

PROJECT REPORT ON EBOOK LIBRARY: QUOTE



In fulfillment of the requirement for
the 5th Semester of
BACHELORS OF COMPUTER APPLICATION

Submitted by:
Kriti Deepa Kalita
Registration no: 22070524
Roll no: UT-221-036-0015
Session: 2024-2025

Under the guidance of:
Binita Deka
CS Department
Rangia- 781354

Name and Address of the Accredited Institute
Rangia College, Rangia-781354

PROFORMA FOR SUBMISSION OF
STUDENT
PROJECT PROPOSAL/SYNOPSIS
(PROJECT TITLE AND GUIDE DETAILS)

Name and Address of the
student

Name: Kriti Deepa Kalita
S/O: Kalpana Kalita
Village: Sadaikuchi
Post Office: Dimu
District: Kamrup
Pin: 781350

Title of the Project

Ebook Library: Quote

Name and Address of the
Guide

Mrs. Binita Deka
Computer science Department
Rangia College
Rangia- 781354

Signature of the Student
(Kriti Deepa Kalita)
Date:

Signature of the Guide
(Binita Deka)
Date:

DECLARATION

I hereby declare that the work done the dissertation entitled “Ebook Library: Quote” carried out by me at Rangia College and submitted in fulfilment of the requirement for 5th Semester BCA program of the same institute under Rangia College.

I also declare that the various contents incorporated in the dissertation have not been submitted in any form from the award of any other degree of other institute or university.

Kriti Deepa Kalita
BCA 5th Semester
Registration no- 22070524
Rangia college, Dist.- Kamrup

CERTIFICATE FROM GUIDE

This is to certify that the project/dissertation report entitled 'Ebook Library: Quote' is a bonafide work done by Kriti Deepa Kalita (GU Registration no- 22070524, 5th semester student of 3RD year BCA course at RANGIA COLLEGE under my personal supervision and guidance. The report is found worthy acceptance for the fulltime fulfillment of the requirements of BCA 5th semester course work for the degree of Bachelors in Computer.

All helps received have been daily acknowledged and no part of these reports has been reproduced for any other degree of diploma.

Signature of Guide/Supervisor Name:
Mrs. Binita Deka, MCA
Computer Science Department
Rangia College
Rangia-781354

ACKNOWLEDGEMENT

While completing the project, several individuals extended their kind co-operation and help with due encouragement. And I owed a lot of allegiance to all of them for enabling me to complete my project work.

Firstly, I would like to thank the Head of Computer Science Department of Rangia college for his kind permission to carry out a project work.

Secondly, I would like to express my deep sense of gratitude to Mrs. Binita Deka, my internal guide. They have advisor all thought out my work.

Thirdly to Naba Prasad Baro my partner in this Project. He completed this project with me.

To those mention above and to those who inspire and encouraged me, I am expressing gratitude once again.

With due regards
Your sincerely
Kriti Deepa Kalita

CONTENT

CHAPTER 1:	INTRODUCTION
1.1	Project Introduction
1.2	Tools and Technology
CHAPTER 2:	INITIAL SYSTEM STUDY
2.1	Drawback of existing system
2.2	Benefit of proposed system
CHAPTER 3:	FEASIBILITY STUDY
3.1	Introduction to SDLC
3.2	Feasibility study
3.2.1	Technical Feasibility
3.2.2	Economical Feasibility
3.2.3	Operational Feasibility
CHAPTER 4:	REQUIREMENT ANALYSIS & SPECIFICATION
4.1	Requirement Analysis
4.2	Requirement Gathering And Analysis
4.2.1	Requirements Gathering
4.2.2	Analysis of Gathered Requirements
4.3	Software Requirement specification

CHAPTER 5:	SYSTEM ANALYSIS
5.1	Introduction
5.2	Entity Relationship Diagram
5.3	Data flow-diagram
5.4	Data Dictionary
CHAPTER 6:	SYSTEM DESIGN
6.1	Introduction
6.2	Architectural design
6.3	Database design
CHAPTER 7:	TESTING
7.1	Testing
CHAPTER 8:	IMPLEMENTATION & MAINTANANCE
8.1	Implementation Plan
8.2	Evaluation
4.2.1	Maintenance
4.2.2	Analysis of Gathered Requirements
CHAPTER 9:	Conclusion
6.1	Conclusion
6.2	Limitation
6.3	Future scope Images of the system
	BIBILIOGRAPHY

Chapter 1

INTRODUCTION

1.1 Project Introduction:

QUOTE is a vibrant and inclusive online platform designed to bridge the gap between creators and consumers of literature. It provides a comprehensive space for publishers and individual writers to publish a variety of creative works such as books, poems, short stories, one-shots, fanfiction, and more. By offering powerful tools for showcasing content, it empowers writers to share their works with a diverse global audience.

For readers, the platform is more than just a place to read content. It allows them to deeply engage with the works, following their favourite authors and publishers, providing feedback through likes and detailed reviews, and interacting with the creative community. Readers can also collaborate with each other to build shared libraries, creating spaces to exchange views on literature, share personal collections, and discuss authors' works.

The platform aims to foster an environment where creativity and community engagement thrive. It's not only about discovering new content but also about building meaningful connections between authors and readers and enabling a dialogue that enhances both the writing and reading experiences.

1.2 Tools and Technology

The following tools and technology are used to develop the system.

Hardware:

- Intel core i5/7th gen processor
- Integrated graphics card
- 8gb DDR4 ram
- 1tb HHDS

Frontend Development

- HTML5: For structuring web pages and ensuring semantic content organization.
- CSS3: For styling and creating visually appealing, responsive designs.
- JavaScript: To add interactivity and dynamic content (e.g., likes, reviews, and collaborative libraries).
- Bootstrap: To simplify responsive design with prebuilt components and grid systems.

Backend Development

- Python Flask: A lightweight framework for server-side logic, managing APIs, and user authentication.

Database

- MySQL: For storing and managing structured data, such as user profiles, creative works, reviews, likes, and collaborative libraries.

Chapter 2

INITIAL SYSTEM STUDY

2.1 Drawback of Existing system:

1. Amazon Kindle

Description: Amazon Kindle is primarily focused on professional authors and publishers. It provides a platform for eBooks and paid content distribution.

Drawbacks:

- Limited Community Engagement: No direct platform for reader-writer interaction, such as collaborative libraries or community discussions.
- Complex User Experience: Overwhelming interface for casual or hobbyist writers.

2. FanFiction.net

Description: FanFiction.net specializes in fan-created works based on existing franchises.

Drawbacks:

- Outdated User Interface: The interface is old-fashioned and not mobile-friendly.
- Limited Genres: Primarily focused on fanfiction, restricting original content creators.
- Poor Community Features: Limited interactivity, with no options for collaborative libraries or personalized content sharing.

2.2 Benefits of Proposed System:

1. **Content Publishing:** The platform supports the publication of a wide range of creative content, from books and poems to one-shots and fanfiction, allowing writers to reach a global audience.
2. **Reader Engagement:** Readers are not passive consumers but active participants who can follow their favourite authors or publishers, provide reviews and likes, and contribute to discussions about the content they love.
3. **Collaborative Libraries:** Readers have the unique ability to create collaborative libraries with other readers. These libraries allow users to share their interests, insights, and favourite works with friends, making the reading experience a shared journey.
4. **Author & Publisher Interaction:** Writers and publishers have dedicated profiles where they can interact with their readers. They can update followers on new releases, respond to feedback, and build a loyal readership.
5. **Community Features:** The platform fosters a sense of community through features that allow readers and writers to join discussions, participate in book clubs, and share recommendations.

Chapter 3

FEASIBILITY STUDY

3.1 Introduction of SDLC (System Development Life Cycle):

The System Development Life Cycle (SDLC) is a systematic framework used to develop, implement, and maintain information systems. It provides a structured approach for managing complex system development projects by dividing them into distinct, manageable phases. SDLC ensures the system meets organizational requirements while being delivered on time, within budget, and with high quality. Following are the Key Phases of the SDLC:

Planning:

- Define project goals and scope.
- Develop a project plan outlining resources, timelines, and risks

System Analysis:

- Gather and document user requirements.
- Create requirement specifications for the new system.

System Design:

- Develop the system architecture and data flow diagrams.
- Prepare a blueprint for system construction.

