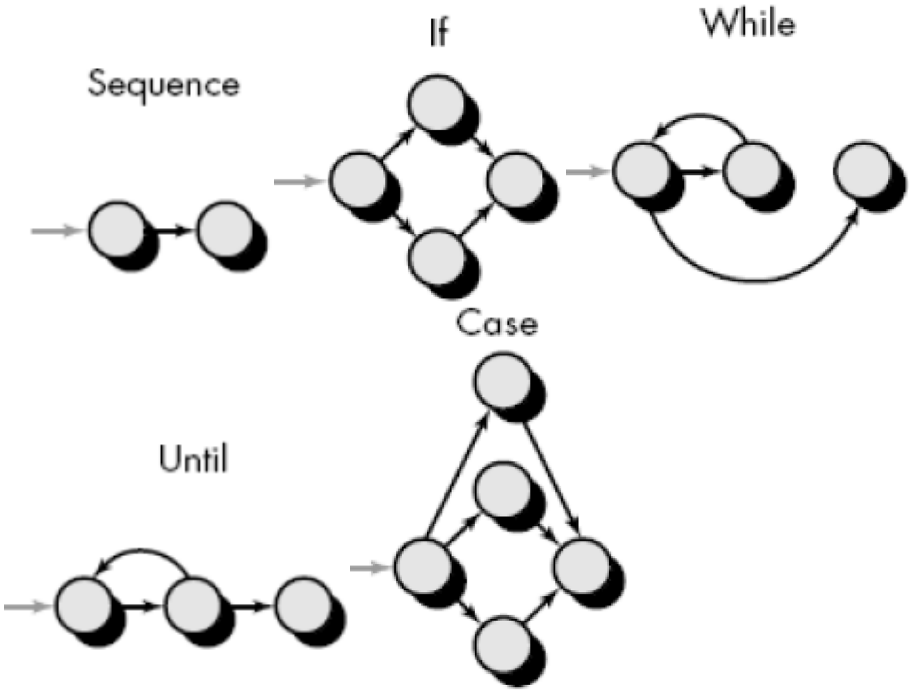


Figure 8.1 Flow graph notations

- When counting regions, we include the area outside the graph as a region.
- Each node that contains a condition is called a predicate node and is characterized by two or more edges emanating from it.

Figure 8.2 Example flowchart and corresponding flow graph

Step 2 : Compute the cyclomatic complexity CC

- *Cyclomatic complexity* is a software metric that provides a quantitative measure of the logical complexity of a program.
- The value computed for cyclomatic complexity defines the number of independent paths in the program.
- Hence gives the number of test cases to ensure that all statements have been executed at least once.
- An *independent path* is any path through the program that introduces at least one new set of processing statements or a new condition.

Complexity is computed in one of three ways:

1. The regions formed in the flow graph correspond to cyclomatic complexity.
2. Cyclomatic complexity, $V(G)$, for a flow graph, G, is defined as
   $$V(G) = E - N + 2$$
   where E is the number of flow graph edges, N is the number of flow graph nodes.
3. Cyclomatic complexity, $V(G)$, for a flow graph, G, is also defined as
   $$V(G) = P + 1$$

where *P* is the number of predicate nodes contained in the flow graph G.

For the example in Figure 8.2 above,

1. The flow graph has four regions.
2. *V*(*G*) = 11 edges - 9 nodes + 2 = 4.
3. *V*(*G*) = 3 predicate nodes + 1 = 4.

Step 3: Determine a basis set (set of independent paths)

- ☐ path 1: 1-11
- ☐ path 2: 1-2-3-4-5-10-1-11
- ☐ path 3: 1-2-3-6-8-9-10-1-11
- ☐ path 4: 1-2-3-6-7-9-10-1-11

Step 4: Prepare test cases that will force execution through each of the basis paths

2. Control Structure Testing

Control structure testing is more comprehensive than basis path testing. This method uses different categories of tests that are listed below.

1. Condition testing (e.g. branch testing) focuses on testing each decision statement in a software module. It is important to ensure coverage of all logical combinations of data that may be processed by the module (a truth table may be helpful).
2. Data flow testing selects test paths based on the locations of variable definitions and uses in the program (e.g. definition use chains).
3. Loop testing focuses on the validity of the program loop constructs (i.e. while, for, go to).

1. Condition testing

Conditional testing strategy can be applied to the different types of conditions that are possible which are listed below.

1. Compound conditions
   Two or more simple conditions connected with AND, OR. Eg : (a>b) AND (c<d)

2. Relational expression
   (E1 rel-op E2) ;E1, E2 --□ arithmetic expr. Eg : (a*b+c) > (a+b+c)

3. Boolean expression
   (B1 AND B2)

Condition testing strategies are described below.

i. Branch Testing

•It is the simplest condition testing strategy.

