**PUSTAK SAHAY: A SECOND-HAND BOOK SHARING PLATFORM**

A PROJECT REPORT

*Submitted by*

|  |  |  |  |
| --- | --- | --- | --- |
| **Roll Number** | **Registration Number** | **Student Code** | **Students Name** |
| **22010301121** | **22013002508** | **BWU/BCA/22/129** | **SWARAJ HALDER** |
| **22010301122** | **22013002510** | **BWU/BCA/22/131** | **DEBJYOTI MONDAL** |
| **22010301123** | **22013002511** | **BWU/BCA/22/132** | **ANTU MALLICK** |
| **22010301128** | **22013002517** | **BWU/BCA/22/138** | **SAMIRAN CHANAK** |
| **22010301130** | **22013002519** | **BWU/BCA/22/140** | **RAUL NASKAR** |
| **22010301131** | **22013002520** | **BWU/BCA/22/141** | **PRIYANKAR MONDAL** |

*in partial fulfillment for the award of the degree*

*of*

**BACHELOR OF COMPUTER APPLICATION (BCA)**

IN

COMPUTATIONAL SCIENCES

**BRAINWARE UNIVERSITY**

398, Ramkrishnapur Road, Barasat, North 24 Parganas, Kolkata - 700 125



MAY,2025

**PUSTAK SAHAY**

***Submitted by***

|  |  |  |  |
| --- | --- | --- | --- |
| **Roll Number** | **Registration Number** | **Student Code** | **Student Name** |
| **22010301121** | **22013002508** | **BWU/BCA/22/129** | **SWARAJ HALDER** |
| **22010301122** | **22013002510** | **BWU/BCA/22/131** | **DEBJYOTI MONDAL** |
| **22010301123** | **22013002511** | **BWU/BCA/22/132** | **ANTU MALLICK** |
| **22010301128** | **22013002517** | **BWU/BCA/22/138** | **SAMIRAN CHANAK** |
| **22010301130** | **22013002519** | **BWU/BCA/22/140** | **RAUL NASKAR** |
| **22010301131** | **22013002520** | **BWU/BCA/22/141** | **PRIYANKAR MONDAL** |

***in partial fulfillment for the award of the degree***

***of***

**BACHELOR OF COMPUTER APPLICATION (BCA)**

***in***

**COMPUTATIONAL SCIENCES**



**BRAINWARE UNIVERSITY**

*398, Ramkrishnapur Road, Barasat, North 24 Parganas, Kolkata - 700 125*



**BRAINWARE UNIVERSITY**

*398, Ramkrishnapur Road, Barasat, North 24 Parganas, Kolkata - 700 125*

**COMPUTATIONAL SCIENCES**

**BONAFIDE CERTIFICATE**

Certified that this project report **“PUSTAK SAHAY: A SECOND-HAND BOOK SHARING PLATFORM”** is the bonafide work of “**ANTU MALLICK, SWARAJ HALDER, DEBJYOTI MONDAL, SAMIRAN CHANAK, RAUL NASKAR, PRIYANKAR MONDAL**” who carried out the project work under my supervision.

**SIGNATURE**

**MR. KOUSIK DEY**

**SUPERVISOR**

ASSISTANT PROFESSOR

COMPUTATIONAL SCIENCES

**SIGNATURE**

**DR. JAYANTA AICH**

**HEAD OF THE DEPARTMENT**

COMPUTATIONAL SCIENCES

*398, Ramkrishnapur Road, Barasat, North 24 Parganas, Kolkata - 700 125*



**BRAINWARE UNIVERSITY**

*398, Ramkrishnapur Road, Barasat, North 24 Parganas, Kolkata - 700 125*

**COMPUTATIONAL SCIENCES**

**ACKNOWLEDGEMENT**

**Project Title:** PUSTAK SAHAY: A WEB PLATFORM FOR BUYING AND SELLING USED BOOKS WITH CART AND REWARD FEATURES  
**Project Group ID:** BCA22C008

We, the project members, would like to sincerely thank **Mr. Kousik Dey**, Assistant Professor in the Department of Computational Sciences at Brainware University, for his invaluable guidance and mentorship as our project guide. His consistent support, expert insights, and constructive feedback were instrumental in the successful completion of our work. We also extend our gratitude to **Mr. Sandip Chakraborty**, Assistant Professor and Project Coordinator, for his encouragement and coordination throughout the project, which helped us stay focused and organized during each phase.

We are also grateful to **Dr. Jayanta Aich**, Head of the Department of Computational Sciences, for facilitating the necessary resources and for his continued support during the course of this project.

Our appreciation also goes to all the faculty members and staff of the Department of Computational Sciences for their cooperation and helpful suggestions, which contributed to the steady progress and quality of our project.

We would like to express heartfelt thanks to our families, friends, and classmates for their unwavering support and motivation throughout this journey, which kept us focused and goal-oriented.

Lastly, we extend our deepest gratitude to Brainware University for giving us the platform to undertake this project as part of our academic curriculum and for nurturing our learning through such practical experiences.

**Project Members:**

1. BWU/BCA/22/129 – SWARAJ HALDER
2. BWU/BCA/22/131 – DEBJYOTI MONDAL
3. BWU/BCA/22/132 – ANTU MALLICK
4. BWU/BCA/22/138 – SAMIRAN CHANAK
5. BWU/BCA/22/140 – RAUL NASKAR
6. BWU/BCA/22/141 – PRIYANKAR MONDAL

**Date:** 21/06/2025  
**Department of Computational Sciences**  
**Brainware University**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Chapter** | **Title** | **Page No.** |
|  | **ABSTRACT** | **i** |
|  | **LIST OF TABLES** | **ii** |
|  | **LIST OF FIGURES** | **iii** |
| **1.** | **INTRODUCTION** | **1** |
| **2.** | **OBJECTIVES** | **2** |
| **3.** | **PLANNING** | **3** |
| **4.** | **REQUIREMENT ANALYSIS** | **5** |
| **5.** | **SYSTEM FLOW** | **7** |
| **6.** | **PROPOSED DESIGN** | **10** |
| **7.** | **EXPERIMENTAL RESULT** | **18** |
| **8.** | **FUTURE SCOPE** | **21** |
| **9.** | **CONCLUSION** | **22** |
|  | **REFERENCES** | **23** |

**ABSTRACT**

Pustak Sahay is an online book distribution & education support platform to provide affordable and accessible educational resources to students of all grades in a medium and vast array of academic institutions. Built on the MERN stack (MongoDB, Express. js, React. js, Node. js), the project seeks to bridge the gap between learners and learning materials by defining an effective user experience for discovering, purchasing and managing educational books and their associated rewards.

With a secure user authentication system, product listing and filtering capabilities, dynamic cart/checkout functionality and coupon-based reward system, the company offers an extensive variety of user account building and search activities, book display and purchase modes, fast and convenient checkout process, and special promotions. The Reward System offers a completely new option: users will be able to apply discount coupons on their account as if they meet the eligibility conditions: the discounts will be implemented through percentage basis or flat rate. These coupons will have predetermined expiry dates, usage limitations, or minimum order values.

