



**GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI, BHAWANIPATNA**  
Address: At- Kandha Bandopala, Po- Risigaon, Via- Bhawanipatna  
Dist.- Kalahandi, Orissa, Pin- 766003

**SUBJECT: SOFTWARE ENGINEERING ( RCS6C201)**

**SEMESTER: 6<sup>th</sup>**

Table of Contents:

<b>Experiment No</b>	<b>Objective of the Experiment</b>	<b>Page Number</b>
01	Develop requirements specification for a given problem (The requirements specification should include both functional and non-functional requirements).	3
02	Develop DFD Model (Level 0, Level 1 DFD and data dictionary) of the sample problem (Use of a CASE tool required).	5
03	Develop structured design for the DFD model developed .	7
04	Develop UML Use case model for a problem (Use of a CASE tool any of Rational rose, Argo UML, or Visual Paradigm etc. is required)	9
05	Develop Sequence Diagrams.	11
06	Develop Class diagrams.	13
07	Develop code for the developed class model using Java.	15
08	Use testing tool such as Junit.	19
09	Use a configuration management tool.	22
10	Use any one project management tool such as Microsoft	24



**GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI, BHAWANIPATNA**  
**Address: At- Kandha Bandopala, Po- Risigaon, Via- Bhawanipatna**  
**Dist.- Kalahandi, Orissa, Pin- 766003**

Project, Gantt Project or ProjectLibre.

*Digital Learning Resources:* Virtual Lab Link: <http://vlabs.iitkgp.e>

### **List of Experiments**

Experiment1: Develop requirements specification for a given problem (The requirements specification should include both functional and non-functional requirements. For a set of about 20 sample problems, see the questions section of Chap 6 of Software Engineering book of Rajib Mall)

Experiment 2: Develop DFD Model (Level 0, Level 1 DFD and data dictionary) of the sample problem (Use of a CASE tool required)

Experiment 3: Develop structured design for the DFD model developed

Experiment 4: Develop UML Use case model for a problem (Use of a CASE tool any of Rational rose, Argo UML, or Visual Paradigm etc. is required)

Experiment 5: Develop Sequence Diagrams.

Experiment 6: Develop Class diagrams.

Experiment 7: Develop code for the developed class model using Java.

Experiment 8: Use testing tool such as Junit.

Experiment 9: Use a configuration management tool.

Experiment 10: Use any one project management tool such as Microsoft Project, Gantt Project or ProjectLibre.

*Digital Learning Resources:* Virtual Lab Link: <http://vlabs.iitkgp.e>



**GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI, BHAWANIPATNA**  
Address: At- Kandha Bandopala, Po- Risigaon, Via- Bhawanipatna  
Dist.- Kalahandi, Orissa, Pin- 766003

**Experiment1: Develop requirements specification for a given problem (The requirements specification should include both functional and non-functional requirements.**

**Objective of the Experiment:** This section should describe where the software would be deployed and how the software would be used.

**Scope of the Experiment:** This section should describe the overall context within which the software is being developed. For example, the parts of a problem that are being automated and the parts that would need to be automated during future evolution of the software.

**Environmental Characteristics of the Experiment:**

This section should describe briefly outline the environment (hardware and Software ) with which the software will interact.

**Product Prospective:**

This section needs to briefly state as to whether the software is intended to be a replacement for a certain existing system. If the software being developed would be used as a component of a larger system, a schematic diagram can be given to show.

**Product feature:**

This section should summarize the major ways in which the software would be used.

**User Classes**

Various user classes that are expected to use this software are identified and described here. The different classes of users are identified by the types of functionalities that they are expected to invoke or their levels of expertise in using computers.

**Operating Environment**

This section should discuss in some detail the hardware platform on which the software would run, the OS, and the application software with which the developed software would interact.

**Design and Implementation Constraints**

In this section different constraints on the design and implementation are discussed I.e. hardware limitation, interface to other applications, tools, database, protocol, security consideration etc.

**User Documents**

This section should list out the type of user documentation such as user manual, on-line help, trouble shooting manuals that will be delivered to the customer along with the software.

**a. Functional SRS:**

**1. User Class1**

**A) Function Requirement 1.1 (Withdraw Cash)**

Description: The withdraw cash function first determines the type of account that the user has and the account number from which the user wishes to withdraw cash. It checks the balance to determine whether the requested amount is available in the account. If enough balance is available, it outputs the required cash, otherwise it generates an error message.

**Prepared By: Dr. Ashok Kumar Bhoi, Assistant Professor**  
**Department Of Computer Science and Engineering.**



**GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI, BHAWANIPATNA**  
**Address: At- Kandha Bandopala, Po- Risigaon, Via- Bhawanipatna**  
**Dist.- Kalahandi, Orissa, Pin- 766003**

CR. 1.1.1 Select withdraw amount option

Input: "Withdraw amount" option selected.

Output: User prompted to enter the account type.

CR.1.1.2 Select account type

Input: User selects option from any one of the followings-savings/checking/deposit.

Output: Prompt to enter amount

CR.1.1.3 Get required amount

Input: Amount to be withdrawn in integer values greater than 100 and less than 10,000 in multiples of 100.

Output: The requested cash and printed transaction statement.

Processing: The amount is debited from the user's account if sufficient balance is available otherwise an error message displayed.

B) Functional Requirement 1.2

2. User Class 2

A). Functional requirement 2.1

B). Functional requirement 2.2

**b. Non-Functional SRS**

**Performance Requirement:**

Aspects such as number of transaction to be completed per second should be specified here.