Development (Implementation):

- Use programming languages, databases, and tools to build the system.
- Conduct unit testing during coding to ensure functionality.

Testing:

- Test the entire system for bugs, compatibility, and performance issues.
- Perform functional, integration, and user acceptance testing.
- Validate that the system meets the defined requirements.

Deployment:

- Install the system in the production environment.
- Train users and provide necessary documentation.
- Roll out the system, either as a pilot or full-scale implementation.

3.2 FEASIBILITY STUDY

Feasibility is the determination of whether or not a project is worth doing. The process followed in making this determination is called feasibility study. This type of study determines if a project can and should be taken. Once it has been determined that a project is feasible, the analyst can go ahead and prepare the project specification which finalizes project requirement.

Normally feasibility studies culminate in a written or oral feasibility report. The contents and recommendations of such a study will be used as a sound basis for deciding whether to proceed, postponed or cancel the project.

3.2.1 Technical Feasibility

In examining technical feasibility, configuration of the system is given more importance than the actual makes of hardware. The configuration should give the complete picture about the systems requirements: This can be used as a basis for the tender document against which dealers and manufacturers can later make their equipment bids. Specific hardware and software products can then be evaluated keeping in view with the logical needs.

The technical issues usually raised during the feasibility stage of the investigation include the following: -

Does the necessary technology exist to do what is suggested?

- Does the proposed equipment's have the technical capacity to hold the data required to use the new system?
- Will the proposed system provide adequate responses to inquiries, regardless of the number or location of users?
- Can the system be upgraded if developed?
- Are there technical guarantees of accuracy, reliability, ease of access and data security?

3.2.2 Economic Feasibility

Economic analyses are used for evaluating the effectiveness of the proposed system. More commonly known as cost/benefit analyses; the procedure is to determine the benefits and saving that are expected from a proposed system and compare them with costs. If benefits outweigh costs, a decision is taken to design and implement the system. Otherwise, further justification or alternative in the proposed system will have to be made if it is to have a chance of being approved. This is an ongoing effort that improves in accuracy at each phase of system lifecycle. As it is the first phase of the project, the cost is estimated.

3.1.3 Operational Feasibility:

Proposed projects are beneficial only if they can be turned out into information systems. That will meet the organizations operating requirements. An operational feasibility aspect of the project is to be taken as an important part of the project implementation.

Some of the important issues raised to test the operational feasibility of a project include the following: -

- Is there sufficient support for the project for management? From users?
- Will the system be used and work properly if it is being developed and implemented?
- Will there be any resistance from the users that will undermine the possible Application benefits?

Chapter 4

REQUIREMENT ANALYSIS AND SPECIFICATION

4.1 REQUIREMENT ANALYSIS:

The requirements analysis and specification phase start once the feasibility study phase is completed and the project is found to be financially sound and technically feasible. The goal of the requirements analysis and specification phase is to clearly understand the customer requirements and to systematically organize these requirements in a specification document called Software Requirement Specification (SRS). This phase consists of the following activities:

- Requirements gathering and analysis
- Requirement specification

4.2 REQUIREMENT GATHERING AND ANALYSIS:

As an analyst one should start the requirement gathering and analysis activity by collecting all information from the customer/user, which could be used to develop the requirements of the system. The two main activities involved in the requirements gathering and analysis phase:

4.3 Software Requirement Specification (SRS):

After the analyst has collect all the requirement information regarding the software to be developed, and has removed all incompleteness, inconsistence, and anomalies from the specification, the analyst systematically organizes the requirements in the form of an SRS document.

1. Contents of the SRS Document:

- a. Functional requirement
- b. Non-functional requirement
- c. Goals of implementation Format of a good SRS document:
 - 1. Introduction: Background, Overall Description, Environmental Characteristic, Hardware, Peripherals, People
 - 2. Goals of Implementation
 - 3. Functional Requirements
 - 4. Non-functional Requirements
 - 5. Behavioural Description
 - o System states
 - o Events and Actions

Chapter 5

SYSTEM ANALYSIS

5.1 INTRODUCTION:

System Analysis is a critical phase in the development of any information system, aiming to ensure that the system is designed to meet the specific needs of the organization. It involves studying the existing systems, gathering requirements, identifying issues, and designing solutions. The primary goal is to create an efficient, scalable, and user-friendly system that enhances business operations and meets user expectations. A key part of system analysis is the use of various modelling tools that help break down complex systems and structure them in a way that is easy to understand, design, and implement.

Among the most essential tools in system analysis are Entity-Relationship Diagrams (ER Diagrams), Data Flow Diagrams (DFD), and the Data Dictionary. These tools help in visualizing the relationships, data flows, and definitions of data elements within the system. Together, they form a comprehensive approach to documenting and analysing a system's structure, data requirements, and processes.

5.2 Entity-Relationship Diagram (ER Diagram):

An Entity-Relationship Diagram (ER Diagram) is a high-level conceptual data model used to represent the structure of a system by showing the system's entities and the relationships between them. This diagram helps in identifying how data is organized and how different components within the system interact with one another. ER diagrams are a powerful tool in visualizing the relationships between real-world entities and the data stored within the system.

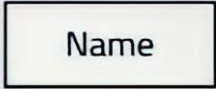
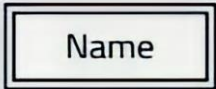
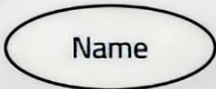
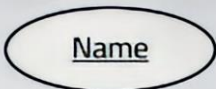
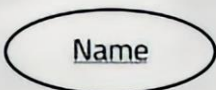

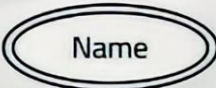
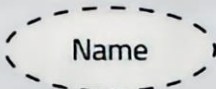
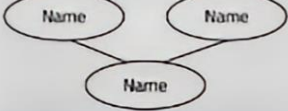

1. Entities: These are the key objects or concepts that are involved in the system, represented by rectangles in the ER diagram. Entities are typically nouns such as "Employee," "Customer," or "Product." Each entity is described by its attributes, which define its characteristics.
- 2.
3. Attributes: Represented by ovals, attributes are the properties or characteristics that describe an entity. For instance, a "Customer" entity may have attributes such as "Customer ID," "Name," "Email," and "Address."

3. Relationships: These are the associations between entities and are represented by diamonds. A relationship defines how two entities are linked together. For example, an "Order" may be related to both "Customer" and "Product," showing which customer placed which order and which products are included in that order.