Unsurprisingly, there is also included an admin panel which makes it easy to monitor products, users, orders and rewards through the back end. Administrators can monitor user activity, manage product inventories, track orders and implement an administration system for coupon distribution and validation. The whole system abides by a role based access model (using JWT authentication) to guarantee security and privacy.

Featuring responsive design and user experience in mind, the frontend is styled using Tailwind CSS, providing a solid and usable experience across all devices. Firebase services are also used (optionally) for real-time features and for hosting of images, which help with application performance and scalability.

Overall Pustak Sahay is the example of an educational project implementing Full-stack Web Development skills in a real world situation. The platform has designed and implemented most modern tools and usability friendly UI/UX that help to deliver efficiently books and also encouraging a continuous learning process through a reward based way. It is a reflection of the work on digital literacy, on education equity, and on innovative software engineering practices.

i

**LIST OF TABLES**

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl. No** | **Table No** | **Table Name** | **Page No** |
| **1** | **3.1** | Work Allocation among Members | **3** |
| **2** | **3.2** | Weekly Progress Report | **3** |
| **2** | **6.1** | Component Interactions | **14** |
| **3** | **6.2** | Overview of Collections | **14** |
| **4** | **8.1** | Test Environment | **20** |
| **5** | **8.2** | Functional Testing | **20** |

**LIST OF FIGURES**

ii

|  |  |  |  |
| --- | --- | --- | --- |
| **Sl. No.** | **Figure No** | **Figure Name** | **Page No** |
| **1** | **5.1** | **Context Level Diagram (Level 0 DFD)** | **6** |
| **2** | **5.2** | **Data Flow Diagram (Level 1)** | **6** |
| **3** | **5.3** | **Use Case Diagram** | **7** |
| **4** | **5.4** | **Book Upload Activity Diagram** | **7** |
| **5** | **5.5** | **Certificate Submission Activity Diagram** | **8** |
| **6** | **5.6** | **Flowchart for Sign Up Process** | **9** |
| **7** | **5.7** | **Flowchart for Login Process** | **10** |
| **8** | **5.8** | **Flowchart for Book upload and verification process** | **11** |
| **9** | **5.9** | **Flowchart for Certificate Upload and Verification Process** | **12** |
| **10** | **8.1** | **Sign up process diagram** | **21** |
| **11** | **8.2** | **Donate Books Form Page** | **22** |
| **12** | **8.3** | **Certificate Submission Form Page** | **22** |
| **13** | **8.4** | **Dashboard with Reward Allocation** | **22** |

iii

**Chapter 1: Introduction**

In today's fast-paced digital age, education has become increasingly resource-intensive, with students and learners constantly needing access to quality study materials. However, the affordability and accessibility of books remain a significant challenge, especially for students from modest backgrounds. To address this issue, Pustak Sahay has been conceptualized and developed as a user-friendly web platform dedicated to the buying and selling of used books.

Pustak Sahay not only encourages the reuse of academic resources but also promotes sustainable practices among students by reducing paper waste. The platform is built with a seamless user experience in mind, incorporating key features like authentication, cart management, order tracking, and a reward system to enhance user engagement.

**Objectives of the Project**

The objective of the project is to offer an affordable and accessible platform that allows students to buy used books, thereby reducing their academic expenses. It also empowers users to sell their old or unused books directly through the web application, creating a convenient peer-to-peer marketplace. A key aim is to promote the reuse of educational materials, encouraging sustainability and reducing paper waste. The platform incorporates a simple and efficient shopping cart system for adding items and placing orders with ease. To further encourage user engagement, a reward or coupon mechanism is integrated, allowing active users to benefit from discounts during checkout**.**

**Key Features**

The project includes several essential features to enhance user experience and system efficiency. User authentication ensures a secure login and registration process, protecting user data and access. The cart functionality allows users to manage their selected items, view total prices, and apply available reward coupons before placing an order. A dynamic rewards system enables users to earn and apply discount coupons based on their activity or eligibility. Additionally, admin functionalities are considered for managing book listings, users, and distributing rewards. The frontend is designed to be responsive and mobile-friendly using React and Tailwind CSS, ensuring a smooth experience across all devices.

**Target Audience**

The target audience for this platform includes school and college students who are in need of cost-effective alternatives to purchasing new books. It also caters to individuals who wish to resell their used books rather than letting them remain unused, thus contributing to a circular educational resource economy.

1

**Chapter 2: Objective**

The primary goal of the Pustak Sahay project is to develop a web-based platform dedicated to facilitating the buying and selling of used books among students and other readers. This initiative aims to enhance affordability, improve access to academic resources, and promote sustainable educational practices by encouraging the reuse of study materials.

**Main Objectives**

1. ***Enable Book Reusability***  
   To reduce the cost burden on students by allowing the purchase of second-hand books in good condition.
2. ***Facilitate Peer-to-Peer Book Selling***To provide an online space where users can upload and sell their used books to other interested buyers.
3. ***Build a Reward-Driven System***To motivate user engagement and loyalty by introducing a reward/coupon system applicable during checkout.
4. ***Ensure Smooth User Experience***To design a responsive, intuitive interface using modern frontend technologies like React and Tailwind CSS.
5. ***Implement Secure Authentication***To ensure secure login and user session handling via JSON Web Tokens (JWT) and proper route protection.
6. ***Provide Order & Cart Management***To allow users to add books to a cart, calculate total price, apply coupons, and place orders seamlessly.
7. ***Promote Educational Sustainability***To contribute to environmental sustainability by encouraging the reuse of physical books and reducing paper waste.

2

**Chapter 3: Planning**

**3.1) Work Allocation Among Team Members**

|  |  |
| --- | --- |
| Team Member Name | Responsibilities Allocated |
| Member 1 | **Frontend development using React, routing setup, reusable components** |
| Member 2 | **UI/UX design, styling with Tailwind CSS, responsive layout** |
| Member 3 | **Backend setup using Node.js & Express, JWT authentication and authorization** |
| Member 4 | **MongoDB schema design, certificate upload logic, database integration** |
| Member 5 | **Admin panel, certificate review functionality, incentive allotment logic** |
| Member 6 | **Testing, debugging, deployment, documentation, weekly tracking** |

Table 3.1: Work Allocation among Members

**3.2) Weekly Progress Report:**

|  |  |  |
| --- | --- | --- |
| **Week** | **Tasks Planned** | **Tasks Completed** |
| **Week 1** | **Team formation, idea discussion, role distribution** | **Project topic finalized, team responsibilities allocated** |
| **Week 2** | **Project planning, tech stack finalization, Git repo setup** | **Repository created, base structure discussed** |
| **Week 3** | **MongoDB schema design, basic Express server setup** | **User and Certificate schema completed** |
| **Week 4** | **React project setup, frontend routing & navbar creation** | **Routes working, basic frontend structure created** |
| **Week 5** | **Authentication system implementation (signup/login), JWT integration** | **Backend auth APIs completed, frontend login integrated** |
| **Week 6** | **Certificate form creation and file upload logic** | **Certificate upload feature completed** |
| **Week 7** | **Admin panel setup and certificate review mechanism** | **Admin dashboard built, review and accept/reject logic integrated** |
| **Week 8** | **Incentive calculation and reward schema logic implementation** | **Reward model done, reward linkage with certificate validated** |
| **Week 9** | **UI finalization, testing across flows, minor bug fixes** | **UI improved, errors handled, edge cases tested** |
| **Week 10** | **Full documentation writing, demo preparation, final deployment** | **Documentation completed, project hosted, demo prepared** |

Table 3.2: Weekly Progress Report

3

**Chapter 4: Requirement Analysis**

Requirement analysis is a critical phase in the software development process. It helps identify what the system is expected to do and the conditions it must satisfy to be successful. For Pustak Sahay, a platform for buying and selling used books, requirements were gathered with a focus on user convenience, security, and functionality.

