Indira Gandhi
National Open University
School of Computer and
Information Sciences

BCSP-165
Project Guidelines

PROJECT GUIDELINES: BCSP-165

Indira Gandhi
National Open University
School of Computer and
Information Sciences

BCSP-165
Project Guidelines

PROJECT GUIDELINES: BCSP-165

PROGRAMME DESIGN COMMITTEE

Prof. Sanjeev K. Aggarwal, IIT, Kanpur
Prof. M. Balakrishnan, IIT , Delhi
Prof Harish Karnick, IIT, Kanpur
Prof. C. Pandurangan, IIT, Madras
Dr. Om Vikas, Sr. Director, MIT
Prof P. S. Grover, Sr. Consultant, SOCIS, IGNOU
Prof. H.M. Gupta Dept. of Elect. Engg. IIT, Delhi
Prof. M.N. Doja, Dept. of CE Jamia Millia, New Delhi
Prof. C. Pandurangan Dept. of CSE, IIT, Chennai
Prof. L. Ramesh Babu Dept. of CSE Acharya Nagarjuna University Nagarjuna Nagar (AP)
Prof. N.S. Gill Dept. of CS, MDU, Rohtak
Prof. Arvind Kalia, Dept. of CS HP University, Shimla
Prof. Anju Sahgal Gupta SOH, IGNOU, New Delhi
Prof. Sujatha Verma SOS, IGNOU, New Delhi
Prof. V. Sundarapandian IITMK, Trivandrum
Prof. Dharamendra Kumar Dept. of CS, GJU, Hissar
Prof. Vikram Singh Dept. of CS, CDLU, Sirsa
Dr. P.K. Mishra, Associate Prof. Dept. of CS, BHU, Varanasi
Prof. M.P. Mishra, SOCIS, IGNOU, New Delhi
Dr. Naveen Kumar, Reader SOCIS, IGNOU, New Delhi
Dr. Subodh Kesharwani, Asst. Prof. SOMS, IGNOU, New Delhi
Prof. P.V. Suresh, SOCIS, IGNOU
Prof. V.V. Subrahmanyam SOCIS, IGNOU
Prof. Akshay Kumar SOCIS, IGNOU
Shri Shashi Bhushan, SOCIS, IGNOU
Prof Manohar Lal, SOCIS, IGNOU

FACULTY OF SOCIS

Prof. Sandeep Singh Rawat, Director, SOCIS, IGNOU
Prof. V.V. Subrahmanyam SOCIS, IGNOU
Prof. Divakar Yadav, SOCIS, IGNOU
Prof. P.V. Suresh, SOCIS, IGNOU
Prof. Akshay Kumar SOCIS, IGNOU
Prof. M.P. Mishra, SOCIS, IGNOU
Dr. Sudhansh Sharma, SOCIS, IGNOU

CONTENT PREPARATION TEAM

Partially Adapted Section – 1,2,3 From MCS-044 Block – 1 Section-1,2,3 Respectively
Partially Adapted Section – 4 From BCSP-064

Vetted By:
Prof. Akshay Kumar & Dr. Sudhansh Sharma

Course Coordinator: *Dr. Sudhansh Sharma & Prof. Akshay Kumar, SOCIS, IGNOU*

July, 2026

CRC was prepared in-house at SOCIS by Mr. Aditya Mawar, (Skilled Worker at SOCIS)

Published (Online) on behalf of the Indira Gandhi National Open University, New Delhi by SOCIS.

COURSE INTRODUCTION

The Project course is designed to enable students to acquire and demonstrate practical competencies in the application of contemporary tools, technologies, and methodologies used in computing. The primary objective is to develop the ability to analyze, design, implement, and evaluate solutions for real-world problems relevant to industry, academic institutions, and computer science research.

A typical computer science project involves systematic stages of the software development life cycle (SDLC), including problem analysis, system design, implementation (coding), integration, and testing of an information system or subsystem. The project may focus on the development of a software-based solution, such as an application, information system, or computational tool. In certain cases, the outcome may also include a comprehensive system design document produced through a detailed design study. Although the design and development of hardware systems or embedded subsystems can constitute valid computing projects, the primary expectation in this course is the development of a software-based system or subsystem.

The project component also aims to strengthen students' analytical thinking, investigative capability, research methodology, and technical documentation skills. It provides an opportunity for students to explore a selected topic in significant depth, apply programming knowledge, and address complex computing challenges through systematic problem-solving approaches. Furthermore, the project enables students to demonstrate the integration of theoretical knowledge with practical implementation in the field of computing.

The course comprises one block titled "PROJECT GUIDELINES BCSP-165," which outlines the fundamental concepts, methodologies, and best practices required for effective project planning, execution, and reporting.

BLOCK INTRODUCTION

Project development constitutes a critical practical component in the field of computer science and information systems education, enabling students to apply theoretical knowledge to real-world computational problems. In many cases, students tend to perceive technical documentation and report preparation as a secondary or time-consuming activity that diverts attention from the final stages of software development. However, comprehensive project documentation is an essential deliverable that records the system design, development methodology, implementation details, and evaluation results. Proper documentation ensures reproducibility, maintainability, and effective communication of the developed solution.

Within the BCA_NEW/BCA_NEWOL programme, the Project course represents a significant academic requirement aimed at consolidating the advanced technical knowledge and practical competencies acquired through earlier coursework. The project typically involves the systematic investigation of a computing problem, followed by the analysis, design, development, and implementation of an appropriate solution. Projects may span diverse areas of computer science such as software systems, data management, networking applications, artificial intelligence, or emerging computing technologies.

Another important objective of the project component is to promote a collaborative and structured development environment, similar to the practices followed in professional software engineering teams within the IT industry. Through this experience, students are exposed to roles and responsibilities analogous to those found in real-world software development settings, including system analysis, software design, coding, testing, and technical documentation.

The Block titled “Project Guidelines BCSP-165” provides a structured overview of the essential processes and standards required for the systematic development and documentation of a software project using a software engineering approach. The block comprises three sections that guide learners through the key stages of project preparation and reporting:

- Section 1: Project Structure: This section describes the standard structure, organization, and formatting requirements of a project report. It outlines the essential components and documentation standards that should be followed while preparing the final project report.

- Section 2: Guidelines and Suggestions: This section presents methodological guidelines and practical recommendations for planning, executing, and documenting the final-year project. It assists students in managing project activities effectively and ensuring that the final report adheres to academic and technical standards.

- Section 3: Software Engineering Concepts and Standards: This section introduces the fundamental software engineering principles, development models, and documentation standards that are essential for systematic software development. The section also includes illustrative examples to demonstrate their practical application in project development.

- Section 4: Guidelines for BCSP-165: This section carries the detailed guidelines for the formulation of Project proposal and it is supported by Annexute-1,2, & 3, available immediately after the Section-4

ignou
THE PEOPLE'S
UNIVERSITY

SECTION 1: PROJECT STRUCTURE

Structure **Page Nos.**

1.0 INTRODUCTION

The project report should be prepared using a systematic software engineering approach that clearly documents the methodology adopted for addressing the identified problem. The report must present a structured and technically sound description of the problem definition, system analysis, design methodology, implementation strategy, and validation procedures employed during the project development process. The documentation should demonstrate the application of appropriate computational techniques, development tools, technologies, and relevant professional standards in the formulation and implementation of the proposed solution.

Students are encouraged to initiate the documentation process concurrently with the early stages of the software development life cycle (SDLC), beginning with problem identification and requirement analysis. Maintaining documentation throughout the development phases facilitates effective tracking of design decisions, implementation challenges, and solution strategies, which can subsequently be incorporated into the final project report.

The report should also incorporate relevant information recorded in the project logbook or development notebook, where the progress of project activities, intermediate results, design modifications, and implementation milestones are systematically documented during the course of the project.

A well-prepared project report must contain sufficient technical details, analytical explanations, and implementation descriptions to enable examiners to effectively evaluate the scope, originality, and technical quality of the work. At the same time, the report should maintain clarity, conciseness, and logical organization to ensure readability and avoid unnecessary verbosity. Key findings, methodologies, and design decisions should be presented prominently within the main body of the report, while supporting materials, extended data, detailed code listings, and supplementary documentation may be included in the appendices.

This section provides a comprehensive overview of the standard structure and essential components of a project report, along with detailed explanations of each section to guide students in preparing a professionally structured and technically sound project document.

1.1 OBJECTIVES

After completing this section, learners should be able to:

- Demonstrate a comprehensive and systematic understanding of the structural components of a software project report, including their purpose and interrelationships.
- Apply appropriate methodologies and professional standards for technical documentation, consistent with accepted software engineering practices.

- Interpret and articulate the significance of various project report components, such as problem definition, system analysis, design specifications, implementation details, and evaluation results.
- Utilize established techniques and best practices for the preparation and development of structured project documentation, ensuring clarity, consistency, and technical rigor in project reporting.

1.2 IMPORTANCE OF THE PROJECT

The Project component constitutes a significant academic requirement that extends beyond routine coursework and serves as a comprehensive mechanism for evaluating a student's technical competence, problem-solving ability, and domain specialization in computer science. It provides an opportunity for students to demonstrate originality, analytical thinking, collaborative skills, systematic planning, and project management capabilities within the context of a software development task. Furthermore, the project enables learners to apply and integrate the concepts, methodologies, tools, and technologies acquired through preceding courses into the development of a practical computing solution.

The project plays a critical role in the academic programme for several reasons. It enables students to:

- **Develop specialized knowledge and practical expertise in selected domains of computer science and information technology.**
- **Prepare for professional evaluation**, as prospective employers often assess candidates' project work during technical interviews to understand their practical experience and problem-solving approaches.
- **Demonstrate the application of a broad range of technical competencies**, including programming, system analysis, design methodologies, implementation strategies, testing procedures, and documentation practices.
- **Integrate theoretical knowledge and practical skills** acquired across multiple courses into a coherent solution addressing a complex computing problem.

The project report represents a critical deliverable of the project and serves as the formal documentation of the entire development process and outcomes. It provides a comprehensive description of the problem domain, system design, implementation methodology, evaluation procedures, and results obtained during the project lifecycle. An effective project report should demonstrate the following:

- **Contextual understanding of the computing domain**, by relating the selected project topic and adopted approach to existing systems, technologies, research developments, or industry practices.
- **Application of theoretical concepts and practical techniques** learned throughout the programme to address the defined problem, along with an explanation of their relevance within the broader computing ecosystem.
- **Ability to critically evaluate and reflect on the developed system**, including identification of limitations, performance considerations, and potential areas for enhancement or future research.
- **Clear and precise technical communication**, demonstrating the ability to present design decisions, implementation strategies, and development processes in a structured, concise, and professional manner through comprehensive project documentation.

1.3 PROJECT: TABLE OF CONTENTS

The project report should present a comprehensive, systematic, and technically coherent documentation of the work carried out during the project development lifecycle. While students may be required to demonstrate the developed software system and provide an oral presentation, the primary evaluation of the project will be based on the technical quality, completeness, and clarity of the written project report.
Students may receive academic guidance, suggestions, and feedback from their BCSP-165 course counsellor during different stages of project development. However, the overall planning, execution, documentation, and submission of the project remain the responsibility of the student. The project report must therefore reflect independent work, methodological rigor, and adherence to professional documentation standards.
To ensure clarity, consistency, and completeness in technical documentation, the project report should follow a structured format consistent with established software engineering documentation practices. The following outline provides a recommended structure for organizing the project report. However, students may adapt the format where necessary in consultation with their course counsellor, depending on the specific requirements and nature of the project.

Suggested Structure of the Project Report

Preliminary Pages

- Title Page
- Original Copy of the Approved Project Proposal Proforma
- Certificate of Authenticated Work
- Role and Responsibility Form
- Abstract
- Acknowledgement
- Table of Contents
- Table of Figures

CHAPTER 1: INTRODUCTION

This chapter introduces the project context, objectives, scope, and overall structure of the report.

1.1 Background

Provides an overview of the problem domain, motivation for the project, and the technological or application context.

1.2 Objectives

Defines the primary goals and expected outcomes of the project.

1.3 Purpose, Scope, and Applicability

- 1.3.1 Purpose – Describes the intended functionality and goals of the system.
- 1.3.2 Scope – Specifies the functional boundaries and limitations of the project.
- 1.3.3 Applicability – Identifies target users, operational environments, or application domains.

1.4 Achievements

Summarizes the major accomplishments and implemented features of the system.

1.5 Organisation of the Report

Provides an overview of the structure and content of the subsequent chapters.

CHAPTER 2: REQUIREMENTS ANALYSIS

This chapter focuses on the system analysis phase, including identification of functional and non-functional requirements.

2.1 Problem Definition
Clearly defines the problem statement and operational requirements addressed by the project.
2.2 Requirements Specification
Provides detailed functional and non-functional system requirements.
2.3 Software and Hardware Requirements
Specifies the software platforms, development tools, frameworks, libraries, and hardware resources required.
2.4 Preliminary Product Description
Provides a high-level description of system features, operational workflow, and user interactions.
2.5 Conceptual Models
Includes conceptual representations such as UML diagrams, data flow diagrams (DFDs), entity–relationship diagrams (ERDs), or system architecture models.

CHAPTER 3: SYSTEM DESIGN
This chapter documents the **architectural and detailed design of the proposed system**.

3.1 Basic Modules
Describes the major system modules and their functional responsibilities.
3.2 Data Design
- **4.2.1 Schema Design** – Defines database schema structures and relationships.
- **4.2.2 Data Integrity and Constraints** – Specifies data validation rules, integrity constraints, and consistency mechanisms.

3.3 Procedural Design
- **4.3.1 Logic Diagrams** – Includes flowcharts or control flow representations.
- **4.3.2 Data Structures** – Describes data organization and structures used in the system.
- **4.3.3 Algorithm Design** – Explains algorithms implemented to achieve system functionality.

3.4 User Interface Design
Describes the graphical or web-based interface architecture, usability considerations, and interaction design.
3.5 Security Considerations
Discusses authentication mechanisms, data protection strategies, and access control measures.
3.6 Test Case Design
Specifies test cases and validation criteria used for system verification.

CHAPTER 4: IMPLEMENTATION AND TESTING
This chapter details the development, coding practices, and system testing processes.

4.1 Implementation Approaches
Describes the software development approach, frameworks, and programming methodologies used.
4.2 Coding Details: include:
- 4.2.1 Structure of Code
- 4.2.2 Code for database implementation
- 4.2.3 Programs with suitable comments for better readability

4.3 Testing Strategy
- **5.3.1 Unit Testing** – Testing of individual modules.
- **5.3.2 Integration Testing** – Testing interactions between modules and subsystems.

4.4 Modifications and Improvements
Documents changes, refinements, and improvements made during system development and testing.