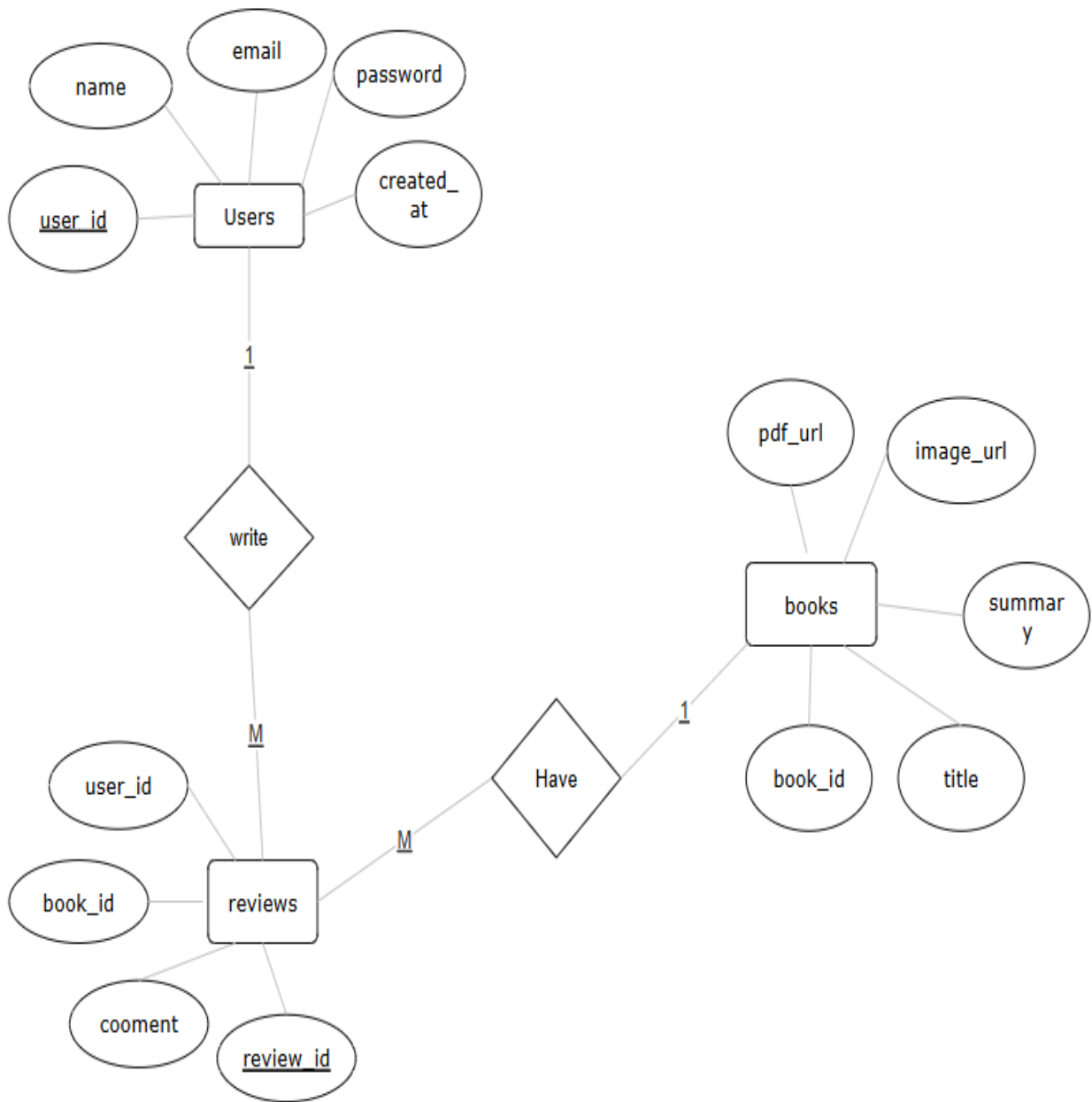
4. Cardinality: Cardinality defines the number of instances of one entity that can be associated with the number of instances of another entity. Common types of cardinality include one-to-one, one-to-many, and many-to-many.

ER diagrams provide clarity by modelling both the data elements and their relationships, making it easier for developers and analysts to design databases and systems that handle data effectively.

ER diagrams often use symbols to represent three different types of information. Boxes are commonly used to represent entities. Diamonds are normally used to represent relationships and ovals are used to represent attributes. The symbols used in the E-R diagram have its own meanings. They are given below:

Meaning	Notation
Entity	
Weak Entity	
Attribute (mandatory)	
Primary identifier attribute	
Partial identifier attribute	
Alternate identifier attribute	
Multi-valued attribute	
Derived attribute	
Composite attribute	
Optional attribute	

ER DIAGRAM



5.3 Data Flow Diagram:

A Data Flow Diagram (DFD) is a graphical representation used to show how data moves through a system. DFDs help illustrate the flow of information between different processes, data stores, and external entities in the system. They provide insight into how data is processed, stored, and used, as well as how it moves through various stages of a system.

1. Processes: Represented by circles or rounded rectangles, processes describe actions that transform input data into output data. For example, a "Process Order" function might take customer order details as input and generate an order receipt as output.
2. Data Flows: Represented by arrows, data flows indicate the movement of data between processes, external entities, and data stores. Each data flow is labelled to show what information is being transferred (e.g., "Order Information" or "Payment Details").
3. Data Stores: Represented by open-ended rectangles, data stores hold data within the system. These can be physical or logical databases, files, or repositories of information that store data temporarily or permanently. For example, an "Inventory Database" could be a data store that holds information about available products.

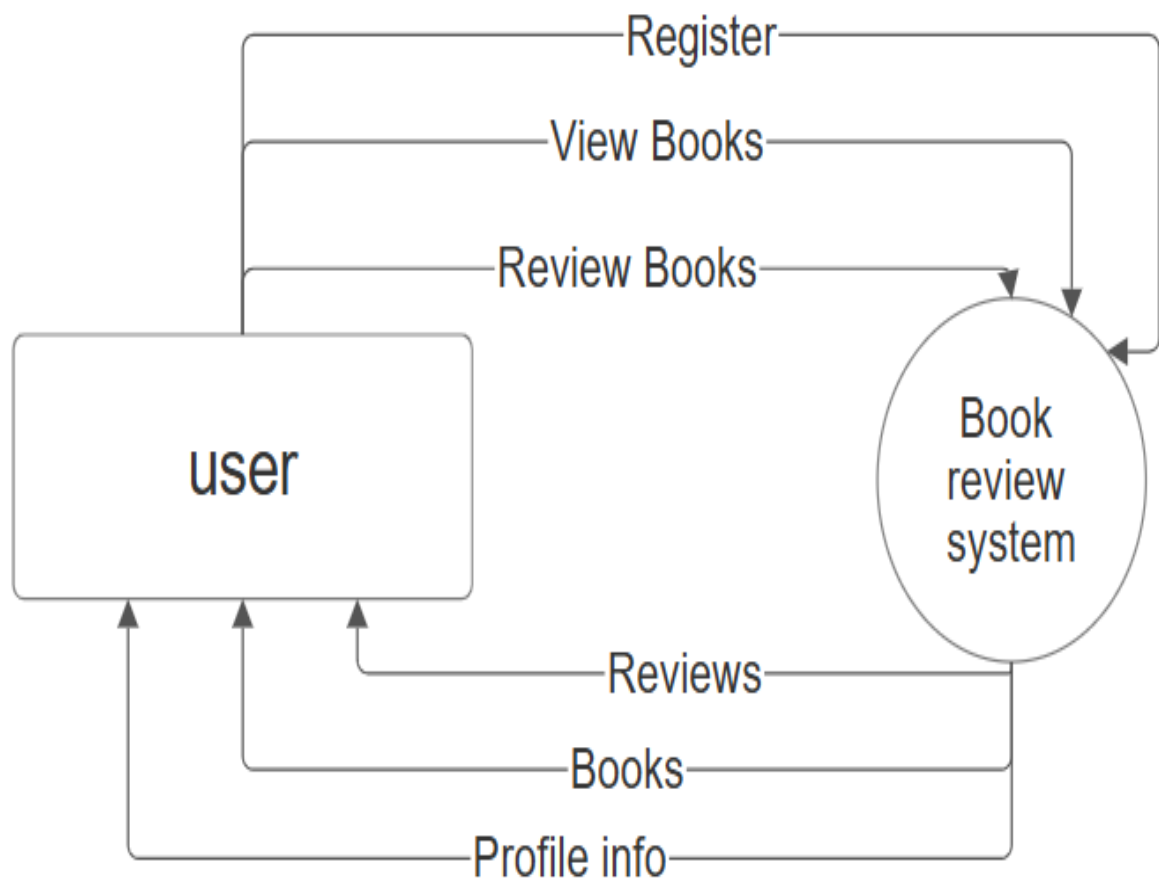
4. External Entities: Represented by squares or rectangles, external entities are sources or destinations of data that exist outside the system. For instance, "Customer" and "Supplier" are external entities that interact with the system by providing or receiving data.