•For a compound condition C, the true and false branches of C and every simple condition need to be executed at least once.

•Eg : if (a>0 && b== null)

condition coverage can be achieved by testing with

a=1 b != null

a=1 b = null

a=0 b != null

a=0 b = null

ii. Domain testing

- For a relational expression, 3 tests are required:

Eg : E1<Rel-op>E2

3 test cases are:

1. E1<E2
2. E1>E2
3. E1=E2

If the relational operator is incorrect , all 3 tests guarantee detection of relational operator error.

iii. Branch and relational operator testing

- Branch and Relational Operator Testing – uses condition constraints

Example 1:

C1: B1 & B2

Where B1, B2 are boolean conditions

Condition constraint of form D1, D2 where D1 and D2 can be true (t) or false (f)

The branch and relational operator test requires the constraint set { (t,t), (f,t), (t,f) } to be covered by the execution of C1

Example 2:

C2: B1 & (E3=E4)

B1 - boolean expr

E3, E4 - arithmetic expr

Condition constraint of C2 is (D1, D2)

D1- (t/f)

D2- >,=,<

Hence the constraint set for C2 is

(t,t)------ (t,=)

(t,f)---- (t,<) and (t,>)

(f,t)--- (f,=)

Example 3:

C3: (E1>E2) & (E3=E4)

E1,E2,E3,E4 ---arithmetic expr.

Constraint set for C3----

{ (>,=) (=,=) (<,=) (>,>) (>,<) }

(t,t) (f,t) (f,t) (t,f) (t,f)

2. Data Flow testing

- Data flow testing selects test paths of a program according to the locations of definitions and uses of variables in the program.
- A USE is a reference to a variable's value. A DEF is an assignment of a new value to the variable.
- A DEF-USE pair (DU chain) is a path from the point the variable is defined to the point the variable is referenced. Data flow testing requires that all DEF-USE pairs be executed.

Example 1

```
if(some_exp)                    //1
      some_var=1;               //2
else                            //3
                some_var=2;             //4
If(some_case)           //5
                P1(some_var);           //6
else                            //7
        P2(some_var);  //8
```

☐ DEF: 2 and 4

USE : 6 and 8

4 DEF-USE pairs ----

DEF-USE-Paths	DEF	USE	PATH
1	2	6	<1-2-5-6>
2	2	8	<1-2-5-8>
3	4	6	<1-4-5-6>
4	4	8	<1-4-5-8>

Test Cases :

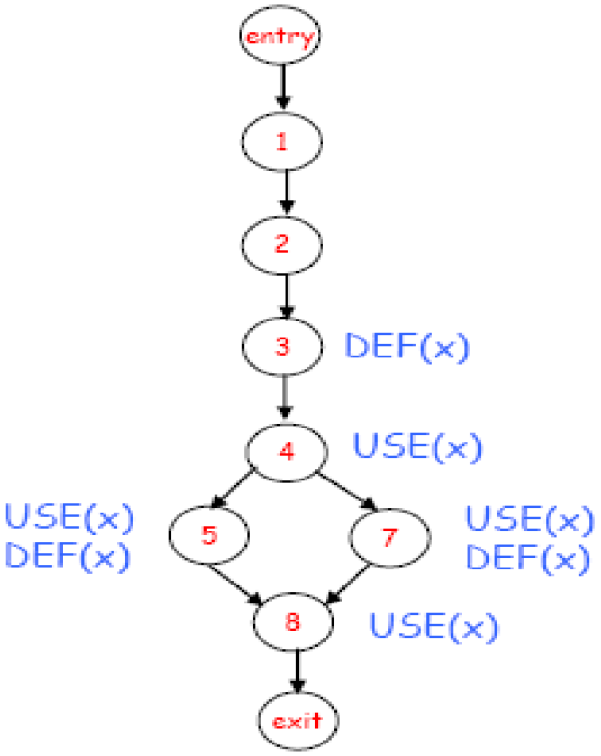
(some_exp=true, some_case=true) path 1

(some_exp=true, some_case=false) path 2

(some_exp=false, some_case=true) path 3

(some_exp=false, some_case=false) path 4

Example 2




```
public void Odd() {
 1     int x;
 2     printf("Enter a number: ");
 3     scanf("%d", &x);
 4     if x%2 == 0
 5       x=x+1;
 6     else
 7       x=2*x;
 8     printf ("x = %d\n", x);
}
```

Figure 8.3 Example for Data Flow Testing

☐ DU Chains of the Odd() Example

– (x, 3, 4), (x, 3, 5), (x, 3, 7)

– (x, 5, 8), (x, 7, 8)

NOTE : (x, 3, 8) is NOT a DU chain since the value of x at Line 3 is redefined at Lines 5 and 7 before it reaches the use at Line 8

Test paths selected according to all use coverage:

path1 1-2-3-4-7-8 cover (x, 3, 4), (x, 3, 7), (x, 7, 8)

path2 1-2-3-4-5-8 cover (x, 3, 4), (x, 3, 5), (x, 5, 8)

3. Loop Testing

Loop Testing is a white box testing technique that focuses exclusively on the validity of loop constructs. Four classes of loops can be defined:

- Simple loops
- Concatenated loops
- Nested loops
- Unstructured loops.

