A REPORT OF SIX MONTH TRAINING

At

JSPIDERS NOIDA

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENT FOR THE AWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY

(INFORMATION TECHNOLOGY)

JULY-DEC, 2024

SUBMITTED BY

NAME: RAJAT SANDHU

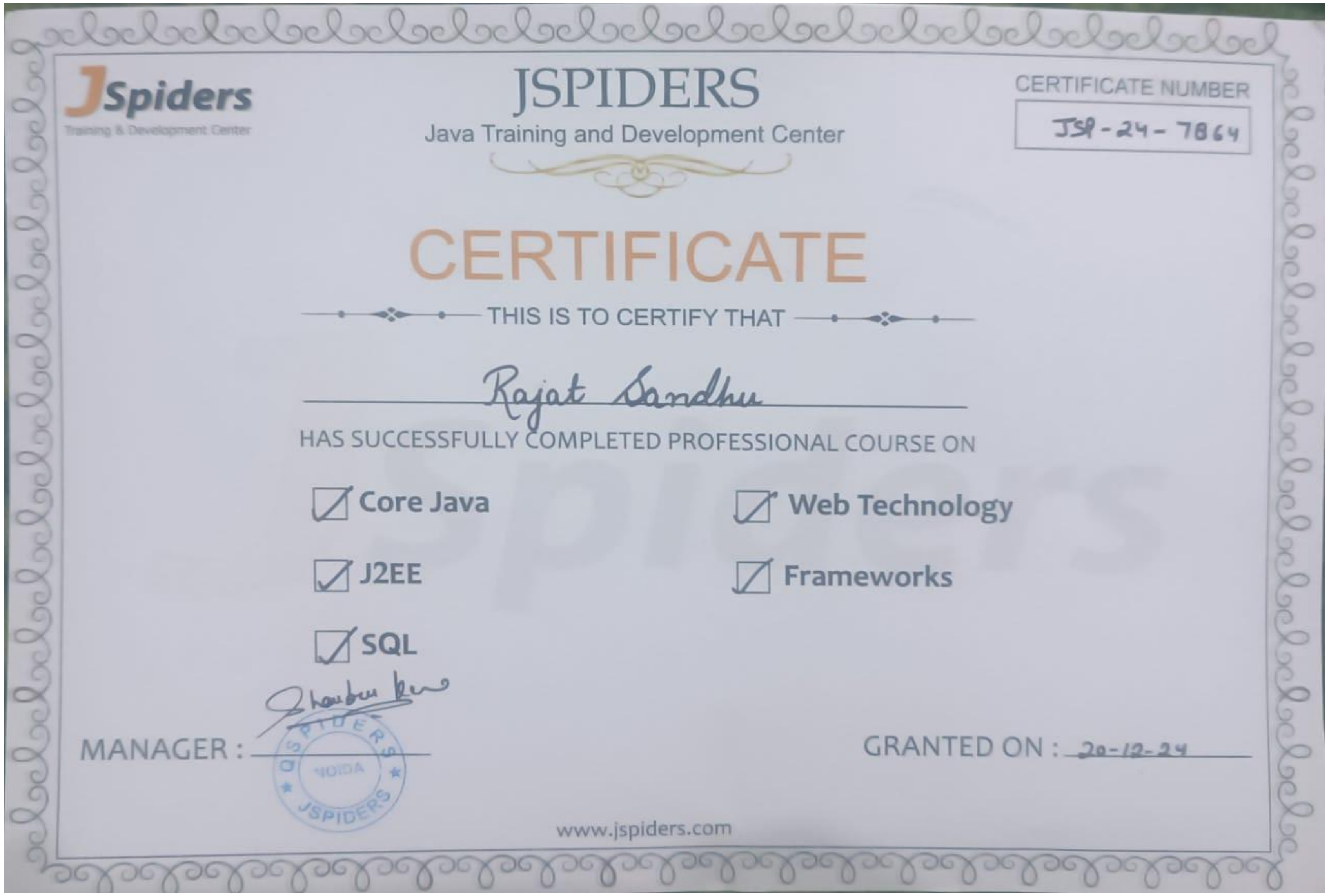
UNIVERSITY ROLL NO: 2104554

DEPARTMENT OF INFORMATION TECHNOLOGY

GURU NANAK DEV ENGINEERING COLLEGE LUDHIANA

(An Autonomous College Under UGC ACT)

CERTIFICATE



JSpiders
Training & Development Center

JSPIDERS
Java Training and Development Center

CERTIFICATE NUMBER
JSP-24-7864

CERTIFICATE

THIS IS TO CERTIFY THAT

Rajat Sandhu

HAS SUCCESSFULLY COMPLETED PROFESSIONAL COURSE ON

- ✓ Core Java
- ✓ J2EE
- ✓ SQL
- ✓ Web Technology
- ✓ Frameworks

MANAGER : [illegible]

JSPIDERS NOIDA

GRANTED ON : 20-12-24

www.jspiders.com

CANDIDATE'S DECLARATION

I Rajat Sandhu hereby declares that I have undertaken six-months training at Jspiders NOIDA during a period from July 2024 to December 2024 in partial fulfilment of requirements for the award of degree of B. Tech (COMPUTER SCIENCE AND ENGINEERING) at GURU NANAK DEV ENGINEERING COLLEGE, LUDHIANA. The work which is being presented in the training report submitted to Department of INFORMATION TECHNOLOGY at GURU NANAK DEV ENGINEERING COLLEGE,LUDHIANA is an authentic record of training work

Signature of the Student

The six-months industrial training Viva–Voce Examination of ____________________ has been held on _____________ and accepted.

Signature of Internal Examiner   Signature of External Examiner

Abstract

The Jspiders Noida 6 months Onsite Training is a focused program designed to provide participants with essential skills in SQL ,core web technologies like HTML,CSS, and JavaScript and advanced java development technologies like JDBC, Hibernate, Servlet, Spring Boot, Spring MVC,JSP and Core Java Conducted by Jspiders Noida, a leading training institute, this course offers a comprehensive introduction to JAVA FULL STACK DEVELOPMENT.

Throughout the six months, participants gain hands-on experience in writing SQL queries, designing databases, and building interactive web pages. The curriculum covers key topics such as data manipulation, responsive web design, and client-side scripting. Real-world projects and exercises ensure that participants can apply their skills practically.

This onsite training provides an engaging learning environment with direct guidance from experienced instructors, fostering collaboration and immediate feedback. The program emphasizes not only technical skills but also problem-solving and best practices in the industry.

Upon completion, participants will be equipped to create efficient databases and develop modern web applications, preparing them for careers in web development and database management.

Acknowledgement

I would like to extend my heartfelt appreciation to both my college and Jspiders Noida for the exceptional opportunity to participate in the 6-months onsite training program. This training experience has been invaluable in enhancing my skills and knowledge in SQL ,web technologies and advance java tools.

First and foremost, I am profoundly grateful to my college for recognizing the importance of providing students with access to such high-quality training programs. This initiative has enabled me and my fellow students to acquire relevant skills and stay competitive in the ever-evolving world of technology.

I would like to express my sincere gratitude to the team at Jspiders Noioda for their dedication and commitment in delivering a comprehensive and well-structured training program. The instructors and mentors demonstrated expertise in their respective fields, making complex concepts accessible and fostering a conducive learning environment.

The hands-on approach of the onsite training allowed me to fully immerse myself in the learning experience. This focused environment was a significant factor in my ability to successfully complete the program and apply the skills I've gained.

This training experience has equipped me with the knowledge and abilities that will undoubtedly benefit my academic and professional pursuits. I am eager to apply what I have learned to make a positive impact in my field.

Once again, I extend my heartfelt thanks to both my college and Jspiders
Delhi for this outstanding learning opportunity.

About Jspiders

JSpiders was established to bridge the gap between academic learning and industry requirements in the field of IT. Founded with the mission to empower students and professionals with technical expertise, JSpiders has grown into a premier software training institute. Recognized for its excellence, JSpiders offers cutting-edge training programs tailored to the evolving demands of the technology sector.

JSpiders Pvt. Ltd. is an India-based training provider specializing in software testing, development, and IT skill enhancement. The company has consistently expanded its reach since its inception, establishing training centers across India and abroad. Featured in prominent educational and tech magazines, JSpiders has gained recognition for its quality education and placement support. The organization has its main headquarters in Bangalore and has expanded its operations to other metropolitan cities to cater to a diverse student base.

In 2008, JSpiders initiated its flagship programs in manual and automation testing, quickly becoming a go-to institute for aspiring testers. By 2013, the institute diversified its offerings, introducing Java, Python, and full-stack development courses. JSpiders has been instrumental in training and placing thousands of students, with an impressive placement record in leading IT companies like Infosys, Wipro, and TCS.

jSpiders is an ISO-certified institution and a registered member of professional training bodies, reflecting its commitment to quality and standards. In 2015, the institute embraced online learning to expand its accessibility and introduced interactive e-learning modules. The company also ventured into corporate training, helping organizations upskill their employees and stay competitive in the technology-driven world.

LIST OF FIGURES/TABLES

Definitions, Acronyms and Abbreviations

JRE (Java Runtime Environment): A subset of the Java Software Development Kit (SDK) that provides the libraries and resources necessary to run Java applications.

JDK (Java Development Kit): A superset of the JRE that includes development tools such as the Java compiler, Javadoc, Jar, and debugger.

RMI (Remote Method Invocation): A Java API that allows an object running in one Java Virtual Machine to invoke methods on an object in another Java Virtual Machine.

JNI (Java Native Interface): A framework that allows Java code running in the Java Virtual Machine to call and be called by native applications and libraries written in other languages like C or C++.

WORA (Write Once, Run Anywhere): A principle of Java that means code written in Java can be run on any platform that has a compatible Java Virtual Machine (JVM) without requiring modification.

API: Application Programming Interface - A set of routines, protocols, and tools for building software and applications. In Java, it refers to the classes and methods provided by the Java Standard Library for developers to use.

IDE: Integrated Development Environment - A software application that provides comprehensive facilities to programmers for software development. In Java, popular IDEs include Eclipse, IntelliJ IDEA, and NetBeans.

JAR: Java Archive - A file format used to package Java classes and associated metadata into a single file, typically used for distributing Java applications and libraries.

Class: A blueprint for creating objects in Java. It defines the properties (attributes) and behaviors (methods) that the objects created from the class can have.

Static: A keyword in Java that indicates a method or variable belongs to the class itself rather than to instances of the class. Static members can be accessed without creating an instance of the class.

Applet: A small application that runs within a web browser or as part of a larger application, allowing interactive features on web pages through the JApplet class.

Servlet: A Java server-side component that generates dynamic web content in response to client requests, often producing HTML or XML.

JavaServer Pages (JSP): A technology for creating dynamically generated web pages by embedding Java code in HTML, which is compiled into servlets for execution.

Generics: A Java feature that allows defining classes and methods with placeholder types, enabling stronger type checks at compile-time and reducing type casting.

Swing: A Java GUI toolkit that provides components for building window-based applications, allowing cross-platform development with a consistent look and feel.

Class Libraries: Compiled bytecodes that support Java application development, including core libraries for data structures, XML processing, and security; integration libraries like JDBC for database access; and user interface libraries such as AWT and Swing.

Javadoc: A documentation system by Sun Microsystems for generating organized documentation in Java, using specific comment tags (/** and */) to distinguish them from regular comments.

Java Virtual Machine (JVM): The runtime environment that executes Java bytecode, allowing applications to run across different platforms without modification.

Java Database Connectivity (JDBC): An API that enables Java applications to connect to and interact with databases, facilitating SQL execution and transaction management.

CONTENTS

CHAPTER 1: INTRODUCTION

1.1 SQL:

SQL Commands
SQL commands are instructions. It is used to communicate with the database. It is also used to perform specific tasks, functions, and queries of data.
SQL can perform various tasks like create a table, add data to tables, drop the table, modify the table, set permission for users.

Types of SQL Commands :

There are five types of SQL commands: DDL, DML, DCL, TCL, and DQL.

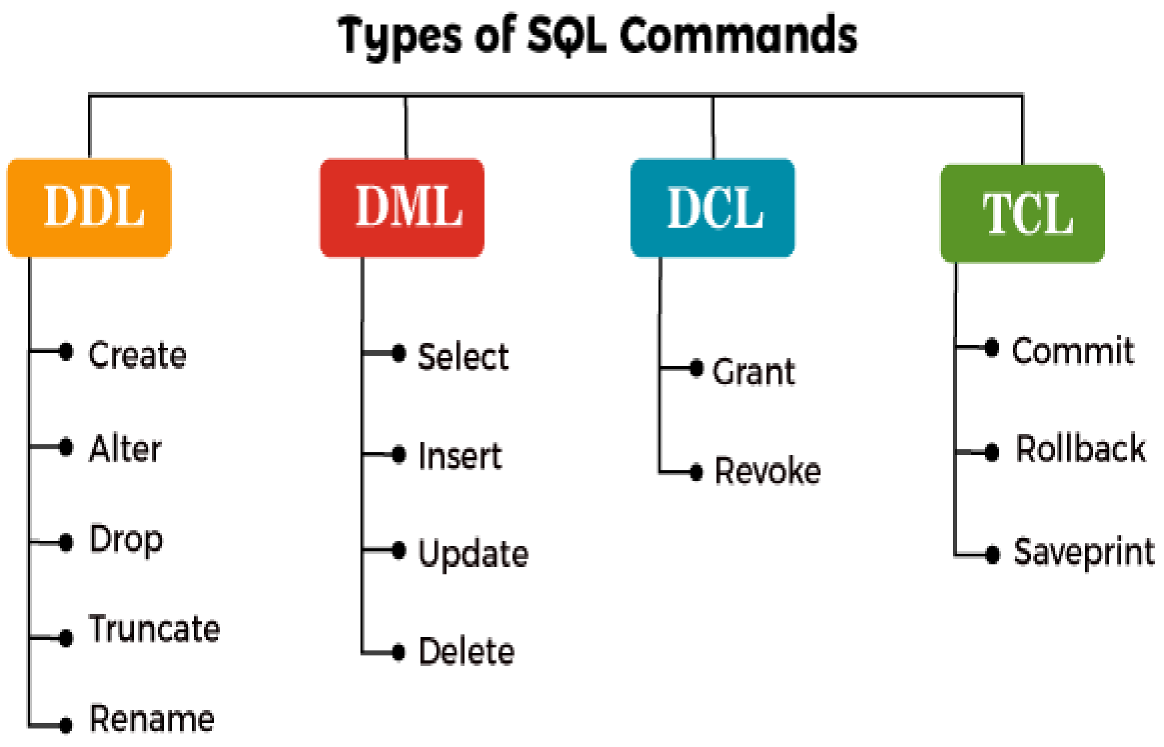


Figure 1.1 Types of Sql commands

1. Data Definition Language (DDL)

DDL changes the structure of the table like creating a table, deleting a table, altering a table, etc. All the command of DDL are auto-committed that means it permanently save all the changes in the database.

Here are some commands that come under DDL:

- CREATE
- ALTER
- DROP
- TRUNCATE

a. CREATE It is used to create a new table in the database.

Syntax:

CREATE TABLE TABLE_NAME (COLUMN_NAME DATATYPES[,....]);

Example:

CREATE TABLE EMPLOYEE(Name VARCHAR2(20), Email VARCHAR2(100), DOB DATE);

b. DROP: It is used to delete both the structure and record stored in the table.

Syntax

DROP TABLE ;

Example

DROP TABLE EMPLOYEE;

c. ALTER: It is used to alter the structure of the database. This change could be either to modify the characteristics of an existing attribute or probably to add a new attribute.

Syntax:

To add a new column in the table

ALTER TABLE table_name ADD column_name COLUMN-definition; To
modify existing column in the table:

ALTER TABLE MODIFY(COLUMN DEFINITION....);

EXAMPLE

ALTER TABLE STU_DETAILS ADD(ADDRESS VARCHAR2(20)); ALTER
TABLE STU_DETAILS MODIFY (NAME VARCHAR2(20));

d. TRUNCATE: It is used to delete all the rows from the table and free the space containing the table.

Syntax:

TRUNCATE TABLE table_name; Example:

TRUNCATE TABLE EMPLOYEE;

2. Data Manipulation Language

DML commands are used to modify the database. It is responsible for all form of changes in the database.

The command of DML is not auto-committed that means it can't permanently save all the changes in the database. They can be rollback.

Here are some commands that come under DML:

- INSERT
- UPDATE
- DELETE

a. INSERT: The INSERT statement is a SQL query. It is used to insert data into the row of a table.

Syntax:

INSERT INTO TABLE_NAME

VALUES (value1, value2, value3, ... valueN);   For

example:

INSERT INTO javatpoint (Author, Subject)

VALUES ("Sonoo", "DBMS");

b. UPDATE: This command is used to update or modify the value of a column in the table.

Syntax:

UPDATE table_name SET [column_name1= value1,...column_nameN = valueN] [WHERE CONDITION]   For example:

UPDATE students

SET User_Name = 'Sonoo'

WHERE Student_Id = '3'

c. DELETE: It is used to remove one or more row from a table.

Syntax:

DELETE FROM table_name [WHERE condition];   For

example:

DELETE FROM javatpoint

WHERE Author="Sonoo";

3. Data Control Language

DCL commands are used to grant and take back authority from any database user.