DFDs are typically represented at multiple levels:

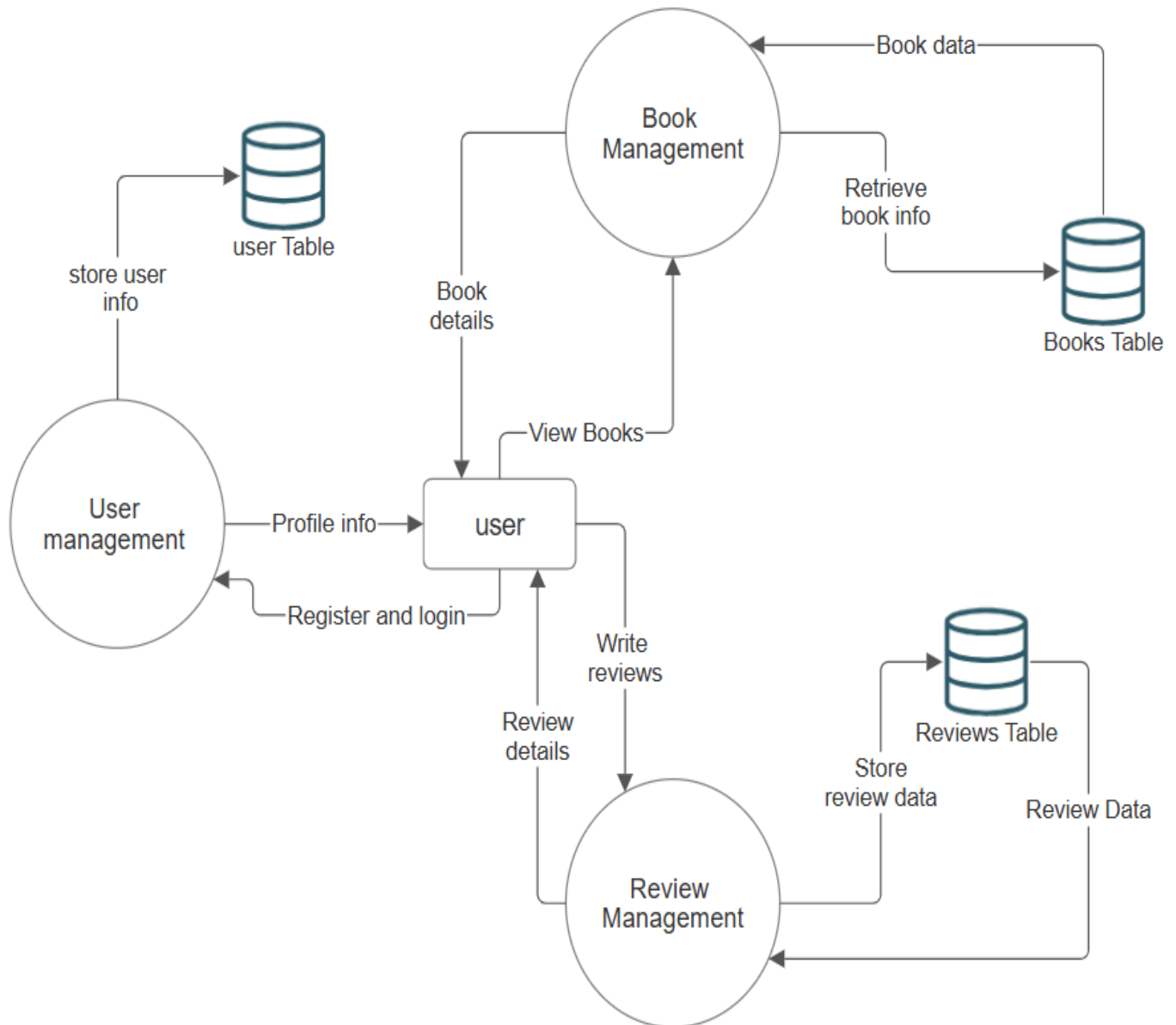
1. Level 0 (Context Diagram): A high-level DFD that shows the system as a whole, illustrating its relationship with external entities and major data flows.
2. Level 1 and beyond: These diagrams break the system into sub-processes, providing a more detailed view of data flow and process interactions.

DFDs are vital for understanding the movement and transformation of data within a system, ensuring that all processes are captured and that the system works

CONTEXT DIAGRAM



LEVEL 1 DFD



5.4 Data Dictionary:

A Data Dictionary is a centralized repository or catalogue that defines all the data elements used in a system. It provides a comprehensive reference for the system's data components, ensuring consistency, clarity, and proper usage across the development and operational teams. The data dictionary defines the structure, meaning, and rules associated with each data element, offering crucial information for system design, implementation, and maintenance.

1. **Data Elements:** These are the smallest units of data, such as fields or variables. For example, a "Customer ID" field could be defined as a data element that holds a unique identifier for each customer.
2. **Attributes and Properties:** The dictionary specifies the attributes and properties of data elements. For example, a "Date of Birth" field might be defined as a date with a specific format (e.g., MM/DD/YYYY).
3. **Relationships Between Data Elements:** The data dictionary also documents how different data elements relate to one another, including the dependencies and associations that impact data integrity and functionality.
4. **Data Types:** Each data element's type is defined in the dictionary, such as integer, string, or Boolean, to ensure that data is used correctly in processing.

The data dictionary acts as a reference for all stakeholders involved in the system's development and provides a clear understanding of data usage, consistency, and structure.

DATA DICTIONARY OF THE PROPOSED SYSTEM

TABLE_NAME	COLUMN_NAME	DATA_TYPE	IS_NULLABLE	COLUMN_KEY	EXTRA
books	book_id	int	NO	PRI	auto_increment
books	title	varchar	NO		
books	summary	text	NO		
books	image_url	varchar	NO		
books	pdf_url	varchar	NO		
reviews	review_id	varchar	NO	PRI	
reviews	user_id	int	NO	MUL	
reviews	book_id	int	NO	MUL	
reviews	comment	longtext	YES		
users	user_id	int	NO	PRI	auto_increment
users	name	varchar	NO		
users	email	varchar	NO	UNI	
users	password	varchar	NO		
users	created_at	timestamp	YES		DEFAULT_GENERATED

Chapter 6

SYSTEM DESIGN

6.1 INTRODUCTION:

The system design for QUOTE provides a robust and scalable foundation to support the platform's goals of creative publishing and interactive community engagement. It focuses on enabling seamless content management for authors and immersive experiences for readers, leveraging modern technologies for efficiency and growth.

Key objectives include:

- **Content Publishing:** Enable authors and publishers to easily upload and manage books, poems, stories, and fanfiction.
- **Reader Interaction:** Support features like likes, reviews, and following to foster connections between readers and authors.
- **Community Building:** Include discussion forums, book clubs, and recommendations to create a vibrant literary community.

This design ensures a user-friendly, scalable platform that meets the diverse needs of a global user base while supporting future enhancements.

6.2 Architectural Design

Three-tier architecture divides the system into three distinct layers, each responsible for specific functions. This modular design enhances scalability, maintainability, and performance.

1) Presentation Layer

a) Function: Manages the user interface and client-side interactions.

b) Technologies:

i. HTML & CSS: Provides the structure and styling for web pages.

ii. JavaScript: Enables interactivity and dynamic behaviour on the client side.
Bootstrap: Ensures responsive and consistent design across devices.

Html code o login cum sign in page “login_page.html”

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Sign in || Sign up from</title>
  <!-- font awesome icons -->
  <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-
awesome/6.1.1/css/all.min.css" integrity="sha512-
KfkfwYDsLkIlwQp6LFn18zNdLGxu9YAA1QvwINks4PhcElQSwqcyVLLD9aMhXd13uQjoXtEKNosOWa
ZqXgel0g==" crossorigin="anonymous" referrerpolicy="no-referrer" />
  <!-- css stylesheet -->
  <link rel="stylesheet" href="../static/css/login_page.css">
</head>
<body>

  <div class="container" id="container">
    <div class="form-container sign-up-container">
      <form action="/signup" method="POST">
        <h1>Create Account</h1>
        <div class="social-container">
          <a href="#" class="social"><i class="fab fa-facebook-
f"></i></a>
          <a href="#" class="social"><i class="fab fa-google-plus-
g"></i></a>
          <a href="#" class="social"><i class="fab fa-linkedin-
in"></i></a>
        </div>
        <span>or use your email for registration</span>
        <div class="infield">
          <input type="text" id="signupName" placeholder="Name"
name="name" required />
          <label></label>
        </div>
        <div class="infield">
          <input type="email" id="signupEmail" placeholder="Email"
name="email" required/>
          <label></label>
        </div>
        <div class="infield">
          <input type="password" id="signupPassword"
placeholder="Password" name="password" required minlength="6" />
          <label></label>
        </div>
      </form>
    </div>
  </div>
</body>
```