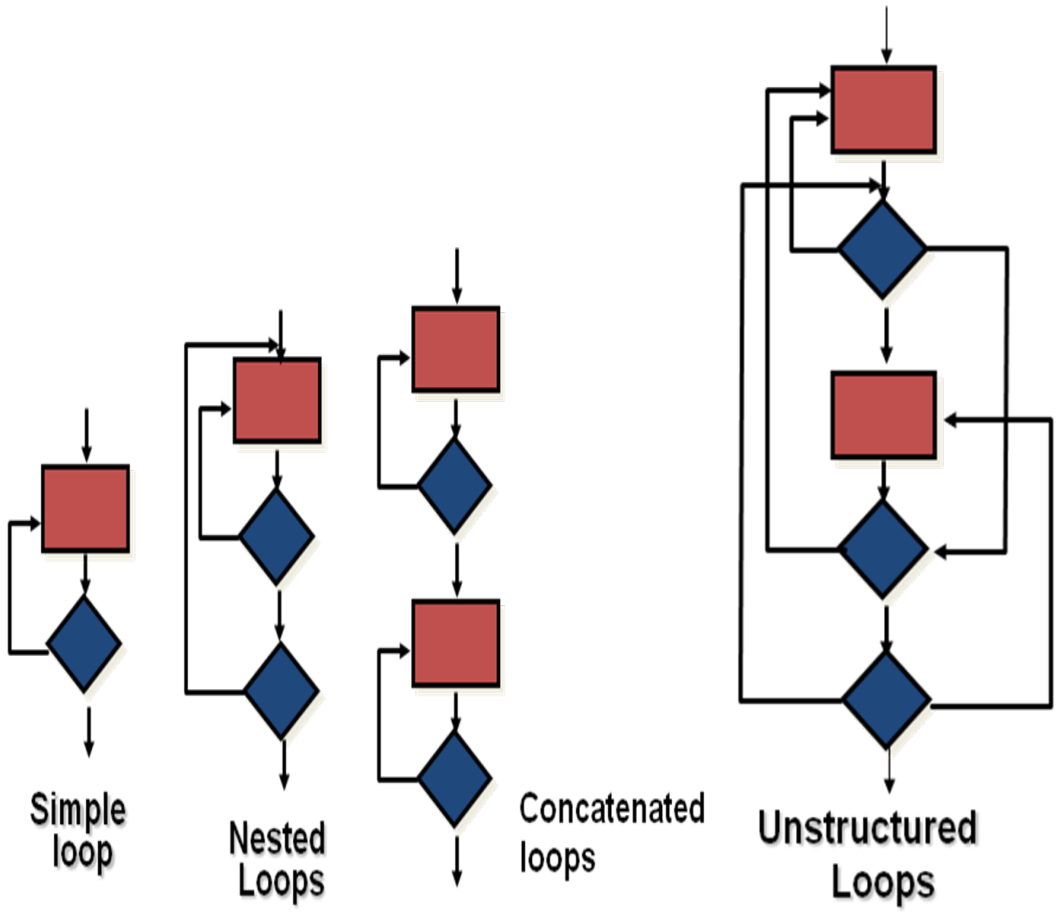


Figure 8.4 Different types of loops

i. Testing simple loops

The following sets of tests can be applied to simple loops, where 'n' is the maximum number of allowable passes through the loop.

1. Skip the loop entirely.

2. Only one pass through the loop.

3. Two passes through the loop.

4. ‘m’ passes through the loop where m is less than n.

5. n-1, n, n+1 passes through the loop.

ii. Nested Loops

If we extend the test approach from simple loops to nested loops, the number of possible tests would grow geometrically as the level of nesting increases.

1. Start at the innermost loop. Set all other loops to minimum values.

2. Conduct simple loop tests for the innermost loop while holding the outer loops at their minimum iteration parameter values.

3. Work outward, conducting tests for the next loop, but keep all other outer loops at minimum values and other nested loops to "typical" values.