**4.1 Functional Requirements**

**User Authentication**  
The system provides secure user authentication where individuals can register and log in using an email and password. It also supports role-based access, distinguishing between regular users and admin roles to ensure proper authorization for different actions.

**Book Listings**  
Users can upload details and images of the books they wish to sell. The platform allows categorization of books based on subjects, academic classes, or streams, enabling easier navigation and search functionality for buyers.

**Cart and Checkout**  
Users can add or remove books from their shopping cart. A dedicated checkout page is provided that summarizes the selected items, displays the total price, and simulates the payment process to complete the order.

**Reward Coupons**  
To incentivize user activity, the platform allows the use of reward points or discount coupons during checkout. These coupons can offer fixed-value or percentage-based discounts based on user eligibility.

**Admin Controls**  
In the current version or future updates, admin-specific features will allow platform administrators to manage users, moderate book listings, and handle reward distribution to maintain system integrity and quality.

**4.2 Non-Functional Requirements**

**Responsiveness**  
The website is designed to function seamlessly across different devices, including desktops, tablets, and smartphones. A mobile-first responsive approach ensures accessibility on all screen sizes.

**Performance**  
To deliver a smooth user experience, the system is optimized for fast loading times. Images and data are processed efficiently to reduce latency and enhance overall performance.

4

**Security**  
Strong security mechanisms are in place, including password encryption using bcrypt and token-based authentication with JSON Web Tokens (JWT). Protected routes ensure that sensitive operations are only accessible to authenticated users.

**Scalability**  
The backend is built using RESTful API principles, making it scalable and capable of handling growing numbers of users, listings, and transactions without a drop in performance.

**Usability**  
The platform offers a clean and intuitive user interface designed for ease of use. It is accessible to users of all age groups, including students and their parents, minimizing the learning curve.

**4.3 Hardware and Software Requirements**

**Hardware Requirements**  
To develop or access the system, a standard computer with a minimum of 4 GB RAM and a stable internet connection is sufficient. These basic requirements ensure smooth functioning during development and usage.

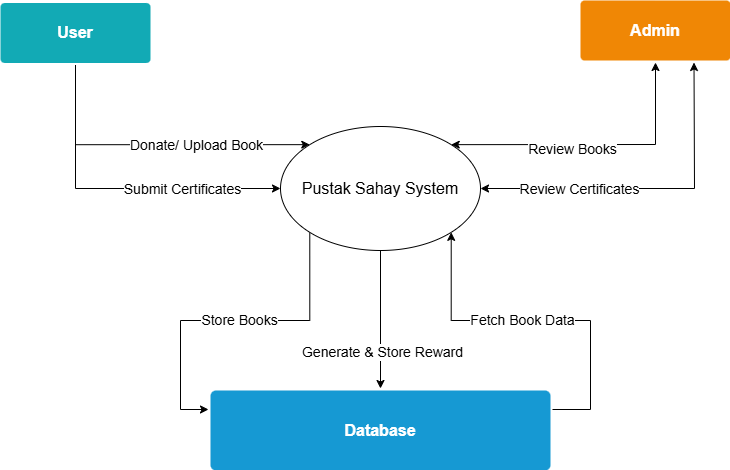
**Software Requirements**  
The frontend of the system is built using **React.js** for the user interface, **Tailwind CSS** for styling, and **Redux Toolkit** for managing global state. On the backend, **Node.js** and **Express.js** are used to create the server-side logic, with **MongoDB** and **Mongoose** handling data storage and retrieval.

**Other Tools**  
Version control is managed using **Git** and **GitHub**, while **Postman** is utilized for API testing and debugging. **Cloudinary** is integrated for secure and optimized image hosting, ensuring smooth media handling across the application.