```

        <button type="submit">Sign Up</button>
    </form>
</div>
<div class="form-container sign-in-container">
    <form action="/signin" method="POST" >
        <h1>Sign in</h1>
        <div class="social-container">
            <a href="#" class="social"><i class="fab fa-facebook-
f"></i></a>
            <a href="#" class="social"><i class="fab fa-google-plus-
g"></i></a>
            <a href="#" class="social"><i class="fab fa-linkedin-
in"></i></a>
        </div>
        <span>or use your account</span>
        <div class="infield">
            <input type="email" id="signinEmail" placeholder="Email"
name="email" required/>
            <label></label>
        </div>
        <div class="infield">
            <input type="password" id="signinPassword"
placeholder="Password" name="password" required />
            <label></label>
        </div>
        <a href="#" class="forgot">Forgot your password?</a>
        <button type="submit">Sign In</button>
    </form>
</div>
<div class="overlay-container" id="overlayCon">
    <div class="overlay">
        <div class="overlay-panel overlay-left">

            <h1>Welcome Back!</h1>
            <p>To keep connected with us please login with your
personal info</p>
            <button type="button" onclick="switchToSignIn()">Sign
In</button>
        </div>
        <div class="overlay-panel overlay-right">
            <h1>Hello, Friend!</h1>
            <p>Enter your personal details and start journey with
us</p>
            <button type="button" onclick="switchToSignUp()">Sign
Up</button>
        </div>
    </div>
    <button id="overlayBtn"></button>

```



```

    </div>
</div>

<!-- js code -->
<script src="../../static/js/login_page.js">

</script>

</body>
</html>

```

CSS code o login cum sign in page "login_page.html"

```

@import
url('https://fonts.googleapis.com/css2?family=Poppins:wght@100;200;300;400;500;600;700;800;900&display=swap');

* {
  padding: 0px;
  margin: 0px;
  box-sizing: border-box;
}

:root {
  --linear-grad: linear-gradient(to right, #141E30, #243B55);
  --grad-clr1: #141E30;
  --grad-clr2: #243B55;
}

body {
  height: 100vh;
  background: #f6f5f7;
  display: grid;
  place-content: center;
  font-family: 'Poppins', sans-serif;
}

.container{
  position: relative;

```

```

width: 850px;
height: 500px;
background-color: #fff;
box-shadow: 25px 30px 55px #5557;
border-radius: 13px;
overflow: hidden;
}

.form-container{
  position: absolute;
  width: 60%;
  height: 100%;
  padding: 0px 40px;
  transition: all 0.0s ease-in-out;
}

.sign-up-container{
  opacity: 0;
  z-index: 1;
}

.sign-in-container{
  z-index: 2;
}

form{
  height: 100%;
  display: flex;
  flex-direction: column;
  align-items: center;
  justify-content: center;
  padding: 0px 50px;
}

h1{
  color: var(--grad-clr1);
}

.social-container{
  margin: 20px 0px;

  a{
    border: 1px solid #DDD;
    border-radius: 50%;
    display: inline-flex;
    justify-content: center;
    align-items: center;
    margin: 0px 5px;
  }
}

```

```

        height: 40px;
        width: 40px;
    }
}

span{
    font-size: 12px;
}

.infield{
    position: relative;
    margin: 8px 0px;
    width: 100%;
}

input{
    width: 100%;
    padding: 12px 15px;
    background-color: #e9e3e3;
    border: none;
    outline: none;
}

label{
    position: absolute;
    left: 50%;
    top: 100%;
    transform: translateX(-50%);
    width: 0%;
    height: 2px;
    background: var(--linear-grad);
    transition: 0.3s;
}

input:focus-label{
    width: 100%;
}

a{
    color: #333;
    font-size: 14px;
    text-decoration: none;
    margin: 15px 0px;
}

a.forgot{
    padding-bottom: 3px;
    border-bottom: 2px solid #EEE;
}

```

```

}

button{
  border-radius: 20px;
  border: 1px solid var(--grad-clr1);
  background: var(--grad-clr2);
  font-size: 12px;
  font-weight: bold;
  padding: 12px 45px;
  letter-spacing: 1px;
  text-transform: uppercase;
  color: #fff;
}

.form-container button{
  margin-top: 17px;
  transition: 80ms ease-in;
}

.form-container button:hover{

  background:#FFF;
  color: var(--grad-clr1);
}

.overlay-container{
  position: absolute;
  top: 0;
  left: 60%;
  width: 40%;
  height: 100%;
  overflow: hidden;
  transition: transform 0.6s ease-in-out;
  z-index:9;
}

#overlayBtn{
  cursor:pointer;
  position: absolute;
  left: 50%;
  top: 304px;
  transform: translateX(-50%);
  width: 143.67px;
  height: 40px;
  border: 1px solid #FFF;
  background: transparent;
  border-radius: 24px;
}

```

```

.overlay{
  position: relative;
  color: #FFF;
  background-color: var(--grad-clr2);
  left: -150%;
  height: 100%;
  width: 250%;
  transition: transform 0.6s ease-in-out;
}

.overlay-panel{
  position: absolute;
  display: flex;
  align-items: center;
  justify-content: center;
  flex-direction: column;
  padding: 0px 40px;
  text-align: center;
  height: 100%;
  width: 340px;
}

.overlay-left{
  right: 60%;
  transform: translateX(-12%);
}

.overlay-right{
  right: 0%;
  transform: translateX(0%);
}

.overlay-panel h1{
  color: #FFF;
}

p{
  font-size: 14px;
  font-weight: 300;
  line-height: 20px;
  letter-spacing: 0.5px;
  margin: 25px 0px 35px
}

```

```

.overlay-panel button{
  border: none;
  background-color: transparent;
}

.right-panel-active .overlay-container{
  transform: translateX(-150%);
}

.right-panel-active .overlay{
  transform: translateX(50%);
}

.right-panel-active .overlay-left{
  transform: translateX(25%);
}

.right-panel-active .overlay-right{
  transform: translateX(35%);
}

.right-panel-active .sign-in-container{
  transform: translateX(20%);
  opacity: 0;
}

.right-panel-active .sign-up-container{
  transform: translateX(66.7%);
  opacity: 1;
  z-index: 5;
  animation: show 0.6s;
}

@keyframes show {
  0%,50%{
    opacity: 0;
    z-index: 1;
  }

  50.1%, 100%{
    opacity: 1;
    z-index: 5;
  }
}

```

```

.btnSclaed{
  animation: scaleBtn 0.6s;
}

@keyframes scaleBtn {
  0%{
    width: 143.67px;
  }
  50%{
    width: 250px;
  }
  100%{
    width: 143.67px;
  }
}

```

JavaScript code o login cum sign in page “login_page.html”

```

const container = document.getElementById('container');
const overlayCon = document.getElementById('overlayCon');
const overlayBtn = document.getElementById('overlayBtn');

overlayBtn.addEventListener('click', () => {
  container.classList.toggle('right-panel-active');
  overlayBtn.classList.remove('btnScaled');
  window.requestAnimationFrame(() => {
    overlayBtn.classList.add('btnScaled');
  });
});

document.querySelector('.sign-in-container form').addEventListener('submit',
async (event) => {
  event.preventDefault();
  const email = document.getElementById('signinEmail').value;
  const password = document.getElementById('signinPassword').value;

  const response = await fetch('/signin', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json'
    },

```

```

        body: JSON.stringify({ email, password })
    });

    const result = await response.json();
    if (result.success) {
        window.location.href = result.redirect;
    } else {
        alert(result.message);
    }
});

document.querySelector('.sign-up-container form').addEventListener('submit',
async (event) => {
    event.preventDefault();
    const name = document.getElementById('signupName').value;
    const email = document.getElementById('signupEmail').value;
    const password = document.getElementById('signupPassword').value;

    const response = await fetch('/signup', {
        method: 'POST',
        headers: {
            'Content-Type': 'application/json'
        },
        body: JSON.stringify({ name, email, password })
    });

    const result = await response.json();
    if (result.success) {
        alert(result.message);
    } else {
        alert(result.message);
    }
});