**Safety Requirement:** Those requirements that are concerned with possible loss or damage that could result from the user of the software are specified here.

**Security Requirement:**

This section should specify any requirements regarding security or privacy requirements on data used or created by the software.

**External Interface Requirement:**

A) **User Interface:** This description include sample screen, images, GUI standard, screen layout, standard push buttons, keyboard shortcut, error message display standard.

B) **Hardware Interface:** It describe interface between the software and hardware components of the system. The nature of data control interactions between this software and hardware and communication protocols to be used.

C) **Software Interface:** This section should describe the connections between this software and other specific software components , including databases, operating systems, tools, libraries , intregated commercial component.

D) **Communicational Interface:** This section Should describe the requirements associated with any type of communicational required by the software such e-mail, web access, network server communicational protocol.



**GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI, BHAWANIPATNA**  
Address: At- Kandha Bandopala, Po- Risigaon, Via- Bhawanipatna  
Dist.- Kalahandi, Orissa, Pin- 766003

## **Experiment 2: Develop DFD Model (Level 0, Level 1 DFD and data dictionary) of the sample problem (Use of a CASE tool required)**

**Objective:** To draw the DFD diagram for a particular problem.

**Data Flow Diagram (DFD)** is a **graphical representation** of data flow in any system. It is capable of illustrating incoming data flow, outgoing data flow and store data. The DFD depicts both **incoming** and **outgoing data flows** and provides a high-level overview of system functionality. It is a relatively simple technique to learn and use, making it accessible for both technical and non-technical stakeholders.

Data Flow Diagram can be represented in several ways. The Data Flow Diagram (DFD) belongs to structured-analysis 1 tools. Data Flow diagrams are very popular because they help us to visualize the major steps and data involved in **software-system processes**.

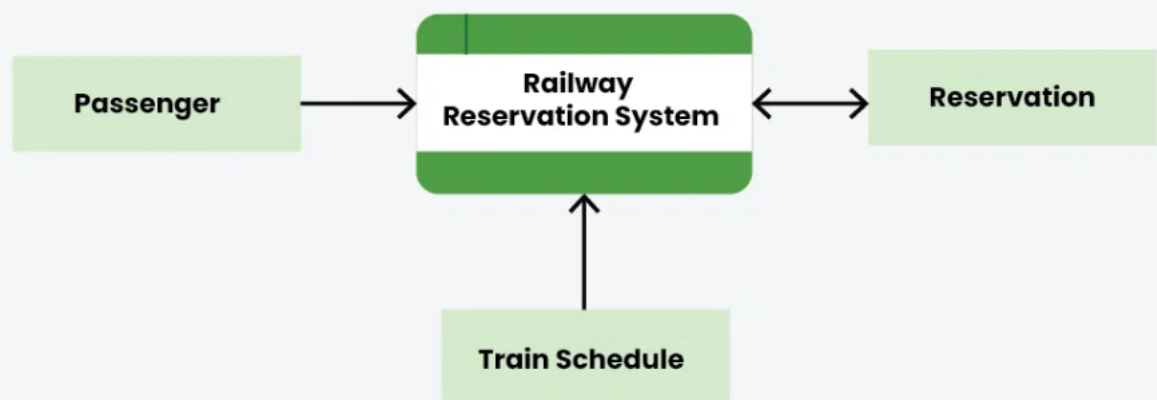
### **1 Example of Levels of Data Flow Diagram (DFD)**

Data Flow Diagram (DFD) uses hierarchy to maintain transparency thus multilevel Data Flow Diagram (DFD's) can be created. **Levels of Data Flow Diagram (DFD)** are as follows:

#### **2 0-level DFD**

It is also known as a context diagram. It's designed to be an abstraction view, showing the system as a single process with its relationship to external entities. It represents the entire system as a single bubble with input and output data indicated by incoming/outgoing arrows.