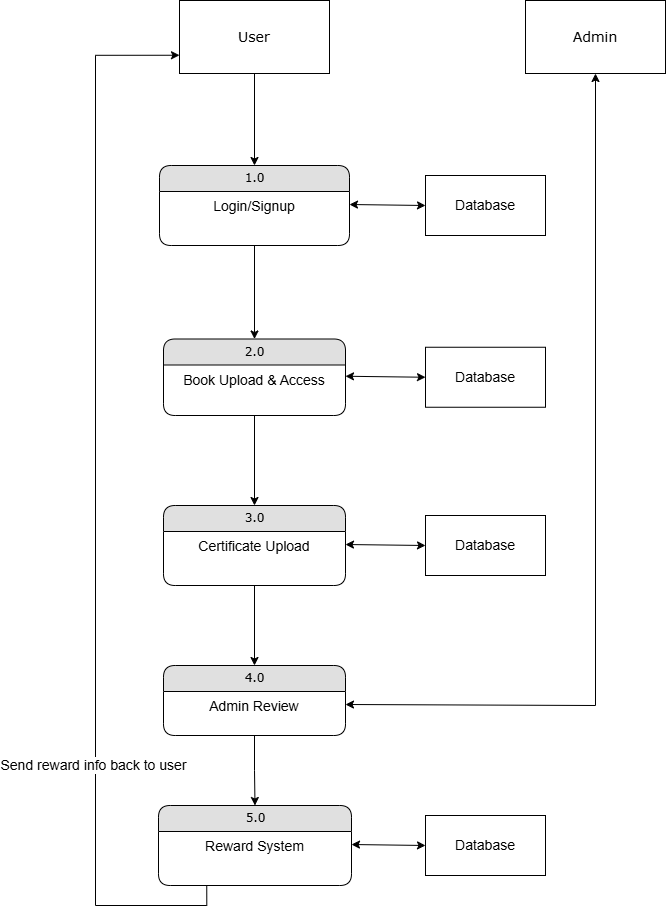
**Chapter 5: System Flow**

5

1. **Context Level Diagram:**

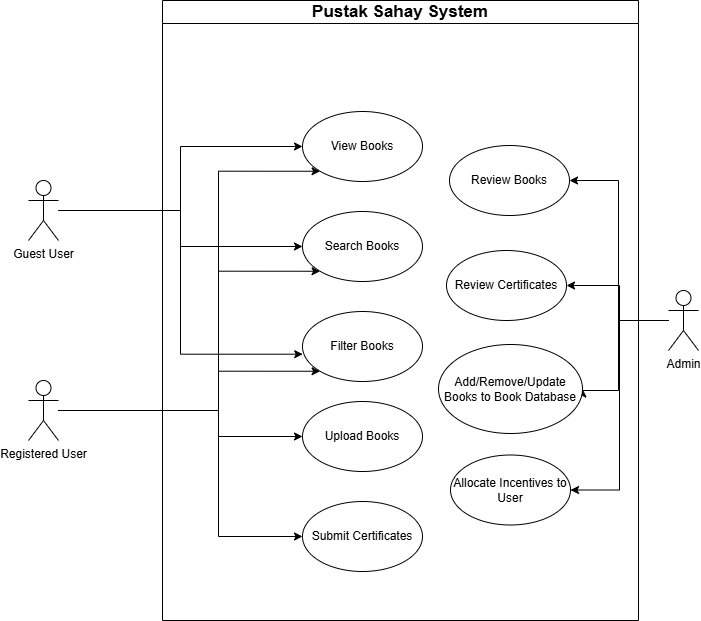
The main Context Level Diagram (CLD) mentioned below

**Fig 5.1: Context Level Diagram (Level 0 DFD)**

1. **Data Flow Diagrams**

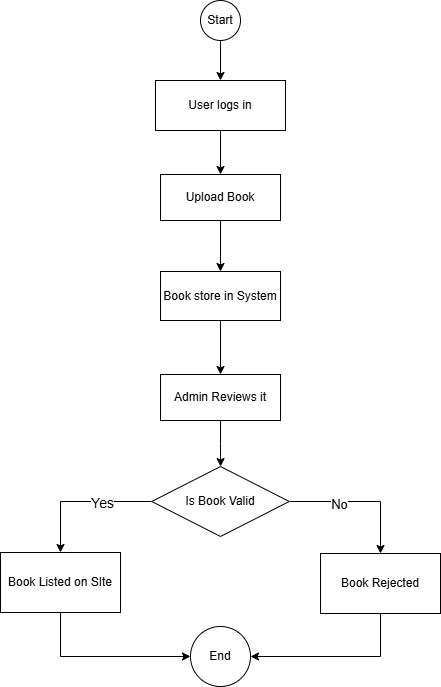
6

**Fig 5.2: Data Flow Diagram (Level 1)**

1. **Use Case Diagram**

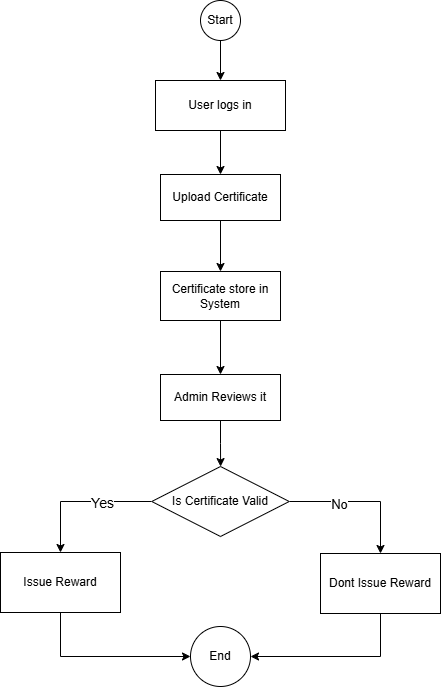
**Fig 5.3: Use Case Diagram**

1. **Activity Diagram**

**i. Book Upload:**

7

**Fig 5.4: Book Upload Activity Diagram**

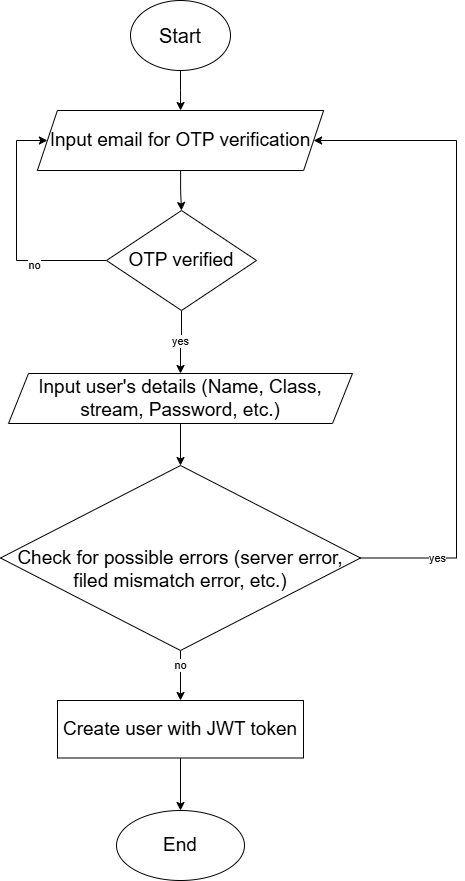
**ii. Certificate Upload:**

**Fig 5.5: Certificate Submission Activity Diagram**

8

1. **Flowcharts:**

Some important system flowcharts are discussed below.

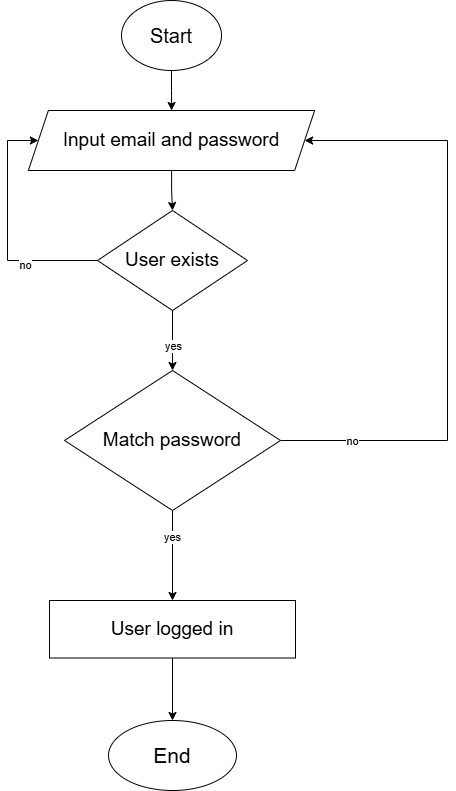
***5.1) First time User creation / Sign Up flow***:

**Fig 5.6: Flowchart for Sign Up process**

The user registration flow in the Pustak Sahay platform begins when the user initiates the signup process by entering their email address for OTP (One-Time Password) verification. Once the OTP is sent and successfully verified, the user proceeds to provide additional details such as name, class, stream, and password. The system then performs validation checks to detect any potential errors, including server issues, input mismatches, or missing fields. If any errors are found, the process halts, and the user is prompted to correct them. Upon successful validation, a new user account is created, and a secure session is initiated using a JWT (JSON Web Token). This completes the registration process.