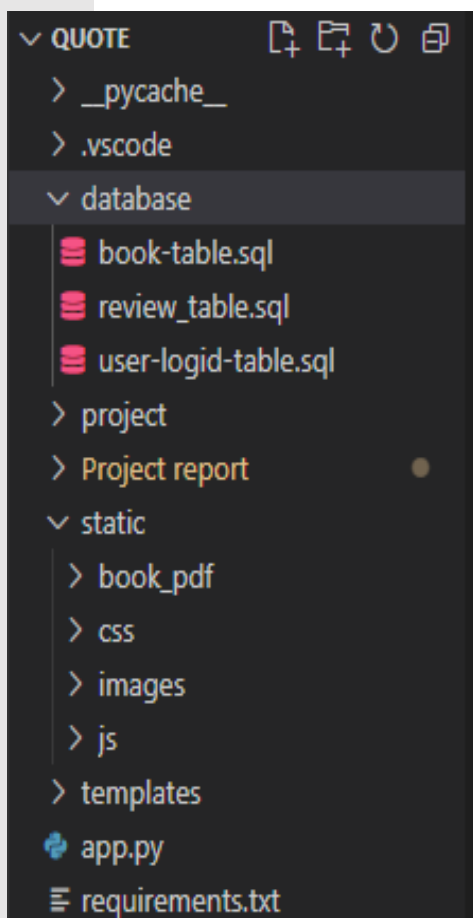
```


2. Application Layer

I. Function: Handles business logic and acts as an intermediary between the presentation and data layers.

II. Technology:

- Flask (Python): Acts as the core backend framework, implementing server-side logic, processing user requests, managing database interactions, and handling RESTful APIs to enable communication between the application and presentation layers.



File Directory for the Flask app of the system:

- Database: store sql table and database creation code.
- Project: python virtual environment. Store python libraries and packages.
- Static: store the static file like CSS, JavaScript codes, images, pdfs.
- Templates: stores the html codes.
- App.py: main flask app of the system.

Python Flask code “app.py”

```
from flask import Flask, request, jsonify, render_template,
redirect,url_for,session
from flask_bcrypt import Bcrypt
from flask_mysqldb import MySQL
import re

app = Flask(__name__)
bcrypt = Bcrypt(app)

# MySQL configuration
app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = 'root'
app.config['MYSQL_DB'] = 'quote_db'
app.secret_key = 'books'
mysql = MySQL(app)

@app.route('/')
def home():
    return render_template('login_page.html')
    # Render your HTML login/signup page

@app.route('/home_real')
def home_real():

    return render_template('home_page.html')

@app.route("/book/api")
def api_book():
    cur = mysql.connection.cursor()
    cur.execute("SELECT book_id,image_url, title, summary FROM books")
    books = cur.fetchall()
    cur.close()

    data = [
        {"id": row[0], "image_url": row[1], "title": row[2], "summary": row[3],
"book_url": url_for('book', book_id=row[0]) }
        for row in books
    ]
    return jsonify(data)
```

```

@app.route('/book/<int:book_id>','methods=['POST', 'GET'])
def book(book_id):
    cur = mysql.connection.cursor()
    cur.execute("SELECT * FROM books where book_id = %s", (book_id,))
    book = cur.fetchone()
    cur.close()

    return render_template('book-display.html', book=book)

@app.route('/signup', methods=['POST', 'GET'])
def signup():
    if request.method == 'GET':
        return render_template('login_page.html') # Render your HTML signup
page

    data = request.get_json()
    name = data['name']
    email = data['email']
    password = data['password']

    # Input validation
    if not re.match(r"^[^@]+@[^@]+\.[^@]+", email):
        return jsonify({"success": False, "message": "Invalid email
address."})
    if len(password) < 6:
        return jsonify({"success": False, "message": "Password must be at
least 6 characters long."})

    # Hash the password
    hashed_password = bcrypt.generate_password_hash(password).decode('utf-8')

    # Insert the user into the database
    cur = mysql.connection.cursor()
    try:
        cur.execute("INSERT INTO users (name, email, password) VALUES (%s, %s,
%s)",
                    (name, email, hashed_password))
        mysql.connection.commit()
        cur.close()
        return jsonify({"success": True, "message": "Account created
successfully."})
    except:

```

```

        return jsonify({"success": False, "message": "Error creating account.
Email might already be registered."})

@app.route('/signin', methods=['POST', 'GET'])
def signin():
    if request.method == 'GET':
        return render_template('login_page.html') # Render your HTML signin
page

    if request.content_type == 'application/json':
        data = request.get_json()
        email = data['email']
        password = data['password']
        print(email)

        # Check if user exists
        cur = mysql.connection.cursor()
        cur.execute("SELECT password FROM users WHERE email = %s", (email,))
        user = cur.fetchone()
        cur.close()

        if user and bcrypt.check_password_hash(user[0], password):
            print("User authenticated successfully")
            cur = mysql.connection.cursor()
            cur.execute("SELECT user_id FROM users WHERE email = %s",
(email,))
            user_id = cur.fetchone()
            session['user_id'] = user_id # Set session variabl
            return jsonify({"success": True, "redirect":
url_for('home_real')})
        else:
            print("Invalid email or password")
            return jsonify({"success": False, "message": "Invalid email or
password."})
        else:
            return jsonify({"success": False, "message": "Request must be JSON and
Content-Type must be 'application/json'."})