CHAPTER 5: RESULTS AND DISCUSSION
This chapter presents the evaluation of the implemented system and interpretation of results.

5.1 Test Reports
Provides test outcomes, validation results, and performance observations.
5.2 User Documentation
Includes operational guidelines, system usage instructions, and user manuals.

CHAPTER 6: CONCLUSION
This chapter summarizes the overall findings and outcomes of the project.

6.1 Conclusion
Provides a summary of achievements and system capabilities.
6.2 Limitations of the System
Discusses constraints, assumptions, and limitations encountered during development.
6.3 Future Scope of the Project
Identifies potential enhancements, scalability considerations, and opportunities for further research or development.

Supporting Sections

- **References** – List of academic papers, books, standards, and online resources cited.
- **Glossary** – Definitions of technical terms and abbreviations used in the report.
- **Appendix A** – Supplementary materials such as detailed diagrams, datasets, or extended documentation.
- **Appendix B** – Additional supporting information such as source code snippets, configuration details, or test case documentation.

1.4 EXPLANATION OF CONTENTS

Title Page

Sample format of *Title page* is given in Appendix 1 of this block. Students should follow the given format.

Original Copy of the Approved Proforma of the Project Proposal

Sample *Proforma of Project Proposal* is given in Appendix 2 of this block. Students should follow the given format.

Certificate of Authenticated work

Sample format of *Certificate of Authenticated work* is given in Appendix 3 of this block. Students should follow the given format.

Role and Responsibility Form

Sample format for *Role and Responsibility Form* is given in Appendix 4 of this block. Students should follow the given format.

Abstract

This should be one/two short paragraphs (100-150 words total), summarising the project work. It is important that this is not just a re-statement of the original project

outline. A suggested flow is background, project aims and main achievements. From the abstract, a reader should be able to ascertain if the project is of interest to them and, it should present results of which they may wish to know more details.

Acknowledgements

This should express your gratitude to those who have helped you in the preparation of your project.

Table of Contents: The table of contents gives the readers a view of the detailed structure of the report. You would need to provide section and subsection headings with associated pages. The formatting details of these sections and subsections you will find in section 2 of this block.

Table of Figures: List of all Figures, Tables, Graphs, Charts etc. along with their page numbers in a table of figures.

Chapter 1: Introduction

The introduction has several parts as given below:

Background: A description of the background and context of the project and its relation to work already done in the area. Summarise existing work in the area concerned with your project work.

Objectives: Concise statement of the aims and objectives of the project. Define exactly what you are going to do in the project; the objectives should be about 30 /40 words.

Purpose, Scope and Applicability: The description of Purpose, Scope, and Applicability are given below:

- *Purpose:* Description of the topic of your project that answers questions on why you are doing this project. How your project could improve the system its significance and theoretical framework.
- *Scope:* A brief overview of the methodology, assumptions and limitations. You should answer the question: What are the main issues you are covering in your project? What are the main functions of your project?
- *Applicability:* You should explain the direct and indirect applications of your work. Briefly discuss how this project will serve the computer world and people.

Achievements: Explain what knowledge you achieved after the completion of your work. What contributions has your project made to the chosen area? Goals achieved - describes the degree to which the findings support the original objectives laid out by the project. The goals may be partially or fully achieved, or exceeded.

Organisation of Report: Summarising the remaining chapters of the project report, in effect, giving the reader an overview of what is to come in the project report.

Chapter 2: Requirements and Analysis

Problem Definition: Define the problem on which you are working in the project. Provide details of the overall problem and then divide the problem in to sub-problems. Define each sub-problem clearly.

Requirements Specification: In this phase you should define the requirements of the system, independent of how these requirements will be accomplished. The Requirements Specification describes the things in the system and the actions that can be done on these things. Identify the operation and problems of the existing system.

Planning and Scheduling: Planning and scheduling is a complicated part of software development. Planning, for our purposes, can be thought of as determining all the small tasks that must be carried out in order to accomplish the goal. Planning also takes into account, rules, known as constraints, which, control when certain tasks can or cannot happen. Scheduling can be thought of as determining whether adequate resources are available to carry out the plan. You should show the Gantt chart and Program Evaluation Review Technique (PERT).

Software and Hardware Requirements: Define the details of all the software and hardware needed for the development and implementation of your project.

- *Hardware Requirement:* In this section, the equipment, graphics card, numeric co-processor, mouse, disk capacity, RAM capacity etc. necessary to run the software must be noted.

- *Software Requirements:* In this section, the operating system, the compiler, testing tools, linker, and the libraries etc. necessary to compile, link and install the software must be listed.

Preliminary Product Description: Identify the requirements and objectives of the new system. Define the functions and operation of the application/system you are developing as your project.

Conceptual Models: You should understand the problem domain and produce a model of the system, which describes operations that can be performed on the system, and the allowable sequences of those operations. Conceptual Models could consist of complete Data Flow Diagrams, ER diagrams, Object-oriented diagrams, System Flowcharts etc.

Chapter 3: System Design

Describes desired features and operations in detail, including screen layouts, business rules, process diagrams, pseudocode and other documentation.

Basic Modules: You should follow the divide and conquer theory, so divide the overall problem into more manageable parts and develop each part or module separately. When all modules are ready, you should integrate all the modules into one system. In this phase, you should briefly describe all the modules and the functionality of these modules.

Data Design: Data design will consist of how you organise, managing and manipulate the data.

- *Schema Design:* Define the structure and explanation of schemas used in your project.

- *Data Integrity and Constraints:* Define and explain all the validity checks and constraints you are providing to maintain data integrity.

Procedural Design: Procedural design is a systematic way for developing algorithms or procedurals.

- *Logic Diagrams:* Define the systematical flow of procedure that improves its comprehension and helps the programmer during implementation. e.g., Control Flow Chart, Process Diagrams etc.
- *Data Structures:* Create and define the data structure used in your procedures.
- *Algorithms Design:* With proper explanations of input data, output data, logic of processes, design and explain the working of algorithms.

User Interface Design: Define user, task, environment analysis and how you intend to map those requirements in order to develop a "User Interface". Describe the external and internal components and the architecture of your user interface. Show some rough pictorial views of the user interface and its components.

Security Issues: Discuss Real-time considerations and Security issues related to your project and explain how you intend avoiding those security problems. What are your security policy plans and architecture?

Test Cases Design: Define test cases, which will provide easy detection of errors and mistakes with in a minimum period of time and with the least effort. Explain the different conditions in which you wish to ensure the correct working of your software.

Chapter 4: Implementation and Testing

Implementation Approaches: Define the plan of implementation, and the standards you have used in the implementation.

Coding Details: Students not need include full source code, instead, include only the important codes (algorithms, applets code, forms code etc). The program code should contain comments needed for explaining the work a piece of code does. Comments may be needed to explain why it does it, or, why it does a particular way.

Testing Approach: Testing should be according to the scheme presented in the system design chapter and should follow some suitable model – e.g., category partition, state machine-based. Both functional testing and user-acceptance testing are appropriate. Explain your approach of testing.

- *Unit Testing:* Unit testing deals with testing a unit or module as a whole. This would test the interaction of many functions but, do confine the test within one module.
- *Integrated Testing:* Brings all the modules together into a special testing environment, then checks for errors, bugs and interoperability. It deals with tests for the entire application. Application limits and features are tested here.

Modifications and Improvements: Once you finish the testing you are bound to be faced with bugs, errors and you will need to modify your source code to improve the system. Define what modification you implemented in the system and how it improved your system.

Chapter 5: Results and Discussion

Test Reports: Explain the test results and reports based on your test cases, which should show that your software is capable of facing any problematic situation and that it works fine in different conditions. Take the different sample inputs and show the outputs.

User Documentation: Define the working of the software; explain its different functions, components with screen shots. The user document should provide all the details of your product in such a way that any user reading the manual, is able to understand the working and functionality of the document.

Chapter 6: Conclusions

Conclusion: The conclusions can be summarized in a fairly short chapter (2 or 3 pages). This chapter brings together many of the points that you would have made in the other chapters.

Limitations of the System: Explain the limitations you encountered during the testing of your software that you were not able to modify. List the criticisms you accepted during the demonstrations of your software.

Future Scope of the Project describes two things: firstly, new areas of investigation prompted by developments in this project, and secondly, parts of the current work that were not completed due to time constraints and/or problems encountered.

REFERENCES

It is very important that you acknowledge the work of others that you have used or adapted in your own work, or that provides the essential background or context to your project. The use of references is the standard way to do this. Please follow the given standard for the references for books, journals, and online material.

GLOSSARY

If you use any acronyms, abbreviations, symbols, or uncommon terms in the project report then their meaning should be explained where they first occur. If you go on to use any of them extensively then it is helpful to list them in this section and define the meaning.

APPENDICES

These may be provided to include further details of results, mathematical derivations, certain illustrative parts of the program code (e.g., class interfaces), user documentation etc.

In particular, if there are technical details of the work done that might be useful to others who wish to build on this work, but that are not sufficiently important to the project as a whole to justify being discussed in the main body of the project, then they should be included as appendices.

1.5 SUMMARY

Project development usually involves an engineering approach to the design and development of a software system that fulfils a practical need. Projects also often form

an important focus for discussion at interviews with future employers as they provide a detailed example of what you are capable of achieving. The next Section *Guidelines and Suggestions* will provide you detailed guidelines and suggestions, which will be useful for you during project development and the preparation of the report.

SECTION 2: GUIDELINES AND SUGGESTIONS

Structure **Page Nos.**

2.0 INTRODUCTION

The objective of this section is to establish a structured set of guidelines and recommendations to support the systematic development of a project as well as the preparation of a comprehensive final year project report. It outlines the standardized format and essential components required in the report, along with conventions related to document layout, organization, and presentation. Additionally, the section provides guidance on appropriate academic writing style, technical language usage, and suggests relevant resources for further study.

The project report should begin with an introductory chapter that clearly defines the project context. This section should present the background of the study, identify the problem statement, and explain the underlying motivation for undertaking the project. The main body of the report should provide a detailed and objective description of the work carried out during the project. It should include explanations of the methodologies adopted, justification of design and implementation decisions, analysis of challenges encountered during development, and the strategies employed to address and resolve those challenges. Furthermore, it should highlight the technical outcomes and results achieved.

The concluding chapter should summarize the overall contributions and achievements of the project. It should also critically evaluate the results obtained and, where appropriate, discuss potential improvements and directions for future research or development related to the project.

2.1 OBJECTIVES

After going through this section you should be able to:

- Explain the key activities of the Project,
- know your Roles and Responsibilities,
- Describe how to select a project topic,
- Steps involved to start the project,
- know the guidelines for proposal and report preparation,

- know the Evaluation Scheme, and
- know the Assessment Guidelines.

2.2 KEY FEATURES OF THE PROJECT

While preparing the project students learn and practice different activities, which help them to develop the ability to solve real life problems related to software development. There are different activities (including the details given in section 1 and section 2) in each phase of software development; however, some of the key activities of the project work are given below:

Analysis

- Framing the Systems Development Life Cycle (SDLC) Model for the related project,
- Understanding and evaluating the specific problem,
- Analysing and evaluating the systems requirements,
- Deciding the Software requirement specifications and Hardware requirement specifications, and
- Designing and constructing the entity-relationship (ER) diagram (for RDBMS related projects), data flow diagrams (level 0,1,2) (OR Object-oriented diagrams, System Flowcharts etc.) and data dictionary.

Design

- Plan the systems design phase and distinguish between logical and physical design requirements,
- Create the systems flow charts and state transition diagrams,
- Describe all the modules and the functionality of modules,
- Design and evaluate system inputs and outputs,
- Design and evaluate user interfaces for input, and validity checks for input data,
- Perform normalisation for the un-normalised tables for RDBMS related projects,
- Design various test cases for different testing techniques/strategies, and
- Decide various data structures.

Coding

- Performing coding according to the requirement and design document,
- Writing comments and description in coding,
- Using of naming convention of variable and functions,
- Explaining the exceptions handling and conditional checking, and
- Maintaining security.

Testing

- Performing various system testing techniques/strategies to include the different phases of testing,
- Identifying key problems with the software and re-implementation with logical justification, and
- Generating various test reports.

2.3 STEPS OF THE PROJECT

We have listed here five general steps in your Project, which may help you to determine the milestones and regulations related to project work.

- Selection of Topic Area,
- Project Report Planning,
- Project Proposal Report,
- Project Final Report, and
- Project Assessment

2.3.1 Selection of Topic Area

The choice of the topic for the project has great importance so it should be discussed with your counsellor in detail. The topic needs to integrate the interests of the student with the specialised expertise of the counsellor, and be of the appropriate level of difficulty. Students are encouraged to think about the areas of their interest in which they would like to undertake a project, and in consultation with a counsellor, an initial topic can be identified.

2.3.2 Project Report Planning

You should provide an overall aim for your project, which is a declaration of what you would like to achieve at the end of the project. This should be followed by a number of objectives, which act as clearly defined stages that make up your project.

Project Plan

A project plan should be included which provides an estimate of the time that you think you will require in order to work and meet each of your objectives. In your project plan avoid scheduling tasks linearly, try to overlap and tackle more than one task at any point of time.

You should begin planning your report from the day you begin your Mini-project. The report is one of the products of your work, in the same way as a program is a product. You should discuss with your counsellor the way in which your work will be reported. You can produce an outline plan of your report after your first meeting with your counsellor. This plan will not be detailed, but you can gradually increase the amount of detail in the plan until it forms the complete basis for writing the report.

The process of planning can help you sort out your ideas, make vague ideas precise and sequential. One common mistake students make is, to believe that a plan cannot be changed or that it is a sign of weakness to change a plan. A plan is another tool to be used to get work completed according to a satisfactory standard. It needs to be treated with as much respect as any other tool. At first, the plan might be in the list of chapter analyse headings. Next, one or more of the chapters can be broken down into sections, then the sections into subsections, and so on until a whole chapter is ready to be written. You may decide to split a chapter into two or more chapters or to merge two or more chapters into one. This means that you can use your plan to decide when to strengthen your report with some extra work, or when to move on to some new work.

A less common mistake that students make is to think that their report has to be written in order, from the first page to the last. It is wise not to begin writing until you have some level of plan for the whole report, but you can write parts as you go along. For instance, when you have the material for your review of previous work, you can write that chapter. It is quite usual to write the inner chapters before the last chapter and then to write the introductory chapter as the last part you write.

2.3.3 Project Proposal Report

Project proposal should be presented to, reviewed by and agreed upon in consultation with the project counsellor to provide constructive feedback on the proposal and

planned programme of the project work. Further, in this section, you can present details regarding the preparation of the project proposal. The preparation of the Project proposal report may be taken to be an opportunity for students to practice their report writing skills. It is expected that this report will contain an overall structure for the project analysis along with a substantial part of the survey of technologies. The survey of technologies, and associated list of cited references, would be complete at this stage. The project proposal should contribute to some of the content of the final report. It also provides the counsellor with an opportunity to make constructive comments on the written work completed at this stage.