4. Continue until all loops have been tested.

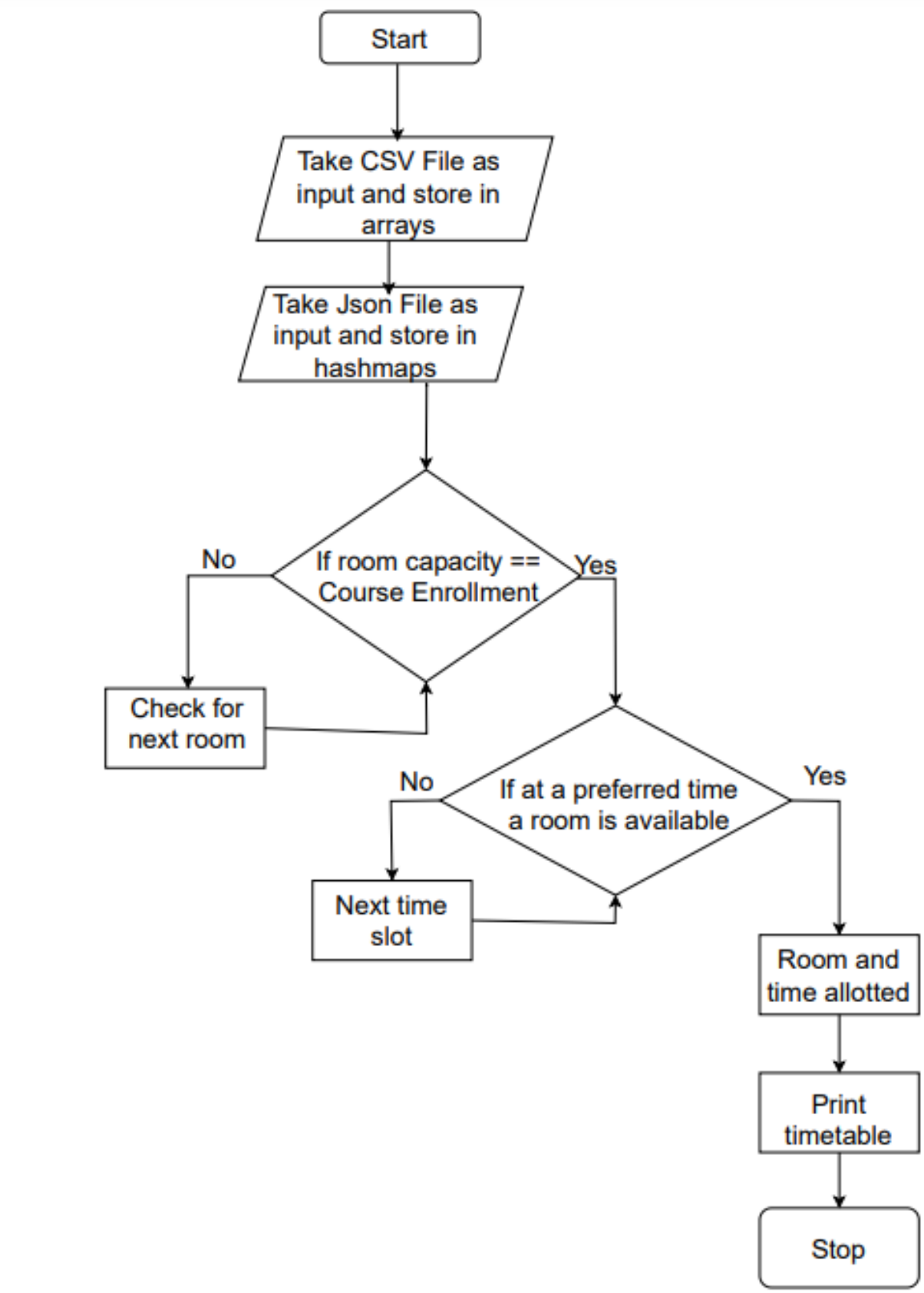
iii. Concatenated Loops

Concatenated loops can be tested using the approach defined for simple loops, if each of the loops is independent of the other.
However, if two loops are concatenated and the loop counter for loop 1 is used as the initial value for loop 2, then the loops are not independent. In this case, treat them as nested loops and perform testing.

iv. Unstructured Loops

Whenever possible, this class of loops should be redesigned to reflect the use of the structured programming constructs.

Flow Chart:


Start
Take CSV File as input and store in arrays
Take Json File as input and store in hashmaps
If room capacity == Course Enrollment
No
Yes
Check for next room
If at a preferred time a room is available
No
Yes
Next time slot
Room and time allotted
Print timetable
Stop

Number of edges: 13

Number of nodes: 10

V(G) = No. of Edges - No. of Nodes + 2
= 13 - 10 + 2
= 5

No. of independent paths :
1-2-3-4-6-8-9-10
1-2-3-4-5-6-8-9-10
1-2-3-4-6-7-8-9-10
1-2-3-4-5-6-7-8-9-10

Testing:

1. First For Loop to check for room available with required capacity
    - Expected Result:
      Allot classroom to course if it's capacity matches the course enrollment or notify the user that a course has no suitable classroom
    - Actual Result:
      The capacity and enrollment are compared to find the most suitable classroom and incase of unavailability has mechanisms to inform the user.