### **Level 0 Diagram of Railway Reservation System**



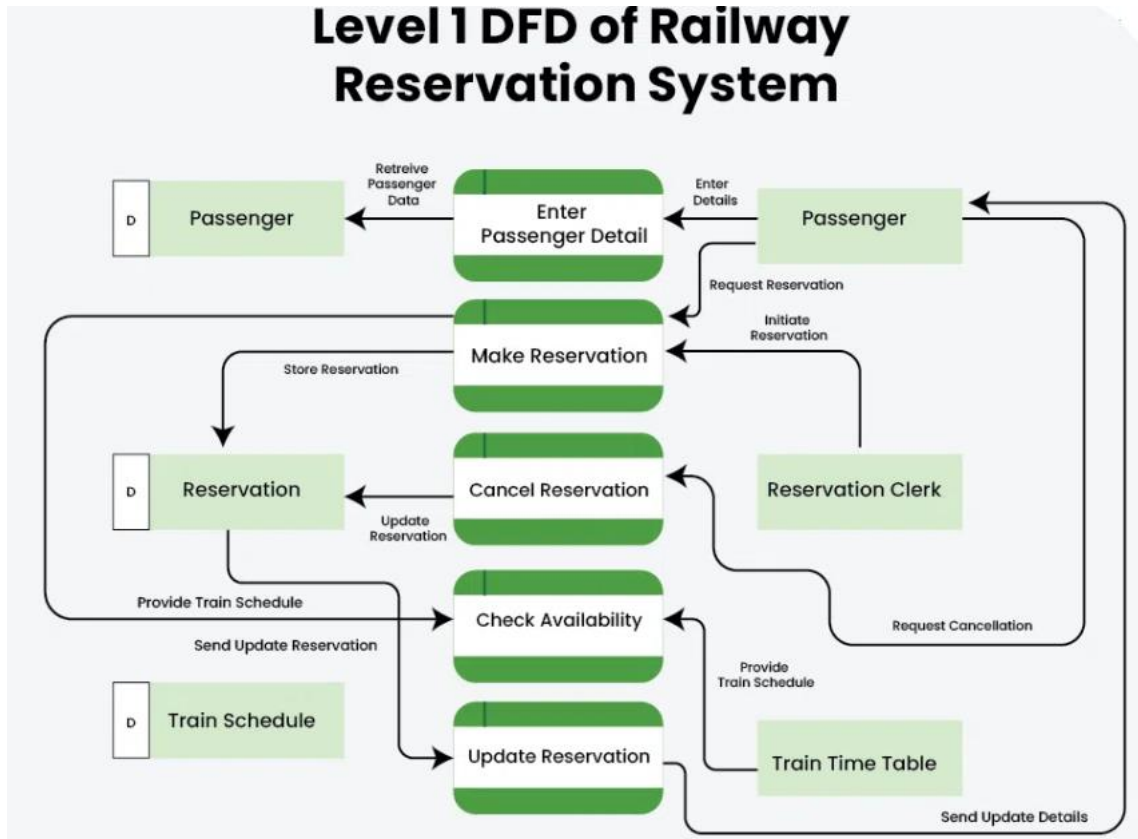
#### **3 1-level DFD**

This level provides a more detailed view of the system by breaking down the major processes identified in the level 0 DFD into sub-processes. Each sub-process is depicted as a separate process on the level 1 DFD. The data flows and data stores associated with each sub-process are also shown.

In 1-level DFD, the context diagram is decomposed into multiple bubbles/processes. In this level, we highlight the main functions of the system and breakdown the high-level process of 0-level DFD into subprocesses.

**Prepared By: Dr. Ashok Kumar Bhoi, Assistant Professor**  
**Department Of Computer Science and Engineering.**

**GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI, BHAWANIPATNA**  
**Address: At- Kandha Bandopala, Po- Risigaon, Via- Bhanuipatna**  
**Dist.- Kalahandi, Orissa, Pin- 766003**



#### 4 2-level DFD

This level provides an even more detailed view of the system by breaking down the sub-processes identified in the level 1 DFD into further sub-processes. Each sub-process is depicted as a separate process on the level 2 DFD. The data flows and data stores associated with each sub-process are also shown.



**GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI, BHAWANIPATNA**  
Address: At- Kandha Bandopala, Po- Risigaon, Via- Bhawanipatna  
Dist.- Kalahandi, Orissa, Pin- 766003

### **Experiment 3: Develop structured design for the DFD model developed** **Steps for Developing Structured Design**

Structured design involves converting the **logical DFD** into a **hierarchical module structure** (like charts, pseudo-code, etc.) that shows how the system will work.

- (a) **Objective:** To develop a structured design for the system based on the DFD model created.
- (b) **Aim:** To convert the DFD (logical design) into a structured design that specifies how the system can be implemented using modular programming principles.
- (c) **Procedure:**
  - (i) Identify all **processes** and **data stores** from the DFD.
  - (ii) Decompose the system into **modules**.
  - (iii) Create a **structure chart** to represent module hierarchy and their relationships.
  - (iv) Write **Module Specifications** for each module.
  - (v) Define **control hierarchy** and data flow between modules.

(d) Example

#### **Level-0 DFD: Library Management System**

- **Processes:**
  - Issue Book
  - Return Book
  - Manage Member
- **Data Stores:**
  - Books Database
  - Member Records

#### **Structure Design:**

##### **1. Structure Chart:**

```
sql
CopyEdit
Library Management System
├── Manage Books
│   ├── Issue Book
│   ├── Return Book
│   └── Update Book Details
├── Manage Members
│   ├── Add Member
│   ├── Update Member Info
│   └── Delete Member
└── Generate Reports
```

##### **2. Module Specifications:**

- **Module Name:** Issue Book
  - **Inputs:** Book ID, Member ID

**Prepared By: Dr. Ashok Kumar Bhoi, Assistant Professor**  
**Department Of Computer Science and Engineering.**



**GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI, BHAWANIPATNA**  
**Address: At- Kandha Bandopala, Po- Risigaon, Via- Bhawanipatna**  
**Dist.- Kalahandi, Orissa, Pin- 766003**

- **Outputs:** Updated Book Database
- **Function:** Check availability, update records, and issue the book.
- **Module Name:** Add Member
  - **Inputs:** Member details
  - **Outputs:** Updated Member Records
  - **Function:** Add new member to the database.
- **Data Dictionary (Optional):**

Data Item Description:  
Book ID: Unique identifier for books  
Member ID Unique identifier for members

Conclusion: The system was decomposed into modules using structured design principles.  
This helps in easy implementation, testing, and maintenance