2.3.4 Project Final Report

The final report will contribute to the assessment and your marks. The format of this report will follow the format, guidelines and suggestions given in this block, but details should also be discussed with your counsellor. The final reports of students doing the project in a group should not be identical. Each student should emphasise on his/her role and responsibilities in the project work.

2.3.5 Project Assessment

The project presentation and viva voce also contribute to the final assessment. The presentation will provide an opportunity for the student to set the work done into context and to identify the main features of the work. Student should have good understanding and knowledge of each and every phase of software development. In addition, the student will be expected to defend successfully the conclusions put forward in the report. The examiners will be looking for clear evidence that the student has a depth of understanding of the subject areas.

2.4 GUIDELINES FOR THE PROJECT PROPOSAL

These guidelines on report preparation gives simple and practical recommendation on the problems of getting started, getting organised, dividing the vast task into less difficult pieces and working on those pieces. It includes a suggested structure and a guide to what should go in each section of the project proposal and the project report.

Project Proposal

As we have discussed earlier the project proposal should be prepared in consultation with your counsellor during the counselling sessions. The project proposal should clearly state the objectives of the project. The project work should compulsorily include the analysis phase of the software development.

Front page

The front page of the proposal should contain the project title, followed by your name and your counsellor's name. The contents of this proposal report should contain the following:

Suggested Structure of the Project Report

Preliminary Pages

- Title Page
- Original Copy of the Approved Project Proposal Proforma
- Certificate of Authenticated Work
- Role and Responsibility Form
- Abstract
- Acknowledgement

- Table of Contents
- Table of Figures

CHAPTER 1: INTRODUCTION

This chapter introduces the project context, objectives, scope, and overall structure of the report.

1.1 Background

Provides an overview of the problem domain, motivation for the project, and the technological or application context.

1.2 Objectives

Defines the primary goals and expected outcomes of the project.

1.3 Purpose, Scope, and Applicability

- 1.3.1 Purpose – Describes the intended functionality and goals of the system.
- 1.3.2 Scope – Specifies the functional boundaries and limitations of the project.
- 1.3.3 Applicability – Identifies target users, operational environments, or application domains.

1.4 Achievements

Summarizes the major accomplishments and implemented features of the system.

1.5 Organisation of the Report

Provides an overview of the structure and content of the subsequent chapters.

CHAPTER 2: REQUIREMENTS ANALYSIS

This chapter focuses on the system analysis phase, including identification of functional and non-functional requirements.

2.1 Problem Definition

Clearly defines the problem statement and operational requirements addressed by the project.

2.2 Requirements Specification

Provides detailed functional and non-functional system requirements.

2.3 Software and Hardware Requirements

Specifies the software platforms, development tools, frameworks, libraries, and hardware resources required.

2.4 Preliminary Product Description

Provides a high-level description of system features, operational workflow, and user interactions.

2.5 Conceptual Models

Includes conceptual representations such as UML diagrams, data flow diagrams (DFDs), entity–relationship diagrams (ERDs), or system architecture models.

CHAPTER 3: SYSTEM DESIGN

This chapter documents the **architectural and detailed design of the proposed system**.

3.1 Basic Modules

Describes the major system modules and their functional responsibilities.

3.2 Data Design

- **3.2.1 Schema Design** – Defines database schema structures and relationships.
- **3.2.2 Data Integrity and Constraints** – Specifies data validation rules, integrity constraints, and consistency mechanisms.

3.3 Procedural Design

- **3.3.1 Logic Diagrams** – Includes flowcharts or control flow representations.

- **3.3.2 Data Structures** – Describes data organization and structures used in the system.
- **3.3.3 Algorithm Design** – Explains algorithms implemented to achieve system functionality.

3.4 User Interface Design
Describes the graphical or web-based interface architecture, usability considerations, and interaction design.
3.5 Security Considerations
Discusses authentication mechanisms, data protection strategies, and access control measures.
3.6 Test Case Design
Specifies test cases and validation criteria used for system verification.

CHAPTER 4: IMPLEMENTATION AND TESTING
This chapter details the development, coding practices, and system testing processes.

4.1 Implementation Approaches
Describes the software development approach, frameworks, and programming methodologies used.
4.2 Coding Details: include:

- **4.2.1** Structure of Code
- **4.2.2** Code for database implementation
- **4.2.3** Programs with suitable comments for better readability

4.3 Testing Strategy

- **5.3.1 Unit Testing** – Testing of individual modules.
- **5.3.2 Integration Testing** – Testing interactions between modules and subsystems.

4.4 Modifications and Improvements
Documents changes, refinements, and improvements made during system development and testing.

CHAPTER 5: RESULTS AND DISCUSSION
This chapter presents the evaluation of the implemented system and interpretation of results.

5.1 Test Reports
Provides test outcomes, validation results, and performance observations.
5.2 User Documentation
Includes operational guidelines, system usage instructions, and user manuals.

CHAPTER 6: CONCLUSION
This chapter summarizes the overall findings and outcomes of the project.

6.1 Conclusion
Provides a summary of achievements and system capabilities.
6.2 Limitations of the System
Discusses constraints, assumptions, and limitations encountered during development.
6.3 Future Scope of the Project
Identifies potential enhancements, scalability considerations, and opportunities for further research or development.

Supporting Sections

- **References** – List of academic papers, books, standards, and online resources cited.
- **Glossary** – Definitions of technical terms and abbreviations used in the report.
- **Appendix A** – Supplementary materials such as detailed diagrams, datasets, or extended documentation.

- **Appendix B** – Additional supporting information such as source code snippets, configuration details, or test case documentation.

The description of these topics are already explained in the previous section, however, the references at this stage of the project proposal may be different from the references of the project report, so you should provide here a list of reference which is related to your project topic: literary and the review, survey of technologies that acts as a good reference to your work. This will demonstrate that your project is current and it is supported by articles, papers or books, which are accessible. (Refer to the reference in further sections). After finalising the proposal report, students should submit the project proposal report along with the proforma.

2.5 GUIDELINES FOR THE PROJECT REPORT

When you are about to begin writing the project report, it seems a lengthy, complicated job. It will seem less discouraging if you write continuously and prepare the documentation of each phase from the start of the project. However, in this case, towards the conclusion, you will even find, an enjoyment, satisfaction in the sense of achievement and pleasure in the enhancement of your technical writing. Let us look at how you should make a start.

Before writing your project report, first write the report outlines containing chapter headings, sub-headings, some figure titles and perhaps some other details, notes and comments. The report should contain a full and coherent account of your work. Although there will be an opportunity to present your work verbally, and demonstrate any software, the major part of the assessment will be based on the written material in your project report.

Project Report Format

The project report documentation should contain 80 to 100 pages for analysis, design, and testing phases, however, the size of complete report may vary depending upon the size of coding /implementation and appendices. The project documentation details should not be too generic in nature. To be more specific, whatever theory in respect of these topics is available in reference books should be avoided as far as possible. The project documentation should be in respect of your project only. You should make sensible use of appendices. For example, software user instructions, detailed code listings, correspondence may be relegated to appendices. Note that spiral bindings are not suitable for handing in the project report.

Project Report Layout

Project report should contain all the details and text should be short and concise, lengthy reports may not be qualitative, and care should be taken to edit the material sensibly. The project report should normally be printed with single line spacing on A4 paper (one side only). Figures should be clearly drawn and all material should be reproducible by normal photocopy. All pages, tables and figures must be numbered and figures should have titles. Detailed information about the layout for the project proposal and report are also listed below:

Font size and margin

1. The report is to be bound with a clear front cover.
2. The text is in 12-point Times New Roman font.

3. The pages are of A4 size, with margins as given below, except for the front cover, which has a specific format given in section 1. Margins of pages should follow the following specifications.

   a. Left margin - 2 inch. from edge of paper.
   b. Right margin - 1 inch. from edge of paper.
   c. Top margin - 1 inch. from edge of paper.
   d. Bottom margin - 1 inch. from edge of paper.

4. The above margins shall be observed on charts, graphs, tables, and drawings. Folded papers or large size paper will not be accepted unless there is absolutely no other way for the material to be presented.

Heading

1. Headings used in the project report should follow the following convention:
2. Main Headings or Chapter Headings

   a. Times Roman, 16 Font size (1,2,3 etc.) numerals.
   b. Capital and Bold.
   c. Must begin a new page and be centered.
   d. Main headings are to be titled names that reflect content of the text that follows. Main headings are not to be identified as chapters.
   e. The number of the headings shall be followed by a period and two spaces.
   f. Must precede the following text material by second heading by three spaces.

3. Second Headings

   a. Times Roman, 14 Font size, Bold, 2.1, 2.2, 2.3, etc.
   b. Must be centered and be typed in capital and lower case (sentence case) letters; i.e., the first letter of each word except conjunctions, prepositions, and articles must be a capital letter. Omit period at the end of the heading.
   c. The letter designation of the heading shall be followed by a period and two spaces.
   d. Must be four spaces below preceding text and three spaces ahead of succeeding text.

4. First sub-headings

   a. Times Roman, 12 Font size, Bold, 2.2.1, 2.2.2, etc.

   b. Must be typed on separate lines beginning at the left margin line of the text, but need not begin on a new page.
   c. Must be typed in capitals and lower case letters except conjunctions, prepositions, and articles.
   d. The number designation of the heading shall be followed by a period and two spaces. Omit period at the end of the heading.
   e. Must be separated from the succeeding text by three spaces.

5. Second sub-headings (second sub-headings should be avoided if possible).

   a. Times Roman, 12 Font size, Bold.
   b. Must be typed on the same line as the text it introduces beginning at the left margin line of the text.
   c. Must be typed in sentence case.
   d. Must be followed by a period at the end of the heading and must be underscored by a line.
   e. The letter designation shall be followed by a period and two spaces.

- Appendices re-start the section numbering, using capital letters as section labels and Arabic numerals as sub-section labels (i.e., A.1, A.2,); appendix headers are in decreasing-sized fonts.
- If a section is divided into sub-sections, it has at least two subsections. Similarly for subsections divided into sub sub-sections, and so on.
- The font matter, Conclusions, Recommendations, Glossary, Acknowledgements, and References sections are not divided into sub-sections. (Include in Main Heading or Chapter Heading).

Figure and Tables

- Each figure has a number and a caption below the figure. As given in the example of a *Figure*.

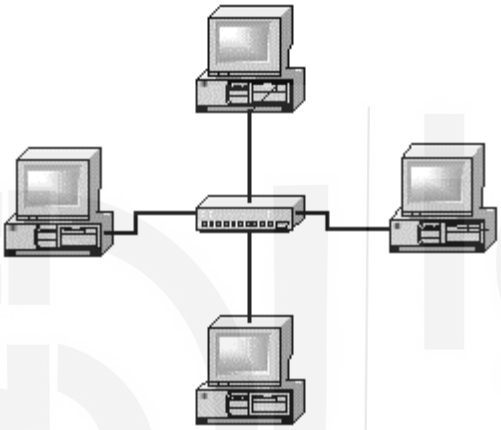


Figure 1: A typical computer network

- Each table has a number and a title above the table. As given in the example of a *Table*.

Table 1: Comparison of various data structures

Operation	Sequential List	Linked List	AVL-Tree
Search	O(log n)	O(n)	O(log n)
Delete	O(n)	O(1)	O(log n)
Insert	O(n)	O(1)	O(log n)

- Figure and table numbering restarts at the beginning of each appendix, using a combination of the appendix label and figure/table number within the appendix (e.g., A-1, A-2).
- Each figure and table is cited (referred to by number) in the report text, either on the same page as the figure/table or on the preceding page.
- Figures and tables are legible.

Paragraphs

Paragraph indentations must be uniformly eight letter spaces long. Series of paragraph items which are to be listed without headings under any of the regular headings may,

for clarity be designated as follows: (A), (B), (C). No period is necessary between the parenthesis and the first sentence. Series of items listed in the interior of a paragraph may be designated as follows: (a), (b), (c). A period must not follow the parenthesis. Each item must begin with a lower case letter and must end with a semi-colon, unless it is a complete sentence. A new paragraph must not begin at the bottom of a page if there is not sufficient space for at least two lines.

Footnotes

Footnotes should be used only if absolutely necessary.

- Footnote references shall be indicated in the text by an Arabic number placed superior to the text and immediately following the word phrase or sentence, on which the footnote is to be appended.
- Footnotes shall be sequential for each page and for the entire report.
- Footnotes shall be placed at the bottom of the page on which they are indicated. They shall be indented from the margin line of the text by eight spaces and placed under a broken line made of 15 dashes.
- Footnotes shall be single spaced typing.

Pagination

Each page in the report is expected to bear a number. Only one side of the paper may be used. The following plan should be used exclusively:

a. The preliminary section, including the title page; copyright page, if any; foreword, preface, or acknowledgements; table of contents; etc., should be numbered, using lower case Roman Numerals, e.g., i, ii, iii, etc. The title page counts as Page i, but the number does not appear. The sequence of the preliminary section is as follows:

Introduction	Page i - number does not appear
Survey of Technologies…….	Page ii, iii, as necessary
Requirements and Analysis	Page iii, iv, as necessary
References .....…….	Page v, vi, as necessary

For the remainder of the report Times numbers are used. Each page must be numbered. Page numbers are to be placed two centimeters from the top and right hand margins on the pages. Include all pages for illustrations, tables, appendices, bibliography, etc. The numbering in the main body of the report should begin with Page 1 and run consecutively to the last page. No punctuation, such as dash or a period, should accompany the page number.

- Page numbering restarts at the main body of the report: pages in the main body and back matter, including appendices, are numbered using Arabic numerals, with the first page of the Introduction as page one.

- Page numbers are centered at the bottom of the page.

Specially Designated Expressions

(1) Specially designated expressions usually mean equations, formulas, etc.

(2) Specially designated expressions shall be centered on the page shall be set below the preceding text and before the succeeding text by three line spaces.

(3) The expressions shall be identified by an arabic number in parenthesis placed opposite the expression and in line with the right margin of the text. They

should be numbered in each chapter in the order of their appearance together with the chapter number, e.g., (6.14).

References

The numerical reference of the material shall be indicated in the text by an arabic numeral in square brackets (e.g., [12]) placed in the text immediately following the name, word, phrase, or sentence which the reference concerns (in most cases this will be the author's name). The number in parenthesis should indicate the order of appearance of the reference in the text. The listing of references in the references shall be in the order in which they are used in the text and shall bear the same number as was used in the reference in the text.

It is very important that you acknowledge any of the work of others that you use or adapt in your own work, or that provides the essential background or context to your project. The use of references is the standard way to do this. Please follow the given standard for the references of books, journals, and online materials.