2. Second For Loop to check if room is available at the required time slot.
    - Expected Result:
      Once a classroom is allotted for a course, the available time slots are checked and a time slot according to the preferences is allotted for that course.
    - Actual Result:
      The time slots are checked and based on preferences an attempt is made to allot the time slot, if not an empty time slot is selected and allotted.

Conclusion

Performed White box testing of our Course Scheduling System to understand and derive test cases based on the internal logic of the system components.

Department of Computer Engineering

Academic Term: First Term 2022-23

Class: T.E /Computer Sem – V / Software Engineering

Practical No:	**10**
Title:	**Version controlling & Risk Analysis of the project**
Date of Performance:	**11/10/2022**
Roll No:	9185, 9193, 9211, 9231
Team Members:	KRIS CORREIA, IVAN DSILVA, MALAIKA MONTEIRO, SARAH ABRAHAM

Rubrics for Evaluation:

Sr. No	Performance Indicator	Excellent	Good	Below Average	Total Score
1	On time Completion & Submission (01)	01 (On Time)	NA	00 (Not on Time)	
2	Theory Understanding(02)	02(Correct)	NA	01 (Tried)	
3	Result Accuracy (03)	03(All used)	02 (Partial)	01 (rarely followed)	
4	Post Lab Questions (04)	04(done well)	3 (Partially Correct)	2(submitted)	

Signature of the Teacher:

EXPERIMENT NO. 10

Version controlling & Risk Analysis of the project.

Aim

1. Version controlling of the project using Git-Hub
2. Develop a risk table for a Course Scheduling System

Description

A **risk table** is a list of all the **risks** that could affect your software project. A **risk** is an event that is not guaranteed to happen (i.e. not 100%) that if **triggered** would **affect** your project positively or negatively. At most times, when people discuss risks in software projects, they are assuming that the risk is **negative**. If the risk event is **triggered**, i.e. comes to pass, then there is a **severity** associated with that event. Risks severity is typically low, medium, high, or catastrophic. You may have a strategy that would **mitigate** the risk. Mitigating strategies are invoked after a risk has **occurred** to reduce the severity of the outcome.

The risk table will at least list the following for each row:

- Risk description
- Probability
- Severity
- Mitigation strategies
- Strategies to reduce the probability

Example of one line of the risk table:

- Description: Chief architect quits during project development
- Probability: Low
- Severity: Catastrophic
- Mitigation strategy:

Identify developer with best architecture skills to work with chief architect

1) Identify recruiters who can find qualified architects quickly

- Strategy to reduce probability of trigger
  1) Make sure chief architect is compensated correctly
  2) Make sure that the architect has good working conditions

Typical risks include:

- Schedule risk
- Key personnel risk
- Requirements risk (i.e. that the requirements are incomplete or inconsistent)
- Learning curve risk (i.e. that your resources learn new things slower than expected)
- Technical risk

A risk table provides a project manager with a simple technique for risk projection.
A sample risk table is illustrated in the next page, Figure 1.Impact values:

l- Catastrophic
2- Critical
3- marginal
4- Negligible

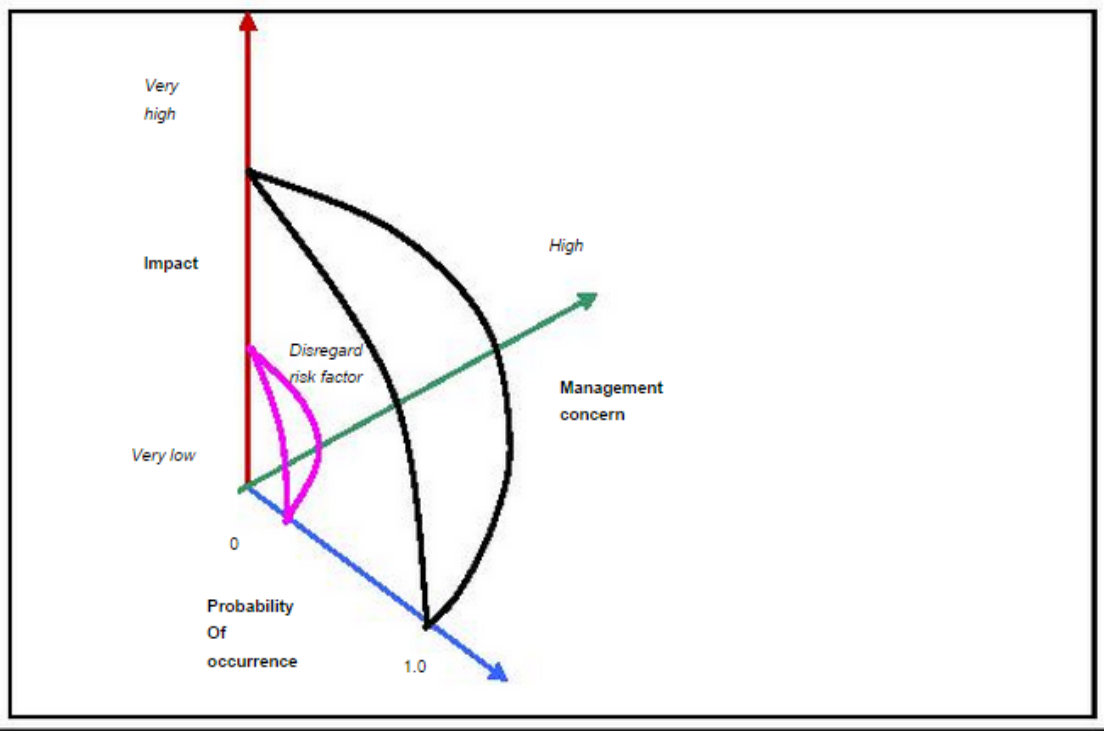
A project team begins by listing all risks (no matter how remote) in the first column of the table. This can be accomplished with the help of the risk item check-lists given earlier. Each risk is categorized in the second column (e.g., PS implies a project size risk, BU implies a business risk). The probability of occurrence of each risk is entered in the next column of the table. The probability value for each risk can be estimated by team members individually. Individual team members are polled in round-robin fashion until their assessment of risk probability begins to converge.

Figure 1:Sample risk table

Risks	Category	Probability	Impact	RMMM
Size estimate may be significantly low	PS	60%	2	
Larger number of users than planned	PS	30%	3	
Less reuse than planned	PS	70%	2	
End-users resist system	BU	40%	3	
Delivery deadline will be lightened	BU	50%	2	
Funding will be lost	CU	40%	1	
Customer will change requirements	PS	80%	2	
Technology will not meet expectations	TE	30%	1	
Lack of training on tools	DE	80%	3	
Staff inexperienced	ST	80%	2	
Staff turnover will be high	ST	60%	2	

Next, the impact of each risk is assessed. Each risk component is assessed using the characterization presented in the sample risk table, and an impact category is determined. The categories for each of the four risk components - performance, support, cost, and schedule - are averaged to determine an overall impact value.

Figure 2: Risk and management concern



Once the first four columns of the risk table have been completed, the table is sorted by probability and by impact. High-probability, high-impact risks percolateto the top of the table, and low-probability risks drop to the bottom. This accomplishes first order risk prioritization. The project manager studies the resultant sorted table and defines a cutoff line. The cutoff line (drawn horizontally at some point in the table) implies that only risks that lie above the line will begiven further attention . Risks that fall below the line are re-evaluated toaccomplish second-order prioritization.

Referring to Figure 2, risk impact and probability have a distinct influence onmanagement concern. A risk factor that has a high impact but a very low probability of occurrence should not absorb a significant amount of management time. However, high-impact risks with moderate to high probability and low-
impact risks with high probability should be carried forward into the risk analysis steps that follow.

All risks that lie above the cutoff line must be managed. The column labeled RMMM contains a pointer into Risk Mitigation, Monitoring and Management Plan or alternatively, a collection of risk information sheets developed for all risksthat lie above the cutoff.

Risk Table

Risks	Category	Probability	Severity	Impact	Risk Mitigation, Monitoring and Management Plan
Improper resource allocation i.e improper classroom allotment	**Schedule Risk**	20%	Critical	1	Mitigation Strategies : • Identify incorrect resource quantities (available classrooms, no of courses,etc) • Rectify resource allocation Strategies to reduce probability of trigger : • Update free and occupied classrooms • Allot only 1 course to 1 classroom for a particular time slot
Unexpected	**Budget Risk**	60%	Critical	3	Mitigation Strategies :

Project Scope expansion					• Compare Scope expansion with user expectations and terminate developing of unnecessary features Strategies to reduce probability of trigger : • Re-evaluate customer requirements and budget after every development iteration
Lack of communication and cooperation	**Operational Risks**	80%	Marginal	1	Mitigation Strategies : • Immediately update staff and students about updates in the system Strategies to reduce probability of trigger : • Staff and students should receive notifications to alert them of changes in the schedule
Application crashes	**Technical Risks**	75%	Catastrophic	1	Mitigation Strategies : • Immediately identify cause of application crash and rectify it • Reboot Application Strategies to reduce probability of trigger : • Avoid Stack overflow Errors
Changes in University policy	**Programmatic Risks**	35%	Marginal	4	Mitigation Strategies : • Identify and implement a efficient and reusable solution Strategies to reduce probability of trigger : • Develop Model in accommodation of possible policy changes