9

***5.2) User login:***

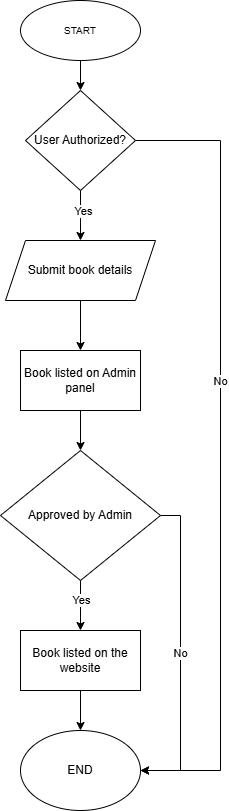
****

**Fig 5.7: Flowchart for Login process**

The login process in the Pustak Sahay platform begins with the user entering their registered email and password. The system first checks whether a user with the provided email exists in the database. If the user does not exist, the login process is terminated. If the user is found, the entered password is then validated against the stored password hash. If the passwords do not match, access is denied. Upon successful verification of both email and password, the user is logged into the system, and a secure session is initiated, allowing access to authorized features.

10

***5.3) Book Upload by User and registered in website flow:***

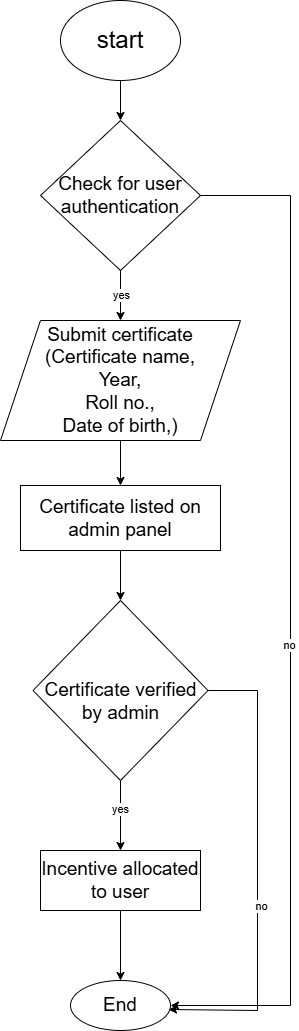


**Fig 5.8: Flowchart for Book upload and verification process**

The book listing process on the Pustak Sahay platform begins with verifying whether the user is authorized to perform the action. If authorized, the user proceeds to submit the book details, including necessary information and images. The submitted book is then listed on the admin panel for review. The admin reviews the listing and either approves or rejects it. If approved, the book becomes publicly visible and listed on the website. If rejected at any stage, the listing does not proceed further, ensuring quality control and adherence to platform guidelines.

11

***5.4) Certificate Upload by User and incentive allocation for that user flow:***

****

**Fig 5.9: Flowchart for Certificate Upload and Verification Process**

The certificate submission process starts with user authentication. Once authenticated, the user submits certificate details such as the certificate name, year, roll number, and date of birth. The submitted certificate is then listed on the admin panel for review. The admin verifies the certificate, and if it is valid, an incentive is allocated to the user. If the certificate fails verification or any previous step, the process is terminated to maintain integrity and prevent misuse.

12

**Chapter 6: Proposed Design**

**6.1 Overview**

The proposed system is a web-based platform designed to manage certificate submissions and automate reward allotment for students or users based on the validation of their achievements. The system is built using the MERN (MongoDB, Express.js, React, Node.js) stack, providing a responsive frontend interface, a scalable backend server, and a flexible NoSQL database structure.

This chapter outlines the architectural design, module-level breakdown, database schema, and workflow involved in the functioning of the system. The objective is to present a clear technical roadmap of how different components interact, how data flows between them, and how the core functionalities such as authentication, certificate handling, and reward management are implemented.

The design focuses on:

Efficient user experience through a structured dashboard, Secure authentication and role-based access.

Seamless uploading and reviewing of certificates, Automated reward generation upon approval.

Admin-level oversight and action capabilities, Diagrams and tables have been included to support clarity in the understanding of system components, their roles, and their relationships.

**6.2 System Architecture**

The architecture of the proposed web application follows a **client-server model** using the MERN stack, ensuring modularity, scalability, and maintainability. It consists of three main layers:

1. **Frontend Layer (Client-Side):** Built with **React.js**, the frontend provides a responsive and intuitive user interface. It handles routing, state management, and data display through protected routes and dynamic components based on user roles (student, admin).
2. **Backend Layer (Server-Side API)**: Developed using **Node.js and Express.js**, the backend handles all business logic, API endpoints, authentication, and authorization. It processes incoming requests from the client, interacts with the database, and returns appropriate responses.
3. **Database Layer**: **MongoDB** serves as the NoSQL database where user data, certificate submissions, and reward records are stored. Mongoose is used to define schemas and manage relationships.

13

**Component Interactions**

| **Component** | **Technology** | **Role** |
| --- | --- | --- |
| Frontend | React.js | Handles UI, API calls, user routing, form submission |
| Backend API | Express.js | Auth, role verification, file handling, reward generation logic |
| Authentication | JWT | Token-based access, protected routes |
| File Storage | Multer/Cloudinary | Stores uploaded certificate files |
| Database | MongoDB | Stores users, certificates, rewards, and metadata |

Table 6.1: Component Breakdown

**Core Functional Modules**

1. **User Module**
   * Signup, login, token management.
   * Dashboard view with user details and certificates.
2. **Certificate Module**
   * File upload (Multer + Cloudinary).
   * Admin view and review flow.
   * Certificate approval/rejection.
3. **Reward Module**
   * Reward schema generation based on certificate criteria.
   * Allotment logic and expiry handling.
4. **Admin Module**
   * Certificate review dashboard.
   * Manual reward assignment and usage tracking.

**6.3 Database Design**

The application uses **MongoDB** as the primary database, offering flexibility through a document-based schema. The design supports scalable and modular data handling for users, certificates, and rewards.

**6.3.1 Overview of Collections**

|  |  |
| --- | --- |
| Collection | Description |
| users | Stores user account information including role and profile |
| certificates | Stores uploaded certificate details and associated metadata |
| rewards | Contains reward schemes assigned to users based on criteria |
| scholarships | Contains ongoing or upcoming scholarships data for that region |

14

Table 6.2: Collections in Project

**6.3.2 Collection Schemas**

Here are some important schema designs used in our project

***i) User Schema***

import mongoose from "mongoose";

import CartItemsSchema from "./CartItems.js";

import RewardSchema from "./Schemas/RewardSchema.js";

const userSchema = new mongoose.Schema(

{

name: { type: String, required: true },

email: {

type: String,

unique: true,

required: true,

match: [/\S+@\S+\.\S+/, "Please use a valid email address"],

},

isStudent: { type: Boolean, default: false },

educationLevel: {

type: String,

enum: ["Class 9 and below", "Class 10", "Class 11", "Class 12", "Under Graduate", "Post Graduate"],

},

stream: {

type: String,

enum: ["Science", "Arts", "Commerce", "Medical", "Engineering", "Law", "Others"],

},

phone: { type: String, unique: true },

password: { type: String },

role: { type: String, enum: ["user", "admin"], default: "user" },

cartItems: [CartItemsSchema],

rewards: [RewardSchema],

orders: [{ type: mongoose.Schema.Types.ObjectId, ref: "Book" }],

},

{ timestamps: true }

);

userSchema.index({ email: 1 });

const User = mongoose.model("User", userSchema);

export default User;

The user schema is designed with essential fields such as name, email, education level, password, and other relevant attributes to ensure a well-structured user model. The **email field is indexed** to optimize search performance and allow for faster retrieval of user records based on email queries.