Here are some commands that come under DCL:

- Grant
- Revoke

a. Grant: It is used to give user access privileges to a database.

Example

```
GRANT SELECT, UPDATE ON MY_TABLE TO SOME_USER, ANOTHER_USER;
```

b. Revoke: It is used to take back permissions from the user.

Example

```
REVOKE SELECT, UPDATE ON MY_TABLE FROM USER1, USER2;
```

4 Data Query Language

DQL is used to fetch the data from the database.

It uses only one command:

- SELECT

a. SELECT: This is the same as the projection operation of relational algebra. It is used to select the attribute based on the condition described by WHERE clause.

Syntax:

```
SELECT expressions
FROM TABLES
WHERE conditions;
```

For example:

```
SELECT emp_name
FROM employee
WHERE age > 20;
```

- **CREATE TABLE**

The SQL CREATE TABLE statement is used to create a new table.

Syntax

The basic syntax of the CREATE TABLE statement is as follows –

```
CREATE TABLE table_name(  column1 datatype,  column2
datatype,  column3 datatype,
.....
columnN datatype,
PRIMARY KEY( one or more columns )
);
CREATE TABLE Employees_details(
ID int,
Name varchar(20),
Address varchar(20)
);
```

Ex:

```
CREATE TABLE CUSTOMERS(
ID   INT    NOT NULL,
NAME VARCHAR (20)    NOT NULL,
AGE  INT      NOT NULL,
ADDRESS  VARCHAR (25) ,
SALARY   DECIMAL (18, 2),
PRIMARY KEY (ID)
);
```

• **DROP TABLE**

The SQL DROP TABLE statement is used to remove a table definition and all the data, indexes, triggers, constraints and permission specifications for that table.

NOTE − You should be very careful while using this command because once a table is deleted then all the information available in that table will also be lost forever.

Syntax

```
DROP TABLE table_name;
```

• **INSERT INTO**

The SQL INSERT INTO Statement is used to add new rows of data to a table in the database.

Syntax

There are two basic syntaxes of the INSERT INTO statement which are shown below.

INSERT INTO TABLE_NAME (column1, column2, column3,...columnN)
VALUES (value1, value2, value3,...valueN);

Here, column1, column2, column3,...columnN are the names of the columns in the table into which you want to insert the data. You may not need to specify the column(s) name in the SQL query if you are adding values for all the columns of the table. But make sure the order of the values is in the same order as the columns in the table.

The SQL INSERT INTO syntax will be as follows −

INSERT INTO TABLE_NAME VALUES (value1,value2,value3,...valueN); Example
:

The following statements would create six records in the CUSTOMERS table.

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (1, 'Ramesh', 32, 'Ahmedabad', 2000.00);
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (2, 'Khilan', 25, 'Delhi', 1500.00);
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (3, 'kaushik', 23, 'Kota', 2000.00);
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (4, 'Chaitali', 25, 'Mumbai', 6500.00);
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (5, 'Hardik', 27, 'Bhopal', 8500.00);
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (6, 'Komal', 22, 'MP', 4500.00);

You can create a record in the CUSTOMERS table by using the second syntax as shown below.

INSERT INTO CUSTOMERS
VALUES (7, 'Muffy', 24, 'Indore', 10000.00);

All the above statements would produce the following records in the CUSTOMERS table as shown below.

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |  |
6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

1.2 FRONTEND DEVELOPMENT

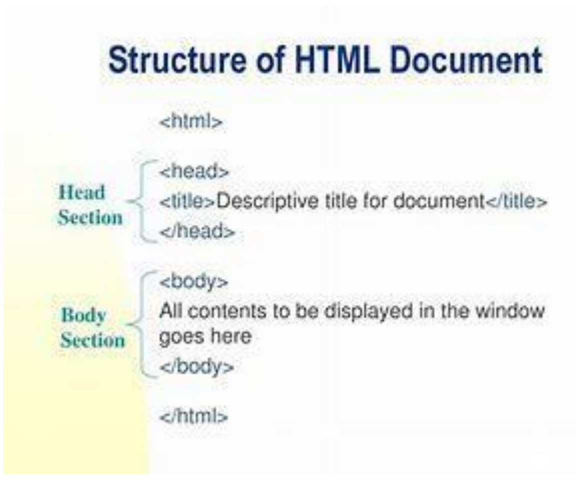
1.2.1 HTML

HTML (Hypertext Markup Language) is the standard markup language used to create web pages. It serves as thebackbone of web content, enabling developers to structure their documents and define how elements are displayed ina web browser. This unit will provide a comprehensive overview of HTML, including its basic concepts, structure,evolution, essential tags, text formatting, and hyperlinks.

Key Characteristics of HTML:

- Markup Language: HTML uses tags to define elements.
- Static Content: HTML is primarily used for static content, dynamic features require additional languages like JavaScript.
- Hyperlinks: HTML allows linking to other documents or resources via hyperlinks, creating a network of interconnected information.

Structure of HTML Documents:
An HTML document is structured hierarchically. At the highestlevel, it consists of various elements, each defined by tags.An HTML document typically includes:



```
<html>

<head>
<title>Descriptive title for document</title>
</head>

<body>
All contents to be displayed in the window
goes here
</body>

</html>
```

Figure 1.2 Structure of HTML

- Doctype Declaration: Specifies the version of HTML being used (e.g., <!DOCTYPE html>).
- HTML Element: The root element that encapsulates the entire document.
- Head Section: Contains metadata about the document, including title, links to stylesheets, and scripts.
- Body Section: Contains the visible content of the webpage, including text, images, and other media.

HTML supports both ordered and unordered lists, which help organize content clearly.

- Ordered List: Created with the <ol> tag, it presents items in a numbered format.
- Unordered List: Created with the <ul> tag, it presents items with bullet points.
- List Items: Each item within a list is defined by the <li>tag.

c) Block-Level Elements

Block-level elements, such as <div>, <header>, <footer>, and<section>, take up the full width available and begin on anew line. They are used to structure the layout of a webpage.

5. Hyperlinks

Hyperlinks are a fundamental feature of HTML, allowing users to navigate between different webpages or resources.

a) Internal Links

Internal links connect to other pages within the same website. They are created using the <a> (anchor) tag, with the href attribute specifying the destination URL.

e.g: <a href =“about.html”> About us</a>

b) External Links

External links point to pages on different websites. Similar to internal links, they are created with the <a> tag, but they navigate users away from the current site.

e.g: <a href=”https://www.w3schools.com/”>Visit W3Schools.com!</a>

c) Mailto Links

Mailto links allow users to send emails directly from the webpage. These links use the mailto: protocol within the href attribute (e.g., <a href=”mailto:someone@example.com”>Email Us</a>). **Working with Images**

Images play a vital role in enhancing the visual appeal of a website. HTML provides the <img> tag to insert images into web pages. Let’s explore how to use it effectively. Inserting Images with the <img> Tag The <img> tag is used to embed images in a webpage. It is a self-closing tag, which means it doesn’t need an end tag. The basic structure of an image tag is:
e.g : <img src=”img_girl.jpg” alt=”Girl in a jacket” width=”500” height=”600”>

- src (source): Specifies the path to the image file. This could either be a relative path (for local images) or an absolute URL (for external images).
- alt (alternative text): Provides a textual description of the image. This is essential for accessibility purposes, such as for screen readers, or when the image fails to load.

Audio and Video:
Embedding Multimedia Elements Multimedia elements like audio and video are essential for interactive websites. HTML5 introduces the <audio> and <video> tags, which make embedding these media types easier and more efficient. Embedding Audio with <audio> The <audio> tag is used to embed audio files into a webpage.

The syntax is simple:

- controls: Adds play, pause, volume, and playback controls to the audio player.
- autoplay: If present, the audio will begin playing automatically when the page loads.
- loop: The audio will restart once it finishes playing.

To provide better cross-browser support, you can include multiple source formats:
```
<audio controls>
  <source src=”horse.ogg” type=”audio/ogg”>
  <source src=”horse.mp3” type=”audio/mpeg”>
 </audio>
```
This ensures that if a browser does not support one format, it will try the next available format.

Embedding Video with <video>:

Similarly, the <video> tag is used to embed video files. Here is an example of how to use it:

- controls: Adds playback controls for the video (play,pause, volume).
- autoplay: The video starts automatically when the page loads.
- muted: Starts the video with no sound.
- poster: Displays a placeholder image before the video starts playing.

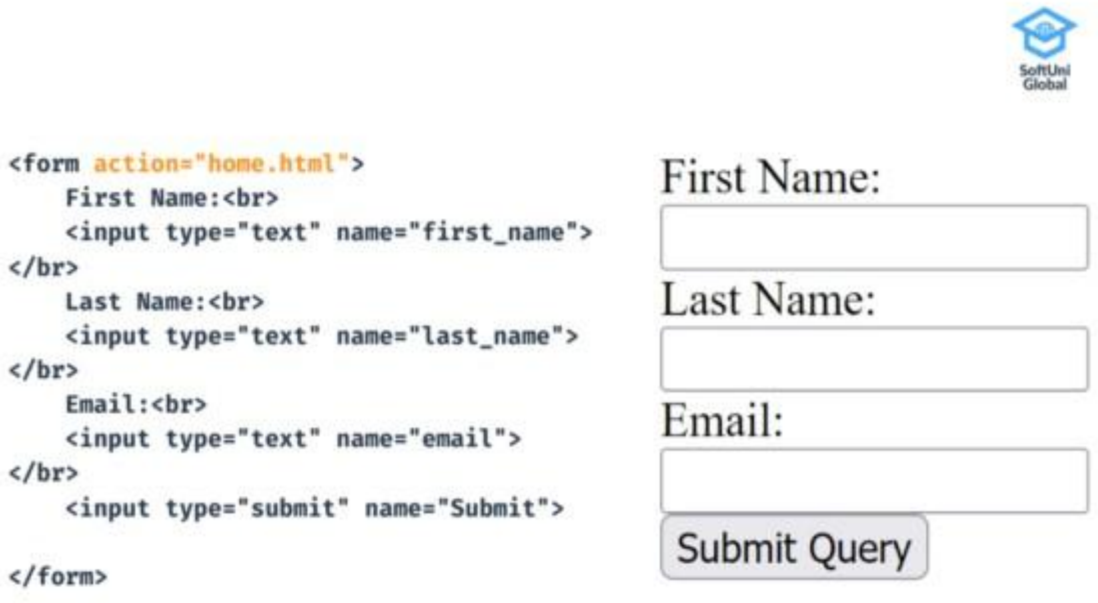
Just like audio, you can include multiple video formats to ensure compatibility across different browsers:

```
<video width="320" height="240" controls>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
 </video>
```

The poster attribute provides an image (thumbnail) to show before the video starts playing.

Forms in HTML

Forms are fundamental in allowing users to interact with websites. HTML provides the <form> tag and several form elements that allow users to input and submit data. Basic Form Structure



```
<form action="home.html">
    First Name:<br>
    <input type="text" name="first_name">
</br>
    Last Name:<br>
    <input type="text" name="last_name">
</br>
    Email:<br>
    <input type="text" name="email">
</br>
    <input type="submit" name="Submit">

</form>
```

Figure 1.3 Forms Structure

The <form> tag is used to define the boundaries of the form, and the action attribute specifies where the form data will be sent for processing. The method attribute defines how the form data will be sent (e.g., GET or POST).

- action: Specifies the URL where the form data will be sent.
- method: Defines the method used to send form data. The two most common methods are GET (appends data to the URL) and POST (sends data in the request body).

Common Form Input Types

Forms in HTML support various input types for different purposes. Some common input types include:

- Text Input: Used for single-line user input (e.g., a name or email).
  <input type="text" id="username" name="username">
- Password Input: Used for passwords, where the entered text is obscured.
  <input type="password" id="pwd" name="pwd">
- Radio Buttons: Allows users to choose one option from a set of choices.
  <input type="radio" id="html" name="fav_language" value="HTML">
  <input type="radio" id="css" name="fav_language" value="CSS">
- Checkboxes: Allows users to select one or more options.
  <input type="checkbox" id="vehicle1" name="vehicle1" value="Bike">
- Submit Button: Submits the form data.
  <input type="submit" value="Submit">
- Reset Button: Resets all form fields to their default values.

Form Validation

HTML5 introduced built-in form validation to ensure that users fill out the form correctly. You can specify required fields and use attributes like pattern, min, and max for validation.

Example of required field validation:

<input type="text" id="username" name="username" required>

In this example, the email field is required. If the user tries to submit the form without filling it in, the browser will prompt them to enter a value. Other attributes like pattern, minlength, and maxlength can help further refine the validation process.

Tables and Frames

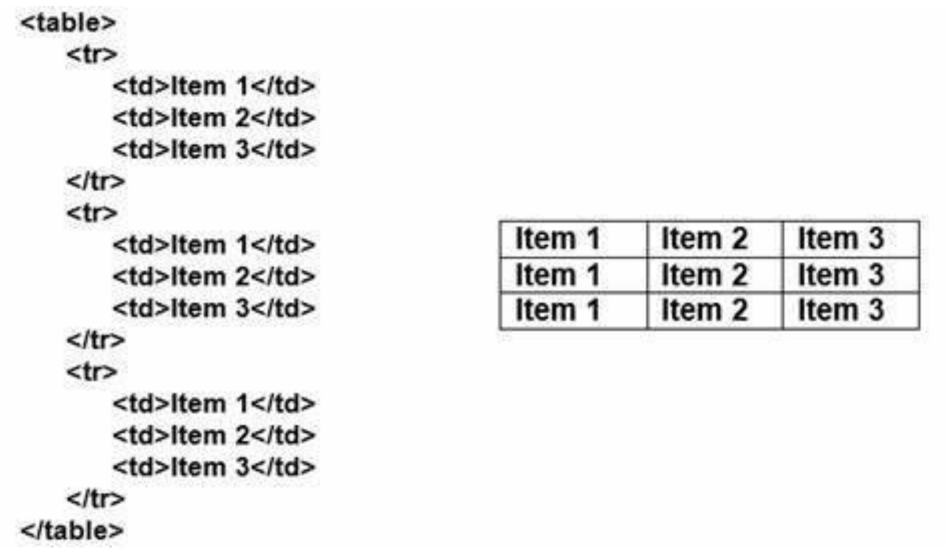
In web development, tables and frames are essential for displaying structured data and organizing content in a more interactive and controlled manner. This unit explores how to create and manipulate tables, work with frames using the <iframe> tag, and combine forms and tables for structured data input.

1. Tables in HTML

Tables are used to organize data into rows and columns,making it easy to present information in a structured format. HTML provides a set of tags that help define the structure and layout of tables.

Basic Table Structure

To create a table in HTML, we use the <table> tag. This tag defines the table structure, while other elements such as<tr>, <td>, and <th> are used to define rows, cells, and headers, respectively.
Here is the basic structure of a table:



```
<table>
    <tr>
        <td>Item 1</td>
        <td>Item 2</td>
        <td>Item 3</td>
    </tr>
    <tr>
        <td>Item 1</td>
        <td>Item 2</td>
        <td>Item 3</td>
    </tr>
    <tr>
        <td>Item 1</td>
        <td>Item 2</td>
        <td>Item 3</td>
    </tr>
</table>
```

Item 1	Item 2	Item 3
Item 1	Item 2	Item 3
Item 1	Item 2	Item 3

Figure 1.4 Table Structure

- <table>: Defines the table container.
- <tr>: Stands for table row, used to define a row in the table.
- <td>: Stands for table data, used to define the data cell in a table row.
- <th>: Stands for table header, used to define header cells (which are bold and centered by default).

Merging Cells

Sometimes you need to merge cells to create a more organized table layout. You can merge cells both horizontally (across columns) and vertically (across rows).

- colspan: Merges cells horizontally (across columns). It defines how many columns the cell will span.
- rowspan: Merges cells vertically (across rows). It defines how many rows

2. Frames in HTML

Frames allow developers to embed multiple documents (webpages) within a single web page. The <iframe> tag is used to embed another webpage within the current webpage. This is useful when you want to display an external webpage inside your website, such as embedding a YouTube video, showing an interactive map, or displaying another website's content.

Basic <iframe> Structure:

```
<iframe src="demo_iframe.html" height="200" width="300" title="Iframe Example"
allowfullscreen></iframe>
```

- src: Specifies the URL of the page to embed.
- width and height: Define the dimensions of the iframe.
- frameborder: Determines whether the iframe shouldhave a border (set to "0" for no border).

Common Attributes for <iframe>

- width and height: Define the size of the embedded frame.
- frameborder: Controls whether a border is shown around the iframe (0 for no border, 1 for a border).
- scrolling: Controls whether scrolling is allowed within the iframe. Options are yes, no, and auto.
- allowfullscreen: If included, the iframe will support full-screen mode (commonly used for video embeds).

Advantages of using <iframe>:

- Embedding external content: Iframes allow you to include content from other websites, such as social media posts, videos, or maps, directly into your webpage.
- Isolation: The content in the iframe is isolated from the rest of the page, meaning it doesn't interfere with the main page's layout or functionality.
- Easy to use: Iframes are easy to implement and provide a simple way to include other content.