The list of cited references is placed between the main text and the appendices. This section should start on a new page, and should not have a chapter or section number, just the heading "References". Each reference should be as complete as possible. However, some schemes for writing references are given below:

- **A Journal Paper**

  name(s) of author(s), year of publication, title of paper, name of journal, volume number issue number (optional), page numbers.

- **A Conference Paper**

  name(s) of author(s), year of conference, title of paper, name(s) of editor(s), name of conference, place of conference (optional), publisher of proceedings, place of publication, page numbers.

- **A Book**

  name(s) of author(s), year of publication, chapters (if only part of the book is relevant), title of book, name of publishers, place of publication (optional), page numbers.

- **A Web Page**

  name(s) of author(s), year of publication, title of paper, url, date last accessed.

Appendices

These should contain detailed information not required on a first reading of the main text, but necessary for closer study of the project and in particular its continuation or replication. The Appendices should also include a program disk/cd when appropriate.

Coding style

In general during coding, most of your development time is spent in debugging, troubleshooting, and practicing general maintenance, especially on larger projects. Even when you work on small projects, a significant amount of time is usually spent analysing and fixing code. The readability of your code is important for your benefit

and the benefit of your team members. When following conventions, you increase readability, which increases workflow and helps find and fix any errors in your code, and all programmers follow a standardised way of writing code, which improves the project in many ways. For C, this involves things like brace placement, indentation, and the way parentheses are used but the coding conventions and style varies from language to language hence, you should follow the coding conventions of the language, which you have used in your project implementation.

Style of English

The report should be written in an objective style, in the third person impersonal tense: e.g., "The following software was developed...."not "I developed the following software....". An impersonal style keeps the technical factors and ideas to the forefront of the discussion and you in the background. Try to be objective and quantitative in your conclusions. For example, it is not enough to say vaguely "because the compiler was unreliable the code produced was not adequate". It would be much better to say "because the A compiler produced code which ran 2-3 times slower than a fast enough scheduler could not be written using this algorithm". The second statement is more precise, clear and informative and gives an impression that the student knows the project and subject very well.

Submission of the Project Report

You have to submit your project report by the given date. One copy of the original project report is to be submitted to the Study Centre concerned. A photocopy of the same project report must be retained by the student, which should be produced before the evaluation.

2.6 ASSESSMENT GUIDELINES

In this section we have given a few general parameters as checkpoints for the assessment of any software development project. You can note these points and emphasize them during the project report preparation and examination. Basically, assessment will be based on the quality of your report, the technical merit of the project and the project execution. Technical merit attempts to assess the quality and depth of the intellectual effort you have put into the project. Project execution is concerned with assessing how much work you have put in.

Analysis

You may know that the software requirement specification (SRS) document is one of the important documents of your project. The indicators for SRS document is whether you have used some standardization like IEEE standards or any other international standard, or whether your SRS has a proper structure based on sound software engineering concepts as given in section 3 or it is given in a running text. Project analysis for DBMS/Application development projects should contain the ER diagram, Data Flow Diagram and other relevant diagrams, so you should include these with the following requirements. However, for other categories of project you should prepare class diagrams, behavior model and /or state transition diagram and details of various data structures used.

- The Entity Relationship diagram (ER Diagram) should have:
    - Proper symbol of attributes, entities and relationship, and
    - Relationship of ER diagram to SRS with strong association
- Data Flow Diagram (DFD) should have:

- All Data flow should be levelled and should have proper input and output.
- Relationship of data flow to data dictionary Context Diagram, Level 1 and Level 2.

- **Data Dictionary:** It should explain each entity and relationship in ER diagram and data flow in DFD.

Design

Project design should include the desired features and operations in detail, including user interface design, program structure, schema design and normalised tables and data integrity and constraints. You should include them with the requirements given below:

- **Program Structure:** It should have the proper modularisations of software and specification of each module.
- **Schema Design and Normalised Tables:** Normalise the table to minimum 3NF. If any demand of Demoralisations clearly explain the reasons.
- **Data Integrity and Constraints:** Explain the referential diagram. Define entity integrity, which should include keys, alternate keys and other keys, value constraints and ranges.
- **Procedural Design:** Explain using Flowchart / Pseudo code / PDL.
- **User Interface Design:** Coherence with dataflow and processor; Consistency of interface and naming convention.
- **Architecture:** Program architecture and explanation on suitability of data structure used.

Coding

Coding phase of software development includes different activities like refining the algorithms for individual components, transferring the algorithms into a programming language (coding), translating the logical data model into a physical one and compiling and checking the syntactical correctness of the algorithm with these activities. You should include the comments and description in code, use the standardisation in coding, use the methodology for error handling and security implementation. These parameters ensure software quality and reliability. You should include them with the requirements given below:

- **Comments and Description:** Should have comments with functional description which include the input, output, total function calls to/from other functions, function parameters, description of main variables, Data type, logic description, etc.
- **Standardisation of Coding:** Use of naming convention of variable and functions, nested depth, naming constant, use of data structure and style.
- **Error Handling:** Explain exceptions handling and conditional checking.
- Parameter passing and calling: Check the technique used for this purpose, how it optimises the coding.
- **Security:** Maintain confidentiality, integrity and authorisation according to the requirement and needs of the system. Also maintain database level security, use of Views, use of revoke and grant, user and access rights and ensure steps taken against hacking of the system.

Testing

Testing is a process of devising a set of inputs to a given piece of software that will cause the software to exercise some portion of its code. The developer of the software can then check if the results produced by the software are in accordance with his or her expectations. It includes, number of activities such as correcting syntactically and semantically erroneous system components, detecting as many errors as possible in the software system, and assuring that the system implementation fulfills system specification.

It ensures the quality, efficiency and reliability of the software, which is measured by the testing methodology and techniques used for section, integrated and system testing. You should give all the reports and results of test cases for section, integrated and system testing. How debugged your code is and what actions you have taken too improve the code, must, be explained. Good testing can be measured by criteria such as correctness, reliability, user friendliness, maintainability, efficiency and portability of software.

Organisation of report

Report organisation improves the professional attitude of writing reports. You should emphasise on the proper binding of the project report, the cover page, page numbering, organisation of content, and proper printout of text and images.

Viva Voce

Student may be asked to write code for problem during viva to demonstrate his coding capabilities and s/he may be asked to write any segment of coding given in project report.

2.7 PITFALLS AND SOME TIPS

The main purpose of this course is to gain experience with the help of the concepts and methods learned from the previous courses particularly, in knowing the practical situation/problem in software development. We have explained most of the components of the project in detail, however you may need some tips that may be helpful and are generally required by students.

In recent years, we have noticed, that projects suffer from the lack of proper analysis and later any review, failure to implement an engineering approach and lack of critical element. A proper analysis and literary review is necessary for you to show that you can place your work in the wider context of computing. An engineering approach is one where you follow some appropriate process or methodology that leads from requirements to design to implementation and testing. By adopting such a process, it is much less likely that, you will fail to take some crucial factor into consideration, which is an important aspect of software engineering.

The critical element involves showing what you independently believe to be good and bad about what you've read, what you have been taught, what you have been asked to do, what you have done, what you have not done and the consequences thereof. It does not involve blindly accepting as fact everything that you have been told or read. However, we have listed a few points that generally are found to be lacking in the projects submitted by our students. These observations may be used by students as valuable feedback.

Sometimes students opt for the wrong project problem/topic without, understanding the depth of the requirements, and their own limitations in that area.

- Few students are over enthusiastic and they don't freeze the requirement or expectations till, the coding and implementation leads to low quality and less reliable software product.
- Most of the students assume that the coding is the only work they have to do and they neglect the importance of other software engineering processes that result in improper knowledge gain of project development and poor results/grades.
- Generally students lack critical work. They assume whatever is given in books or taught to them in software development is correct.
- Most of the students do not use the software engineering approach and methodologies for software development.
- Most of the students do not use standardisation in different phases of software development.
- Some students do not prepare documents such as the SRS, SDD, and Test Cases etc. in different phases of software development.
- Some students never consider the importance of notes and documentation, which creates problems for them during report writing.
- Generally students don't give proper acknowledgement of material used in the reference however, few are not aware of material which should be acknowledged in the reference. You must include the following material in the reference.
    - Written, printed, or published (electronic or paper both) material.
    - The algorithms, models, hypotheses and paradigms that are used.
    - The software, which you used, including the development environment, compilers, libraries, etc. you should given the web reference of the website so that other student can collect information about it.
- Most of the students are interested in developing website projects or some system development project, but very few student are involved in research oriented work.
- A few students are too dependent on the counsellor. Remember you are responsible for coding and documentation of your project.
- A few students also copy the projects from some sources. This practice creates a major problem, as it does not encourage the student's own creativity/ knowledge/potential.
- There is no need to write a detail history of Java or visual basic etc. On the other hand, it is important to mention what you have used in the project and to explain anything that might be unusual to the reader. For example, five years ago it was important to introduce the reader to Java, but now it is not required.
- Students may think, "So what if it's wrong as long as the meaning is clear". But poor spellings leave a negative impression (of carelessness) and sometimes cause confusion.

Report Format and Layout

In the previous sections, we have explained the specification of project report format and layout however, it is also important to discuss additional tips for the preparation of the report. That's why we have given you a few tips and points as given below:

- Use proper spelling and capitalisation of the programming languages, tools and other proprietary software that you have used. For example, some students write Unix and some writes UNIX, but which one of the two is correct?
- Similarly, do not leave a space before a full stop or other punctuation mark.
- Also, do not leave a space after an open bracket or before a closed bracket. You should do it (like this).
- Use round brackets only for parentheses. Keep square brackets for references.
- Commas and full stops should be placed inside quotation marks. Even at the end of sentences like "this."
- It is a good practice to leave two spaces after a full stop.
- Spell out numbers (from one to nine) in text, i.e., write "two" not "2" except where you are numbering bullet points. For number 10 and above, use numerals. Remember numbers should be spelled out when they begin a sentence.
- Spell out per cent; do not use %, and write per cent as two words without a period.
- Do not use the pronoun "I", always use "We", better use passive voice.

2.8 SUMMARY

In this section, an effort has been made to address the major technical and structural aspects involved in the preparation of a comprehensive project report**.** The section emphasizes the importance of systematic technical documentation and encourages learners to devote adequate effort to the preparation of a well-organized and professionally structured report. To facilitate this, a set of recommended guidelines and formatting standards for the layout, organization, and presentation of project reports has been outlined.

These guidelines are intended to promote consistency, clarity, and technical rigor in project documentation and are progressively evolving into accepted standards for project report preparation within the programme. The recommendations presented in this section may continue to be refined, updated, and enhanced based on emerging practices, technological advancements, and accumulated academic and professional experience in software project development and documentation.

SECTION 3: SOFTWARE ENGINEERING CONCEPTS AND STANDARDS

Structure **Page Nos.**

3.0 INTRODUCTION

Software Engineering is a discipline concerned with the systematic development, deployment, and maintenance of software systems. Therefore, a clear understanding of the fundamental concepts of software and software development methodologies is essential prior to initiating any project work. Before commencing the development phase of a project, it is important to revisit and reinforce foundational knowledge related to software engineering principles and practices.

It is important to distinguish between programming and software engineering. Programming primarily involves the implementation phase, specifically writing source code to solve computational problems. In contrast, software engineering encompasses the entire software development life cycle (SDLC). This includes activities such as requirement analysis, feasibility analysis, system design, coding, documentation, software testing, quality assurance, deployment, and long-term maintenance. Thus, software engineering adopts a holistic and process-oriented approach toward software development.

One of the most critical components of the software development process is the creation and maintenance of documentation. Documentation refers to the systematic recording of information related to the design, functionality, implementation, and usage of a software system. It facilitates effective communication among developers, stakeholders, and end users, and plays a vital role in ensuring that the system can be understood, operated, and maintained efficiently. Comprehensive and accurate documentation significantly contributes to the overall success and sustainability of a software system.

Documentation activities are integrated into every stage of the system development process, often beginning even before formal documentation procedures are initiated. Throughout the development lifecycle, various reports, technical notes, and supporting materials are prepared to capture design decisions, development progress, and system specifications. These records assist the development team and other stakeholders in understanding the system architecture and the processes involved in its development.

Software documentation can be broadly categorized into several types, including requirements or analysis documentation, design documentation, interface

documentation, internal program documentation, and user-oriented documentation. Each category serves a distinct purpose in describing different aspects of the software system. This section focuses on the methodologies and best practices for developing these documentation artifacts, in the context of this project course, the focus extends beyond merely writing code.

The objective is to gain practical exposure to real-world software engineering practices, including handling development complexities, addressing practical challenges, and applying systematic engineering principles to build robust and well-documented software systems.

3.1 OBJECTIVES

After going through this section, you should be able to:

- learn about various documents and process of documentation;
- understand application of various standards of documentation processes;
- differentiate between various documentation processes;
- understand the relation between documentation and quality of software, and
- understand the good practices for documentation process.

3.2 ANALYSIS

Analysis describes the "What" of a system. The objectives that are to be achieved in Software process development, are the requirements. In the requirements analysis phase, the requirements are properly defined and noted down. The output of this phase is SRS (Software Requirements Specification) document written in the natural language. According to IEEE, requirements analysis may be defined as (1) the process of studying user's needs to arrive at a definition of system hardware and software requirements (2) the process of studying and refining system hardware or software requirements.

The main activities in the analysis phase are cost estimation, project planning and scheduling, feasibility report preparation, and the development of SRS (software requirement specifications) document of the project.

System Analysis Report : You may include the following contents in the system analysis report given in the *Table* shown below:

Topic	**Some of the contents of the topic**
Introduction	Include the project background and report layout here.
Terms of Reference	Define the expectations of the project, the Time Frame and Available Resources
Existing System	Define here the results of Fact Finding, working of the Current System and the problems in the Current System
System Requirements	Give the system requirements here, after discussion with the System User
Proposed System	Define the outline of the Proposed System, key Input and Output to and from the system

Normally IEEE standard is followed. A typical format of SRS is as follows:

TABLE OF CONTENTS

Introduction

- Purpose
- Scope
- Definition
- Product and its function
- Benefits and Goals
- Overview.

Overall Description

- Product Description
- Product Functioning
- Functions of Project
- Users of Project
- Assumptions made.

Specific Requirements

- Interface Requirements
- User Requirements
- Hardware Requirements
- Software Requirements
- Logical Database Requirements.

Basic Processing Actions of the System
Appendices

- Input/Output Formats
- Instruction for Security
- Results of Fact Finding
- Data Model
- Functional Model
- Process Specification.