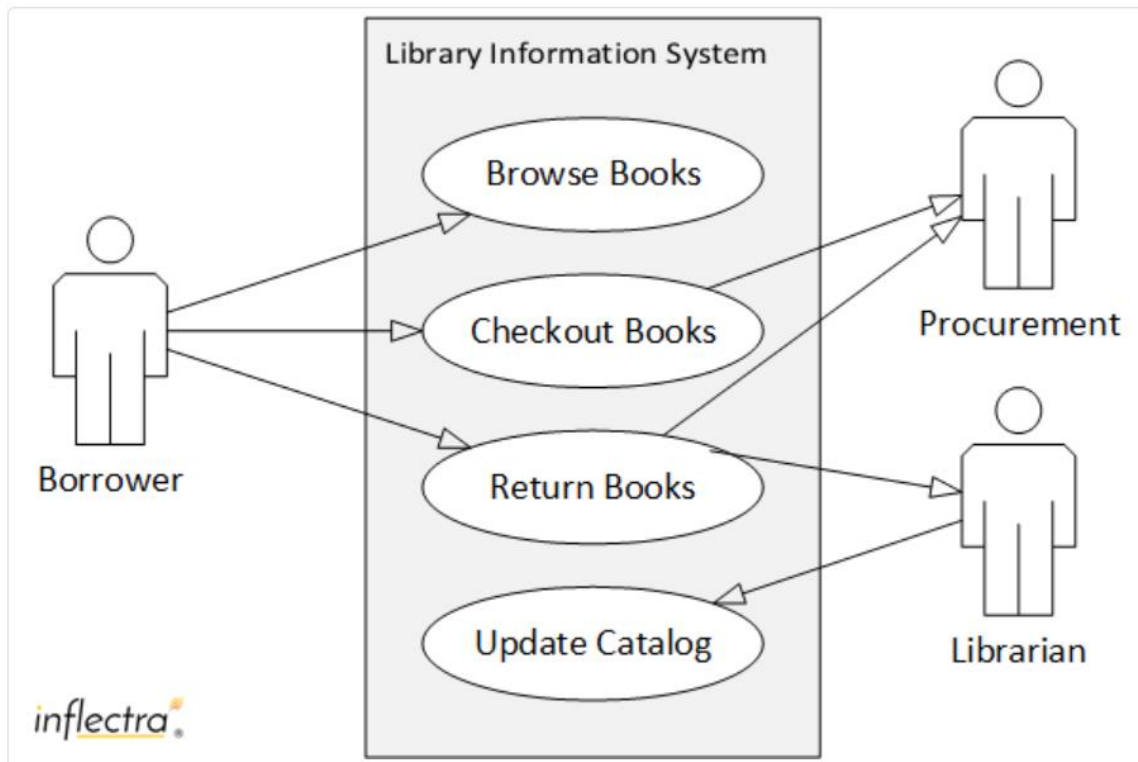


**GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI, BHAWANIPATNA**  
**Address: At- Kandha Bandopala, Po- Risigaon, Via- Bhawanipatna**  
**Dist.- Kalahandi, Orissa, Pin- 766003**

**Experiment 4: Develop UML Use case model for a problem (Use of a CASE tool any of Rational rose, Argo UML, or Visual Paradigm etc. is required)**

- (a) Objective: To analyze the requirements of a system and develop a UML Use Case model for it.
- (b) Aim: To identify actors and use cases of the system and represent their interactions using a UML Use Case diagram.
- (c) Tools:  StarUML / Visual Paradigm / Lucidchart / Any UML tool Paper and pencil (for hand-drawn diagrams)
- (d) A Use Case Diagram shows the functional requirements of a system. It consists of:
- **Actors** (users or other systems interacting with the system)
  - **Use Cases** (functions or services provided by the system)
  - Relationships like **include**, **extend**, and **generalization**
- (e) Procedure:
- I. Identify the problem domain and describe the system briefly.
  - II. Identify **actors** (users or external systems).
  - III. Identify **use cases** (functions performed by the system).
  - IV. Draw relationships between actors and use cases.
  - V. Represent the diagram using a UML tool or draw it manually.
- (f) Example:
- (g) "**Develop a Use Case model for a Library Management System.**"
- (h) Identify The Actors
- **Librarian**: Manages books and members.
  - **Member**: Borrows and returns books.
  - **Admin**: Maintains system settings.
- (i) **Identify Use cases**
- • Issue Book
  - • Return Book
  - • Search Book
  - • Add New Member
  - • Generate Reports

**GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI, BHAWANIPATNA**  
Address: At- Kandha Bandopala, Po- Risigaon, Via- Bhawanipatna  
Dist.- Kalahandi, Orissa, Pin- 766003



(j) Conclusion:

The system's functional requirements were identified and represented visually using a Use Case diagram, which helps in understanding user interactions with the system.



**GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI, BHAWANIPATNA**  
**Address: At- Kandha Bandopala, Po- Risigaon, Via- Bhawanipatna**  
**Dist.- Kalahandi, Orissa, Pin- 766003**

### **Experiment 5: Develop Sequence Diagrams.**

Objective :

To model the dynamic behavior of a system using UML Sequence Diagram by identifying the interactions between objects in a time sequence.

Aim: To design a UML Sequence Diagram for the given problem to represent object interactions and message flow in the system.

Tools Required :

- UML Modeling Tool (StarUML, Visual Paradigm, Lucidchart, etc.)
- Paper & Pencil (for manual sketching)

Description: A **Sequence Diagram** shows how objects interact in a particular scenario of a use case.

It displays:

<b>Objects</b>	(represented	as	lifelines)
<b>Messages</b>	(represented	as	arrows)

**Activation bars** (to show when an object is active).

Sequence Diagrams help in understanding **system behavior over time**.

Procedure:

- Identify the system and the scenario to model.
- Identify all the objects involved in the scenario.
- List all the messages exchanged between objects.
- Draw lifelines for objects.
- Represent the sequence of messages using arrows in top-to-bottom order (time flows from top to bottom).
- Model activation bars on lifelines where required.