Disadvantages of using <iframe>:

- Performance issues: Loading external content inside iframes can slow down the page load time, especially if the embedded content is large or contains ads.
- SEO challenges: Search engines may not index the content inside an iframe properly, potentially limiting SEO benefits.
- Security risks: If you are embedding external content from unreliable sources, it can pose security risks, such as cross-site scripting (XSS) attacks.

1.2.2 CSS:

CSS (Cascading Style Sheets) is a vital technology in web development that enables the separation of content (HTML) and presentation (styling). It provides a flexible way to control the layout, color, fonts, and other visual elements of a website, making it an essential tool for creating visually appealing and user-friendly websites. In this unit, we will explore the basics of CSS, how to create layouts with CSS, the importance of responsive design, and how to integrate CSS with HTML to create modern, mobilefriendly web pages.

CSS controls the visual presentation of HTML elements. It allows web designers to style the HTML structure of a website and apply specific styles to make the website look appealing and functional.

a) **Types of CSS**

CSS can be implemented in three ways:

1. Inline CSS: This type of CSS is used within an HTML element using the style attribute. It is applied directly to a specific element, which overrides any external or internal CSS.

2. Internal CSS: This type of CSS is written within the <style> tag in the <head> section of the HTML document. It is used to style elements within that specific HTML page only.

3. External CSS: External CSS is written in a separate .css file, which is linked to the HTML file using the <link> tag. This method is the most efficient for styling large websites because it allows for reusable styles across multiple pages.

b) **Basic CSS Syntax**

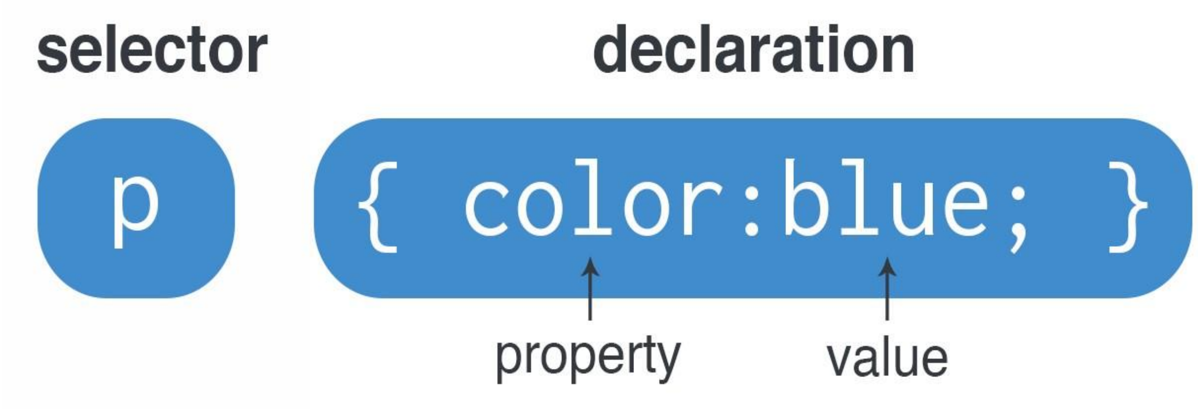


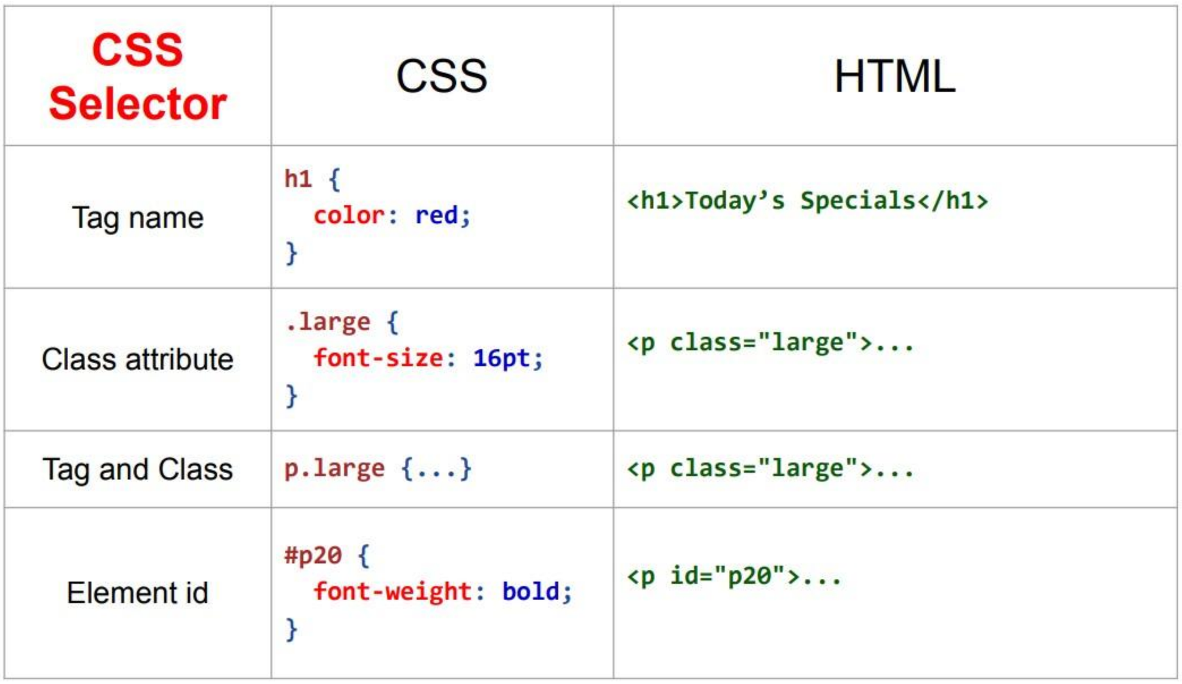
Figure 1.5 CSS Syntax

CSS is composed of selectors and declarations.

• Selector: The HTML element that you want to style (e.g.,p, div, h1).

• Declaration: The style to be applied to the selected element, enclosed in curly braces {}. A declaration contains a property and a value separated by a colon :.

c) **CSS Selectors**



CSS Selector	CSS	HTML
Tag name	`h1 { color: red; }`	`<h1>Today's Specials</h1>`
Class attribute	`.large { font-size: 16pt; }`	`<p class="large">...`
Tag and Class	`p.large {...}`	`<p class="large">...`
Element id	`#p20 { font-weight: bold; }`	`<p id="p20">...`

Figure 1.6 CSS Selectors

<u>CSS Pseudo Selectors</u>

hover - Apply rule when mouse is over element (e.g. tooltip)

p:hover, a:hover { background-color: yellow;

} a:link, a:visited - Apply rule when link has been visited or not visited

(link) a:visited { color: green;

} a:link {

color: blue;

}

Box Model:

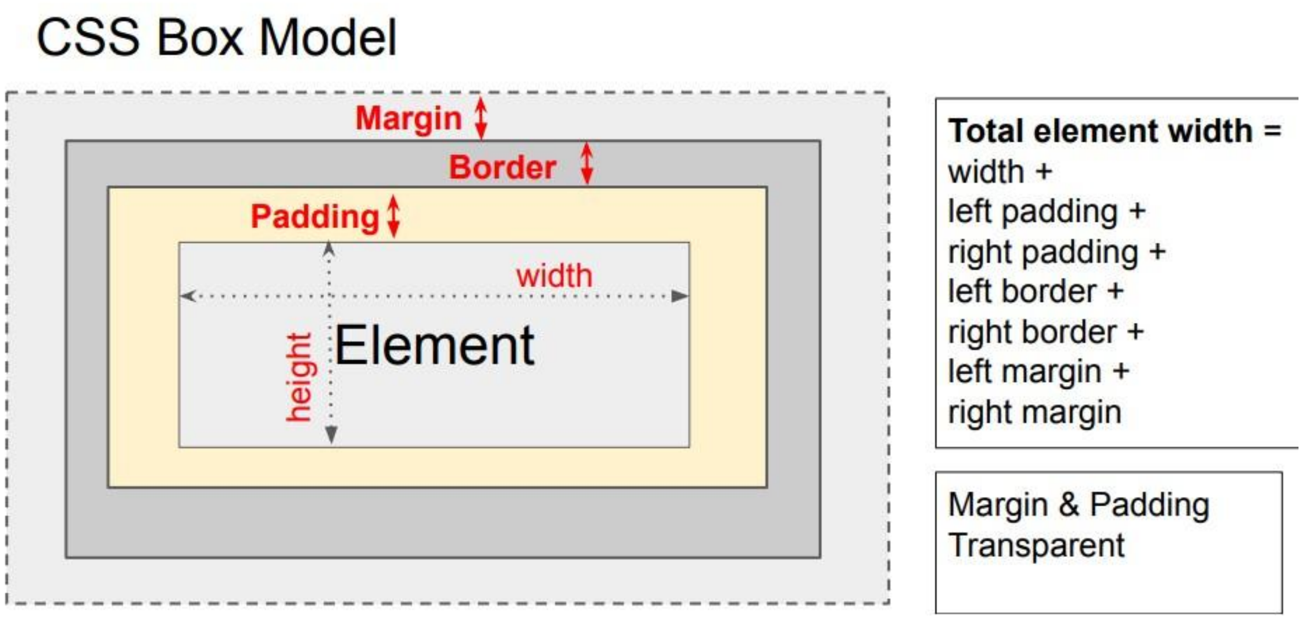


Figure 1.7 Box Model

position property

- position: static; (default) - Position in document flow
- position: relative; Position relative to default position via top, right, bottom, and left properties
- position: fixed; Position to a fixed location on the screen via top, right, bottom, and left properties
- position: absolute; Position relative to ancestor absolute element via top, right, bottom, and left properties

<u>Flexbox and Grid layout</u>

- display: flex; (Flexbox)
- display: grid; (Grid) newer layout method
- Items flex to fill additional space and shrink to fit into smaller spaces.
- Useful for web app layout:
- Divide up the available space equally among a bunch of elements
- Align of different sizes easily
- Key to handling different window and display sizes

- Flexbox - Layout one dimension (row or column) of elements • Grid - Layout in two dimensions (rows and columns) of elem

1.2.3 JavaScript:

JavaScript (js) is a light-weight object-oriented programming language which is used by several websites for scripting the webpages. It is an interpreted, full-fledged programming language that enables dynamic interactivity on websites when applied to an HTML document. It was introduced in the year 1995 for adding programs to the webpages in the Netscape Navigator browser. Since then, it has been adopted by all other graphical web browsers. With JavaScript, users can build modern web applications to interact directly without reloading the page every time. The traditional website uses js to provide several forms of interactivity and simplicity.

Although, JavaScript has no connectivity with Java programming language. The name was suggested and provided in the times when Java was gaining popularity in the market. In addition to web browsers, databases such as CouchDB and MongoDB uses JavaScript as their scripting and query language.

Features of JavaScript

There are following features of JavaScript:

1. All popular web browsers support JavaScript as they provide built-in execution environments.
2. JavaScript follows the syntax and structure of the C programming language. Thus, it is a structured programming language.
3. JavaScript is a weakly typed language, where certain types are implicitly cast (depending on the operation).
4. JavaScript is an object-oriented programming language that uses prototypes rather than using classes for inheritance.
5. It is a light-weighted and interpreted language.
6. It is a case-sensitive language.
7. JavaScript is supportable in several operating systems including, Windows, macOS, etc.
8. It provides good control to the users over the web browsers.

Application of JavaScript

- JavaScript is used to create interactive websites. It is mainly used for:
- Client-side validation,
- Dynamic drop-down menus,
- Displaying date and time,
- Displaying pop-up windows and dialog boxes (like an alert dialog box, confirm dialog box and prompt dialog box),
- Displaying clocks etc.

Variables

Variables in JavaScript are containers for storing data. JavaScript allows the usage of variables in the following three ways:

Variable	**Description**	**Example**
var	Used to initialize to value, redeclared and its value can be reassigned.	var x= value;
let	Similar to var but is block scoped	let y= value;
const	Used to declare a fixed value that cannot be changed.	const z= value;

There are following primitive and non-primitive datatypes in JavaScript:

Datatype	**Description**	**Example**
Number	Numeric values can be real number or integers.	var x= number;
String	Series of multiple characters written in quotes.	var x= “characters”;

<u>Boolean</u>	Has only two values true or false.	var x= true/false;
<u>Null</u>	Special value that represents that the variable is empty.	var x= null;
Datatype	**Description**	**Example**
<u>Undefined</u>	Represents a variable which is declared but not assigned any value.	let x; / let x= undefined;
<u>Object</u>	Complex data type that allows us to store a collection of data.	Var x= { key: “value”; key: “value”; }
<u>Array</u>	Stores multiple values of same type in a single variable.	var x =[‘y1’, ‘y2′,’y3′,’y4’]; y: any datatype
<u>Function</u>	Functions are objects that can be called to execute a block of code.	function x(arguments){ block of code }

e.g: // String let str =

"hello geeks";

console.log(str);

```
// Number const
num = 10;
console.log(num);

// Boolean const
x = "true";
console.log(x);

// Undefined let
name;
console.log(na
me );


// Null const number
= null;
console.log(number);

// Symbol const value1 =
Symbol("hello"); const value2 =
Symbol("hello");
console.log(value1);
console.log(value2);

// Here both values are different
// as they are symbol type which
// is immutable object
const object = {
firstName: "geek",
lastName: null,     batch:
2,
}; console.log(object);
```

Functions

A JavaScript function is a block of code designed to perform a particular task. It is executed when invoked or called. Instead of writing the same piece of code again and again you can put it in a function and invoke the function when required. JavaScript function can be created using the functions keyword. Some of the functions in JavaScript are:

Function	**Description**
parseInt()	Parses an argument passed to it and returns an integral number.
parseFloat()	Parses the argument and returns a floating-point number.
Function	**Description**
isNaN()	Determines if a given value is Not a Number.
Number()	Returns an argument after converting it to number.
eval()	Used for evaluating JavaScript programs presented as strings.
prompt()	Creates a dialogue box for taking input from the user.
encodeURI()	Encodes a URI into a UTF-8 encoding scheme.
match()	Used to search a string for a match against regular expression.

Arrays

In JavaScript, array is a single variable that is used to store different elements. It is often used when we want to store list of elements and access them by a single variable. Arrays use numbers as index to access its "elements".

Declaration of an Array: There are basically two ways to declare an array.

Example:

var House =[];

var House = new Array(); // Method 2

There are various operations that can be performed on arrays using JavaScript methods. Some of these methods are:

Method	**Description**
push()	Adds a new element at the very end of an array.
pop()	Removes the last element of an array.
Method	**Description**
concat()	Joins various arrays into a single array.
shift()	Removes the first element of an array
unShift()	Adds new elements at the beginning of the array
reverse()	Reverses the order of the elements in an array.

slice()	Pulls a copy of a part of an array into a new array.
splice()	Adds elements in a particular way and position.
toString()	Converts the array elements into strings.
valueOf()	Returns the primitive value of the given object.
indexOf()	Returns the first index at which a given element is found.
lastIndexOf()	Returns the final index at which a given element appears.
join()	Combines elements of an array into one single string and then returns it
sort()	Sorts the array elements based on some condition.

e.g :

```
// Number Array let arr = [10,
20, 30, 40, 50]; let arr1 = [110,
120, 130, 140];

// String array
let string_arr = ["Alex", "peter", "chloe"];

// push: Adding elements at the end of the array
arr.push(60); console.log("After push op " +
arr);
```

```
// unshift() Adding elements at the start of the array
arr.unshift(0, 10); console.log("After unshift op " +
arr );

// pop: removing elements from the end of the array
arr.pop(); console.log("After pop op" + arr);

// shift(): Removing elements from the start of the array
arr.shift(); console.log("After shift op " + arr);

// splice(x,y): removes x number of elements
// starting from index y arr.splice(2,
1); console.log("After splice op" +
arr);

// reverse(): reverses the order of elements in array arr.reverse();
console.log("After reverse op" + arr);

// concat(): merges two or more array console.log("After
concat op" + arr.concat(arr1));
```
Strings

Strings in JavaScript are primitive and immutable data types used for storing and manipulating text data which can be zero or more characters consisting of letters, numbers or symbols. JavaScript provides a lot of methods to manipulate strings. Some most used ones are:

Methods	**Description**
concat()	Used for concatenating multiple strings into a single string.
match()	Used for finding matche of a string against a provided pattern.
replace()	Used for finding and replacing a given text in string.

substr()	Used to extract length characters from a given string.
slice()	Used for extracting an area of the string and returs it
lastIndexOf()	Used to return the index (position) of the last occurrence of a specified value.
charAt()	Used for returning the character at a particular index of a string
valueOf()	Used for returning the primitive value of a string object.
split()	Used for splitting a string object into an array of strings.
toUpperCase()	Used for converting strings to upper case.
toLoweCase()	Used for converting strings to lower case.