15

***ii) Book Schema:***

import mongoose from "mongoose";

const bookSchema = new mongoose.Schema(

{

title: { type: String, required: true, index: true },

author: { type: String, required: true },

publisher: { type: String, required: true },

category: {

type: String,

enum: [

"Engineering", "Medical", "Finance", "Science", "School",

"Competitive Exams", "Literature", "Reference", "Self-Help", "Fiction"

],

required: true,

index: true,

},

subCategory: { type: String },

condition: {

type: String,

enum: ["New", "Like New", "Used"],

default: "Used",

},

originalPrice: { type: Number, required: true },

sellingPrice: { type: Number, required: true },

discount: { type: Number, default: 0 },

description: { type: String, required: true },

imageUrl: { type: String, required: true },

uploadedBy: { type: String, required: true },

approvedBy: { type: String, required: true },

binding: {

type: String,

enum: ["Paperback", "Hardcover", "Board Book", "Leather Bound"],

},

status: {

type: String,

enum: ["Available", "Sold"],

default: "Available",

},

},

{ timestamps: true }

);

bookSchema.index({ title: "text" });

bookSchema.index({ author: "text" });

bookSchema.index({ category: 1, subCategory: 1 });

export default mongoose.model("Book", bookSchema);

The book schema contains title, author, publisher, category, condition, original price., image URL and other essential fields. The **title, author, category and sub category combinedly** **indexed** to make the searches for books faster.

16

***iii) Certificate Schema:***

import mongoose from "mongoose";

const certificateSubmissionSchema = new mongoose.Schema(

13

  {

    name: { type: String, required: true },

    phone: { type: String, required: true, unique: true },

    dob: { type: Date },

    institution: { type: String, required: true },

    certificateName: { type: String, required: true },

rollno: {type: String, required: true},

    issuedBy: { type: String, required: true },

    issueYear: { type: String, required: true },

    certificateUrl: { type: String, required: true },

    uploadedBy: {

      type: mongoose.Schema.Types.ObjectId,

      ref: "User",

      required: true,

    },

  },

  { timestamps: true }

);

const Certificate = mongoose.model("Certificate", certificateSubmissionSchema);

export default Certificate;

The certificate Schema is well structured to store the certificate details uploaded by User. It is mainly use to assign incentives or rewards to that user.

***iv) Reward Schema:***

import mongoose from "mongoose";

const RewardSchema = new mongoose.Schema({

name: String,

isPercentage: Boolean,

value: String,

terms: [String],

minimumPrice: Number,

usageLimit: Number,

durationInMonths: Number,

createdAt: Date,

expiresAt: Date

});

export default RewardSchema

The Reward schema is integrated into the User schema through a rewards attribute, establishing a structured relationship between users and their associated incentives.

17

**6.4) Main Code functions**

The core functions of important code blocks are illustrated below

***6.4.1) Sign Up:***

try {

    const existingUser = await User.findOne({ email });

    if (existingUser) {

      return res.status(400).json({ message: "Email already registered!" });

    }

    const salt = await bcrypt.genSalt(10);

    const hash = await bcrypt.hash(password, salt);

    const user = await User.create({name, email, password: hash, isStudent, educationLevel, stream, phone, role: role || undefined});

    if (!user) {

      return res.status(500).json({ message: "Failed to create user" });

    }

    const token = createToken(user.\_id);

    const sendUserDetails = {

      name: user.name,

      email: user.email,

      role: user.role,

      cartItemCount: user.cartItems,

      isStudent: user.isStudent,

      educationLevel: user.educationLevel,

      stream: user.stream,

      token,

    };

    return res.status(201).json({

      sendUserDetails,

      message: "Signed Up successfully",

    });

  } catch (error) {

    return res.status(400).json({ message: error.message });

  }

***6.4.2) Sending OTP:***

const transporter = nodemailer.createTransport({

  service: "gmail",

  secure: true,

  port: 465,

  auth: {

    user: process.env.EMAIL\_ADDRESS,

    pass: process.env.EMAIL\_PASSWORD,

  },

});

const sendOtpEmail = async (to, otp) => {

  const mailOptions = {

    from: `"PUSTAK SAHAY" <${process.env.EMAIL\_ADDRESS}>`,

    to: to,

    subject: "OTP Verification - Pustak Sahay",

    text: `Your One-Time Password (OTP) is: ${otp}

  If you did not request this OTP, please ignore this email.

18

  Best regards,

  Team Pustak Sahay`,

  };

  try {

    const info = await transporter.sendMail(mailOptions);

    return true;

  } catch (error) {

    console.error("Error sending OTP:", error);

    return false;

  }

};

***6.4.3) Searching Books:***

const searchBooks = async (req, res) => {

  try {

    const { type, query } = req.query;

    if (!query) {

      return res.status(400).json({ message: "Query is required" });

    }

    const searchWords = query

      .split(" ")

      .map((word) => word.trim())

      .filter((word) => word);

    let searchQuery = {};

    if (type === "title") {

      searchQuery = { $text: { $search: query } };

    } else if (type === "author") {

      searchQuery = { author: { $regex: query, $options: "i" } };

    } else if (type === "category") {

      // Create regex queries for each search word

      const regexQueries = searchWords.map((word) => ({

        $or: [

          { category: { $regex: word, $options: "i" } },

          { subCategory: { $regex: word, $options: "i" } },

        ],

      }));

      searchQuery = { $and: regexQueries };

    } else {

      return res.status(400).json({ message: "Invalid search type" });

    }

    const books = await Book.find(searchQuery).lean();

    res.status(200).json(books);

  } catch (error) {

    res.status(500).json({ message: error.message });

  }

};

19

**Chapter 7: Experimental Result**

**7.1 Objective**

The goal of this section is to evaluate the performance, usability, and functionality of the implemented system. Various experiments were conducted to test the core modules of the website, particularly focusing on certificate submission and verification, reward allotment and tracking, authentication and authorization flow, and user profile management.