if __name__ == '__main__':
    app.run(debug=True)

```

3. Data Layer:

a) Function: Manages the user interface and client-side interactions.

b) Technologies:

iii. MySQL: A relational database that stores structured data and supports complex queries for content and user management

Note: Data layer further explained in the next section “Database Design”

6.2 Database Design:

The general purpose of database is to handle information as an integrated form. A database is a collection of interrelated data, stored with minimum redundancy. In database design, several objectives are considered.

(a) Controlled redundancy's unique aspect of database design is starting data only once, which redundancy and improves performance.

(b) Easy to learn and use

(c) Data independence

(d) Accuracy and integrity

(e) Privacy and security

(f) Performance improvement

I have designed the database to eliminate the redundant data as much as possible. The integration of the data files has been done with proper care. Care has been taken to share the data among all users but some of the database has been kept private beyond the display and manipulation of the records.

Table Descriptions

1. Review Table

Field	Type	Null	Key	Default	Extra
review_id	varchar(255)	NO	PRI	NULL	
user_id	int	NO	MUL	NULL	
book_id	int	NO	MUL	NULL	
comment	longtext	YES		NULL	

2. User Table

Field	Type	Null	Key	Default	Extra
user_id	int	NO	PRI	NULL	auto_increment
name	varchar(100)	NO		NULL	
email	varchar(100)	NO	UNI	NULL	
password	varchar(255)	NO		NULL	
created_at	timestamp	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED

3. Book Table

Field	Type	Null	Key	Default	Extra
book_id	int	NO	PRI	NULL	auto_increment
title	varchar(255)	NO		NULL	
summary	text	NO		NULL	
image_url	varchar(255)	NO		NULL	
pdf_url	varchar(255)	NO		NULL	

Chapter 7

TESTING

7.1 Testing:

In the testing process the Demo version of the software i.e. actual replica of the existing system will be installed so that the users can use it as they like and give their valuable suggestion and advice. There after security can be incorporated in the system. In this phase we will be using both alpha and beta test, which will enable the system to check the whole system thoroughly. The said demo version software can be used for a period of 15 days to 1 months and during this period only training of the proposed software will be imported.

Any engineered product can be tested in one of two ways:

1. Knowing the specified function that a product has been designed to perform, tests can be conducted that demonstrate each function is fully operational while at the same time searching for errors in each function.
2. Knowing the internal workings of a product, tests can be conducted to ensure that "all gears mesh" that is, internal operation are performed according to specifications and all internal components have been adequately exercised.

7.2 Black Box Testing:

The first test approach is called black box testing and it alludes to tests that are conducted at the software interface. It is used to demonstrate that software functions are operational, that is the input is correctly accepted and the output is correctly produced, and the integrity of the external information (e. g. a database) is maintained. A black-box test examines some fundamental aspect of a system with little regard for the internal logical structure of the software. During the black-box tests, those errors, which came to the light, were removed and validation were put through to allow correct input of data and accordingly the correct output is displayed. To prevent occurrence of wrong input of data, a check constraint is added to prompt the user.

7.3 White Box Testing:

White box testing also known as Glass box testing is next approach of testing where the control structure of the procedural design is derived to test case. With the help of this test we can derive test cases that-

1. Guarantee that all independent paths within a module have been exercised at the least once.
2. Exercised all logical decision on their true or false sides,
3. Execute all loops at their boundaries and within their operational bonds and Exercised internal data structures to ensure their validity.

7.4 Integration Testing:

It is a systematic technique for construction of the program structure while at the same time conducting test to uncover error associated with interfacing. The objective is to take unit-tested module and built a program structure that has been dictated by design.

7.5 Unit testing: Unit testing verifies the smallest module of the software designed. Using this testing the entire module can be debugged very easily. The relative complexity of test and the error detected as a result its limited by the constrained scrap established for unit testing. The unit test is always white box oriented and the step can conduct in parallel for multiple modules. Unit testing is considered an adjunct to the coding step. After source code has been developed and verified for the syntax connection, unit test case designed starts.

7.6 Nature of Test Data: The proper choice of data is as useful as the test itself. If the test data inputs are not valid then the reliability of the output is suspect. Test data may be artificial or live. Unlike data, which are based towards typical values, artificial data provides extreme values first testing the limits of the proposed system.

7.7 Output Testing:

No system is useful if it does not provides the required output. Asking the user about the required format must match the output generated by the system under consideration. This test is also done by matching the required formats with the output formats obtained by the system and the users are very much satisfied by the reports generated by the software.

7.8 Debugging:

As testing method has certain disadvantages, the order method used to identify and remove false (bugs) in the system debugging. Debugging includes certain activities performed for programmers during execution of the system. Special programmer run each program one by one in simulated environment using simulated data in order to detect bugs in the programs. Debugging is similar to the process of verification testing. The only difference between verification testing and debugging is that verification is performed to test entire system while debugging is performed to test individual system.

Chapter 8

IMPLEMENTATION AND MAINTANANCE

8.1 Implementation Plan:

Implementation involves testing and deploying the new system, training users, installing applications, and transitioning data from old systems. It may replace a manual or automated system or modify an existing one. Proper implementation is crucial, as poor installation can hinder success. System analysts must handle this phase with precision. Key activities include:

- Training: Teaching users' tools and techniques for system operation.
- Hardware Installation: Preparing and installing the system hardware.
- Procedural Manual: Creating guides for system operation.
- Testing: Verifying programs process data correctly.
- File Conversion: Transferring existing data to the new system.
- Parallel Operation: Running the new and old systems simultaneously to verify accuracy.

8.2 Evaluation:

System evaluation identifies strengths and weaknesses across four dimensions:

1. Operational Evaluation: Assesses usability, reliability, response time, and system utilization.
2. Organizational Impact: Measures benefits like financial gains and operational efficiency.
3. User Manager Assessment: Gauges user and managerial satisfaction with the system.
4. Development Performance: Evaluates development time, cost, and adherence to standards.

8.3 Maintenance:

Maintenance ensures the system meets specifications through error corrections and updates. This phase improves quality by promptly addressing issues and reducing redundant efforts.

Maintenance includes:

1. Adaptive Maintenance: Updating the system for new platforms, OS, or hardware compatibility.
2. Corrective Maintenance: Fixing bugs and improving performance.
3. Perfective Maintenance: Adding new features or modifying functions based on user demands

Chapter 9

Conclusion

9.1 Conclusion:

In the end of this project we are able develop the 1st working prototype for the eBook website Quote. The website is able to achieve meet a substantial amount of its proposed objective with a wide scope for further development and scalability before hitting the real world. The development of QUOTE marks the initial step in creating a dynamic and inclusive literary platform. By integrating features for publishing, engaging, and collaborating, it aspires to reshape the way literature is experienced online. Despite its promising concept and objectives, the current version reflects the efforts of an amateur developer and remains underdeveloped. However, it lays the foundation for a future where readers and writers can connect meaningfully and creatively.

9.2 Limitations:

1. Underdeveloped Features: Some features, such as advanced search, filtering, and collaborative libraries, are not fully functional or optimized, limiting user experience.
2. Technical Constraints: The platform's design and performance may suffer from inefficiencies due to the developer's lack of expertise.
3. User Interface: The current UI lacks polish, which may affect the platform's ability to attract and retain users.

9.3 Future Scope:

Feature Enhancement: Expanding functionalities, improving user engagement tools, and refining the collaborative libraries.

Performance Optimization: Enhancing technical aspects for faster load times, robust data management, and scalability.

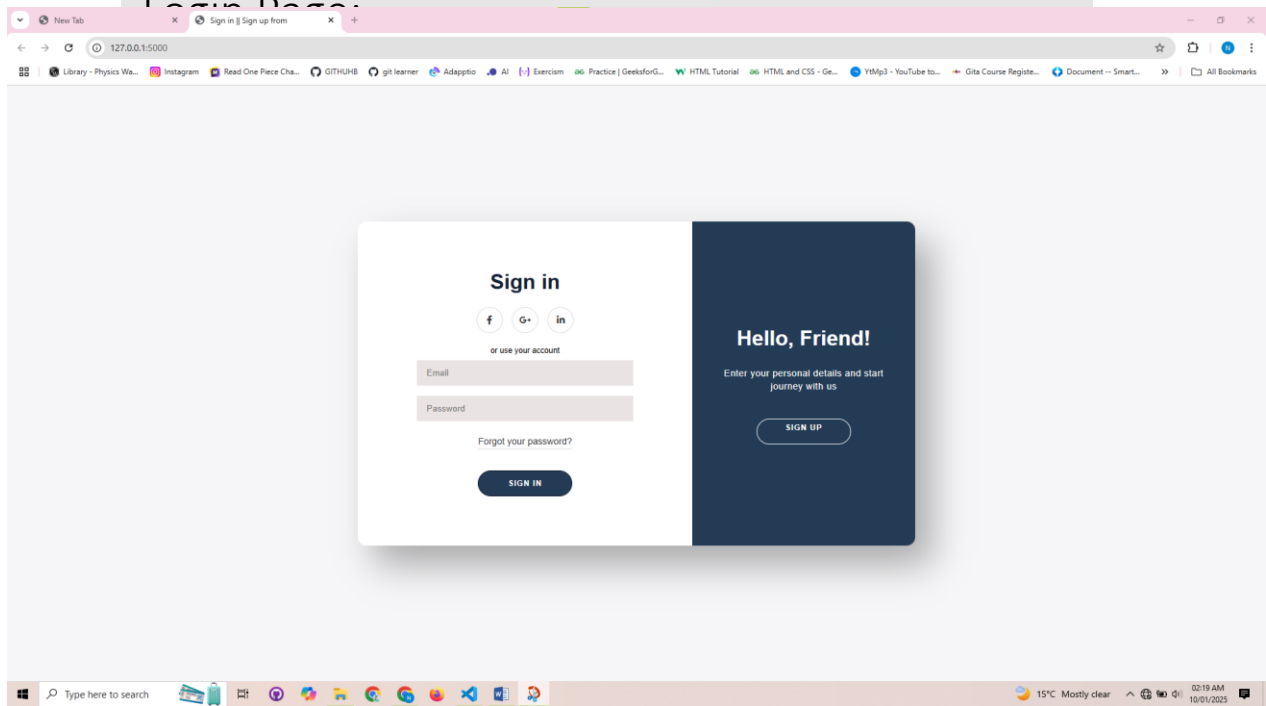
UI/UX Improvements: Redesigning the interface for a smoother, more visually appealing user experience.

AI Integration: Incorporating AI for personalized recommendations, advanced content discovery, and automated moderation.

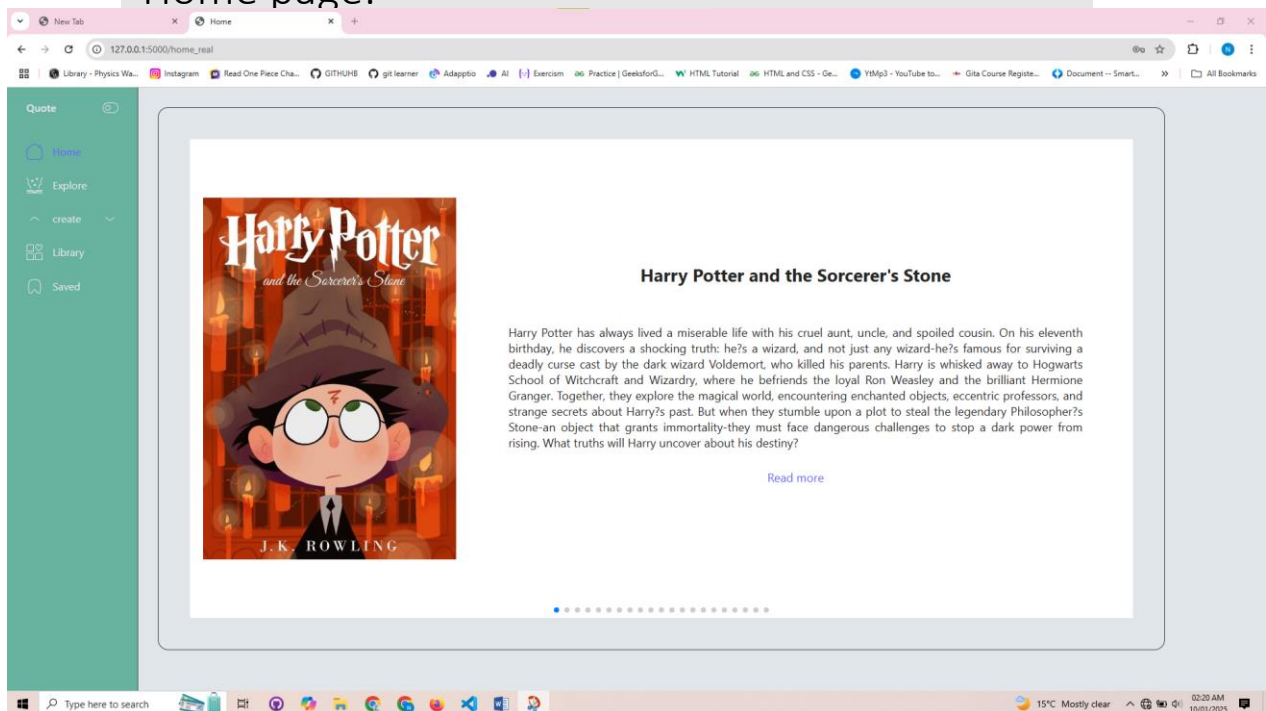
Community Building: Introducing forums, live chat, and interactive events like author Q&A sessions to strengthen the community aspect.

Mobile Optimization: Developing dedicated mobile applications for a seamless user experience across devices.

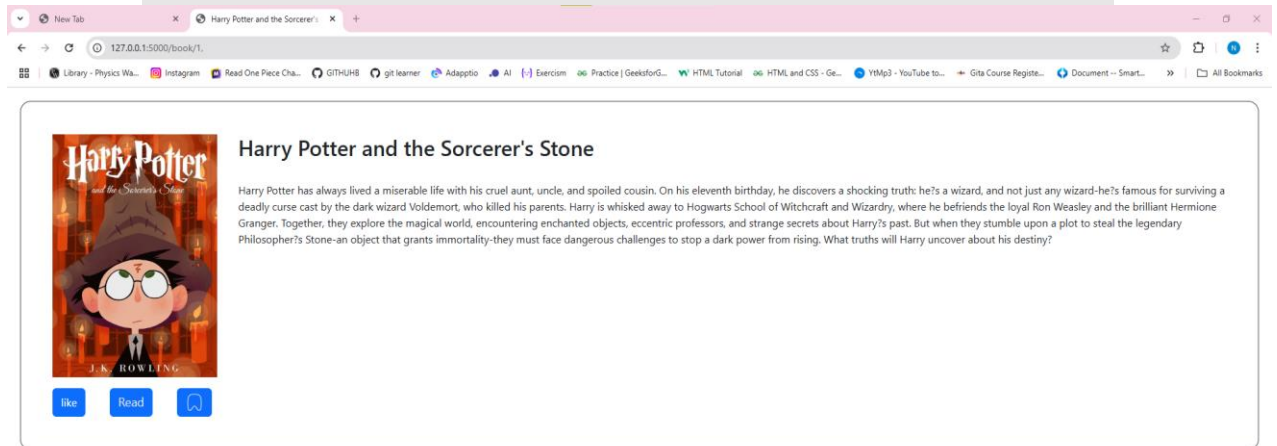
IMAGES OF THE SYSTEM



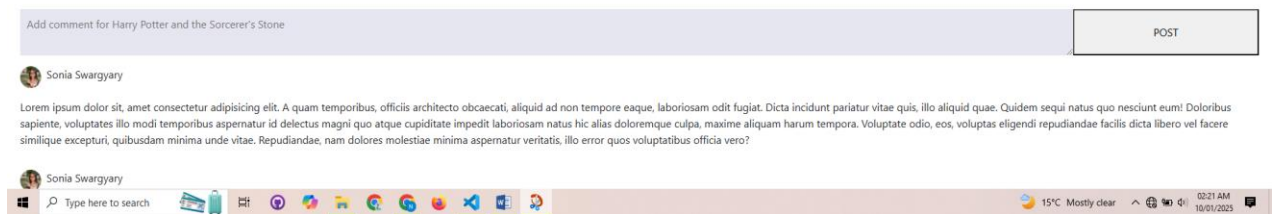
Home page:



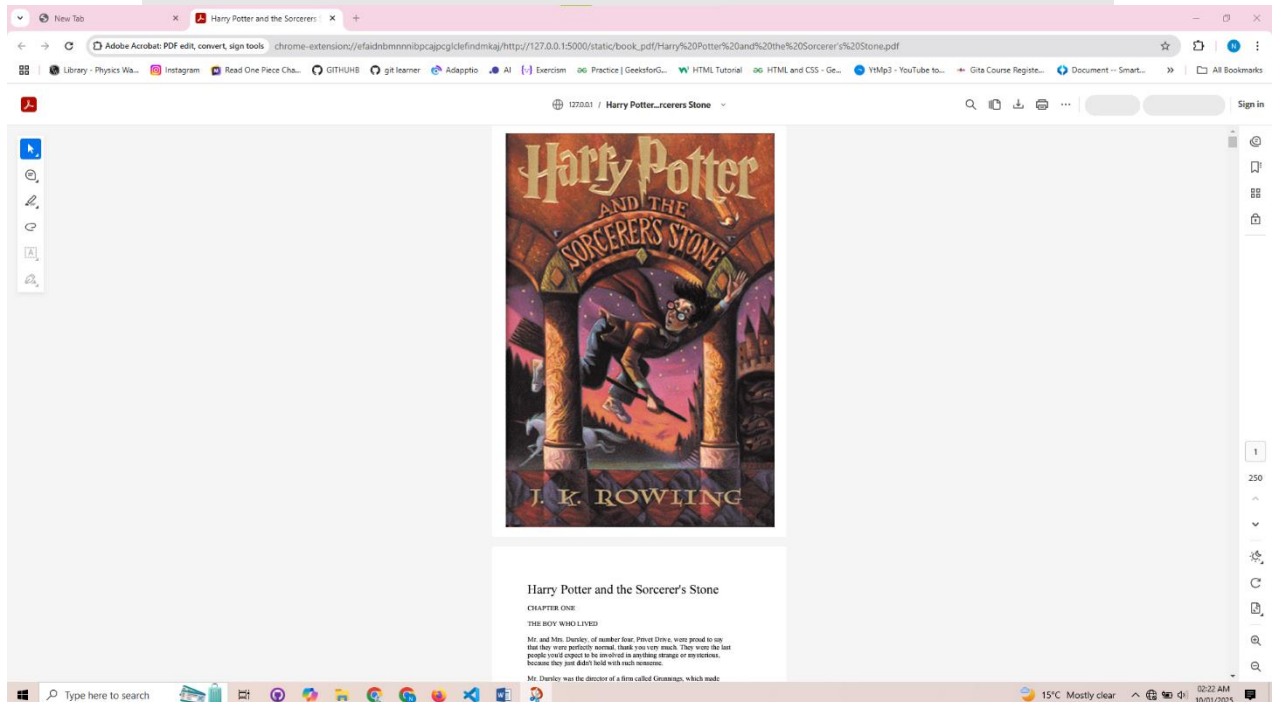
Book display page:



Review



Open book:



BIBILIOGRAPHY

1. https://www.w3schools.com/bootstrap/boostrap_ver.asp
2. <https://www.geeksforgeeks.org/flask-tutorial/>
3. <https://getbootstrap.com/docs/5.3/getting-started/contents/>
4. <https://getbootstrap.com/docs/5.3/getting-started/contents/>
5. Shradha Khapra Javascript full course 2024(youtube)
6. Code with Harry Flask tutorial (youtube)
7. Chatgpt
8. Microsoft Copilot
9. Smartdraw.com
10. Simplelearn css tutorial (yotube)

THANK YOU