e.g:

```
let gfg = 'GFG '; let geeks = 'stands-
for-GeeksforGeeks';

// Print the string as it is
console.log(gfg); console.log(geeks);

// concat() method console.log(gfg.concat(geeks));

// match() method console.log(geeks.match(/eek/));

// charAt() method console.log(geeks.charAt(5));
// valueOf() method console.log(geeks.valueOf());


// lastIndexOf() method console.log(geeks.lastIndexOf('for'));


// substr() method console.log(geeks.substr(6));


// indexOf() method console.log(gfg.indexOf('G'));


// replace() method console.log(gfg.replace('FG',
'fg'));


// slice() method console.log(geeks.slice(2,
8));


// split() method console.log(geeks.split('-'));
// toUpperCase method
console.log(geeks.toUpperCase(geeks));


// toLowerCase method console.log(geeks.toLowerCase(geeks));
```

Loops

Loops are a useful feature in most programming languages. With loops you can evaluate a set of instructions/functions repeatedly until certain condition is reached. They let you run code blocks as many times as you like with different values while some condition is true. Loops can be created in the following ways in JavaScript:

Loop	Description	Syntax
for	Loops over a block of with conditions specified in the beginning.	for (initialization tion; testing condition;increment/decrement) { statement(s) }
while	Entry control loop which executes after checking the condition.	while (boolean condition) { loop statements… }
do-while	Exit Control Loop which executes once before checking the condition.	do { statements.. } while (condition);
for-in	Another version of for loop to provide a simpler way to iterate.	for (variableName in Object) {
Loop	**Description**	**Syntax**
		statement(s) }

```
// Illustration of for loop let
x;

// for loop begins when x=2 //
and runs till x <=4 for (x = 2; x
<= 4; x++) {
console.log("Value of x:" + x);
}

// Illustration of for..in loop
// creating an Object let
languages = {     first: "C",
second: "Java",     third:
"Python",     fourth:
"PHP",     fifth:
"JavaScript",
};

// Iterate through every property of
// the object languages and print all
// of them using for..in loops for
(itr in languages) {
console.log(languages[itr]);
}

// Illustration of while loop
let y = 1;

// Exit when x becomes greater than 4
while (y <= 4) {
console.log("Value of y:" + y);

   // Increment the value of y for
```

```
    // next iteration
x++;
}

// Illustration of do-while loop let
z = 21;

do {

    // The line while be printer even if
    // the condition is false
console.log("Value of z:" + z);

    z++;
} while (z < 20);
```

DOM

DOM stands for **Document Object Model.** It defines the logical structure of documents and the way a document is accessed and manipulated. JavaScript can not understand the tags in HTML document but can understand objects in DOM.Below are some of the methods provided by JavaScript to manipulate these nodes and their attributes in the DOM:

Method	**Description**
appendChild()	Adds a new child node as the last child node.
Method	**Description**
cloneNode()	Duplicates an HTML element.
hasAttributes()	Returns true If an element has any attributes otherwise,returns false.

removeChild()	Removes a child node from an element using the Child() method.
getAttribute()	Returns the value of an element node’s provided attribute.
getElementsByTagName()	Returns a list of all child elements.
isEqualNode()	Determines whether two elements are same.

1.3 CORE JAVA & ADVANCE JAVA

1.3.1 CORE JAVA

Java is a programming language and platform that has been widely used since its development by James Gosling in 1991. It follows the Object-oriented Programming concept and can run programs written on any OS platform. Java is a high-level, object-oriented, secure, robust, platform-independent, multithreaded, and portable programming language all those words are collectively called Java Buzzwords.

Java Basics:

Now, we will explore some of the fundamental concepts often utilized in the Java programming language.

Object – An object refers to an entity that possesses both behavior and state, such as a bike, chair, pen, marker, table, and car. These objects can be either tangible or intangible, including the financial system as an example of an intangible object.

There are three characteristics of an object:

- **State:** The data (value) of an object is represented by its state.
- **Behaviour:** The functionality of an object, such as deposit, withdrawal, and so on, is represented by the term behaviour.
- **Identity:** A unique ID is often used to represent an object's identification. The value of the ID is hidden from the outside user. The JVM uses it internally to uniquely identify each object.

Class – A class is a collection of objects with similar attributes. It's a blueprint or template from which objects are made. It's a logical thing. It can't be physical. In Java, a class definition can have the following elements:

- **Modifiers:** A class can be private or public, or it can also have a default access level
- **class keyword:** To construct a class, we use the class keyword.
- **class name:** The name of the class should usually start with a capital letter.
- **Superclass (optional):** If the class has any superclass, we use the extends keyword and we mention the name of the superclass after the class name.

- **Interface (optional):** If the class implements an interface, we use the implements keyword followed by the name of the interface after the class name.

Constructors: In Java, a constructor is a block of code similar to a method. Whenever a new class instance is created, the constructor is called. The memory allocation for the object only happens when the constructor is invoked.
There are two types of constructors in Java. They are as follows:-
Default Constructor – A default constructor is a type of constructor that does not require any parameters. When we do not declare a constructor for a class, the compiler automatically generates a default constructor for the class with no arguments.
Parameterised Constructor – A parameterized constructor is a type of constructor that requires parameters. It is used to assign custom values to a class's fields during initialization.

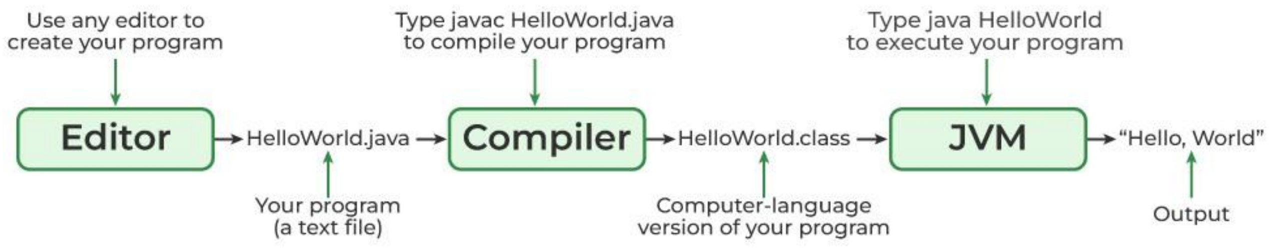

Use any editor to create your program
Type javac HelloWorld.java to compile your program
Type java HelloWorld to execute your program
Editor
HelloWorld.java
Compiler
HelloWorld.class
JVM
"Hello, World"
Your program (a text file)
Computer-language version of your program
Output

Figure 1.8 Compiling a program

Data Types in Java
Data Types in Java are the different values and sizes that can be stored in the variable according to the requirements.
Java Datatype are further two types:-
1. Primitive Data Type in Java
In Java, primitive data types serve as the foundation for manipulating data. They are the most basic types of data that the Java programming language uses. Java has several primitive data types, including:

Type	Size	Example Literals	Range of values
boolean	1 bit	true, false	true, false

byte	8 bits	(none)	-128 to 127
Type	**Size**	**Example Literals**	**Range of values**
char	16 bits	?’, ‘\u0041’, ‘\101’, ‘\\’, ‘\’, ‘\n’, ‘β’	characters representation of ASCII values 0 to 255
short	16 bits	(none)	-32,768 to 32,767
int	32 bits	-2,-1,0,1,2	-2,147,483,648 to 2,147,483,647
long	64 bits	-2L,-1L,0L,1L,2L	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
float	32 bits	1.23e100f , -1.23e-100f , .3f ,3.14F	upto 7 decimal digits
double	64 bits	1.23456e300d , -123456e-300d , 1e1d	upto 16 decimal digits

2. Non-primitive Data Type

Non-primitive datatypes are created from primitive datatypes. Examples of non-primitive datatypes include arrays, stacks, and queues.

Control Flow in Java

- **If, else-if, else**

```
     import java.io.*;
     class Flow{     public static void
main(String[] args)
   {        int a = 1, b
= 2;         if (a < b)
System.out.println(
b);        else if (a >
b)
System.out.println(
a);        else
        System.out.println(a + "==" + b);
   }
}
```

<u>Output</u>

```
2
```

- **Switch Statement**

```
public class Test{     public static void
main(String[] args)
   {        int day = 5;
String dayString;
switch (day) {
case 1:
        dayString = "Monday";
break;        case 2:
        dayString = "Tuesday";
break;        case 3:
        dayString = "Wednesday";
break;        case 4:
```

```
            dayString = "Thursday";
            break;         case 5:
dayString = "Friday";
break;         case 6:
dayString = "Saturday";
break;         case 7:
            dayString = "Sunday";
break;         // Default case
default:
            dayString = "Invalid day";
        }
        System.out.println(dayString);
    }
}
```

Output

```
Friday
```

Loops in Java

Loops are used for performing the same task multiple times. There are certain loops available in Java as mentioned below:

1. For Loop
2. While Loop
3. do-while

e.g:

```
import java.io.*; class
Driver{
    public static void main(String[] args)
    {
        int i = 16;
while (i != 0) {
```

```
            System.out.println(i);
if (i > 4)              i -= 4;
else              i -= 1;
        }
    }
}
```

Output:

```
16
12
8
4
3
2
1
```

Methods in Java

Java Methods are collections of statements that perform some specific task and return the result.

Syntax of Method

```
<access_modifier><return_type><method_name>(list_of_parameters)
{
   //body
}
```

Eg:

```
import java.io.*; class
Main{
   public static int sum(int i, int j) { return i + j; }
public static void main(String[] args)
   {        int n = 3, m
= 3;        for (int i =
0; i < n; i++) {
for (int j = 0; j < m;
j++) {
System.out.print(su
m(i, j) + " ");
```

```
            }
            System.out.println();
        }
    }
}
```

Output

```
Salary : 70000
Benefits : 15000
```

Scanner Class Syntax:

```
Scanner scn = new Scanner(System.in);
```

```
import java.io.*; class
ABC {
    // Sum Method with int returning value
public int sum(int x, int y) { return x + y; }

    // Sum Method with float returning value     public
double sum(double x, double y) { return x + y; }
} class Driver{     // main function
public static void main(String[] args)
    {
        ABC temp = new ABC();         System.out.println(temp.sum(1, 2));
        System.out.println(temp.sum(3.14, 4.23));
    }
}
```

Output**:**

```
ABC
65
Name:ABC
Marks :65
```

Java Polymorphism

Polymorphism: It is the ability to differentiate between entities with the same name efficiently.

Compile-time polymorphism**:** Static polymorphism, also called compile-time polymorphism, is achieved through function overloading in Java. Static polymorphism, also called compile-time polymorphism, is achieved through function overloading in Java.

Method Overloading**:** If there are multiple functions that have the same name but different parameters, it is known as overloading. Alterations in the number or type of arguments can result in the overloading of functions. **Example:** class ABC {

```
   // Sum Method with int returning value
public int sum(int x, int y) { return x + y; }

   // Sum Method with float returning value     public
double sum(double x, double y) { return x + y; }
}
Class Driver {
   public static void main(String[] args)
   {
      ABC temp = new ABC();
      System.out.println(temp.sum(1, 2));
      System.out.println(temp.sum(3.14, 4.23));
   }
}
```

Output

```
3
7.370000000000001
```

Java Inheritance

Inheritance: It is the mechanism in Java by which one class is allowed to inherit the features (fields and methods) of another class.

```
import java.io.*;
class Employee {
int salary = 70000;
} class Engineer extends Employee
{     int benefits = 15000;
} class Test{     public static void
main(String args[])
   {
       Engineer E1 = new Engineer();
       System.out.println("Salary : " + E1.salary
                       + "\nBenefits : " + E1.benefits);
}
}
```

Output

```
Salary : 70000
Benefits : 15000
```

Arrays in Java

Arrays are the type of data structure that can store data of similar data types. Arrays are allocated in contiguous memory allocation with a fixed size.

Syntax:

```
intarr[]= new int[20]; int[]
arr=new int[20];
```

Strings in Java

Strings are the type of objects that can store the character of values. A string acts the same as an array of characters in Java Syntax:

```
String abc=" ";
```

Java Exception Handling

Exception:– An Exception is an unexpected error that occurs during the run-time of a program.

Error vs Exception: What is the difference?

When a program encounters an error, it means there is a significant issue that a well-designed program should not try to fix. On the other hand, an exception indicates a particular situation that a well-designed program should try to handle.

Types of Exceptions:-

Checked Exception:- This mainly includes IO Exceptions and Compile time Exceptions

Unchecked Exceptions:– This covers both Runtime Exceptions and Null Pointer Exceptions.

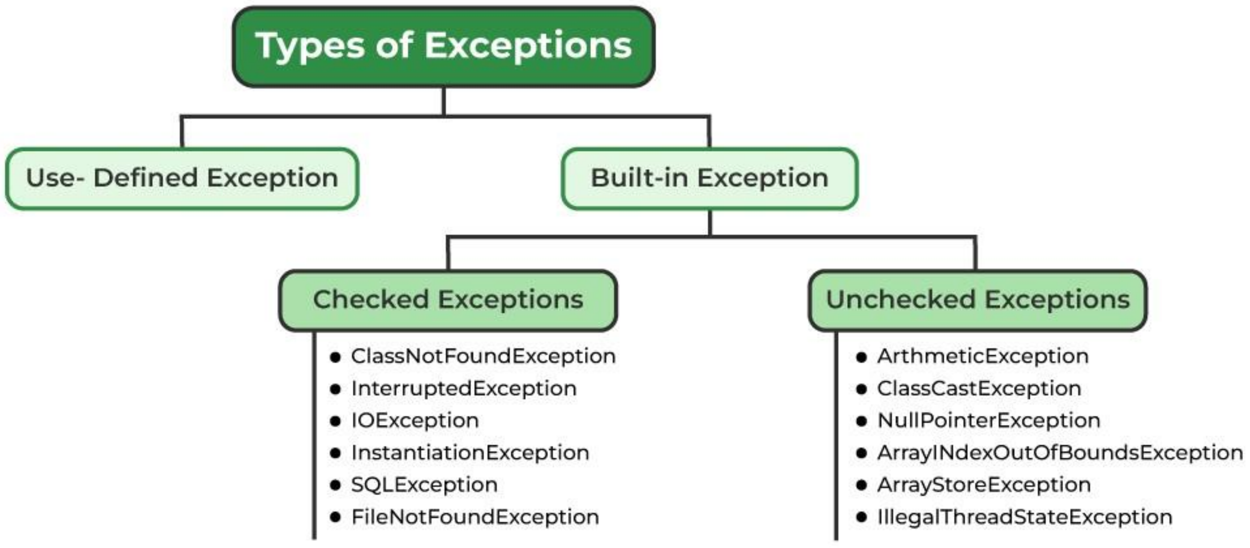


Figure 1.9 Types of Exception

Handle Exceptions try block: The try block comprises a set of statements that may throw an exception.

Catch Block: When there is an uncertain condition in the try block, the catch block comes in handy to handle it. It is mandatory to have a catch block right after the try block to handle any exceptions that may be thrown by the try block.

Finally Block: When programming in Java, the finally block is a section of code that will always run, regardless of whether or not an exception has been caught. If there is a catch block following a try block, it will be executed before the finally block. However, if there is no catch block, the finally block will be executed immediately after the try block.

final vs finally vs finalize final

- A final keyword can be used with classes, methods, and variables.
- A final class cannot be inherited.
- A final method cannot be overridden.
- A final variable cannot be reassigned.

finally

- A finally block is always executed, regardless of whether an exception is thrown or not.
- The finally block is executed after the try block and catch block, but before the program control returns to the calling method.
- The finally block can be used to close resources, such as files and database connections. finalize
- The finalize() method is called by the garbage collector when an object is no longer needed.
- The finalize() method can be used to perform cleanup operations, such as releasing resources or writing data to a file.
- The finalize() method is not guaranteed to be called, so it should not be used to perform critical operations.

throw keyword:- When using Java, the throw keyword is utilized to throw an exception from a block of code or a method. It is possible to throw either a checked or unchecked exception. The throw keyword is frequently used for throwing custom exceptions.

throws keyword:- If a try/catch block is not present, the throws keyword can be used to manage exceptions. This keyword specifies the specific exceptions that a method should throw if an exception happens.

What are Interfaces in Java?

The interface in Java is *a* mechanism to achieve abstraction. Traditionally, an interface could only have abstract methods (methods without a body) and public, static, and final variables by default. It is used to achieve abstraction and multiple inheritances in Java. In other words, interfaces primarily define methods that other classes must implement. Java Interface also represents the IS-A relationship. In Java, the abstract keyword applies only to classes and methods, indicating that they cannot be instantiated directly and must be implemented.

Syntax for Java Interfaces interface{

```
   //declare constant fields
   //declare methods that abstract
//by default.
}
```

To declare an interface, use the interface keyword. It is used to provide total abstraction. That means all the methods in an interface are declared with an empty body and are public and all fields are public, static, and final by default. A class that implements an interface must implement all the methods declared in the interface. To implement the interface, use the implements keyword.

Relationship Between Class and Interface

A class can extend another class, and similarly, an interface can extend another interface. However, only a class can implement an interface, and the reverse (an interface implementing a class) is not allowed.