A sample portion of SRS is given below:

INTRODUCTION

Purpose

SRS contains details of the proposed software system, sufficient enough for the designers to design the system. Thus, SRS is a means of communicating the findings of the analysis stage to the design stage. The SRS includes:

Interface,

Logical Database,

Hardware, and

Performance and other constraints.

It also contains the assumptions made by the analyst and any systemic dependency.

Scope

The scope of SRS contains all the areas related to the project. The scope of SRS includes:

Proposed software description,

Users of the proposed software,

Functional requirements of the software,

Assumptions and dependencies in the system, and

Constraints.

Definition

……..

Product and its function

……..

Benefits and Goals

……..

Overview

..……

Overall Description

Product Description

The Client should be able to upload the raw data in Excel files into the Database. The raw data is then validated using ……

Product Functioning

The Raw data from the Clients is put into the database.

……..

Functions in the Project

There are five functions of the system.

User Verification

The User is asked to enter the Username and Password. The system checks the validity of Username and Password, otherwise the User is asked to do so again. A maximum of three valid attempts are given to the user.

Upload Raw Data

…….

Validate Data

…….

Put the Validated Data

…….

Generating Reports

……..

Users of the Product

……

Assumptions

……

SPECIFIC REQUIREMENTS

Interface Requirements

The Interface requirements include:

Easy to follow Interface,
Very few graphics,
No hidden buttons, and
Relevant error messages.
…….

User Requirements

After a careful study of the requirements of the Client, analysts have come up with a set of requirements.
……..

Hardware and Software Requirements

There are three environments that have been created for the project, viz.,

Development environment,
Quality Control environment, and
Production environment.

The hardware requirements for all the platforms may be indicated here.

Logical Database Requirements

The following information is to be stored in the database.

The Clients Raw data,
The Clients Validated data, and
Username and Password.

BASIC PROCESSING ACTIONS OF THE SYSTEM

The basic processing actions of the system are.

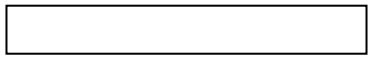
Verification of the User
……..
Upload Data

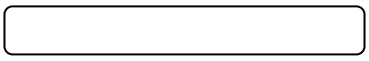
APPENDICES

APPENDIX A

INPUT / OUTPUT FORMATS

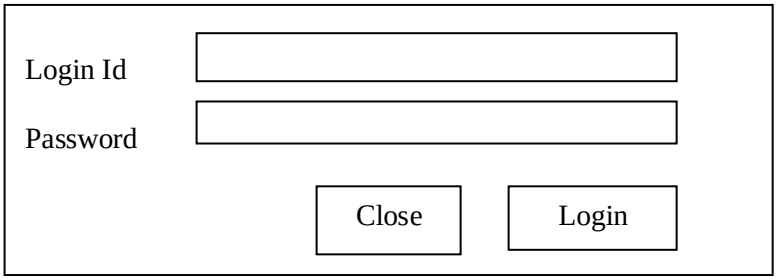
The input formats for the system contains the following screens.
The convention used while making the input formats are.


Square box is used for user input


Rounded Square box is used for system display

Login Screen

The following screen that inputs the Username and Password from the User for authentication of the User to the system is:



APPENDIX B

Instructions For Security

Security is an integral part of any system. Clients send their confidential data with trust and it is our foremost duty to protect the security and privacy of the data of the Clients.

……

APPENDIX C
RESULTS OF FACT FINDING

Working of the Current System and its Problems

APPENDIX D

DATA MODEL

Classes involved in the Project

Clients

Excel Data Files

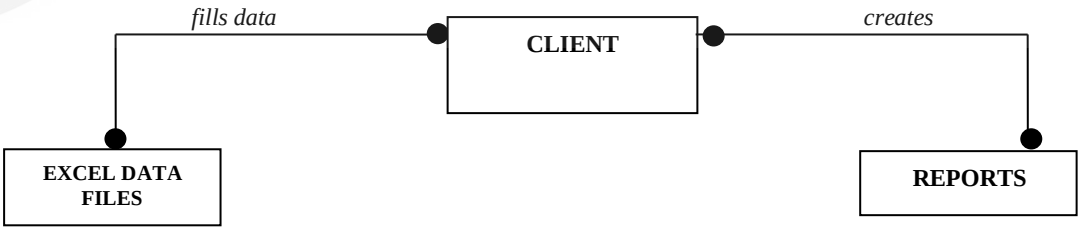
Reports

Association between the classes

Clients fill data in the Excel Data Files (M .. M)

Clients generate Reports (M … M)

E-R/Object Diagram for the System



Attributes of the Entities are:

Object Classes	Attribute
Clients	number name address phone number fax email

Excel data Files	excel file number client number
Reports	report number report name client number

FUNCTIONAL MODEL

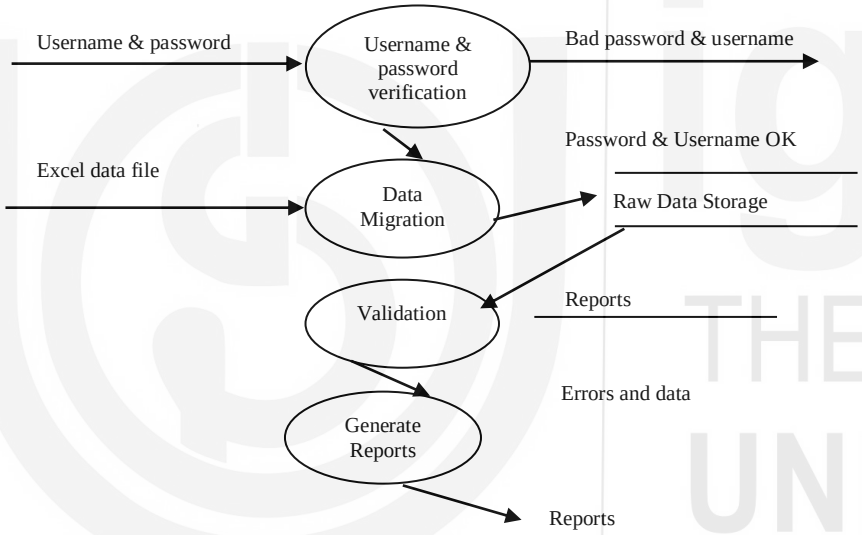
One sample DFD at level 1 contains the following processes:

Verification of username and Password

Data Migration,

Validation of Migrated data, and

Generating Reports.
You can make various levels of DFDs



Process Specification

Let us give one sample process verification.

Process: Username and Password Verification

Input: Username & Password

Output: Access granted or denied message

Processing

The Client enters the Username and Password and clicks the Login button.

The system connects with the DBMS and verifies them in the related database. If both are valid: allow user to enter the system with the allowed access rights for that user. Otherwise prompt wrong Username-Password message and take the user to the screen where s/he can re-enter the Username-Password.

Interface Description

The interface contains the text boxes where the user can enter Username and Password. It also contains a Login button for login and a Close button on closing the application.

Internal Data Structure

The process uses the encrypted username and password table that also contains information on access permission.

3.3 DESIGN

Software design is all about developing the blue print for designing workable software. The goal of a software designer is to develop models that can be translated into software. Unlike design in civil and mechanical engineering, software design is a new and evolving discipline contrary to classical building design etc. In early days, software development mostly concentrated on writing codes. Software design is central to the software engineering process. Various design models are developed during the design phase. The design models are further refined to develop detailed design models, which are closely related to the program.

Software Design Process

Software design is the process of applying various software engineering techniques for developing models in order to define a software system, which provides sufficient details for the actual realisation of the software. The goal of software design is to translate user requirements into an implementable program.

Software design is the only way through which we can translate user requirements to workable software. In contrary to designing, a building software design is not a fully developed and mature process. Nevertheless the techniques available provides us with tools for a systematic approach to the design of software. *Figure 6* depicts the process of software design.

Figure 6: Process of software Design

During the process of software design, the information model is translated in to data design. Functional model and behavioural model are translated to architectural design, which defines major component of the software. Keeping in view the importance of design, it should be given due weightage before rushing to the coding of software.

Software design forms the foundation for implementation of a software system and helps in the maintenance of software in the future too. Software quality and software design process are highly interrelated. Quality is built into the software during the design phase.

High level design gives a holistic view of the software to be built, whereas low level refinements of the design are very closely related to the final source code. A good design can make the work of programmer easy and hardly allows the programmer to forget required details. Sufficient time should be devoted to the design process to ensure good quality software.

The following are some of the fundamentals of design:

The design should follow a hierarchical organisation,

Design should be modular or logically partitioned into modules, which are relatively independent and perform independent task,

Design leading to interface that facilitates interaction with the external environment,

Step-wise refinement to more detailed design, which provides necessary details for the developer of the code, and

Modularity is encouraged to facilitate parallel development, but, at the same time, too many modules lead to the increase of effort involved in integrating the modules.

System Design Specification

The system design specification or software design specification as referred to, has a primary audience, the system implementer or coder. It is also an important source of information the system verification and testing. The system design specification gives a complete understanding of the details of each component of the system, and its associated algorithms, etc.

The system design specification documents the requirements of the system be implemented. It consists of the final steps of describing the system in detail before coding begins.

The system design specification is developed in a two stage process. In the first step, design specification generally describes the overall architecture of the system at a higher level. The second step provides the technical details of low-level design, which will guide the implementer. It describes exactly what the software must perform to meet the requirements of the system.

Tools for Describing Design

Various tools are used to describe the higher level and lower level aspects of system design. The following are some of the tools that can be used in the System Design Specification to describe various aspects of the design.

Data Dictionary

Definition of all the data and control elements used in the software product or sub-system. Complete definition of each data item and its synonyms are included in the data dictionary. A data dictionary may consist of description of data elements and definitions of tables.

Description of Data Element:

Name and aliases of data item (its formal name).

Uses (which processes or modules use the data item; how and when the data item is used).

Format (standard format for representing the data item).

Additional information such as default values, initial value(s), limitations and constraints that are associated with the data elements.

Table Definitions

Table name and Aliases.

Table owner or database name.

Key order for all the tables, possible keys including primary key and foreign key.

Information about indexes that exist on the table.

Database Schema: Database schema is a graphical presentation of the whole database. Data retrieval is made possible by connecting various tables through keys. Schema can be viewed as a logical section from the programmer's point of view.

E-R Model: Entity-relationship model is database analysis and design tool. It lists real-life application entities and defines the relationship between real life entities that are to be mapped to the database. E-R model forms the basis for database design.

Security Model: The database security model associates users, groups of users or applications with database access rights.

Trade-off Matrix

A matrix that is used to describe decision criteria and relative importance of each decision criterion. This allows comparison of each alternative in a quantifiable term.

Decision Table

A decision table shows the way the system handles input conditions and subsequent actions on the event. A decision table is composed of rows and columns, separated into four separate quadrants**.**

Input Conditions	Condition Alternatives
Actions	Subsequent Action Entries

Timing Diagram

Describes the timing relationships among various functions and behaviours of each component. They are used to explore the behaviours of one or more objects throughout a given period of time. This diagram is specifically useful to describe logic circuits.

State Machine Diagram

State machine diagrams are good at exploring the detailed transitions between states as the result of events. A state machine diagram is shown in *Figure 7*.

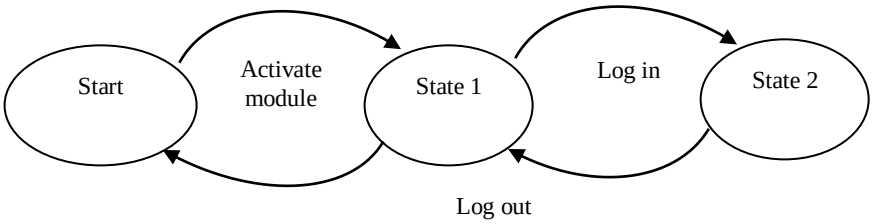


Figure 7: A state machine diagram

Object Interaction Diagram

It illustrates the interaction between various objects of an object-oriented system. Please refer to *Figure 8.* This diagram consists of directed lines between clients and servers. Each box contains the name of the object. This diagram also shows conditions for messages to be sent to other objects. The vertical distance is used to show the life of an object.

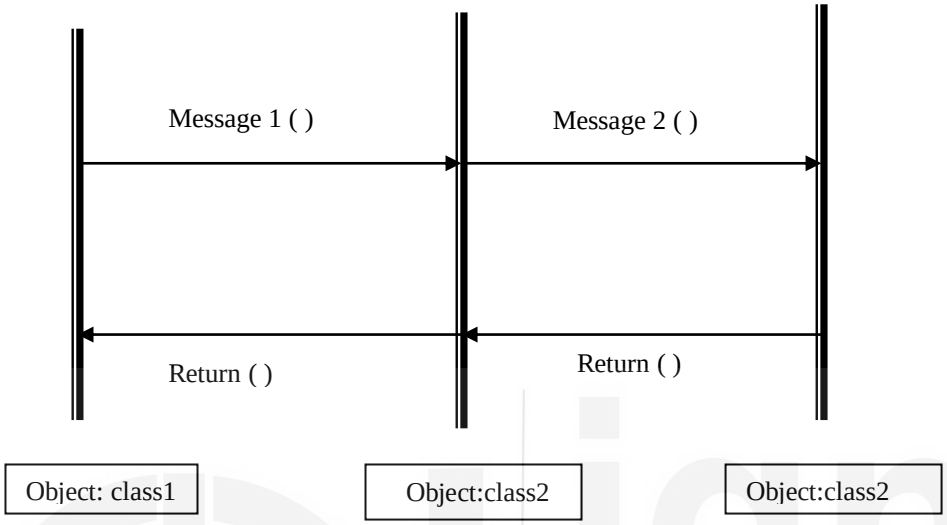


Figure 8: An object interaction diagram

A Flow chart shows the flow of processing control as the program executes. Please refer to *Figure 9.*

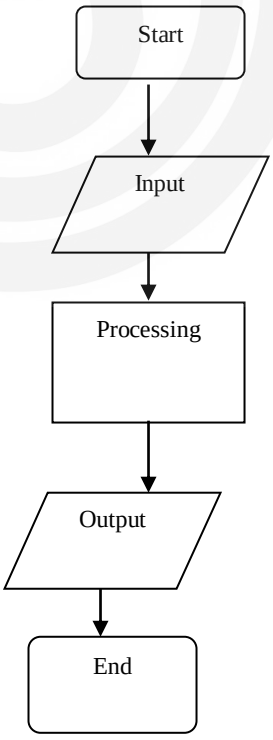


Figure 9: Flow chart

Inheritance Diagram

It is a design diagram work product that primarily documents the inheritance relationships between classes and interfaces in object-oriented modeling. The standard notation consists of one box for each class. The boxes are arranged in a hierarchical tree according to their inheritance characteristics. Each class box includes the class name, its attributes, and its operations. *Figure 10* shows a typical inheritance diagram.