Problem Statement:

**"Develop a Sequence Diagram for the 'Issue Book' process in a Library Management System."**

Identified Object:

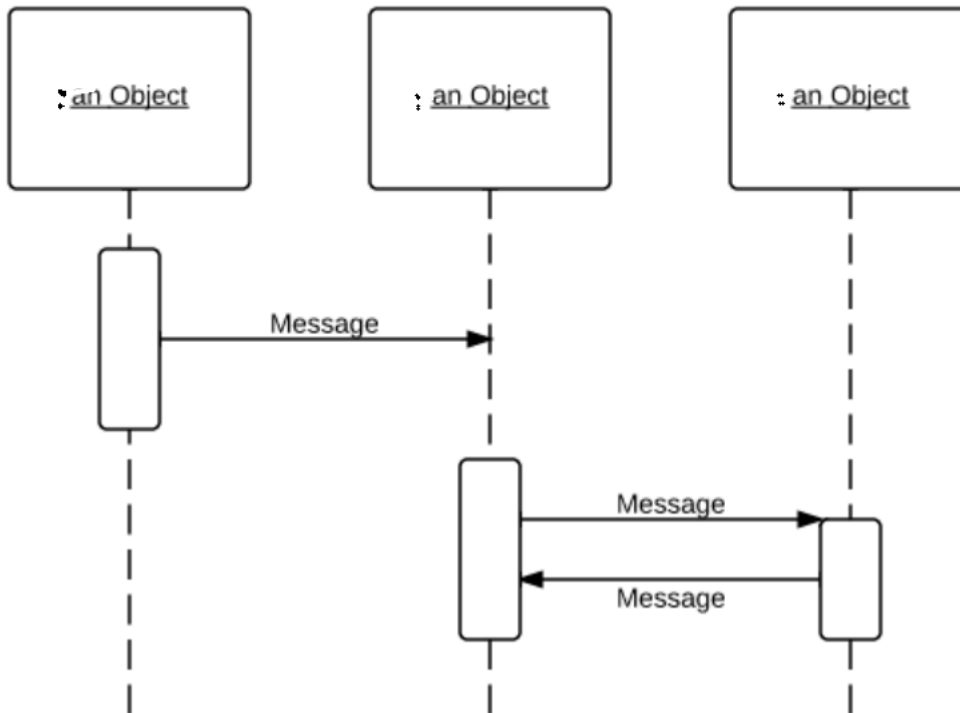
- Member
- Librarian
- Library System

Message In the Scenario

- • Member requests to issue a book.
- • Librarian verifies member credentials.
- • Library System checks book availability.
- • Library System updates records.
- • Librarian confirms the issue to the member.



**GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI, BHAWANIPATNA**  
**Address: At- Kandha Bandopala, Po- Risigaon, Via- Bhawanipatna**  
**Dist.- Kalahandi, Orissa, Pin- 766003**



**Conclusion:**

A Sequence Diagram for the given system was developed, effectively representing object interactions in a time-ordered sequence.



**GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI, BHAWANIPATNA**  
**Address: At- Kandha Bandopala, Po- Risigaon, Via- Bhanuaniapatna**  
**Dist.- Kalahandi, Orissa, Pin- 766003**

### **Experiment 6: Develop Class diagrams.**

**Objective:** To analyze the system requirements and develop a UML Class Diagram representing the static structure of the system.

**Aim:** To model the classes, attributes, operations, and relationships of a system using a UML Class Diagram.

- Tools: UML Modeling Tool (StarUML, Lucidchart, Visual Paradigm, etc.)
- Paper and pencil (for hand-drawn diagrams)

**Description:** A Class Diagram is a type of UML diagram that shows:

- (a) Classes (blueprints for objects)
  - (b) Attributes (data members of a class)
  - (c) Methods (operations or functions)
  - (d) Relationships like Association, Aggregation, Composition, and Inheritance.
- It helps in understanding the object-oriented structure of the system.

**Procedure:**

(i) Identify the key entities (classes) in the system.

(ii) List down their attributes and operations.

(iii) Determine relationships between classes:

- Association: “uses” relationship.
- Aggregation: “has-a” relationship (weak).
- Composition: “owns” relationship (strong).
- Generalization: inheritance (is-a).
- Draw the class diagram using UML notations.

**Example:**

Develop a Class Diagram for a Library Management System."

**Identify Class:**

1. Member

- Attributes: memberID, name, address, phoneNumber
- Methods: register(), updateProfile()

2. Book

- Attributes: bookID, title, author, status
- Methods: checkAvailability(), updateStatus()