Version Control

Initialize git

```
Sarah@DESKTOP-SVL88CJ MINGW64 ~/OneDrive/Documents/Sem 5 tings/Software Engineer
ing/Course Scheduling System
$ git init
Initialized empty Git repository in C:/Users/Sarah/OneDrive/Documents/Sem 5 ting
s/Software Engineering/Course Scheduling System/.git/
```

Add files

```
Sarah@DESKTOP-SVL88CJ MINGW64 ~/OneDrive/Documents/Sem 5 tings/Software Engineer
ing/Course Scheduling System (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        .idea/
        Course Scheduling System.iml
        Course-Scheduling-System/
        src/

nothing added to commit but untracked files present (use "git add" to track)
```

```
Sarah@DESKTOP-SVL88CJ MINGW64 ~/OneDrive/Documents/Sem 5 tings/Software Engineer
ing/Course Scheduling System (master)
$ git add .
warning: adding embedded git repository: Course-Scheduling-System
hint: You've added another git repository inside your current repository.
hint: Clones of the outer repository will not contain the contents of
hint: the embedded repository and will not know how to obtain it.
hint: If you meant to add a submodule, use:
hint:
hint:   git submodule add <url> Course-Scheduling-System
hint:
hint: If you added this path by mistake, you can remove it from the
hint: index with:
hint:
hint:   git rm --cached Course-Scheduling-System
hint:
hint: See "git help submodule" for more information.
```

Commiting the files

```
Sarah@DESKTOP-SVL88CJ MINGW64 ~/OneDrive/Documents/Sem 5 tings/Software Engineer
ing/Course Scheduling System (master)
$ git commit
hint: Waiting for your editor to close the file...
[main 2022-10-28T19:01:19.635Z] update#setState idle
[main 2022-10-28T19:01:21.125Z] Starting extension host with pid 6360 (fork() to
ok 21 ms).
[main 2022-10-28T19:01:49.650Z] update#setState checking for updates
[main 2022-10-28T19:01:50.075Z] update#setState idle
[main 2022-10-28T19:09:33.202Z] Waiting for extension host with pid 6360 to exit
.
[main 2022-10-28T19:09:33.246Z] Extension host with pid 6360 exited with code: 0
, signal: null.
[master (root-commit) b4d840a] initializing repository
 9 files changed, 693 insertions(+)
 create mode 100644 .idea/.gitignore
 create mode 100644 .idea/dbnavigator.xml
 create mode 100644 .idea/misc.xml
 create mode 100644 .idea/modules.xml
 create mode 100644 .idea/runConfigurations.xml
 create mode 100644 .idea/vcs.xml
 create mode 100644 Course Scheduling System.iml
 create mode 160000 Course-Scheduling-System
 create mode 100644 src/InputFiles.java
```

COMMIT_EDITMSG

C: > Users > Sarah > OneDrive > Documents > Sem 5 tings > Software Engineering > Course Scheduling System > .git > COMMIT_EDITMS

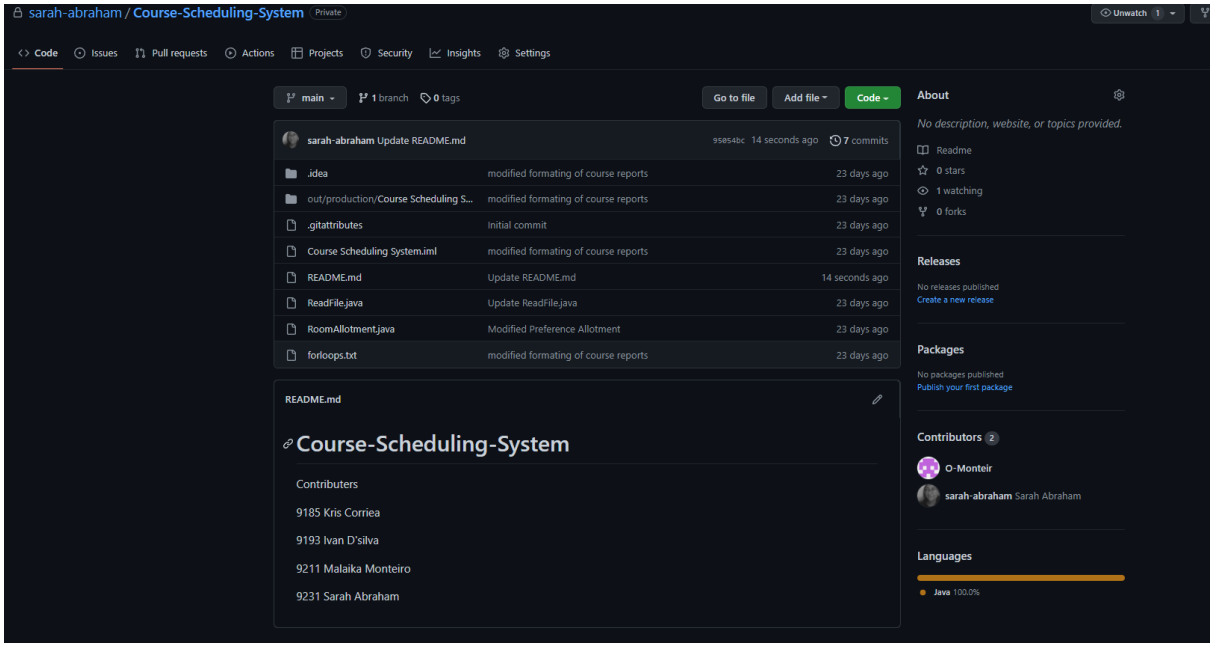
```
initializing repository
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
#
# Initial commit
#
# Changes to be committed:
#   new file:   .idea/.gitignore
#   new file:   .idea/dbnavigator.xml
#   new file:   .idea/misc.xml
#   new file:   .idea/modules.xml
#   new file:   .idea/runConfigurations.xml
#   new file:   .idea/vcs.xml
#   new file:   Course Scheduling System.iml
#   new file:   Course-Scheduling-System
#   new file:   src/InputFiles.java
#
```

```
Sarah@DESKTOP-SVL88CJ MINGW64 ~/OneDrive/Documents/Sem 5 tings/Software Engineering/Course Scheduling System (master)
$ git status
On branch master
nothing to commit, working tree clean
```