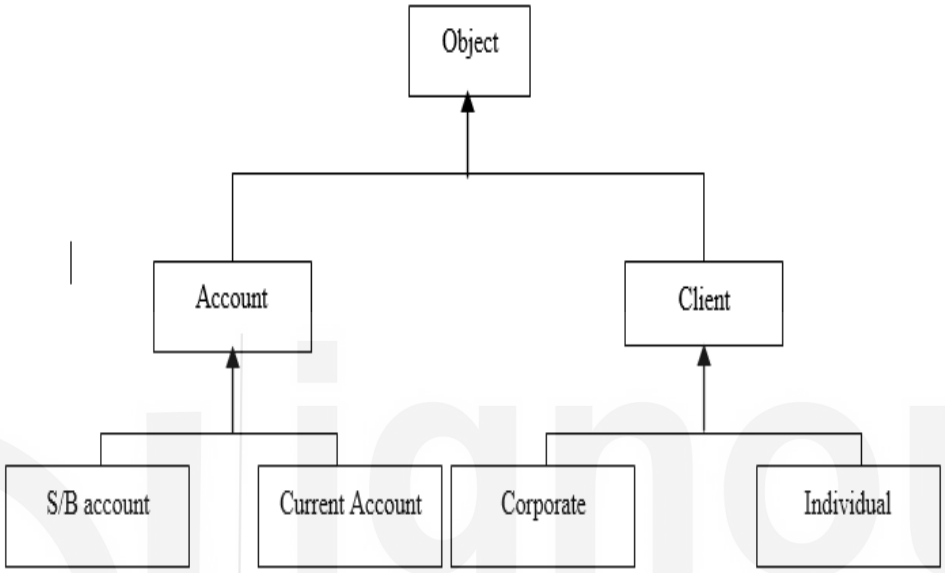


Figure 10: A typical inheritance diagram

Aggregation Diagram

The E-R model cannot express relationships among relationships. An aggregation diagram shows relationships among objects. When a class is formed as a collection of other classes, it is called an aggregation relationship between these classes. Each module will be represented by its name. The relationship will be indicated by a directed line from container to container. The directed line is labelled with the cardinality of the relationship. It describes "has a" relationship. *Figure 11* shows an aggregation between two classes (circle *has a* shape).

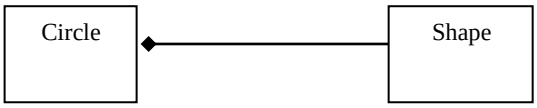


Figure 11: Aggregation

Structure Chart

A structure chart is a tree of sub-routines in a program (Refer to *Figure 12*). It indicates the interconnections between the sub-routines. The sub-routines should be labelled with the same name used in the pseudo code.

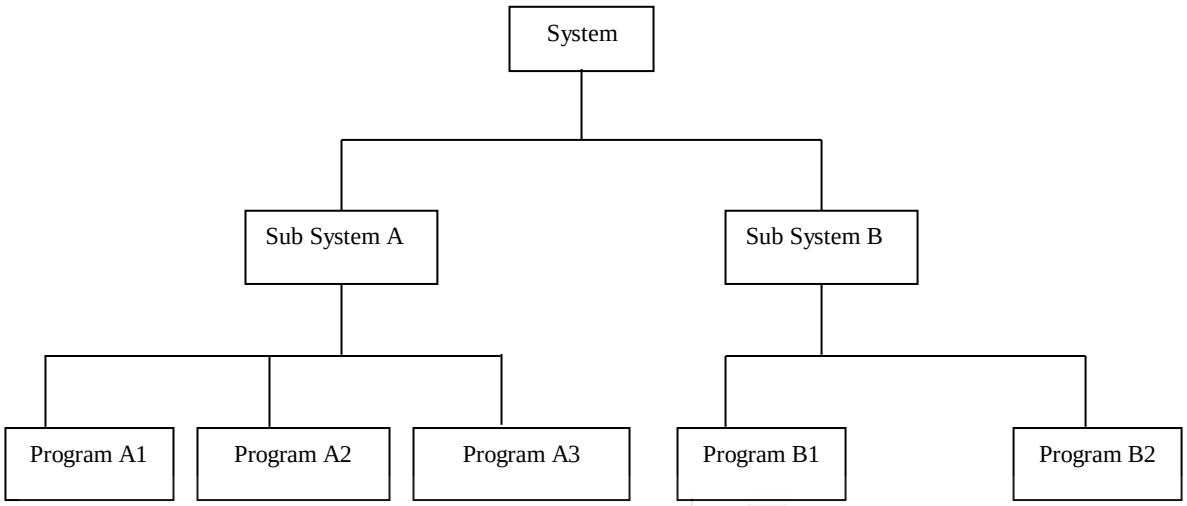


Figure 12: A structures chart

Pseudocode

Pseudocode is a kind of structured English for describing algorithms in an easily readable and modular form. It allows the designer to focus on the logic of the algorithm without being distracted by details of language syntax in which the code is going to be written. The pseudocode needs to be complete. It describes the entire logic of the algorithm so that implementation becomes a routine mechanical task of translating line by line into the source code of the target language. Thus, it must include flow of control.

This helps in describing how the system will solve the given problem.

"Pseudocode" does not refer to a precise form of expression. It refers to the simple use of Standard English. Pseudocode must use a restricted subset of English in such a way that it resembles a good high level programming language. *Figure 13* shows an example of pseudo code.

```
IF HoursWorked > MaxWorkHour THEN
Display overtime message
ELSE
Display regular time message
ENDIF
```

Figure 13: Example of a pseudocode

Contents of a Typical System Design Specification Document

1. Introduction

    1.1 Purpose and scope of this document:
    Full description of the main objectives and scope of the SDS document is specified.

    1.2 Definitions, acronyms, abbreviations and references
    Definitions and abbreviations used are narrated in alphabetic order. This section will include technical books and documents related to design issues. It must refer to the SRS as one of the reference book.

2. System architecture description

    2.1 Overview of modules, components of the system and sub-systems Structure and relationships. Interrelationships and dependencies among various components are described. Here, the use of structure charts can be useful.

3. Detailed description of components

3.1 Name of the component
3.2 Purpose and function
3.3 Sub-routine and constituents of the component
3.4 Dependencies, processing logic including pseudocode
3.5 Data elements used in the component.
4. Appendices.

3.4 SAMPLE DESIGN DOCUMENT

SCOPE

Define the System's Objectives here

……

Architecture Design

Give the architecture of the system based on the analysis and flow models.

DATA DESIGN

Refine the E-R diagram if needed and make one table for each entry and data store and table for all M.M relationships.

For example, the Client table may be:

COLUMN HEADING	*CONSTRAINTS*
client_no	Primary Key
client_number	Not Null
client_addr	Not Null
…….	

INTERFACE DESIGN

Human-Machine Interface Design Rules

Follow the basis rules for the interface design. These can be classified into three main types:

External Interface design

Interface to the External Data

Interface to the external system or devices

External Interface Design

Easy to follow Interface

Zero or very less graphics (as not being used commercially)

No hidden buttons

Proper error messages

……..

Interface to the External Data

The system has to use proper external data. The system makes a connection with the DBMS to do so. The rules that have to be followed while interfacing with the external data are:

You must do the type checking.

Field overrun that is maximum size should be enough to accommodate the largest data of that type

It is always better to encrypt the data and then store it in the database.

Interface to the External System or Device

……..

PROCEDURAL DESIGN

Verification of the User

Algorithm

Input: ……..

Output: …….

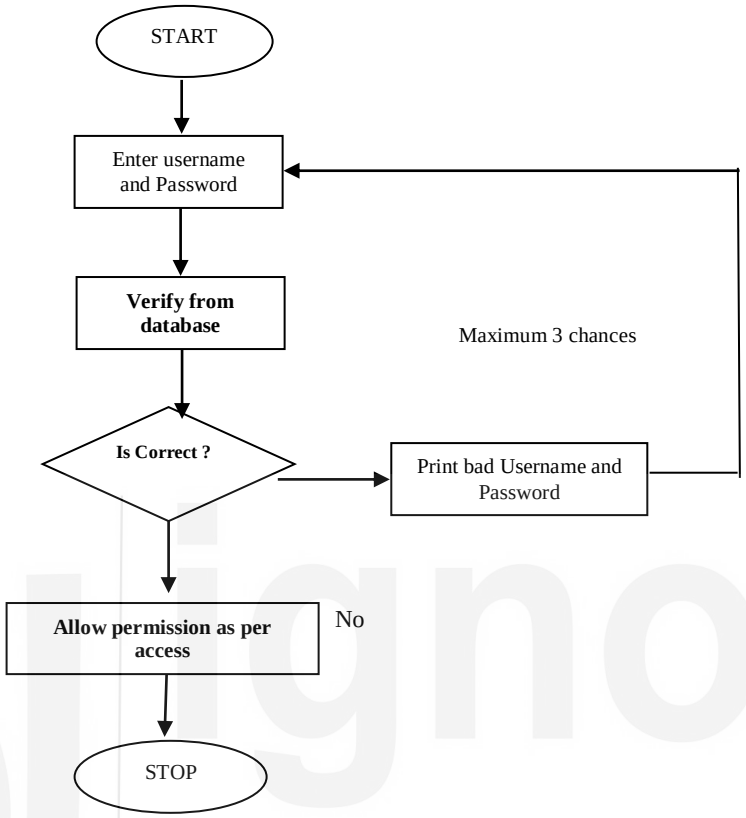
STEPS

Ask the Client to enter Username-Password.

Check Username and Password combination in the encrypted database.
If the Username and Password is correct then give permission with allowed access rights.

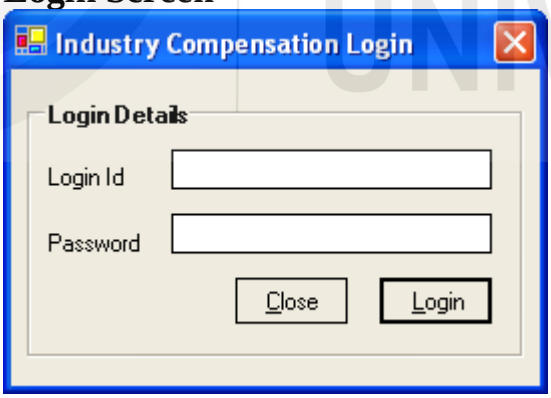
Else, access is denied and the enter Username and Password screen is shown again. This cycle may be repeated for a maximum of three times.

Flow Chart



Interface Design

Login Screen



Output / Reports Design

Design your output reports

Data Security and Rights

Security is the integral part of any system. You can use encryption and authentication or any other security mechanism as the need may be.

3.5 CODING

The input to the coding phase is the SDD document. In this phase, the design document is coded according to the module specification. This phase transforms the SDD document into a high-level language code. Currently, major software companies adhere to some well-specified and standard style of coding called coding standards. Good coding standards improve the understanding of the code. Once a module is developed, a check is carried out to ensure that coding standards are followed. Coding standards generally give guidelines on the following:

i) Name of the module
ii) Internal and External documentation of source code
iii) Modification history
iv) Uniform appearance of codes.

It must contain proper comments. The coding should be structured as per software architecture.

VALIDATION CHECKS

Verification of the User

Both User Id and Password are mandatory
The size of User Id must be 10 characters long

ERROR AND EXCEPTION HANDLING

A proper mechanism of Error Handling is necessary in any system.

COMMENTS

Comments are an integral part of any system. Like any properly developed and maintained system with high quality, this system too has sufficient comments and description of the processes, functions, classes, and data structures.

CODING STANDARDS

The defined coding standards of software developers may be followed. A sample clause in coding standard may be: "A consistent naming pattern is one of the most important elements of predictability and discoverability in a managed code. For each type, you must follow the defined capitalisation styles, case sensitivity and word choice".

3.6 TESTING

Testing

Testing is the process of running the software on manually created inputs with the intention to detect errors. In the process of testing, an attempt is made to detect errors, to correct the errors in order to develop error free software. Testing is performed keeping the user's requirements in mind and before the software is actually launched on a real system, it is tested.

Normally, while developing the code, a software developer also carries out some testing. This is known as debugging. This unearths the defects that must be removed from the program. Testing and debugging are different processes. Testing is meant for finding the existence of defects while debugging stands for locating the place of errors

and correcting the errors during the process of testing. The following are some guidelines for testing:

i) Test the modules thoroughly, cover all the access paths, generate enough data to cover all the access paths arising from conditions.
ii) Test the modules by deliberately processing/entering wrong data.
iii) Specifically create data for conditional statements. Enter data in test file, which will satisfy the conditions and test the script again.
iv) Test for locking by invoking multiple concurrent processes.

The following objectives are to be kept in mind while performing testing:

i) It should be done with the intention of finding errors.
ii) Good test cases should be designed with a probability of detecting undiscovered errors.
iii) A successful test is one that uncovers yet undiscovered error(s).

The following are some of the principles of testing:

i) All tests should be performed according to user requirements.
ii) Planning of tests should be done long before testing.
iii) Starting with a small test, it should proceed towards large tests.

The following are different levels of testing:

Large systems are built out of subsystems, subsystems are made up of modules, of procedures and functions. Thus, in large systems, testing is performed at various levels, like unit level testing, module level testing, subsystem level, and system level testing.

In all levels, testing is performed to check interface integrity, information content, and performance.

The following are some of the strategies for testing: This involves design of test cases. Test case is a set of designed data for which the system is tested. Two testing strategies are present.

i) **Code Testing:** The code testing strategy examines the logic of the system. In this, the analyst develops test cases for every instruction in the code. All the paths in the program are tested. This test does not guarantee against software failures. Also, it does not indicate whether the code is according to requirements or not.

ii) **Specification Testing:** In this, testing with specific cases is performed. The test cases are developed for each condition or combination of conditions and submitted for processing.

The objective of testing is to design test cases that systematically uncover different classes of errors and do so with the minimum amount of time and effort. Testing cannot show the absence of errors. It can only find the presence of errors. The test case design is as challenging as software development. Still, however effective the design is, it cannot remove 100% errors. Even, the best quality software are not 100 % error free. The reliability of software is closely dependent on testing.

Some testing techniques are the black box and the white box methods.

White Box Testing: This method, also known as glass box testing, is performed early in the testing process. Using this, the software engineer can derive tests that

guarantees exercises of all independent paths within the module at least once. It has the following features:

i) Exercises all logical decisions on their true and false sides.
ii) Executes all loops at their boundaries and within their operational bounds.
iii) Exercises internal data structures to assure their validity.

Black Box Testing: This is applied during the later stage of testing. It enables the software developer to derive a set of input conditions that will fully exercise the functional requirements of a program. It enables him/her to detect errors like incorrect or missing functions, interface errors, data structures or external data base access errors and performance errors etc.