3. Librarian

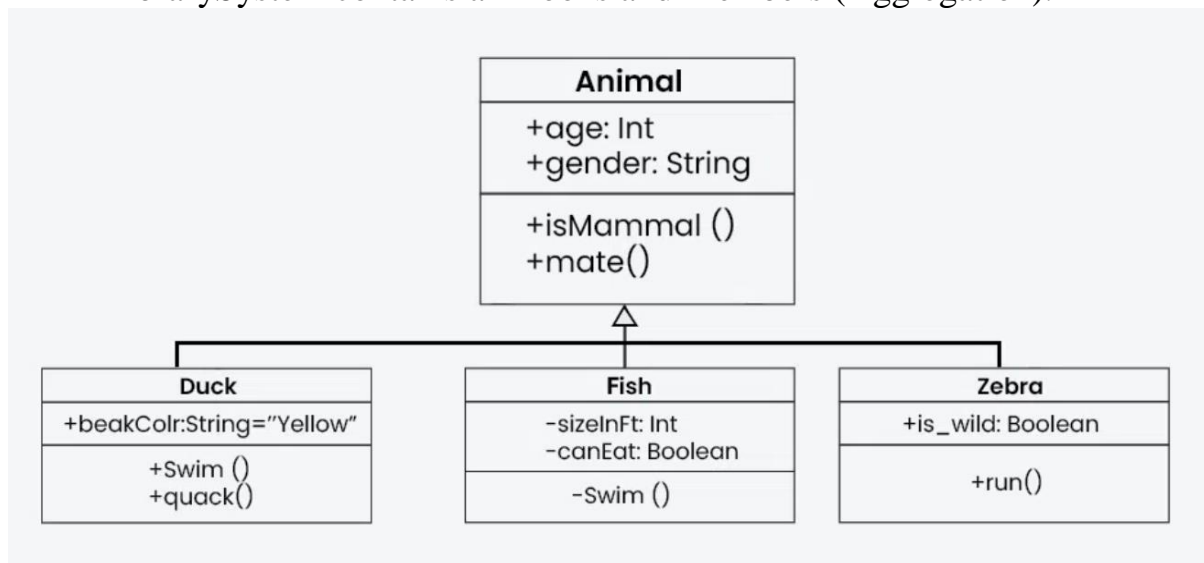
- Attributes: employeeID, name, designation
- Methods: addBook(), removeBook(), issueBook()



**GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI, BHAWANIPATNA**  
Address: At- Kandha Bandopala, Po- Risigaon, Via- Bhawanipatna  
Dist.- Kalahandi, Orissa, Pin- 766003

#### 4. LibrarySystem

- Attributes: systemName
- Methods: searchBook(), generateReport()
- Relationships:
  - Member borrows Book (Association).
  - Librarian manages Book (Association).
  - LibrarySystem contains all Books and Members (Aggregation).



The system's static structure was modeled using a Class Diagram, helping to understand the relationships between different system components.

Experiment 7: Develop code for the developed class model using Java.

Objective: To implement the developed UML Class Model using Java programming language.

Aim: To convert the static design (class diagram) into Java classes, defining their attributes, methods, and relationships.

- Tools: Java Development Kit (JDK)
- IDE like Eclipse / IntelliJ IDEA / NetBeans / BlueJ
- Command prompt for compilation (optional)

Description:

- I. The **class diagram** acts as a blueprint for developing Java code.
- II. Each **class** in the diagram becomes a Java class.
- III. **Attributes** are implemented as instance variables.
- IV. **Methods** are implemented as class functions.
- V. Relationships like **association**, **aggregation**, and **inheritance** are implemented using references and extends/implements.

**Prepared By: Dr. Ashok Kumar Bhoi, Assistant Professor**  
**Department Of Computer Science and Engineering.**



**GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI, BHAWANIPATNA**  
**Address: At- Kandha Bandopala, Po- Risigaon, Via- Bhawanipatna**  
**Dist.- Kalahandi, Orissa, Pin- 766003**

### **Experiment-7 Develop code for the developed class model using Java.**

**Objective: To generate code from class model**

Example:

"Write Java code for the Library Management System based on the developed class diagram."

```
class Book {
    private String bookID;
    private String title;
    private String author;
    private boolean isAvailable;

    public Book(String bookID, String title, String author) {
        this.bookID = bookID;
        this.title = title;
        this.author = author;
        this.isAvailable = true; // By default, the book is available
    }

    public void checkAvailability() {
        if (isAvailable) {
            System.out.println("Book " + title + " is available.");
        } else {
            System.out.println("Book " + title + " is not available.");
        }
    }

    public void updateStatus(boolean status) {
        this.isAvailable = status;
    }

    public String getTitle() {
        return title;
    }
}
java
Copy
```

**Prepared By: Dr. Ashok Kumar Bhoi, Assistant Professor**  
**Department Of Computer Science and Engineering.**



**GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI, BHAWANIPATNA**  
**Address: At- Kandha Bandopala, Po- Risigaon, Via- Bhawanipatna**  
**Dist.- Kalahandi, Orissa, Pin- 766003**

Edit

// Class: Member

```
class Member {
    private String memberID;
    private String name;
    private String address;
    private String phoneNumber;

    public Member(String memberID, String name, String address, String
phoneNumber) {
        this.memberID = memberID;
        this.name = name;
        this.address = address;
        this.phoneNumber = phoneNumber;
    }

    public void register() {
        System.out.println("Member " + name + " registered successfully.");
    }

    public void updateProfile(String newAddress, String newPhone) {
        this.address = newAddress;
        this.phoneNumber = newPhone;
        System.out.println("Profile updated for " + name);
    }
}
}
java
Copy
```

Edit

// Class: Librarian

```
class Librarian {
    private String employeeID;
    private String name;

    public Librarian(String employeeID, String name) {
        this.employeeID = employeeID;
        this.name = name;
    }
}
```



**GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI, BHAWANIPATNA**  
**Address: At- Kandha Bandopala, Po- Risigaon, Via- Bhanuani Patna**  
**Dist.- Kalahandi, Orissa, Pin- 766003**

```
public void issueBook(Book book, Member member) {
    if (book != null) {
        book.updateStatus(false);
        System.out.println("Book " + book.getTitle() + " issued to " +
member.name);
    }
}

public void returnBook(Book book) {
    if (book != null) {
        book.updateStatus(true);
        System.out.println("Book " + book.getTitle() + " returned.");
    }
}
}
}
java
Copy
Edit
// Main class to demonstrate functionality
public class LibrarySystem {
    public static void main(String[] args) {
        // Creating objects
        Book book1 = new Book("B001", "Data Structures", "Narasimha
Karumanchi");
        Member member1 = new Member("M001", "Alice", "123 Main St",
"9876543210");
        Librarian librarian1 = new Librarian("L001", "Mr. John");

        // Performing actions
        member1.register();
        book1.checkAvailability();
        librarian1.issueBook(book1, member1);
        book1.checkAvailability();
        librarian1.returnBook(book1);
        book1.checkAvailability();
    }
}
```



**GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI, BHAWANIPATNA**  
**Address: At- Kandha Bandopala, Po- Risigaon, Via- Bhawanipatna**  
**Dist.- Kalahandi, Orissa, Pin- 766003**

The UML Class Model was converted into executable Java code, demonstrating object-oriented principles like encapsulation and relationships.



**GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI, BHAWANIPATNA**  
Address: At- Kandha Bandopala, Po- Risigaon, Via- Bhawanipatna  
Dist.- Kalahandi, Orissa, Pin- 766003

### **Experiment 8: Use testing tool such as JUnit.**

Objective: To test Java programs using JUnit testing framework.

Aim: To perform unit testing on Java classes and methods using JUnit.

- I. Tools:  Java Development Kit (JDK)
- II. JUnit (version 4 or 5)
- III. Eclipse / IntelliJ IDEA / NetBeans IDE

Description

- I. **JUnit** is a Java framework used to write and run **repeatable unit tests**.
- II. It helps in **testing individual units** (methods) of a program automatically.
- III. Basic JUnit concepts:
  - (a)  **@Test**: Marks a method as a test case.
  - @BeforeEach**: Code executed before each test case.
  - assertEquals()**: Used to check expected vs actual output.

Procedure:

1. Write Java classes and methods that you want to test.
2. Add JUnit library to your project.
3. Create a test class and annotate test methods with @Test.
4. Write assertions to compare expected and actual results.
5. Run test cases and analyze results.

Example:

**Write JUnit test cases for the Book class in a Library Management System."**

```
public class Book {
    private String title;
    private boolean isAvailable;

    public Book(String title) {
        this.title = title;
        this.isAvailable = true; // default status
    }

    public String getTitle() {
        return title;
    }
}
```

**Prepared By: Dr. Ashok Kumar Bhoi, Assistant Professor**  
**Department Of Computer Science and Engineering.**



**GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI, BHAWANIPATNA**  
**Address: At- Kandha Bandopala, Po- Risigaon, Via- Bhawanipatna**  
**Dist.- Kalahandi, Orissa, Pin- 766003**

```
}

public boolean isAvailable() {
    return isAvailable;
}

public void updateStatus(boolean status) {
    this.isAvailable = status;
}
}

JUnit Test Code (BookTest.java):
java
Copy
Edit
import static org.junit.jupiter.api.Assertions.*; // For assertions
import org.junit.jupiter.api.Test; // For @Test annotation

public class BookTest {

    @Test
    public void testBookCreation() {
        Book book = new Book("Java Programming");
        assertEquals("Java Programming", book.getTitle());
        assertTrue(book.isAvailable()); // Default status should be true
    }

    @Test
    public void testUpdateStatus() {
        Book book = new Book("Python Basics");
        book.updateStatus(false);
        assertFalse(book.isAvailable()); // Now status should be false
    }
}
}
```

Steps to Run the Test (In IDE like Eclipse):

Create a Java project.

Add JUnit 5 library to the project (Right-click → Build Path → Add Libraries → JUnit 5).



**GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI, BHAWANIPATNA**  
**Address: At- Kandha Bandopala, Po- Risigaon, Via- Bhawanipatna**  
**Dist.- Kalahandi, Orissa, Pin- 766003**

Create the Book class.

Create the BookTest class in a separate folder (usually src/test/java).

Right-click on BookTest.java → Run As → JUnit Test.

Verify that all tests pass (green bar) or fail (red bar).

JUnit framework was used to perform unit testing on Java methods, ensuring they behave as expected under different scenarios.



**GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI, BHAWANIPATNA**  
Address: At- Kandha Bandopala, Po- Risigaon, Via- Bhawanipatna  
Dist.- Kalahandi, Orissa, Pin- 766003

### **Experiment 9: Use a configuration management tool.**

**Objective:** To learn how to use a configuration management tool to manage and track changes in software projects.

**Aim:** To demonstrate the use of a configuration management tool (e.g., Git) for version control in software development.

**Tools:**  Git (Installed locally)

GitHub account (for remote repository)

Command-line interface (Terminal/Command Prompt)

IDE like Eclipse, VS Code (optional)

**Description:**

A **Configuration Management (CM) Tool** helps manage changes to software artifacts over time.