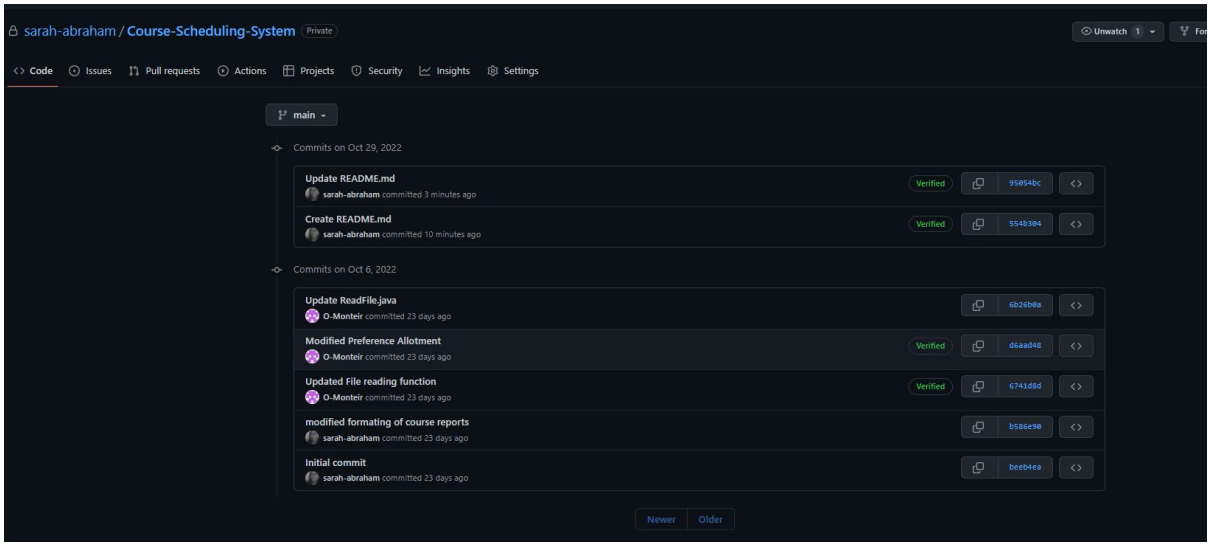
Create Repository

```
Sarah@DESKTOP-SVL88CJ MINGW64 ~/OneDrive/Documents/Sem 5 tings/Software Engineering/Course Scheduling System (master)
$ git remote add origin https:/github.com/sarah-abraham/course-scheduling-system

Sarah@DESKTOP-SVL88CJ MINGW64 ~/OneDrive/Documents/Sem 5 tings/Software Engineering/Course Scheduling System (master)
$ git remote -v
origin  https:/github.com/sarah-abraham/course-scheduling-system (fetch)
origin  https:/github.com/sarah-abraham/course-scheduling-system (push)
```



Through the repository we can view the commits made and see who made these changes, and other details about the changes made.



Conclusion : As part of project management, risk analysis consists of identifying the factors that could affect the success or failure of a project. Among these processes are the identification of risks, the analysis of risks, and the management and control of risks. A proper risk analysis is more proactive than reactive, and it assists in controlling possible future events that may harm the overall project.