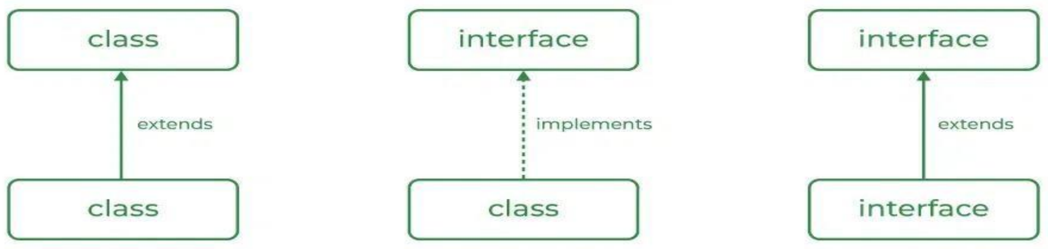


Figure 1.10 Relaionship b/w class and interface

```
import java.io.*;

interface Vehicle {

   // all are the abstract methods.
   void changeGear(int a);
void speedUp(int a);     void
applyBrakes(int a);
}
```

```
class Bike implements Vehicle {
int speed;     int gear;
@Override
   public void changeGear(int newGear){
gear = newGear;
   }
   @Override     public void
speedUp(int increment){         speed =
speed + increment;
   }
   @Override     public void
applyBrakes(int decrement){         speed =
speed - decrement;
   }
       public void printStates() {
System.out.println("speed: " + speed
           + " gear: " + gear);
   }

} class
Driver{

   public static void main (String[] args) {
// creating instance of the bike.         Bike
bike = new Bike();
bike.changeGear(1);
bike.speedUp(4);
bike.applyBrakes(3);

       System.out.println("Bike present state :");
bike.printStates();
   }
}
```

Output:

```
Bike present state :speed: 1 gear:
```

1.3.2: Advance Java :

Collections in Java

Any group of individual objects that are represented as a single unit is known as a Java Collection of Objects. In Java, a separate framework named the *"Collection Framework"* has been defined in JDK 1.2 which holds all the Java Collection Classes and Interface in it.

In Java, the Collection interface (java.util.Collection) and Map interface (java.util.Map) are the two main "root" interfaces of Java collection classes.

What is a Framework in Java?

A framework is a set of classes and interfaces which provide a ready-made architecture. In order to implement a new feature or a class, there is no need to define a framework. However, an optimal objectoriented design always includes a framework with a collection of classes such that all the classes perform the same kind of task.

Hierarchy of the Collection Framework in Java

The utility package, (java.util) contains all the classes and interfaces that are required by the collection framework. The collection framework contains an interface named an iterable interface which provides the iterator to iterate through all the collections. This interface is extended by the main collection interface which acts as a root for the collection framework.

Before understanding the different components in the below framework, let's first understand a class and an interface.

- **Class:** A class is a user-defined blueprint or prototype from which objects are created. It represents the set of properties or methods that are common to all objects of one type.

- **Interface:** Like a class, an interface can have methods and variables, but the methods declared in an interface are by default abstract (only method signature, nobody). Interfaces specify what a class must do and not how. It is the blueprint of the class.

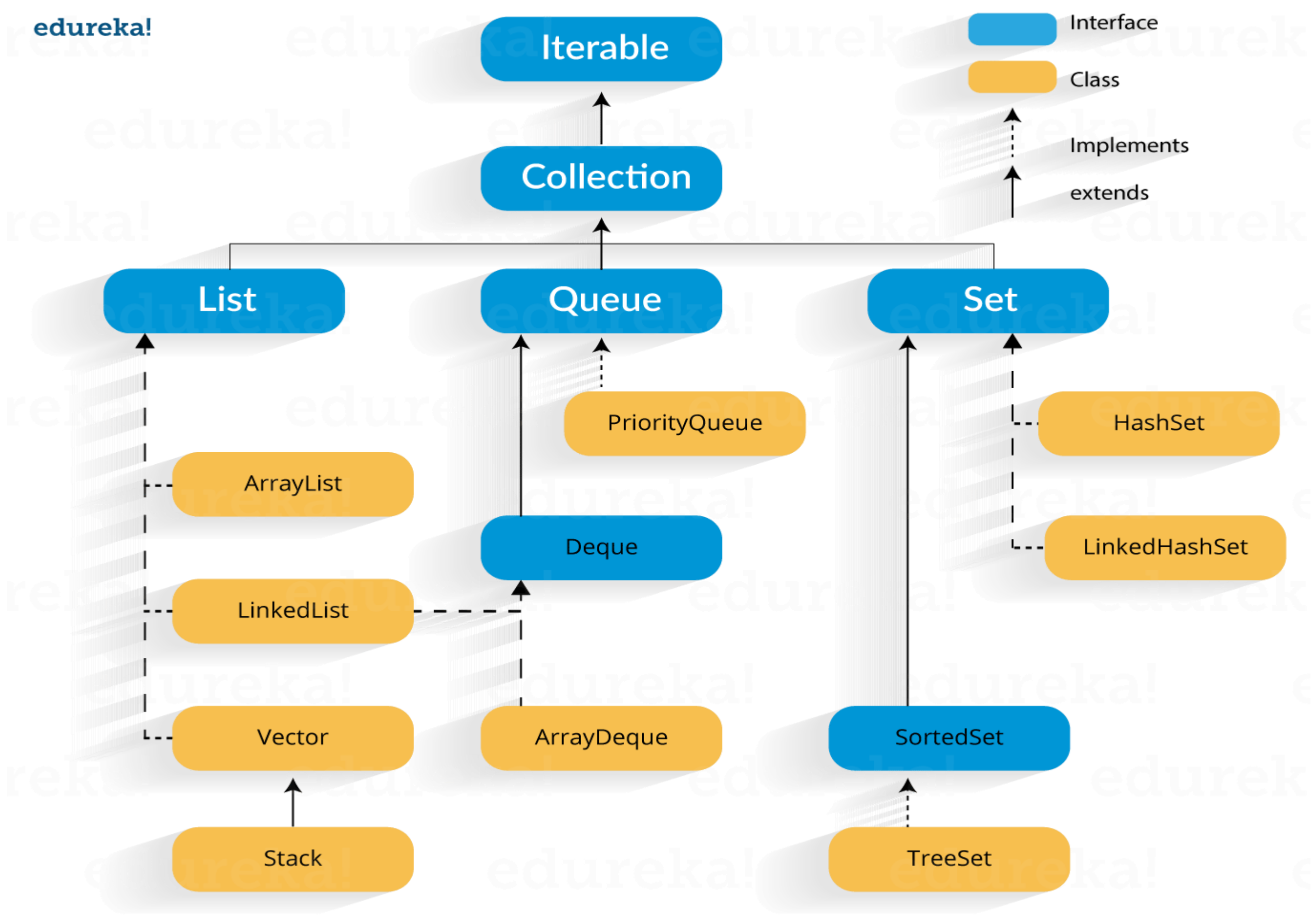


Figure 1.11 Hierarchy of Collection framework

Methods of the Collection Interface

This interface contains various methods which can be directly used by all the collections which implement this interface. They are:

Method	**Description**
add(Object)	This method is used to add an object to the collection.
addAll(Collection c)	This method adds all the elements in the given collection to this collection.

clear()	This method removes all of the elements from this collection.

Method	**Description**
contains(Object o)	This method returns true if the collection contains the specified element.
containsAll(Collection c)	This method returns true if the collection contains all of the elements in the given collection.
equals(Object o)	This method compares the specified object with this collection for equality.
hashCode()	This method is used to return the hash code value for this collection.
isEmpty()	This method returns true if this collection contains no elements.
iterator()	This method returns an iterator over the elements in this collection.
max()	This method is used to return the maximum value present in the collection.

remove(Object o)	This method is used to remove the given object from the collection. If there are duplicate values, then this method removes the first occurrence of the object.
removeAll(Collection c)	This method is used to remove all the objects mentioned in the given collection from the collection.
Method	**Description**
retainAll(Collection c)	This method is used to retain only the elements in this collection that are contained in the specified collection.
size()	This method is used to return the number of elements in the collection.
spliterator()	This method is used to create a Spliterator over the elements in this collection.
toArray()	This method is used to return an array containing all of the elements in this collection.

What is Java Servlet?

Java Servlets are the Java programs that run on the Java-enabled web server or application server. They are used to handle the request obtained from the web server, process the request, produce the response, and then send a response back to the web server.

Properties of Java Servlet

The properties of Servlets are as follows:

- Servlets work on the server side.
- Servlets are capable of handling complex requests obtained from the web server.

Execution of Java Servlets

Execution of Servlets basically involves Six basic steps:

1. The Clients send the request to the Web Server.
2. The Web Server receives the request.
3. The Web Server passes the request to the corresponding servlet.
4. The Servlet processes the request and generates the response in the form of output.
5. The Servlet sends the response back to the webserver.
6. The Web Server sends the response back to the client and the client browser displays it on the screen.

Life Cycle of Servlet :

The entire life cycle of a Servlet is managed by the Servlet container which uses the **javax.servlet.Servlet** interface to understand the Servlet object and manage it. So, before creating a Servlet object, let's first understand the life cycle of the Servlet object which is actually understanding how the Servlet container manages the Servlet object.

Stages of the Servlet Life Cycle:

- Loading a Servlet.
- Initializing the Servlet.
- Request handling.
- Destroying the Servlet.

Let's look at each of these stages in details:

1. **Loading a Servlet**: The first stage of the Servlet lifecycle involves loading and initializing the Servlet by the Servlet container. The Web container or Servlet Container can load the Servlet at either of the following two stages :

- Initializing the context, on configuring the Servlet with a zero or positive integer value.
- If the Servlet is not preceding stage, it may delay the loading process until the Web container determines that this Servlet is needed to service a request.

The Servlet container performs two operations in this stage :

- **Loading :** Loads the Servlet class.
- **Instantiation :** Creates an instance of the Servlet. To create a new instance of the Servlet, the container uses the no-argument constructor.

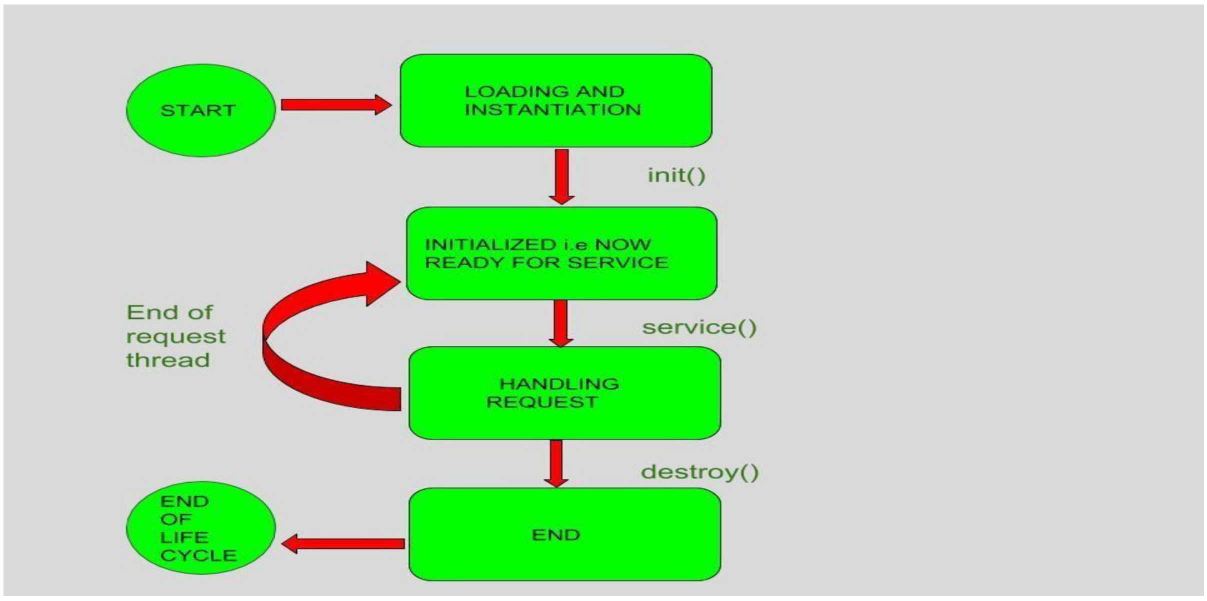


Figure 1.12 Life cycle of Servlet

2. **Initializing a Servlet**: After the Servlet is instantiated successfully, the Servlet container initializes the instantiated Servlet object. The container initializes the Servlet object by invoking the **Servlet.init(ServletConfig)** method which accepts ServletConfig object reference as parameter.

The Servlet container invokes the Servlet.init(ServletConfig) method only once, immediately after the Servlet.init(ServletConfig**)** object is instantiated successfully. This method is used to initialize the resources, such as JDBC datasource.

Now, if the Servlet fails to initialize, then it informs the Servlet container by throwing the ServletException or UnavailableException.

3. **Handling request**: After initialization, the Servlet instance is ready to serve the client requests. The Servlet container performs the following operations when the Servlet instance is located to service a request :

- It creates the ServletRequest and ServletResponse objects. In this case, if this is a HTTP request, then the Web container creates HttpServletRequest and HttpServletResponse objects which are subtypes of the ServletRequest and ServletResponse objects respectively.
- After creating the request and response objects it invokes the Servlet.service(ServletRequest, ServletResponse) method by passing the request and response objects.
- The **service()** method while processing the request may throw the **ServletException** or **UnavailableException** or **IOException**.

4. **Destroying a Servlet**: When a Servlet container decides to destroy the Servlet, it performs the following operations,
    - It allows all the threads currently running in the service method of the Servlet instance to complete their jobs and get released.
    - After currently running threads have completed their jobs, the Servlet container calls the **destroy()** method on the Servlet instance.

After the **destroy()** method is executed, the Servlet container releases all the references of this Servlet instance so that it becomes eligible for garbage collection.

JSP:

JSP stands for Jakarta Server Pages **(**formerly JavaServer Pages**)**. It is a programming tool that is used on the Application Server Side to create dynamic web pages. JSP basically used to support Platform–Independent and Dynamic methods to build web-dependent applications based on HTML, XML, SOAP, etc. JSP allows developers to embed Java code as well as pre-defined tags. JSP pages are like ASP (Active Server Pages) in that they are compiled on the server, instead of the user's web browser.
Note: *In JSP, Java code can be embedded with special tags enclosed in "<% %>" or "<%= %>"* JSP was developed by Sun Microsystems Company in 1999. For the development of the JSP, languages are used all the functions built into it have been created in the Java programming language.

Life cycle of JSP

A Java Server Page life cycle is defined as the process that started with its creation which later translated to a servlet and afterward servlet lifecycle comes into play. This is how the process goes on until its destruction.

Lifecycle of JSP

Following steps are involved in the JSP life cycle:

1. Translation of JSP page to Servlet
2. Compilation of JSP page(Compilation of JSP into test.java)
3. Classloading (test.java to test.class)
4. Instantiation(Object of the generated Servlet is created)
5. Initialization(jspInit() method is invoked by the container)
6. Request processing(_jspService()is invoked by the container)
7. JSP Cleanup (jspDestroy() method is invoked by the container)

We can override jspInit(), jspDestroy() but we can't override _jspService() method. **Translation of JSP page to Servlet:**

This is the first step of the JSP life cycle. This translation phase deals with the Syntactic correctness of JSP. Here test.jsp file is translated to test.java.

1. **Compilation of JSP page:** Here the generated java servlet file (test.java) is compiled to a class file (test.class).
2. **Classloading:** The classloader loads the Java class file into the memory. The loaded Java class can then be used to serve incoming requests for the JSP page.
3. **Instantiation:** Here an instance of the class is generated. The container manages one or more instances by providing responses to requests.
4. **Initialization:** jspInit() method is called only once during the life cycle immediately after the generation of the Servlet instance from JSP.
5. **Request processing:** _jspService() method is used to serve the raised requests by JSP. It takes request and response objects as parameters. This method cannot be overridden.
6. **JSP Cleanup:** In order to remove the JSP from the use by the container or to destroy the method for servlets jspDestroy()method is used. This method is called once, if you need to perform any cleanup task like closing open files, or releasing database connections jspDestroy() can be overridden.

Difference between Servlet and JSP

Servlet	**JSP**
Servlet is a java code.	JSP is a HTML-based compilation code.
Writing code for servlet is harder than JSP as it is HTML in java.	JSP is easy to code as it is java in HTML.
Servlet plays a controller role in the ,MVC approach.	JSP is the view in the MVC approach for showing output.
Servlet	**JSP**
Servlet is faster than JSP.	JSP is slower than Servlet because the first step in the JSP lifecycle is the translation of JSP to java code and then compile.
Servlet can accept all protocol requests.	JSP only accepts HTTP requests.
In Servlet, we can override the service() method.	In JSP, we cannot override its service() method.
In Servlet by default session management is not enabled, user have to enable it explicitly.	In JSP session management is automatically enabled.

Modification in Servlet is a time-consuming compiling task because it includes reloading, recompiling, JavaBeans and restarting the server.	JSP modification is fast, just need to click the refresh button.
There is no method for running JavaScript on the client side in Servlet.	While running the JavaScript at the client side in JSP, client-side validation is used.

JDBC (Java Database Connectivity)

JDBC stands for Java Database Connectivity**.** JDBC is a Java API to connect and execute the query with the database. It is a specification from Sun Microsystems that provides a standard abstraction(API or Protocol) for Java applications to communicate with various databases. It provides the language with Java database connectivity standards. It is used to write programs required to access databases. JDBC, along with the database driver, can access databases and spreadsheets. The enterprise data stored in a relational database(RDB) can be accessed with the help of JDBC APIs.