**Git** is a distributed version control system used for tracking changes in source code during software development.

	Key	Git	Concepts:
--	-----	-----	-----------

**Repository (Repo):** A project folder tracked by Git.

**Commit:** A snapshot of project changes.

**Branch:** A parallel version of the project for development.

**Merge:** Combining changes from different branches.

**Push/Pull:** Sending/receiving changes to/from remote servers like GitHub.

1. **Install Git** on your system and set up a GitHub account.
2. Initialize a new Git repository in your project folder using:  
csharp  
CopyEdit  
git init
3. Create a new file (e.g., Book.java) and add it to the repository:  
csharp  
CopyEdit  
git add Book.java
4. Commit the changes:  
sql  
CopyEdit  
git commit -m "Added Book class"
5. Create a remote repository on GitHub.
6. Link your local repository to GitHub:  
csharp



**GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI, BHAWANIPATNA**  
Address: At- Kandha Bandopala, Po- Risigaon, Via- Bhawanipatna  
Dist.- Kalahandi, Orissa, Pin- 766003

CopyEdit

```
git remote add origin <GitHub repository URL>
```

7. Push your changes to GitHub:

```
css
```

CopyEdit

```
git push -u origin main
```

8. Modify the file, commit again, and push changes to demonstrate version tracking.

---

## 5 Example Commands:

```
bash
```

CopyEdit

```
git init
```

```
git add Book.java
```

```
git commit -m "Initial commit with Book class"
```

```
git branch -M main
```

```
git remote add origin https://github.com/username/library-system.git
```

```
git push -u origin main
```

---

## 6 Expected Output:

Your project is uploaded to GitHub.

Changes to files can be tracked with commit history.

Example:

```
sql
```

CopyEdit

```
$ git log
```

```
commit d4f7a3c Initial commit with Book class
```

```
Author: Ashok <ashok@example.com>
```

```
Date: Thu Jul 10 10:00 2025 +0530
```

---

## 7 Result:

The configuration management tool (Git) was successfully used to initialize a repository, commit changes, and manage versions of the software project.

---

## 8 Conclusion:

Version control was achieved using Git, enabling efficient collaboration, change tracking, and management of the software project.



**GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI, BHAWANIPATNA**  
Address: At- Kandha Bandopala, Po- Risigaon, Via- Bhawanipatna  
Dist.- Kalahandi, Orissa, Pin- 766003

**Experiment 10: Use any one project management tool such as Microsoft Project, Gantt Project or ProjectLibre.**

Objective: To plan and manage a software project using a project management tool.

Aim: To learn how to create and manage a project schedule using tools like Microsoft Project, GanttProject, or ProjectLibre.

- Tools: Any one of the following:
  - Microsoft Project
  - GanttProject
  - ProjectLibre
- A computer system with the tool installed.

Description:

**Project Management Tools** help manage software projects by assisting in:

- Planning tasks and activities
- Assigning resources
- Scheduling and tracking project progress
- Visualizing timelines using **Gantt Charts** and **Task Dependencies**

**Gantt Chart:** A visual representation of the project schedule, showing tasks along a timeline.

Procedure:

1. Install and open the selected project management tool.
2. Create a **new project** and set project details:
  - Project name
  - Start date
  - Duration
3. Identify the major tasks in the project.
4. Break down tasks into subtasks (Work Breakdown Structure).
5. Set dependencies between tasks (e.g., Task B starts after Task A).
6. Assign resources (e.g., developers, testers).
7. Generate and view the **Gantt Chart** to visualize the schedule.
8. Save and export the project plan for future tracking.

**9 Problem Statement (Example):**

Prepared By: **Dr. Ashok Kumar Bhoi**, Assistant Professor  
Department Of Computer Science and Engineering.



**GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI, BHAWANIPATNA**  
Address: At- Kandha Bandopala, Po- Risigaon, Via- Bhawanipatna  
Dist.- Kalahandi, Orissa, Pin- 766003

**"Plan the Library Management System project using a project management tool."**

#### **10 Identified Tasks:**

<b>Task No</b>	<b>Task Description</b>	<b>Duration</b>	<b>Predecessor Task</b>
1	Requirement Analysis	3 days	-
2	System Design	4 days	1
3	Database Design	2 days	2
4	Implementation	5 days	3
5	Testing	3 days	4
6	Deployment	2 days	5

#### **11 Expected Output (Gantt Chart):**

A Gantt Chart showing tasks with their start and end dates, duration, and dependencies.

pgsql  
CopyEdit  
Requirement Analysis |  
System Design |  
Database Design |  
Implementation |  
Testing |  
Deployment |

#### **12 Result:**

The software project was successfully planned and managed using the selected project management tool. The project schedule was visualized using a Gantt Chart.

#### **13 Conclusion:**

Project management tools like Microsoft Project, GanttProject, or ProjectLibre provide an efficient way to plan, schedule, and monitor software projects. They help ensure timely delivery and resource optimization.

#### **14 Diagram for Lab Record:**

*(Include this if you need to submit a hand-drawn chart or screenshot)*

pgsql  
CopyEdit



**Prepared By: Dr. Ashok Kumar Bhoi, Assistant Professor**  
**Department Of Computer Science and Engineering.**



**GOVERNMENT COLLEGE OF ENGINEERING KALAHANDI, BHAWANIPATNA**  
**Address: At- Kandha Bandopala, Po- Risigaon, Via- Bhawanipatna**  
**Dist.- Kalahandi, Orissa, Pin- 766003**

+-----+	
Gantt Chart	
+-----+	
Task	Timeline
-----	
Req. Analysis	
System Design	
Database Design	
Implementation	
Testing	
Deployment	
+-----+	

**Do you want me to:**

-  Create a **ready-made Gantt Chart screenshot** (from *GanttProject* or *ProjectLibre*) for you to paste in your lab record?
-  Or prepare a **step-by-step guide with screenshots** for **Microsoft Project** (since it's most commonly used in labs)?

Which tool do you need to use in your lab: **Microsoft Project**, **GanttProject**, or **ProjectLibre**?

Prepared By:

Dr. Ashok Kumar Bhoi,

Assistant Professor