3.7 TEST DESIGN DOCUMENT

During system development, this document provides the information needed for adequate testing. It also lists approaches, procedures and standards to ensure that a quality product meets the requirement of the user. This document is generally supplemented by documents like schedules, assignments and results. A record of the final result of the testing should be kept externally.

This document provides valuable input for the maintenance phase.

The following IEEE standards describe the standard practices on software test and documentation:

829-1998 IEEE Standard for Software Test Documentation
1008-1987 (R1993) IEEE Standard for Software Unit Testing
1012-1998 IEEE Standard for Software Verification and Validation

The following is the typical content of the Test Design Document:

1. Introduction

Purpose

The purpose of this *document* and its intended audience are clearly stated.

Scope

Give an overview of testing process and major phases of the testing process. Specify what is not covered in the scope of the testing such as, supporting or not third party software.

Glossary

It defines technical terms used in this document.

References

Any references to other external documents stated in this document including references to related project documents. They usually refer to the System Requirement Specification and the System Design Specification documents.

Overview of Document

Describe the contents and organisation of the document.

Test Plan

A test plan is a document that describes the scope, approach, resources and schedule of intended testing activities. It identifies test items, the features to be tested, the testing tasks, and the person who will do each task, and any risks that require contingency planning.

Schedules and Resources

An overview of the testing schedule in phases along with resources required for testing is specified.

Recording of Tests

Specify the format to be used to record test results. It should very specifically name the item to be tested, the person who did the testing, reference of the test process/data and the results expected by that test, the data tested. If a test fails, the person responsible for correcting and re-testing is also documented. The filled out format would be kept with the specific testing schedule. A database could be used to keep track of testing.

Reporting Test Results

The summary of what has been tested successfully and the errors that still exist which are to be rectified is specified.

Verification Testing

Unit Testing: *For each unit/component, there must be a test, which will enable the tester to know the accurate functioning of that unit.*

Integration testing: *Integration test is done on modules or sub-systems.*

Validation Testing

System Testing : This is top level integration testing. At this level, requirements are validated as described in the SRS.

Acceptance and Beta Testing

List test plans for acceptance testing or beta testing. During this test, real data is used for testing by the development team (acceptance testing/alpha testing) or the customer (beta testing). It describes how the results of such testing will be reported and handled by the developers.

A sample test case may be:

Login Screen

S.No.	Test Case ID	Do	Expected Result
1.	QT1-001	Enter user id in the text box specified. User id must not be more than 510 characters and it should not contain any special characters and no spaces including in the start.	Successful login in to the system if the values are found in the database.

		Enter password in the text box specified. Password must not be more than 10 characters. Click on the Login button.	
…..			

Test Log

Function	Purpose of Set of Test Cases per Area	Test Case ID(s)	No of Test Cases run	Number of Test Cases Successful
Login	The verification of username and password	QT1-001	8	100%
……				

3.8 SUMMARY

Software Engineering is a discipline focused on the systematic development, deployment, and maintenance of software systems. It requires a solid understanding of fundamental software concepts and development methodologies before beginning any project work.

Unlike programming, which mainly involves writing source code, software engineering covers the entire Software Development Life Cycle (SDLC). This includes activities such as requirement analysis, system design, coding, documentation, testing, quality assurance, deployment, and maintenance, ensuring a structured and process-oriented approach to software development.

Documentation is a critical component of the development process. It involves systematically recording information about the system's design, functionality, and implementation, enabling effective communication among developers, stakeholders, and users. Documentation is created throughout the development lifecycle and helps in understanding system architecture, development progress, and technical decisions.

Different types of documentation include requirements (analysis), design, interface, internal program, and user documentation. The goal of this project course is not only to develop software but also to gain practical experience in real-world software engineering practices, addressing development challenges and producing well-structured and properly documented software systems.

Section 4: *Guidelines for BCSP-165*

BCSP-165

BACHELOROFCOMPUTERAPPLICATIONS

(BCA_NEW/BCA_NEWOL)

BCA_NEW /

BCA_NEWOL6th

Semester BCSP-165

PROJECTGUIDELINES

SCHOOL OF COMPUTER AND INFORMATION S CIENCES
INDIRA GANDHI NATIONAL OPEN UNIVERSITY
MAIDAN GARHI, NEW DELHI – 110 068

CONTENTS

MESSAGE FROM THE PROJECT CO-ORDINATOR

The Bachelor of Computer Applications(BCA_NEW/BCA_NEWOL) programme prepares the students to take up positions as Programmers, Systems Analysts, Systems Designers in the field related to computer science and information technology, and ITES or students may go for higher studies in this area. We had therefore imparted the comprehensive knowledge covering the skills and core areas of computer science courses with equal emphasis on the theory and practice in BCA_NEW/BCA_NEWOL programme.

The BCA_NEW/BCA_NEWOL students are encouraged to involve themselves completely on the project work in their final semester. It is advised to students to develop their project for solving problems of software industry or any research organization. Doing this will give more exposure to handle real life problems of project development.

The courses studied by you during your BCA_NEW/BCA_NEWOL programme provide you the basic background to work on diverse application domains. The theoretical background of various courses provides you the necessary foundation, principles, and practices to develop effective ways to solve computing problems. The hands-on experience gained from the practical courses provide you the knowledge to work with various operating systems, programming languages, and software tools.

This project work is kept in BCA_NEW/BCA_NEWOL program to give you opportunity to develop quality software solution. During the development of the project, you should involveinall the stages of the software development life cycle (SDLC) like requirements analysis, systems design, software development/coding, testing and documentation, with an overall emphasis on the development of reliable software systems. The primary emphasis of the project work is to understand and gain the knowledge of the principles of software engineering practices, and develops good understanding of SDLC.

Students should take this project work very seriously. BCSP-165 project should be taken as an opportunity to develop software, which gives exposure to SDLC. Topics selected, should be complex and large enough to justify as a BCA_NEW/BCA_NEWOL project. The project should be genuine and original in nature and should not be copied from anywhere else. If found copied, the project report will be forwarded to the Exam Discipline Committee of the University as an Unfair means case for necessary action. Students should strictly follow and adhere to the BCSP-165 project guidelines.

I wish you all the success.

BCA_NEW/BCA_NEWOL COORDINATOR

Email: BCA@ignou.ac.in /BCAOL@ignou.ac.in

I. CALENDAR FOR THE PROJECT

Sl.No.	*Topic*	*Date*
1.	Submission of a Guide‟s bio-data and project proposal at the following address: The Regional Director of your Regional Centre	**Twice a year as shown below:** **1st April to 30th June** **or** **1st October to 31st December**
2.	Approval of Project	**30** days after the project Proposal is received.
3.	Submission of the Project Report(**one copy**) **in Bound form to:** The Regional Director of your Regional Centre	**Twice a year as shown below:** **1st July to 30th September (For Project Proposal that have been approved during the 1st April to 30th June slot)** **or** **1st January to 31st March (For Project Proposal that have been approved during 1st October to 31st December slot)**
4.	Viva-Voce to be conducted	**In July**(For project reports submitted during 1st January - 31st March slot) **In January**(For project reports submitted during 1st July– 30th September slot)

SCHOOL OF COMPUTER AND INFORMATION SCIENCES
IGNOU, MAIDAN GARHI, NEW DELHI – 110 068

II. PROFORMA OF BCA_NEW/BCA_NEWOL PROJECT PROPOSAL (BCSP-165)

Enrolment No.:……Regional Centre Code:……..….Study Centre: ……..……

1. Name and Address of the student

……..……..……
……
……
E-mail:…….....Telephone No.……..

2. Title of the Project.……..……

3. Name and Address of the Guide

……
……
……

4. Qualification of the Guide (Attach bio-data also)

Ph.D.	M.Tech.	B.Tech.	MCA	Any other
☐	☐	☐	☐	☐

(Note: i. All the above mentioned Degrees must have been awarded in Computer Science/IT only
ii. A Guide should not guide more than 8 students of BCA_NEW/BCA_NEWOL at any point of time)

5. Industrial/Teaching experience of the Guide (in Years) ……..

6. Software Used for this Project: ……

NOTE:

- *Refer to the list of broad areas of application and related tools (Section VII)*
- *Demonstrate/ execution of your Project at the time of Viva is must*
- *Synopsis comments should be implemented in the report*

Signature of the Student — Signature of the Guide

Date:…… — Date:……

Important: 1. Attach this Proforma along with Guide"s Biodata and Project Synopsis in the Project Report.
2. Not more than one student is permitted to work on a project.

For Office Use Only

☐ Approved ☐ Not approved

……
Signature, Designation, Stamp of the Project Proposal Evaluator
Date:……..

Suggestions for reformulating the Project:

Ensure that you include the following while submitting the Project Proposal:

1. **Proforma for Approval of Project Proposal duly filled and signed by both the student and the Project Guide with date.**

2. **Bio-data of the project guide with her/his signature and date.**

3. **Synopsis of the project proposal (12-15 pages).**

4. **A self-addressed envelope with duly affixed postage stamps (to send it by ordinary post only) on it.**

A photocopy of the complete Project Proposal (along with Project Proforma, Project Synopsis, Biodata of the guide) submitted to your Regional Centre, should be retained by the student for future reference.

SCHOOL OF COMPUTER AND INFORMATION SCIENCES IGNOU,
MAIDAN GARHI, NEW DELHI – 110 068

III. GUIDELINES FOR PROJECT FORMULATION

The project work constitutes a major component in most of the professional programmes and it is to be carried out with due care and should be executed with seriousness by the candidates.

TYPE OF PROJECT

As majority of the students are expected to work out a real life project in some industry/research and development laboratories/educational institutions/software companies, it is suggested that the project is to be chosen which should have some direct relevance in day-to-day activities of the candidates in his/her institution. Students are encouraged to work in the areas listed at the end. However, it is not mandatory for a student to work on a real life project. The student can formulate a project problem with the help of Guide.

PROJECT PROPOSAL(SYNOPSIS)

The project proposal should be prepared in consultation with your guide. The project proposal should clearly state the project objectives and the environment of the proposed project to be undertaken. **The project work should compulsorily include the software development.** The project proposal should contain complete details in the following form:

1. Title of the Project
2. Introduction and Objectives of the Project
3. Project Category (RDBMS/OOPS/Networking/Multimedia/ArtificialIntelligence/ExpertSystems etc.)
4. Analysis (DFDs at least up to second level, ER Diagrams/Class Diagrams/Database Designetc. as per the project requirements).
5. A complete structure which includes:
   - Number of modules and their description to provide an estimation of the student"s effort on the project.
   - Data Structures as per the project requirements for all the modules.
   - Process Logic of each module.
   - Testing process to be used.
   - Reports generation (Mention tentative content of report)
6. Tools/ Platform, Hardware and Software Requirement specifications
7. Are you doing this project for any Industry/Client? Mention Yes/No. If Yes, Mention the Name and Address of the Industry or Client
8. Future scope and further enhancement of the project.

ignou
THE PEOPLE'S
UNIVERSITY

IV. PROJECT PROPOSAL SUBMISSION AND APPROVAL

After finalising the topic and the selection of the guide, students should send the Project Proposal Proforma given on page no. 6 along with the synopsis and bio-data of the guide to The Regional Director of the Regional Centre concerned**. Incomplete project proposals in any respect will be summarily rejected.**

COMMUNICATION OF APPROVAL

Communication regarding the Approval / Non-approval of the project will be sent to you within four weeks after the receipt of the project proposal by the Regional Centre concerned.

RESUBMISSION OF THE PROJECT PROPOSAL IN CASE OF NON-APPROVAL

In case of non-approval, the suggestions for reformulating the project will be communicated to you. The revised project synopsis along with a new Performa, should be re-submitted along with a copy of the earlier synopsis and non-approval project proposal Performa in the next slot. For example, if the student submitted the synopsis during the 1st April to 30th June slot and is not approved due to some reasons, s/he is eligible to resubmit the revised project synopsis only during the next slot i.e., 1st October to 31st December. These guidelines are applicable for earlier batch students also whose project work is pending.

ELIGIBILITY OF PROJECT GUIDE

1. A person having Ph.D./M.Tech. in Computer Science.

   OR

2. A person having B.E/B.Tech (Computer Science), MCA, M.Sc (Computer Science) with minimum 2 years experience in Industry / Teaching.

V. PROJECT REPORT FORMULATION

ITEMS TO BE INCLUDED IN THE PROJECT REPORT

The following items should be included in the Project Report:

1. The project report must contain the following:
   - Introduction
   - Objectives
   - Tools/Environment Used
   - Analysis Document (This should include SRS in proper structure based on Software Engineering concepts, E-R diagrams/Class diagrams/any related diagrams (if the former are not applicable), Data flow diagrams/other similar diagrams (if the former is not applicable), Data dictionary)
   - Design Document (Modularization details, Data integrity & constraints including database design, Procedural design, User interface design)
   - Program code (Complete code (well indented)/Detailed specification instead of code*, Comments & Description. The program8code should always be developed in such a way that it includes complete error handling, passing of parameters as required, placement of procedure/function statements as needed.)
   - Testing(Test case designs are to be included separately for Unit testing, Integration testing, System testing; Reports of the outcome of Unit testing, Integration testing, System testing are to be included separately. Also, details of debugging and code improvement are to be included.)
   - Input and Output Screens
   - Implementation of Security for the Software developed (In case, you have set up a User Name and Password for your software, you should ensure the security of User Name and Password during transmission to server)
   - Limitations of the Project
   - Future Application of the Project
   - Bibliography

   *Students who have done their project for any organization are permitted to attach detailed algorithm/specification instead of code, in case, the organization doesn‟t permit them to attach the code. Student needs to attach letter in the project report from the Project Manager of the project in the organization that they are not permitting student to attach the code. In the absence of such letter, the student needs to attach the code compulsorily.

   The project report should be hard bound; should consist of a Contents page; all pages of report should be numbered; content should be well organized in a meaningful manner; printouts of text & screen layouts should be original and should not be xeroxed)

2. **Original copy of the Approved Project Proposal Proforma, Synopsis and Guide‟s Bio-data.**
3. **Certificate of Originality (Formatgivenonpageno.13).**
4. **The Project Report may be about 100 double spaced A-4 size typed pages (excluding program code). However, 10% variation on either side is permissible.**

SUBMISSION OF PROJECT REPORT

Only one copy of the project report is to be submitted to the following address by registered insured post: **Regional Director**, **Concerned Regional Centre** by the date mentioned in the Calendar for the project (refer to page-3).

TYPE OF PROJECT

The majority of the students are expected to work on a real-life project preferably in some industry/ Research and Development Laboratories / Educational Institution / Software Company. Students are encouraged to work in the areas listed at the end (Refer page no.15). However, it is not mandatory for a student to work on a real-life project. The student can formulate a project problem with the help of her/his Guide and submit the project proposal of the same. If approved, the student can commence working on it and complete it.