JDBC Drivers

JDBC drivers are client-side adapters (installed on the client machine, not on the server) that convert requests from Java programs to a protocol that the DBMS can understand. **JDBC drivers** are the software components which implements interfaces in JDBC APIs to enable java application to interact with the database.

Thin driver – Type 4 driver (fully Java driver)

Type-4 driver is also called native protocol driver. This driver interact directly with database. It does not require any native database library, that is why it is also known as Thin Driver.

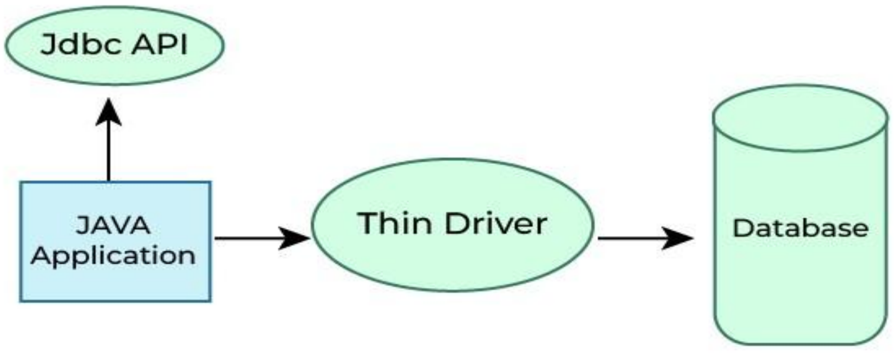


Figure 1.13 JDBC Driver

Advantages

- Does not require any native library and Middleware server, so no client-side or server-side installation.
- It is fully written in Java language, hence they are portable drivers. **Disadvantage**
- If the database varies, then the driver will carry because it is database dependent.

Steps to Connect Java Application with Database

Below are the steps that explains how to connect to Database in Java**:**

Step 1 – Import the Packages

Step 2 – Load the drivers using the *forName() method*

Step 3 – Register the drivers *using DriverManager*

Step 4 – Establish a connection *using the Connection class object*

Step 5 – Create a statement

Step 6 – Execute the query

Step 7 – Close the connections

Java Database Connectivity Step 1: Import the Packages

Step 2: Loading the drivers

In order to begin with, you first need to load the driver or register it before using it in the program. Registration is to be done once in your program. You can register a driver in one of two ways mentioned below as follows: **2-A Class.forName()**

Here we load the driver's class file into memory at the runtime. No need of using new or create objects. The following example uses Class.forName() to load the Oracle driver as shown below as follows:

Class.forName("oracle.jdbc.driver.OracleDriver");

2-B DriverManager.registerDriver()

DriverManager is a Java inbuilt class with a static member register. Here we call the constructor of the driver class at compile time. The following example uses DriverManager.registerDriver()to register the Oracle driver as shown below:

DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver())

Step 3: Establish a connection *using* the *Connection class object*

After loading the driver, establish connections as shown below as follows: Connection con = DriverManager.getConnection(url,user,password)

- **user: U**sername from which your SQL command prompt can be accessed.
- **password:** password from which the SQL command prompt can be accessed.
- **con:** It is a reference to the Connection interface.
- **Url**: Uniform Resource Locator which is created as shown below:

String url = " jdbc:oracle:thin:@localhost:1521:xe"

Where oracle is the database used, thin is the driver used, @localhost is the IP Address where a database is stored, 1521 is the port number and xe is the service provider. All 3 parameters above are of String type and are to be declared by the programmer before calling the function. Use of this can be referred to form the final code.

Step 4: Create a statement

Once a connection is established you can interact with the database. The JDBCStatement,

CallableStatement, and PreparedStatement interfaces define the methods that enable you to send SQL commands and receive data from your database.
Use of JDBC Statement is as follows:
Statement st = con.createStatement();
Note: *Here, con is a reference to Connection interface used in previous step .*

Step 5: Execute the query
Now comes the most important part i.e executing the query. The query here is an SQL Query. Now we know we can have multiple types of queries. Some of them are as follows:

- The query for updating/inserting a table in a database.
- The query for retrieving data.

The executeQuery() method of the **Statement interface** is used to execute queries of retrieving values from the database. This method returns the object of ResultSet that can be used to get all the records of a table.
The executeUpdate(sql query) method of the Statement interface is used to execute queries of updating/inserting.

Step 6: Closing the connections
So finally we have sent the data to the specified location and now we are on the verge of completing our task. By closing the connection, objects of Statement and ResultSet will be closed automatically. The close() method of the Connection interface is used to close the connection. It is shown below as follows:
con.close();
Java Program to Establish Connection in JDBC

```
import java.sql.*;
import java.util.*; class
Main {

        // Main driver method
public static void main(String a[])
        {
                // Creating the connection using Oracle DB
                // Note: url syntax is standard, so do grasp
                String url = "jdbc:oracle:thin:@localhost:1521:xe";

                // Username and password to access DB
                // Custom initialization
                String user = "system";
                String pass = "12345";

                // Entering the data
                Scanner k = new Scanner(System.in);

                System.out.println("enter name");
                String name = k.next();

                System.out.println("enter roll no");
int roll = k.nextInt();

                System.out.println("enter class");
                String cls = k.next();

                // Inserting data using SQL query
                String sql = "insert into student1 values('" + name
                                        + "'," + roll + ",'" + cls + "')";

                // Connection class object
                Connection con = null;

                // Try block to check for exceptions
                try {
```

```
                              // Registering drivers
DriverManager.registerDriver(
new oracle.jdbc.OracleDriver());

                              // Reference to connection interface
con = DriverManager.getConnection(url, user,pass);

                              // Creating a statement
                              Statement st = con.createStatement();

                              // Executing query
int m = st.executeUpdate(sql);
                              if (m == 1)
System.out.println(
                                                    "inserted successfully : " + sql);
                    else
                                        System.out.println("insertion failed");

                              // Closing the connections
con.close();
                    }

                    // Catch block to handle exceptions
          catch (Exception ex) {
                        // Display message when exceptions occurs
                                       System.err.println(ex);
                    }
          }
}
```

CHAPTER 2
TRAINING WORK UNDERTAKEN

2.1 INTRODUCTION

The advent of the internet has transformed the way consumers purchase goods and services, and the book industry is no exception. Online bookstores have emerged as a convenient alternative to traditional brick-and-mortar stores, offering a vast selection of titles, competitive pricing, and the comfort of shopping from home. This chapter provides an overview of the online bookstore project, outlining its significance in the current digital landscape, the target audience, and the core functionalities that will be developed to enhance the user experience.

The rise of the internet has fundamentally altered consumer behavior, particularly in how people shop for books. The transition from traditional brick-and-mortar bookstores to online platforms has not only expanded access to literature but has also introduced new dynamics in the publishing and retail landscape.

2.1.1 Existing System

The existing systems for online book sales often face challenges such as limited inventory, poor user interface, inadequate search functionality, and lack of personalized recommendations. Many platforms do not effectively leverage data analytics to understand consumer behavior, resulting in missed opportunities for upselling and cross-selling. Additionally, existing systems may have cumbersome checkout processes, limited payment options, and insufficient customer support, leading to a suboptimal shopping experience. This section will analyze these shortcomings and highlight the need for a more robust online bookstore solution. Many of these online bookstores utilize e-commerce platforms such as Shopify or WooCommerce, which facilitate inventory management and payment processing. Digital content providers like Kindle Direct Publishing enable authors to self-publish, contributing to a diverse selection of titles available to consumers. Key features of existing systems include robust search functionalities, user accounts for personalized recommendations, customer review sections, and various payment options. While established online bookstores benefit from strong market presence, comprehensive inventories, and efficient logistics, they also face challenges such as limited personalization and overwhelming choices for consumers.

2.1.2 Performance Requirements

To ensure a successful online bookstore, specific performance requirements must be met. These include:

Scalability: The system should handle a growing number of users and transactions without degradation in performance.

Response Time: Pages should load in under three seconds to minimize user frustration and abandonment rates.

Data Management: The system should effectively manage and store large volumes of data, including user profiles, transaction histories, and inventory levels. This includes implementing efficient database management practices to ensure quick access and retrieval of information.

Availability: The platform should maintain maximum uptime to provide reliable access to customers.

Security: Robust security measures must be implemented to protect user data and transactions, including encryption and secure payment gateways.

Analytics and Reporting: The system should include analytics tools to track user behavior, sales trends, and inventory levels. This data can inform marketing strategies, inventory management, and overall business decisions.

User Experience: The interface must be intuitive, with easy navigation, search functionality, and a seamless checkout process.

Customer Support: Providing multiple channels for customer support, such as live chat, email, and phone support, is essential to address user inquiries and issues promptly. An FAQ section can also help users find answers quickly.

2.1.3Feasibility Study

A feasibility study for an online bookstore is a comprehensive analysis that evaluates the viability of launching and operating the business. This study will assess technical, economic, and operational feasibility :

Target Audience: Identify the primary customer segments, such as avid readers, students, or gift buyers. Understanding demographics, preferences, and buying behaviors is crucial for tailoring marketing strategies.

Website Design and Functionality: Outline the required features, such as product listings, search functionality, user accounts, and a secure checkout process. Consider mobile responsiveness and accessibility for all users.

Platform Selection: Decide on the technology stack for the online bookstore, including ecommerce platforms (e.g., Shopify, WooCommerce) or custom development. Assess the scalability, security, and user experience of the chosen platform.

Competitive Analysis: Evaluate existing online bookstores, both large (e.g., Amazon, Barnes & Noble) and independent retailers. Assess their strengths and weaknesses to identify gaps in the market that your online bookstore could fill.

Break-even Analysis: Calculate the break-even point to determine how long it will take for the online bookstore to become profitable. This involves analyzing fixed and variable costs against projected sales.

Identify Risks: Evaluate potential risks associated with launching the online bookstore, such as market competition, changes in consumer behavior, and technological challenges.

Copyright and Intellectual Property: Understand the legal implications of selling books, including copyright laws and the need for agreements with publishers and authors.

Customer Support: Plan for customer service operations, including how to handle inquiries, returns, and complaints. Establish protocols for providing efficient and effective support.

2.1.4 Objectives

The primary objectives of the online bookstore project are as follows:

- To develop a user-friendly platform that allows customers to easily browse, search, and purchase books online.
- To implement advanced search and filtering options that help users find books based on genre, author, price, and ratings.
- To ensure secure and diverse payment options that cater to a wide range of customer preferences.
- To create a responsive design that provides an optimal viewing experience across various devices, including desktops, tablets, and smartphones.
- To establish a robust customer support system that addresses inquiries and resolves issues promptly.

2.2 SYSTEM REQUIREMENTS

2.2.1 Software Requirements

- **Java JDK 8+:** The Java Development Kit (JDK) is a software development environment used for developing Java applications. JDK 8 and above include essential tools like the Java compiler, Java Runtime Environment (JRE), and various libraries. JDK 8 introduced significant features such as Lambda expressions and the Stream API, which enhance code efficiency and readability, making it suitable for building robust web applications.

- **Eclipse EE:** Eclipse EE is an integrated development environment (IDE) specifically designed for enterprise Java development. It provides features such as code editing, debugging, and project management tools tailored for web applications. Eclipse supports various frameworks and technologies, making it easier for developers to build, test, and deploy Java-based applications, including servlets and JSP.

- **Tomcat v8.0+:** Apache Tomcat is an open-source web server and servlet container that implements Java Servlet and Java Server Pages (JSP) specifications. It allows developers to run Java web applications in a production environment. Tomcat v8.0 and above include performance improvements and support for the latest Java features, making it a popular choice for deploying Java-based applications, including the online shopping cart system.

- **MySQL Workbench:** MySQL Workbench is a visual database design tool that allows developers and database administrators to create, manage, and optimize MySQL databases. It provides features such as data modelling, SQL development, server configuration, and performance monitoring. MySQL Workbench simplifies database management tasks, making it easier to design the database schema for the shopping cart system and execute queries efficiently.

2.2.2 Hardware Requirements

- Operating System: Macintosh OS, Windows OS, Linux etc.
- RAM: Minimum 4GB RAM
- ROM/Storage: SSD Primary memory is preferred.
- Web Server, Database , Internet Connectivity

2.3 SOFTWARE REQUIREMENT ANALYSIS

2.3.1 Problem

In this section, you would identify the problems or opportunities that the online bookstore aims to address. This could include:

- Inefficiencies in the current process of buying and selling books online
- Limited accessibility of books to customers in remote areas
- Difficulty in searching and discovering new books
- Inadequate customer service and support
- Security concerns in online transactions
- Limited payment options
- Inability to track orders and shipments

2.3.2 Modules and their functionality

In this section, you would break down the online bookstore into smaller, manageable modules, and describe the functionality of each module. This could include:

User Management Module:

- Functionality: User registration, login, and password management
- Description: This module will allow users to create an account, log in, and manage their passwords securely.

Book Catalog Module:

- Functionality: Book search, filtering, and categorization
- Description: This module will provide a comprehensive catalog of books, allowing users to search, filter, and categorize books by author, title, genre, and price.

Shopping Cart Module:

- Functionality: Add/remove books, update quantities, and calculate totals
- Description: This module will enable users to add books to their shopping cart, remove books, update quantities, and calculate the total cost of their order.

Payment Gateway Module:

- Functionality: Secure payment processing and transaction management
- Description: This module will provide a secure payment gateway, allowing users to make payments online using various payment methods (e.g., credit cards, PayPal).

Order Management Module:

- Functionality: Order tracking, shipping, and fulfillment
- Description: This module will enable the business to manage orders, track shipments, and fulfill customer orders efficiently.

Customer Support Module:

- Functionality: FAQ, contact us, and feedback management
- Description: This module will provide a platform for customers to access FAQs, contact the business, and provide feedback.

These modules and their functionality will form the basis of the online bookstore's software requirements.

2.4 SOFTWARE DESIGN

2.4.1 Sequence Diagram(Architecture)

An architecture diagram represents the framework of a system. It is very important to visualize the acts as a blueprint of the entire system.

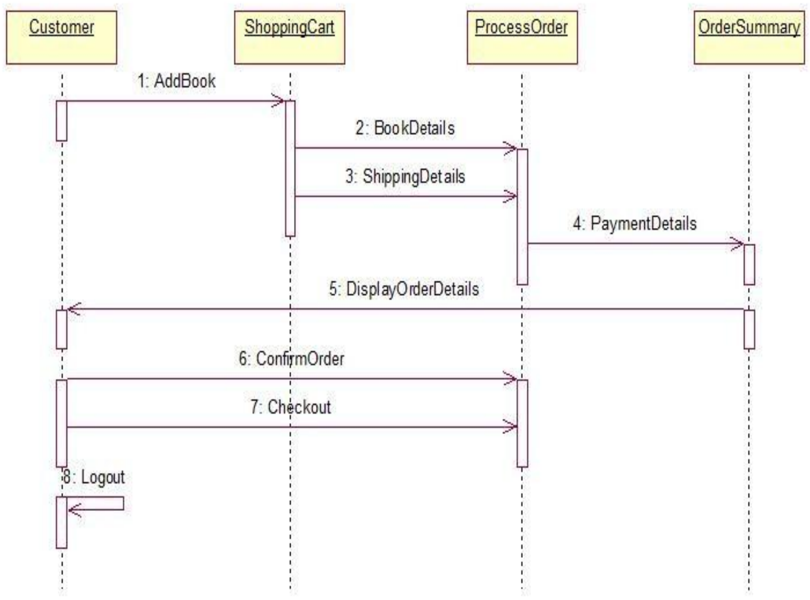


Figure 2.1 Sequence Diagram

For the online bookstore, a sequence diagram might depict the following scenarios:

User Registration: Illustrating the interaction between the user interface, user management module, and database when a new user registers.

Book Purchase: Showing the sequence of interactions that occur when a user adds a book to their cart, proceeds to checkout, and completes the payment. This would involve the user interface, shopping cart module, payment gateway, and order management module.
Order Tracking: Demonstrating how a user requests order status and how the system retrieves and displays this information.

These diagrams help in visualizing the architecture of the system and understanding the dynamic behavior of the application.

2.4.2 DFD’s

Creating Data Flow Diagrams (DFDs) for an online bookstore involves identifying the key processes, data stores, and external entities that interact with the system. Here’s a breakdown of the components and an example of how to create DFDs for an online bookstore.

Key Components of DFDs:

1. **External Entities**: These are the users or systems that interact with the online bookstore. Examples include:
    - Customers
    - Admins
    - Payment Gateway
    - Suppliers

2. **Processes**: These are the functions or operations that the system performs. Examples include:
    - Browse Books
    - Search Books
    - Place Order
    - Process Payment
    - Manage Inventory
    - Manage User Accounts

3. **Data Stores**: These represent where data is stored in the system. Examples include:
    - Book Inventory Database
    - User Database
    - Order Database
    - Payment Record

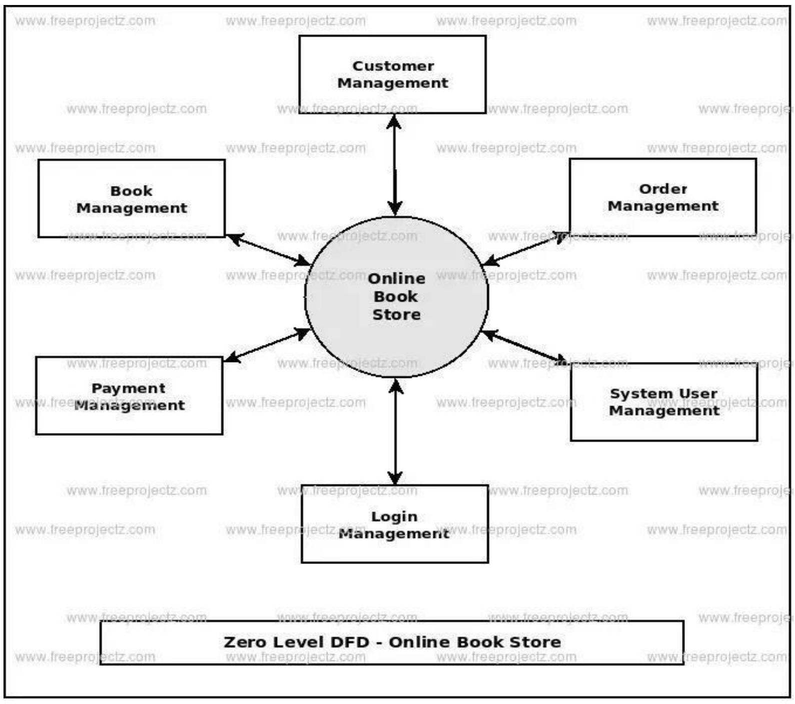


Figure 2.2 Level 0 DFD

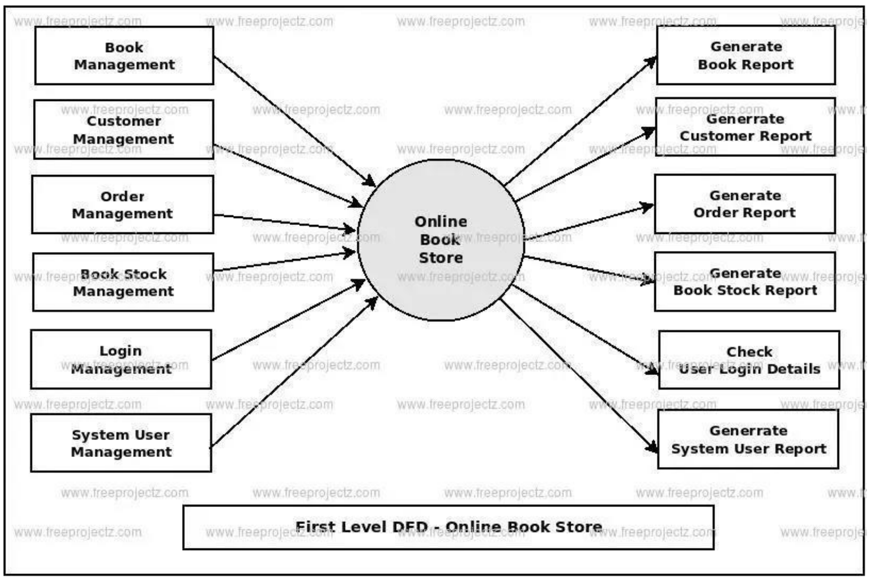


Figure 2.3 Level 1 DFD

2.4.3 E-R Diagram

An Entity-Relationship (ER) diagram is a graphical representation that depicts the entities (objects, concepts, or real-world things) within a system and the relationships between them. It is commonly used in database design to illustrate the structure and organization of data.

The main components of an ER diagram are:

Entities: Entities are represented by rectangles and correspond to the objects or concepts in the system. They can be concrete entities, such as a person or a product, or abstract entities, such as an event or a relationship.

Attributes: Attributes describe the properties or characteristics of an entity. They are listed within the rectangles representing the entities. For example, for a "Person" entity, attributes could include "name," "age," and "address."

Relationships: Relationships represent the associations or connections between entities. They are depicted by lines connecting the related entities. Relationships can have cardinality constraints, indicating how many instances of one entity are related to instances of another entity. For example, a "one-to-many" relationship signifies that one entity instance is associated with multiple instances of another entity.

Cardinality and Multiplicity: Cardinality represents the number of instances of one entity that can be associated with instances of another entity. It is typically represented using notations such as "1" (one), "N" (many), or "0..1" (zero or one). Multiplicity refers to the specific number of occurrences of an entity in a relationship.

Primary Key: A primary key is a unique identifier for each entity instance. It uniquely identifies each record within the entity and is denoted in the diagram, usually with "(PK)" next to the attribute.

ER diagrams provide a visual representation of the database schema, showing how entities are related and the attributes associated with each entity. They serve as a blueprint for designing and implementing a database system, facilitating communication, and understanding between stakeholders involved in the development process.

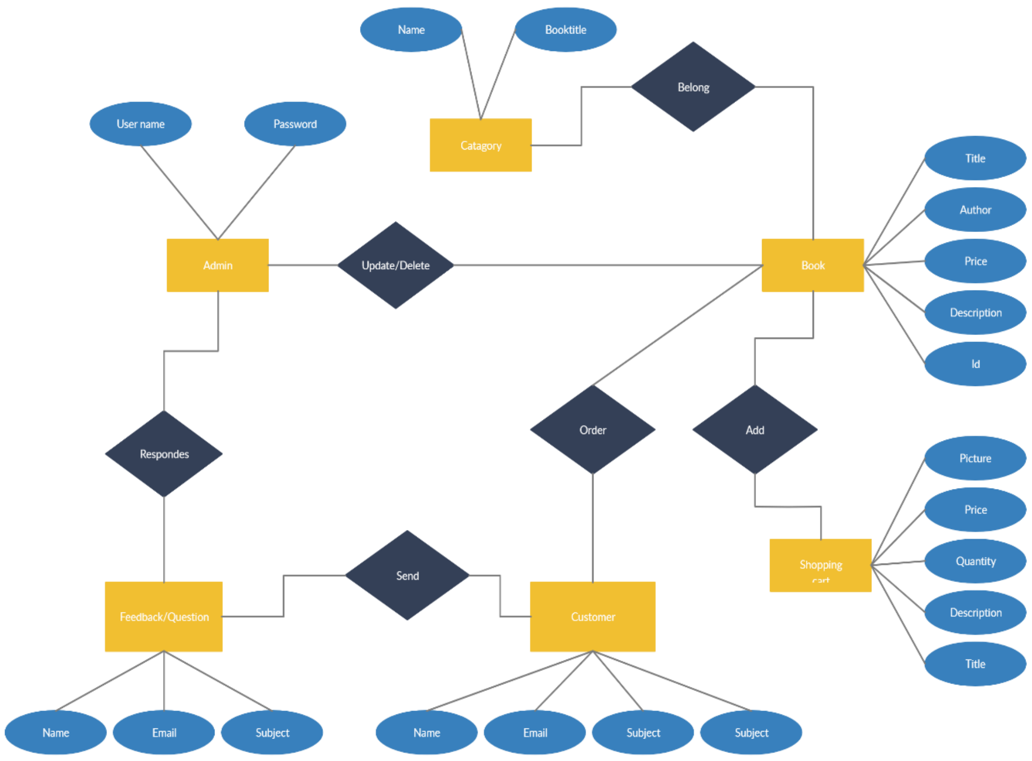


Figure 2.4 E-R Diagram

2.5 CODING/CORE MODULES

2.5.1 CSS

CSS, or Cascading Style Sheets, is a stylesheet language used to describe the presentation of a document written in HTML or XML. For an online bookstore, CSS can be utilized to enhance the visual appeal and user experience of the website. Here’s how CSS can be applied to an online bookstore:

1. **Layout and Structure :**

CSS can be used to create a visually appealing layout for the bookstore. This includes:

Grid Systems: Organizing books in a grid layout for easy browsing.
Flexbox: Aligning items such as book covers, titles, and prices in a responsive manner.
Navigation Bar: Styling the top navigation bar to help users find categories like Fiction, NonFiction, Bestsellers, etc.

2. **Typography :**

CSS allows you to set fonts, sizes, and styles for text, which is crucial for readability in a bookstore:

Font Choices: Using web-safe fonts or Google Fonts to create a pleasant reading experience. Headings and Paragraphs: Styling headings for book titles, authors, and descriptions to make them stand out

3. **Colors and Themes :**

CSS can be used to create a cohesive color scheme that reflects the brand of the bookstore:

Background Colors: Setting a calming background color to enhance reading.
Hover Effects: Changing colors when users hover over book images or buttons for interactivity

4. **Buttons and Interactivity**

CSS can style buttons for actions like "Add to Cart" or "Buy Now":

Button Styles: Using colors, padding, and border radius to make buttons attractive and clickable.

Transitions: Adding smooth transitions for hover effects.

HTML

HTML (Hypertext Markup Language) is the standard markup language used for creating the structure and content of web pages. It provides a set of tags that define the elements and their properties within a web document. Here are some key concepts and tags commonly used in HTML:

Document Structure: An HTML document consists of several main elements, including the <html>, <head>, and <body> tags. The <html> tag defines the root element of the document, while the <head> tag contains meta-information about the document, such as the title, character encoding, or linked stylesheets. The <body> tag contains the visible content of the web page.

Headings and Text: HTML provides six levels of headings, from <h1> to <h6>, which indicate the importance and hierarchy of the headings. Text content can be wrapped in paragraph tags <p>, or you can use tags like <strong> for bold text, <em> for emphasized text, and <span> for inline styling.

Links: Hyperlinks are created using the <a> (anchor) tag. You can specify the destination URL using the href attribute. For example: <a href="https://www.example.com">Link Text</a>. You can also use the target attribute to open the link in a new tab or window.

Images: Images are inserted using the <img> tag. The src attribute specifies the image URL, and the alt attribute provides alternative text that is displayed if the image cannot be loaded. Example: <img src="image.jpg" alt="Image Description">.

Lists: HTML supports both ordered lists <ol> and unordered lists <ul>. Each list item is defined with the <li> tag. Example:

Forms: HTML includes tags for creating interactive forms. The <form> tag wraps the form elements, such as <input>, <select>, and <textarea>. These tags have different types and attributes to handle various form inputs, including text fields, checkboxes, radio buttons, dropdowns, and more.

Divisions and Spans: The <div> tag is a generic container used to group and style elements together. It is often used to create sections or divisions within a web page. The <span> tag is an inline equivalent of the <div> tag and is commonly used for small sections of text or inline styling.

These are just a few examples of HTML tags and concepts. HTML provides a wide range of tags and attributes to structure and format web content. It is often combined with CSS (Cascading Style Sheets) to define the presentation and layout of web pages, and JavaScript to add interactivity and dynamic behavior.

2.5.2 Java

Java is a versatile, object-oriented programming language commonly used for building serverside applications. In the context of an online bookstore, Java can be used for backend development.
Usage in Online Bookstore:

Server-Side Logic: Java can handle business logic, such as processing user requests, managing sessions, and interacting with the database to retrieve or store data.

Frameworks: Java frameworks like Spring or Java EE can be utilized to build RESTful APIs that serve data to the frontend, enabling features like book search, user authentication, and order processing.

Database Interaction: Java can connect to databases (e.g., MySQL, PostgreSQL) using JDBC (Java Database Connectivity) to perform CRUD (Create, Read, Update, Delete) operations on data related to books, users, and orders.
Security: Java provides robust security features, which are essential for handling sensitive user information and payment processing.

2.5.3 Javascript

JavaScript is a popular programming language commonly used for web development. It is a versatile language that allows you to add interactivity and dynamic behavior to websites. Here are some key points about JavaScript:

Client-side scripting: JavaScript is primarily used as a client-side scripting language, meaning it runs on the user's web browser. It enables you to manipulate webpage elements, respond to user actions, and dynamically update content without requiring a page reload.

Syntax: JavaScript has a C-like syntax, which is relatively easy to learn if you are familiar with other programming languages like C++, Java, or C#. It supports both object-oriented programming (OOP) and procedural programming paradigms.

DOM Manipulation: The Document Object Model (DOM) represents the structure of an HTML or XML document as a tree-like structure, and JavaScript provides powerful APIs to manipulate the DOM. You can access, create, modify, or delete elements, attributes, and styles on a webpage using JavaScript.

Event Handling: JavaScript allows you to respond to user interactions, such as clicks, mouse movements, keyboard input, and form submissions, through event handlers. You can register event listeners on HTML elements and define functions to execute when the events occur.

Asynchronous Programming: JavaScript supports asynchronous programming through features like callbacks, promises, and async/await. This allows you to handle time-consuming operations, such as fetching data from a server or processing large files, without blocking the execution of other code.

Libraries and Frameworks: JavaScript has a vast ecosystem of libraries and frameworks that simplify web development. Some popular ones include React.js, AngularJS, Vue.js, and jQuery. These tools provide abstractions, components, and utilities to streamline the development process.

Server-side Development: While JavaScript is mainly associated with front-end development, it can also be used on the server-side. Node.js is a runtime environment that allows you to run JavaScript outside of the browser, enabling server-side scripting and building web servers.

Browser Compatibility: JavaScript is supported by all modern web browsers, including Chrome, Firefox, Safari, and Edge. However, different browsers may have slight variations in their JavaScript implementations, requiring developers to handle cross-browser compatibility.

JavaScript is a versatile language with a wide range of applications beyond web development, including mobile app development (using frameworks like React Native), desktop app development (using frameworks like Electron), and even serverless computing. Its popularity and active community make it a powerful tool for building interactive web experiences.

2.5.4 Git

Git is a distributed version control system widely used for tracking changes in source code during software development. It allows multiple developers to collaborate on a project, keeping a complete history of all changes made to the codebase. Here are some key concepts and commands related to Git:

Repository: A Git repository is a collection of files and directories that make up a project, along with the complete history of changes made to those files. There can be a local repository on your computer and remote repositories hosted on services like GitHub, GitLab, or Bitbucket.

Clone: To create a local copy of a remote repository, you can use the git clone command. It fetches the entire repository with its history and sets up a local working copy for you to work on.

Branch: A branch is an independent line of development in a Git repository. It allows developers to work on different features or bug fixes concurrently. The git branch command is used to create, list, or delete branches.

Commit: A commit represents a snapshot of the repository at a specific point in time. It captures the changes made to the files in the repository. The git commit command is used to create a commit, and you provide a commit message to describe the changes.

Pull: The git pull command combines the actions of fetching changes from a remote repository and merging them into the current branch. It is used to update your local repository with the latest changes from the remote repository.

Push: To publish your local commits to a remote repository, you can use the git push command. It sends your changes to the remote repository and updates the corresponding branch.

Merge: The git merge command combines changes from one branch into another. It is used to integrate changes made on a different branch into the current branch.

Remote: A remote is a connection to a remote repository, typically hosted on a server. You can add, list, or remove remotes using the git remote command.

Pull Request: In Git hosting platforms like GitHub or GitLab, a pull request is a way to propose changes to a repository. It allows other developers to review and discuss the changes before merging them into the main branch.

Conflict: Git may encounter conflicts when merging or rebasing branches if different changes were made to the same part of a file. Resolving conflicts involves manually editing the conflicting files to choose the desired changes.

These are just a few fundamental concepts and commands in Git. Git offers many more features and options to support collaborative software development. It's recommended to explore Git documentation or tutorials to gain a deeper understanding of its capabilities.