PROJECT EVALUATION

The **Project Report** is evaluated for 150 marks and the **Viva-Voce** is for 50 marks. To be declared successful, the student should secure at least 50% marks separately in both project report evaluation (i.e.75/150) and viva-voce (i.e.50/100). Students will be duly intimated about the schedule of viva-voce by a letter from the respective Regional Centre. An unsuccessful student can either submit the same project after following comments on the assessment sheet or s/he can do a different project. Always, ensure that the BCSP-165 project guidelines are followed.

Unfair cases of copied versions of the project synopsis and project reports, they will be awarded very low score and may be sent to Unfair Means Committee of IGNOU for action.

RESUBMISSION OF THE BCA_NEW/BCA_NEWOL PROJECT IN CASE OF FAILED STUDENTS

If the student is unsuccessful in the project, s/he should „re-do" the whole cycle, right from the submission of the project synopsis. Students are advised to select a new topic for the project and should prepare and submit the project synopsis to the Regional Centre concerned as per the project guidelines. There are no separate slots for the submission of the project synopsis / project reports for the failed students. Respective submissions of the project synopsis and the project reports should be done strictly as per the "calendar for the BCA_NEW/BCA_NEWOL project" given in the project guidelines. Along with the resubmission of the project report, the student is required to remit the pro-rata fee (subject to change as per university rule – contact your concerned Regional Centre for necessary update). The fee should be remitted to the Regional Centre at the time of resubmission of the project report by the way of Demand Draft favouring IGNOU and payable at the city where your Regional Centre is located.

ENQUIRIES

Enquiries regarding the Project Report and Viva-Voce should be addressed to the **Regional Director, Concerned Regional Centre**. In all correspondence, please quote your Enrolment No.

VI. IMPORTANT POINTS WHILE PREPARING THE PROJECT REPORT

1. The Project Report should be submitted in A-4 size typed in double space. The Project Report should be hard bound.

2. Ensure that it contains the following:

- Project Proposal Proforma. All the items should be filled. The signatures of both student and Guide should be present.
- Project Synopsis. Both Guide and student should sign on the Project Synopsis.
- Guide"s Bio data. The Biodata should consist of signature of the Guide.
- Certificate of Originality (Format given on Page no.13)
- All signatures should be accompanied by the date of signature.

It should include all items mentioned in Section IV.

3. **If any project report is received in the absence of the items listed above, it will be rejected and returned to students for compliance. Also, violation of Project Guidelines may lead to rejection of the Project .**

4. **Only One hard bound original copy** of the project report is to be submitted to **„The Regional Director**, **Concerned Regional Centre"** by registered insured post. **One copy of the same Project Report is to be retained with the student and the student is supposed to carry his copy while appearing for viva voce. Spiral binding of Project Report is not permitted.**

5. Xerox copy of the project report is not acceptable.

6. Not more than one student is permitted to work on a Project.

7. If the title of the Project differs from the title mentioned in the Project Proposal, the Project Report will be rejected and will be returned back to the student.

8. Kindly mention on the top of the envelope **"BCA_NEW/BCA_NEWOL PROJECT REPORT (BCSP-165)".**This will facilitate sorting out project reports at the Regional Centre.

9. The envelope containing the remuneration form for the Project Guide duly signed by the Guide and the student, should be sent to "Regional Director, Concerned Regional Centre."

10. In case, students require any letter from the University for doing a project in any organization, they may request the Regional Director of the concerned Regional centre for the issuance of the same in the format indicated under **Project Trainee** in this document. (Refer to Page number:15).

VII. LIST OF BROAD AREAS OF APPLICATION AND RELATED TOOLS

Technology Area	Tools / Platforms	Application Examples
Front End / GUI / Web & App Frameworks	React.js, Angular, Vue.js, Next.js, Flutter, React Native, SwiftUI, Kotlin, Tailwind CSS, Bootstrap etc.	Web applications, mobile apps, dashboards, enterprise UI systems
RDBMS / Databases	PostgreSQL, MySQL, MariaDB, Oracle, Microsoft SQL Server, IBM Db2, SQLite etc.	Enterprise data storage, transaction systems, ERP/CRM databases
NoSQL / Distributed Databases	MongoDB, Cassandra, Redis, CouchDB, Neo4j, DynamoDB, Firebase etc.	Big data applications, real- time apps, graph analytics, caching systems
Programming Languages	Python, Java, C#, Go, Rust, Kotlin, JavaScript etc.	Software development, AI systems, enterprise applications, scientific computing
Scripting Languages	Python, PHP, Bash, PowerShell, JavaScript (Node.js) etc.	Automation, system administration, backend scripting, DevOps workflows
Middleware / Microservices Technologies	Docker, Kubernetes, gRPC, Apache Kafka, RabbitMQ, Nginx, Spring Boot, .NET Core, GraphQL etc.	Microservices platforms, scalable enterprise applications, event streaming systems
Software Architecture & Design	Microservices Architecture, Service-Oriented Architecture (SOA), Event Driven Architecture, REST APIs, GraphQL APIs etc.	Distributed systems, cloud-native applications, enterprise platforms
Internet & Web Technologies	HTML5, CSS3, JavaScript, Node.js, Express.js, Django, Flask, FastAPI, Spring Boot, ASP.NET Core, REST APIs, WebSockets etc.	Web platforms, APIs, SaaS applications, dynamic websites
Cloud & DevOps Technologies	AWS, Microsoft Azure, Google Cloud Platform (GCP), Docker, Kubernetes, Terraform, Jenkins, GitHub Actions, GitLab CI/CD etc.	Cloud deployment, CI/CD pipelines, scalable infrastructure management
Networking Technologies	TCP/IP, IPv6, Software Defined Networking (SDN), Network Function Virtualization (NFV), 5G, Edge Networking etc.	Telecommunication systems, internet infrastructure, enterprise networking
Wireless & Mobile Technologies	5G, Wi- Fi 6, Bluetooth Low Energy (BLE), IoT Protocols (MQTT, CoAP), Android Development, iOS Development etc.	Mobile apps, IoT devices, smart city applications
Operating Systems	Windows, Linux (Ubuntu, RedHat, Debian), macOS, Android, iOS, Unix variants etc.	Desktop systems, servers, cloud platforms, mobile devices
Real-Time / Embedded Systems	FreeRTOS, Zephyr RTOS, Embedded Linux, ARM Development Tools, NVIDIA Jetson, Raspberry Pi etc.	Robotics, IoT devices, industrial automation, smart hardware
Data Science & Analytics Tools	Python (Pandas, NumPy, SciPy), R, Apache Spark, Apache Hadoop, Jupyter Notebook, etc.	Data analysis, visualization, business intelligence, big data analytics
Machine Learning / Deep Learning Frameworks	TensorFlow, PyTorch, Keras, Scikit- learn, XGBoost, LightGBM, Transformers, MXNet etc.	Predictive models, computer vision, NLP systems
Generative AI & LLM Development	OpenAI API, Transformers, LangChain, LlamaIndex, Ollama, vLLM, DeepSpeed, PEFT/LoRA etc.	Chatbots, AI assistants, document analysis systems, text generation

Technology Area	Tools / Platforms	Application Examples
Agentic AI Frameworks	LangGraph, AutoGen, CrewAI, Semantic Kernel, Haystack Agents etc.	Autonomous AI agents, task automation, multi-agent collaboration systems
Federated Learning Frameworks	TensorFlow Federated, Flower (FLWR), PySyft, NVIDIA FLARE etc.	Privacy- preserving AI, distributed healthcare models, cross- organization ML
AI/ML Development	MLflow, Weights & Biases, DVC, Kubeflow, Airflow etc.	Model lifecycle management, ML pipelines, experiment tracking
Vector Databases (RAG Applications)	FAISS, Pinecone, Weaviate, Milvus, ChromaDB etc.	Semantic search, retrieval- augmented generation, knowledge base systems
Generative AI Application Development	LangChain, LlamaIndex, Haystack, Streamlit, Gradio, FastAPI etc.	AI chat interfaces, LLM apps, prototype AI tools
Application Domains	AI, Data Science, Cloud Computing, IoT, Cybersecurity, FinTech, HealthTech, Smart Manufacturing, Multimedia Systems, E- Commerce, ERP, Mobile & Web Apps, etc.	Industry applications, research systems, enterprise platforms

Note: *The choice of Technology area, use of tools/platforms is not restricted to the table mentioned above, always try to use the latest ones.*

VIII. REMUNERATION FORM FOR THE BCA_NEW/BCA_NEWOL(BCSP-165) PROJECTGUIDE

INDIRA GANDHI NATIONAL OPEN UNIVERSITY MAIDAN GARHI, NEW DELHI – 110068

REMUNERATION BILL FOR THE BCA_NEW/BCA_NEWOL PROJECT GUIDE

1. Course Code : **BCA_NEW/BCA_NEWOL(BCSP-165)**

2. Name of the Guide : _____

3. Residential Address : _____

_____

_____

4. Designation : _____
5. Office address : _____

This is to certify that I have Guided the following student/(s) for their project work:

S.No	Enrolment Number of the student	PR No. (to be filled by the Regional Centre)	Name of the student	Title of the Project	Amount Claimed

Signature of the Guide
Date :.....…….

NOTE: ***Project guide cannot guide more than eight students in any given point of time. This form duly signed by the guide placed in a separate envelope, should be submitted along with the project report. Remuneration Bill not accompanied with the project report will not be considered for payment.ForBCSP-165(8 Credits) course, the remuneration to the project guide (for guiding one project) is as per the university norm.***

FOR USE IN THE EXAMINATION BRANCH

Registrar (SR&ED) Verified by Dealing Asstt.

FOR USE IN FINANCE BRANCH

Passed for payment of Rs. : _____

Debit Head: Examination _____

: Contingent _____

: T.A. _____

Dated : _____ Section Officer/A.R.(Accounts)

Dealing Asstt.

Paid by Cheque No. _____ Dated _____

Assistant Registrar

IX CERTIFICATE OF ORIGINALITY

This is to certify that the project report entitled _____

Submitted to **Indira Gandhi National Open University** in partial fulfilment of the requirement for the award of the degree of **BACHELOR OF COMPUTER APPLICATIONS (BCA_NEW/BCA_NEWOL) ,** is an original work carried out by Mr./ Ms._____

Enrolment No.: _____under the guidance of Mr./ Ms._____

The matter embodied in this project is a genuine work done by the student and has not been submitted whether to this University or to any other University / Institute for the fulfilment of the requirement of any course of study.

Signature of the Student

Name and Address
of the student :

Signature of the Guide

Name, Designation and
Address of the Guide

Enrolment No.:

Grams : IGNOU

School of Computer and Information Sciences
INDIRA GANDHI NATIONAL OPEN UNIVERSITY
Maidan Garhi, NEWDELHI–110068

X. PROJECT TRAINEE LETTER

Date:

Subject: Project Trainee

Sir,

This is to certify that Mr/Ms_____whose Enrolment No._____is a student of BCA_NEW/BCA_NEWOL Programme of Indira Gandhi National Open University and has to do a project in his/her final year starting from January / July session. The project is compulsory for BCA_NEW/BCA_NEWOL programme. S/he has to do a project for 3-6 months in Industry/Research Laboratories under the supervision of a guide preferably from the same organization. During his course, the student has gone through / will go through several theoretical papers such as Data Structures, Database Management System, Programming Languages (C, C++, and Java), TCP/IP Programming, Intranet Administration, Computer Networks, Software Engineering etc. The student also attended / will also attend practical sessions in all courses in which practical sessions were prescribed for various subjects.

Looking forward for your positive response.

Signature & Name of Project Coordinator
with Date and Stamp

Note: This letter may also be signed by Regional Director / Asst. Regional Director of Regional Centre concerned.

APPENDIX 1: SAMPLE TITLE PAGE

INDIRA GANDHI NATIONAL OPEN UNIVERSITY

PROJECT TITLE

by

Student's Full Name

Under Guidance
of

Guide's Full Name

Submitted to the School of Computer and Information Sciences
in partial fulfilment of the requirements
for the degree of

Bachelor of Computer Applications
PROJECT COURSE(BCSP-165)

Indira Gandhi National Open University
Maidan Garhi
New Delhi – 110068.

Annexure-1

ASSESSMENT PARAMETERS FOR PROJECT EVALUATION BCSP-064/BCSP-165

1) **Project Report:** The submitted project report is evaluated on the basis of the following components:
   - **ANALYSIS:** This section evaluates the clarity and completeness of system analysis. It includes the preparation of the Software Requirements Specification (SRS) with a proper structure. Additionally, it assesses the inclusion of analysis models/diagrams such as ER diagrams, Class diagrams, Data Flow Diagrams, and other relevant diagrams/models along with a Data Dictionary.

   - **DESIGN:** This section focuses on the design aspects of the project. It includes modularization of the system, maintenance of data integrity and constraints including database design and procedural design, and the design of the user interface.

   - **CODING:** This section evaluates the coding practices followed in the project. It includes proper commenting and description within the code, adherence to coding standards or providing detailed specifications in place of code, and handling of errors, parameter passing, and function calling. Please note, you must attach complete code.

   - **TESTING:** This section assesses the testing methodologies used in the project. It includes test case design covering unit testing, integration testing, and system testing. It also evaluates test reports including the type of testing performed (viz. unit testing, integration testing, system testing etc.), debugging, and code improvement.

   - **Report Organization:** Some marks are also allotted to the overall presentation of the project report. It includes proper binding of the report, inclusion of a content page and page numbering, logical organization of content, and proper printing of text and screen layouts.

2) **Project Viva Voce:** The performance of the student in project viva, is evaluated on the basis of following Components:
   - **Presentation:** It evaluates the student's ability to present the project effectively, including clarity, confidence, and communication skills. This includes evaluation of Analysis, Design, Coding, Coding Demonstration, and Testing.

   Note: *Coding demonstration is must and caries significant weightage.*

   - Also, the **Explanation of Subject Area** Related to the Project is considered for evaluation of the students' performance.

Annexure-2

BCSP-165 PROJECT SYNOPSIS
TABLE OF CONTENTS

Important Note:

1. Always Map the Entities of Entity Relationship Diagram with the data stores of Data Flow Diagram and Structure of the Tables in the Database
2. Always map the modules in your project with the processes in the first level of the Data Flow Diagram
3. Always Extend the processes identified in the first level to the second level, respectively

Annexure-3

BCSP-165 PROJECT REPORT
TABLE OF CONTENTS

Important Note:

1. Always Map the Entities of Entity Relationship Diagram with the datastores of Data Flow Diagram and Structure of the Tables in the Database
2. Always map the modules in your project with the processes in the first level of the Data Flow Diagram
3. Always Extend the processes identified in the first level to the second level, respectively