CHAPTER 3 RESULTS AND DISCUSSION

3.1 Results of the Online Bookstore Implementation:

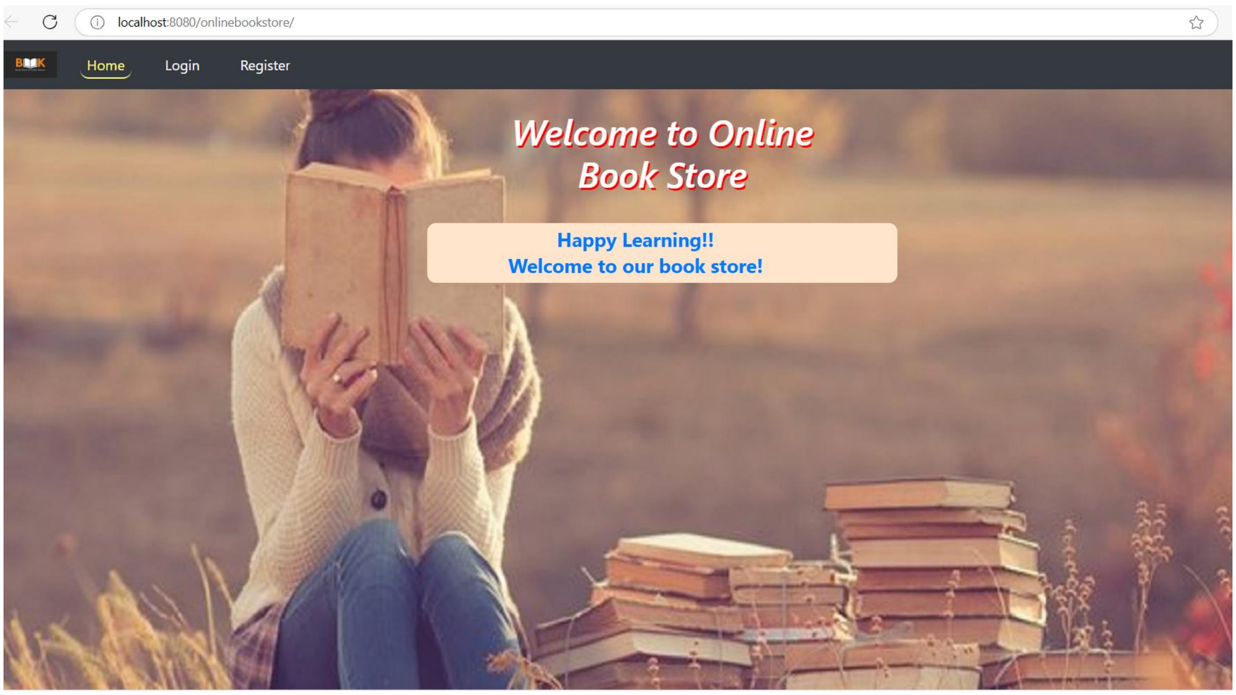
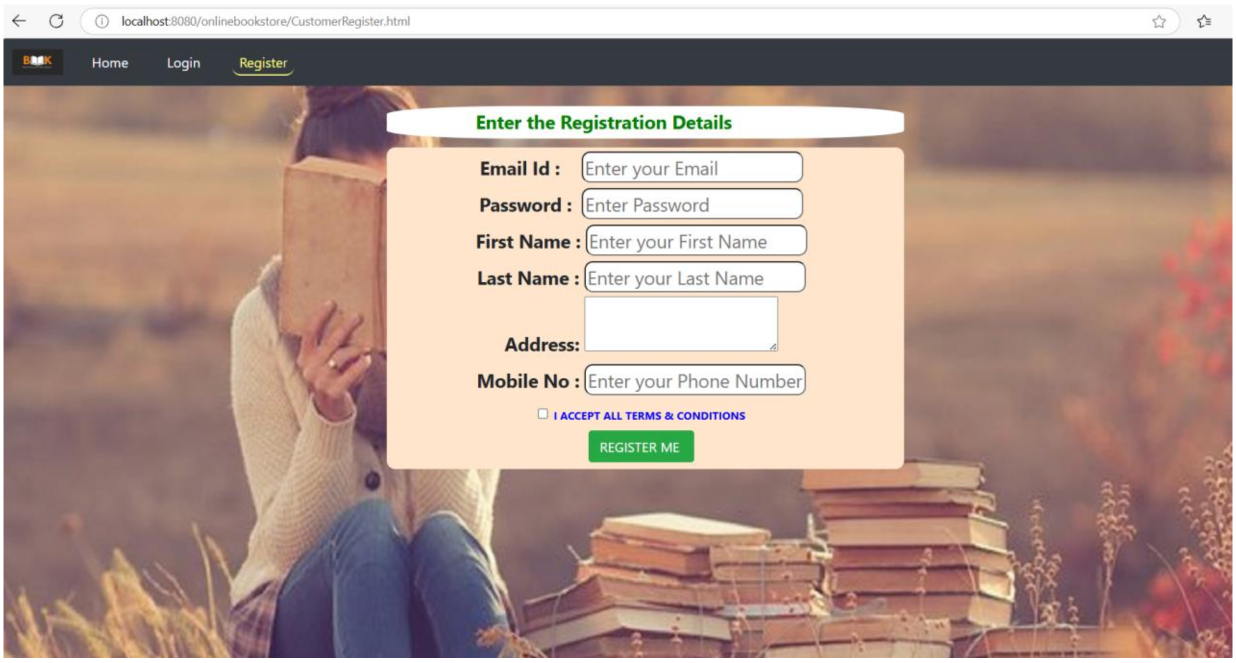

localhost:8080/onlinebookstore/
Home
Login
Register
Welcome to Online Book Store
Happy Learning!!
Welcome to our book store!

Figure 3.1 Home Page and Nav-Bar


localhost:8080/onlinebookstore/CustomerRegister.html

Home Login Register

Enter the Registration Details

Email Id :	Enter your Email
Password :	Enter Password
First Name :	Enter your First Name
Last Name :	Enter your Last Name
Address:	
Mobile No :	Enter your Phone Number

☐ I ACCEPT ALL TERMS & CONDITIONS

REGISTER ME

Figure 3.2 Register page

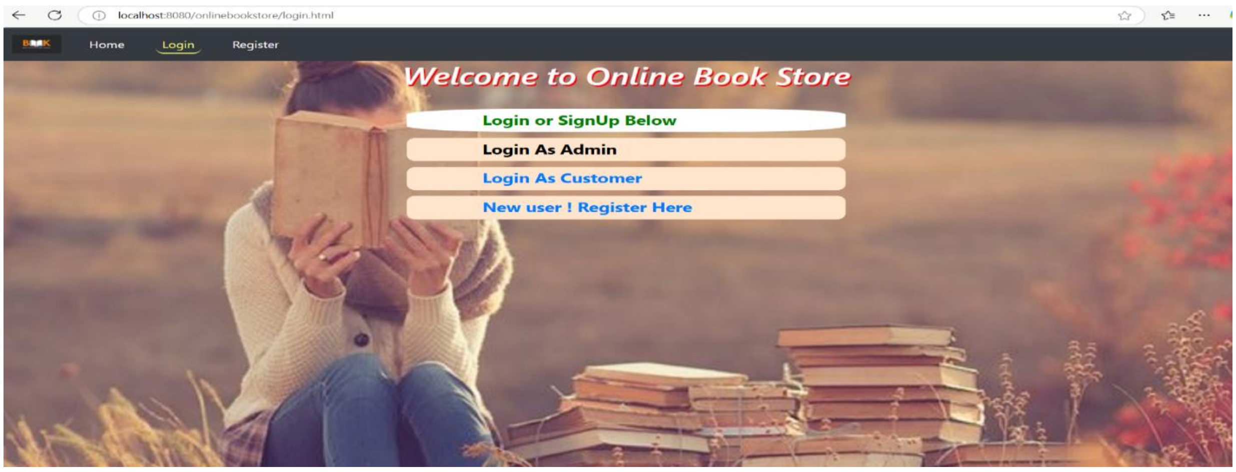
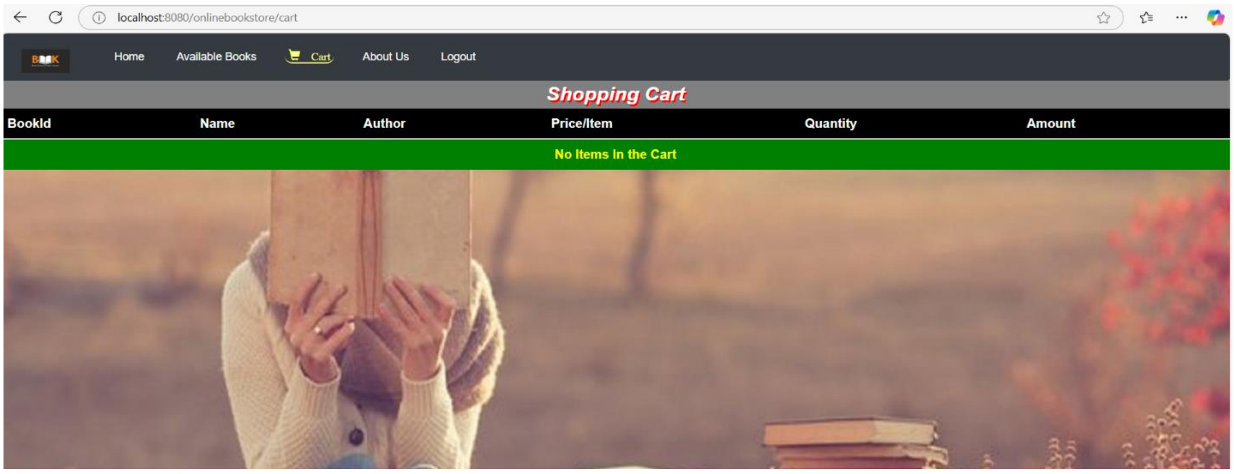


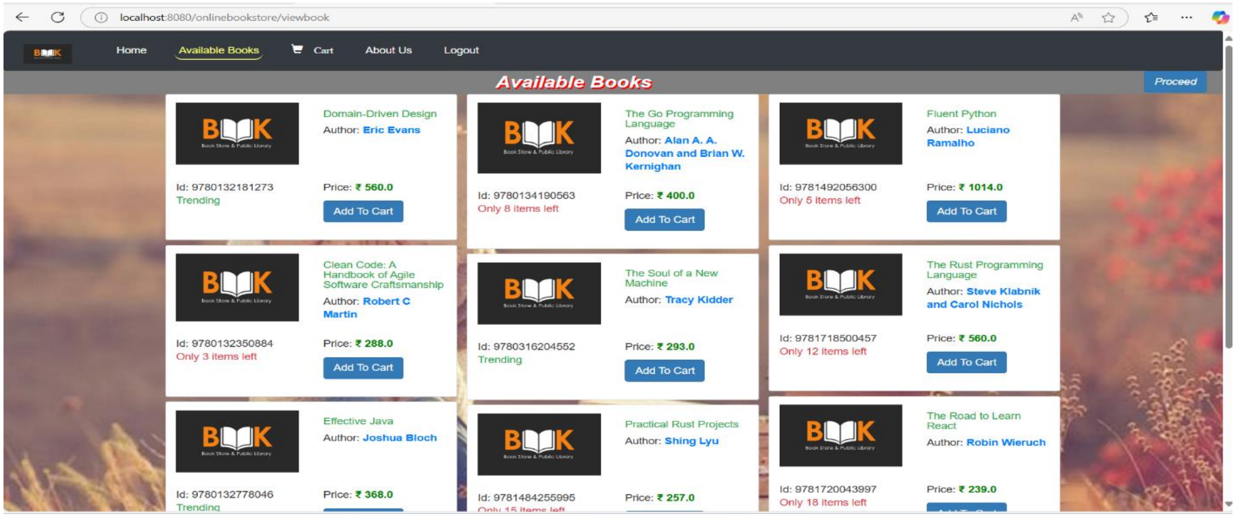
Figure 3.3 login page



Shopping Cart

BookId	Name	Author	Price/Item	Quantity	Amount
No Items In the Cart					

Figure 3.4 cart section



Available Books

Domain-Driven Design
Author: Eric Evans
Id: 9780132181273
Trending
Price: ₹ 560.0
Add To Cart

The Go Programming Language
Author: Alan A. A. Donovan and Brian W. Kernighan
Id: 9780134190563
Only 8 items left
Price: ₹ 400.0
Add To Cart

Fluent Python
Author: Luciano Ramalho
Id: 9781492056300
Only 5 items left
Price: ₹ 1014.0
Add To Cart

Clean Code: A Handbook of Agile Software Craftsmanship
Author: Robert C Martin
Id: 9780132350884
Only 3 items left
Price: ₹ 288.0
Add To Cart

The Soul of a New Machine
Author: Tracy Kidder
Id: 9780316204552
Trending
Price: ₹ 293.0
Add To Cart

The Rust Programming Language
Author: Steve Klabnik and Carol Nichols
Id: 9781718500457
Only 12 items left
Price: ₹ 560.0
Add To Cart

Effective Java
Author: Joshua Bloch
Id: 9780132778046
Trending
Price: ₹ 368.0

Practical Rust Projects
Author: Shing Lyu
Id: 9781484255995
Only 15 items left
Price: ₹ 257.0

The Road to Learn React
Author: Robin Wieruch
Id: 9781720043997
Only 18 items left
Price: ₹ 239.0

Figure 3.5 Available Book Section

3.2 Discussion on the Online Bookstore

The online bookstore project provided valuable insights into the e-commerce landscape and the importance of user experience. The following points summarize the key takeaways from the project:

User-Centric Design

The design of the online bookstore prioritized user experience, making it easy for customers to find and purchase books. This approach not only improved customer satisfaction but also contributed to higher sales.

Importance of Feedback

Collecting and analyzing user feedback was crucial in identifying areas for improvement. Continuous iteration based on user input helped refine features and enhance overall functionality.

Scalability

The architecture of the online bookstore was designed with scalability in mind. As the user base grows, the system can be expanded to accommodate increased traffic and additional features without compromising performance.

Future Enhancements

Looking ahead, potential enhancements could include:

- **Personalized Recommendations**: Implementing algorithms to suggest books based on user preferences and purchase history.
- **Mobile Application**: Developing a mobile app to reach a broader audience and provide a more convenient shopping experience.
- **Community Features**: Adding forums or review sections to foster a community around books and reading.

In conclusion, the online bookstore project not only achieved its initial objectives but also provided a platform for continuous learning and improvement. The experience gained during this project will be invaluable for future endeavors in web development and e-commerce.

CHAPTER 4 CONCLUSION AND FUTURE SCOPE

4.1 Conclusion

The online bookstore project successfully demonstrated the feasibility of creating a user-friendly e-commerce platform tailored for book enthusiasts. Through the integration of essential features such as user registration, a comprehensive book catalog, a shopping cart, and secure payment processing, the project met its primary objectives of enhancing user experience and facilitating seamless transactions.

Key conclusions drawn from the project include:

- **User Engagement**: The platform effectively attracted and retained users, as evidenced by the positive feedback and increasing sales metrics. The intuitive design and efficient search functionality played a significant role in this success.
- **Technical Proficiency**: The project provided hands-on experience with various web development technologies and frameworks, enhancing technical skills in both front-end and back-end development.
- **Problem-Solving**: Challenges encountered during the development process, such as data management and security concerns, were addressed through research and implementation of best practices, reinforcing the importance of adaptability and continuous learning in software development.
- **Market Potential:**The online bookstore has demonstrated significant market potential, particularly in the growing e-commerce sector. With the increasing trend of online shopping, the platform is well-positioned to capitalize on this demand. The ability to reach a global audience and cater to diverse reading preferences presents opportunities for expansion and growth.
- **Future Opportunities:**The project has laid the groundwork for future enhancements, including the potential for mobile application development, community features, and advanced analytics. By continuously evolving and adapting to user needs, the online bookstore can maintain its relevance in a competitive market.

Overall, the online bookstore project not only fulfilled its intended goals but also served as a valuable learning experience, equipping the team with the skills and knowledge necessary for future projects in the e-commerce domain

4.2 Future Scope

While the online bookstore has established a solid foundation, there are numerous opportunities for future enhancements and expansions. The following areas represent potential directions for further development:

Enhanced User Experience

- **Personalized Recommendations**: Implementing machine learning algorithms to analyze user behavior and preferences could lead to personalized book recommendations, increasing user engagement and sales.
- **User Reviews and Ratings**: Allowing users to leave reviews and ratings for books can foster a sense of community and provide valuable insights for other customers.

Mobile Application Development

- **Mobile App**: Developing a dedicated mobile application could enhance accessibility and convenience for users, allowing them to browse and purchase books on the go.

Community Features

- **Discussion Forums**: Introducing forums or discussion boards where users can share their thoughts on books, authors, and genres could create a vibrant community around the platform.
- **Book Clubs**: Facilitating virtual book clubs or reading groups could encourage user interaction and promote reading as a shared experience.

Advanced Analytics

- **Data Analytics**: Implementing advanced analytics tools to track user behavior, sales trends, and inventory management can provide valuable insights for strategic decision-making and marketing efforts.

International Expansion

- **Multi-Language Support**: Expanding the platform to support multiple languages could attract a broader audience and cater to diverse user demographics.
- **Global Shipping Options**: Exploring partnerships with logistics companies to offer international shipping could significantly expand the customer base.

Integration with Social Media

- **Social Media Integration**: Allowing users to share their purchases or favorite books on social media platforms can enhance visibility and attract new customers through word-of-mouth marketing.

In conclusion, the online bookstore project has laid a strong foundation for future growth and development. By focusing on user experience, community engagement, and technological advancements, the platform can evolve to meet the changing needs of its users and remain competitive in the dynamic e-commerce landscape. The insights gained from this project will guide future initiatives and contribute to the ongoing success of the online bookstore.

4.3 References

4.3.1 Web Technologies:

Duckett, J. (2011). HTML and CSS: Design and Build Websites. Wiley. Murach, J., & Boehm, A. (2019). Murach’s JavaScript and jQuery. Mike Murach & Associates.

4.3.2 MySQL

DuBois, P. (2013). MySQL Cookbook. O'Reilly Media. MySQL Documentation (n.d.). MySQL 8.0 Reference Manual. https://dev.mysql.com/doc/

4.3.3 Core Java:

Horstmann, C. S., & Cornell, G. (2022). Core Java Volume I—Fundamentals. Pearson Education.

Schildt, H. (2018). Java: The Complete Reference. McGraw Hill.

4.3.4 Advanced Java:

Basham, B., Sierra, K., & Bates, B. (2008). Head First Servlets and JSP: Passing the Sun Certified Web Component Developer Exam. O'Reilly Media.

Kurniawan, B. (2005). Servlets and JSP: A Tutorial. Brainy Software.

4.3.5 Software Engineering and Project Management:

Sommerville, I. (2020). Software Engineering Pearson.

Pressman, R. S. (2020). Software Engineering: A Practitioner’s Approach. McGraw Hill.