**7.2 Test Environment**

|  |  |
| --- | --- |
| Parameter | Value |
| Frontend Framework | React.js (with Tailwind CSS) |
| Backend Framework | Node.js with Express |
| Database | MongoDB (Cloud-hosted) |
| Authentication Method | JWT (JSON Web Token) |
| OS | Windows/Linux |
| Browser | Chrome, Firefox |

Table 8.1: Test Environment

**7.3 Functional Testing**

|  |  |  |
| --- | --- | --- |
| Functionality | Expected Result | Status |
| User Registration/Login | User should be able to register and login | Pass |
| Certificate Upload | Certificate should be submitted and saved | Pass |
| Certificate Review | Reviewer should see and access uploaded entries | Pass |
| Reward Allotment | Admin can assign reward to a verified certificate | Pass |
| Token-based Protected Routes | Unauthorized users get 401 | Pass |
| Payment Gateway | Users can directly make payment in website | In Progress |

Table 8.2: Functional Testing

**7.4 Sample Outputs**

Some important responses for http requests are mentioned

**i) Login Response:**

{

"message": "Login successful",

"user": {

"\_id": "6642d3fca948b9a123456789",

"name": "Shrabani Halder",

"email": "shrabanihalder596@gmail.com",

"educationLevel": "Under Graduate",

"stream": "Engineering",

"isStudent": true,

"role": "user",

"cartItemCount": []

20

},

"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6..."

}

**ii) Certificate Upload Response:**

{

"message": "Certificate uploaded successfully",

"certificate": {

"name": "Shrabani Halder",

"certificateName": "Web Development Workshop",

"issuedBy": "ABC Institute",

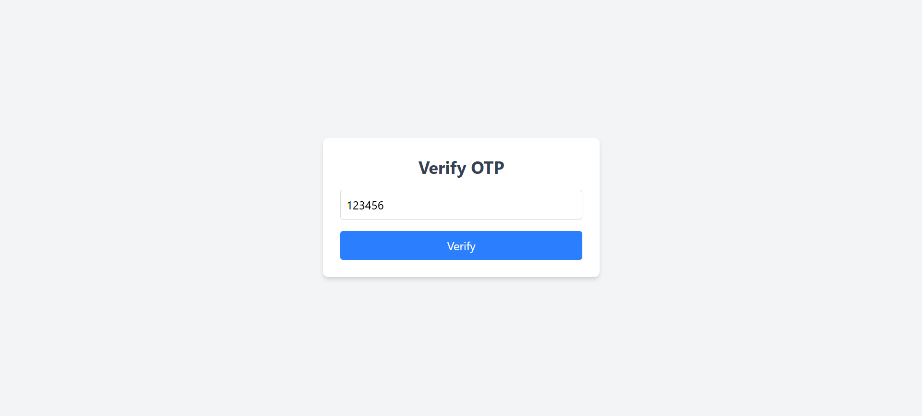
"createdAt": "2025-05-07T14:23:00Z"

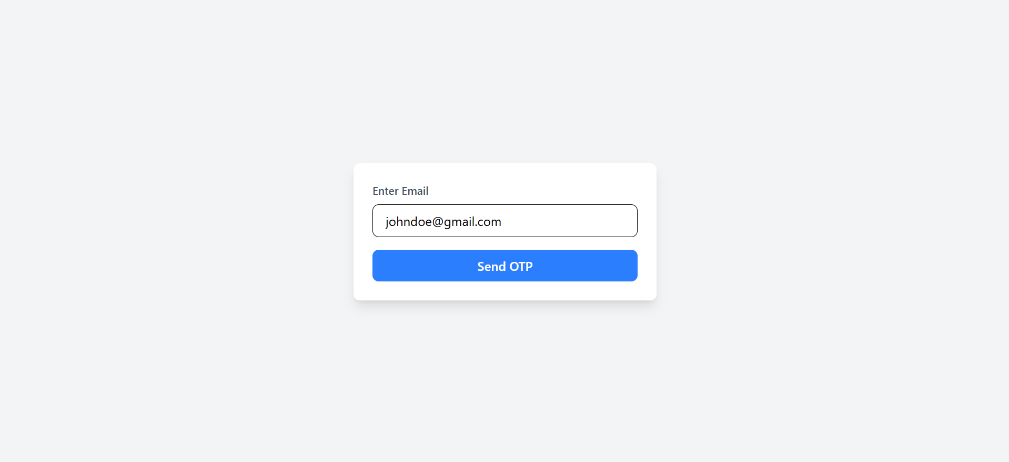
}

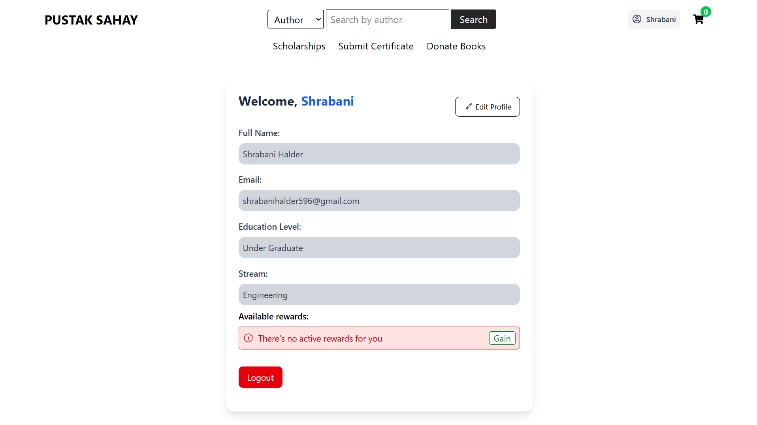
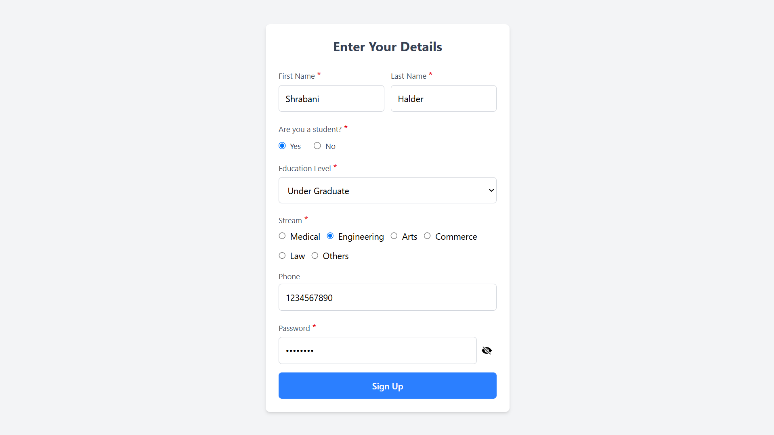
}

**7.5 Visual Evidence**

Here are some diagrams to illustrate some main function flows

* ***Sign up process***:

******



**3rd Step**: Enter basic details of the user

**4th Step**: After successful Sign Up redirected to Dashboard

**2nd Step**: Enter the OTP sent to the Email

**1st Step**: Enter Email to get OTP

Figure 8.1: Sign up process diagram

21

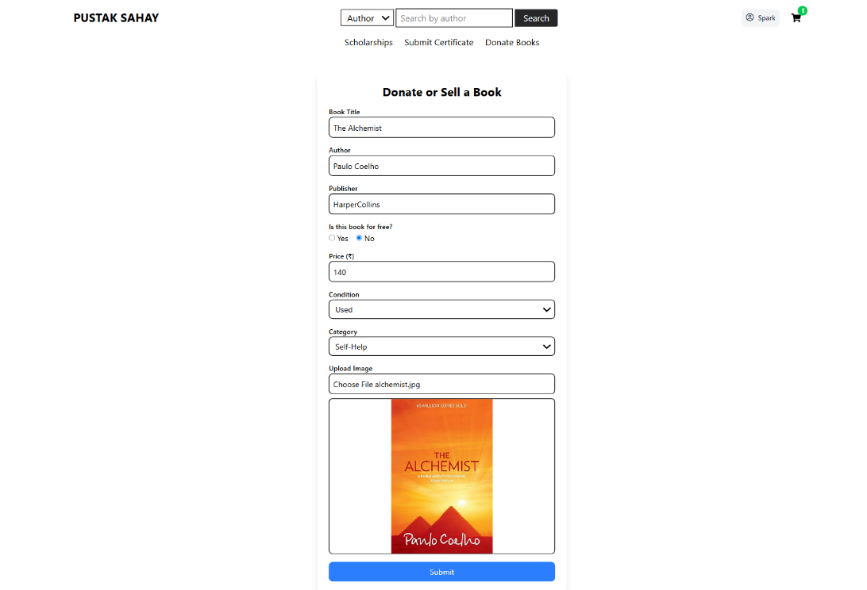
* ***Donate books page***

Figure 8.2: Donate Books Form Page

* ***Certificate Submission page:***

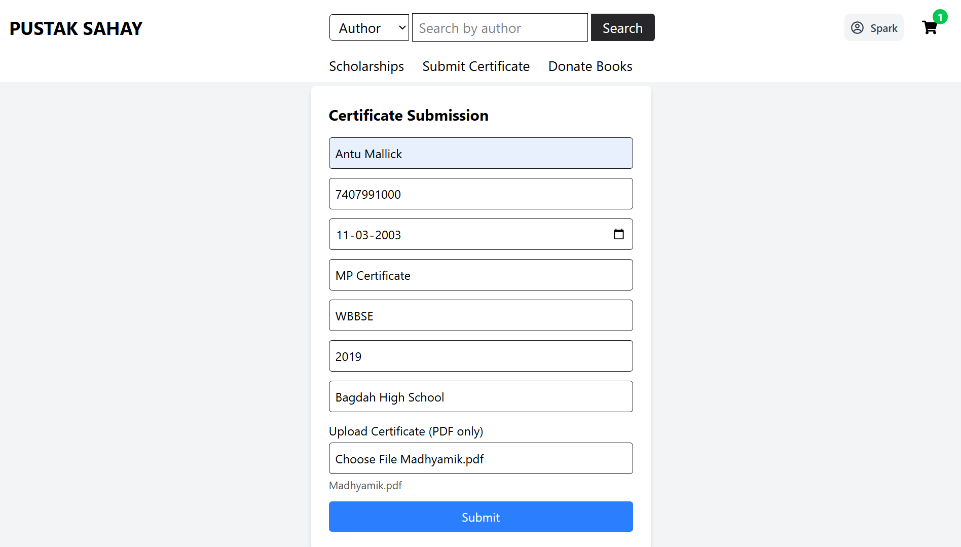


Figure 8.3: Certificate Submission Form Page

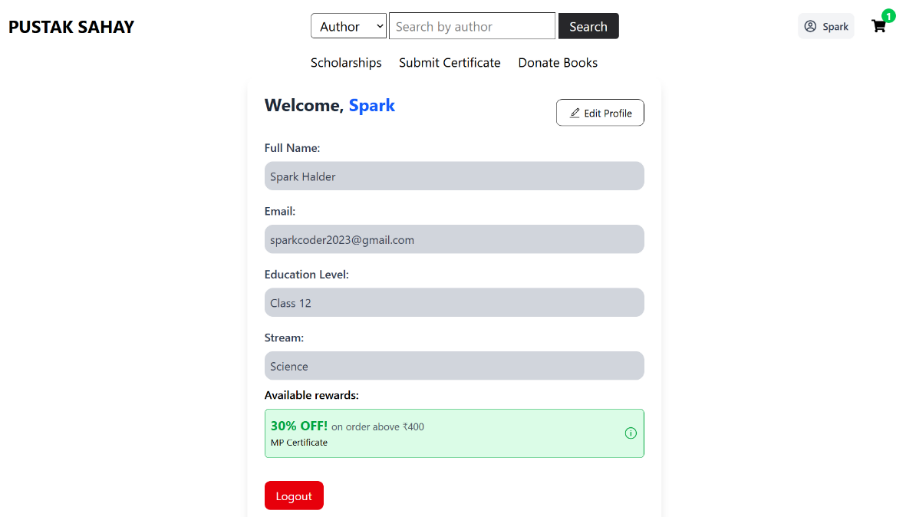
* ***User dashboard with listed Reward***

Figure 8.4: User Dashboard with Reward Allocation

22

**Chapter 8: Future Scope**

The current system lays the foundation for a robust certificate management and reward distribution platform targeted towards students and academic institutions. However, there are several areas where the system can be enhanced in the future to increase scalability, efficiency, and user experience:

**1. Admin Analytics Dashboard**Introducing a comprehensive analytics panel for administrators will provide deeper insights into certificate submission trends, reward usage frequency, and user engagement over time. This will help institutions take data-driven decisions.

**2. Reward Auto-Generation**Future iterations can include intelligent reward generation based on certificate type and issuing authority, user’s past activity and engagement, and predefined reward rules and academic performance**.**

**3. User Notifications**Implementing email or in-app notifications for certificate review status updates, reward allotment alerts, and reward expiry reminders will enhance user experience and engagement.

**4. Mobile App Integration**A mobile app version can be developed using React Native or Flutter, making the platform more accessible for students on-the-go, especially for uploading certificates through the camera.

**5. Blockchain-Based Verification**Future enhancements may include blockchain integration to ensure authenticity and tamper-proof storage of certificates.

**6. Multi-Language Support**To reach a broader user base across different educational regions, support for multiple languages will make the platform more inclusive.

**7. Institution-Level Dashboards**Allowing institutions to onboard themselves, manage their users, and issue institution-specific rewards would evolve the system into a SaaS platform.

**8. Payment Gateway Integration**To monetize rewards or introduce a premium model, secure integration with payment gateways such as Razorpay or Stripe can enable users to purchase additional incentives, donate to educational causes, or upgrade to premium features.

23

**Chapter 9: Conclusion**

The certificate management and reward allotment system developed serves as an organized and automated platform for granting the near-instant award of student achievement recognition. In light of user-friendly frontends, secure authentication processes, and flexible reward methodologies, the system overcomes impediments faced at both the administration and student levels in educational institutions.

This tool, in bringing about digitalization into the certificate review process, along with an incentive-motivational system, helps improve efficiency while instilling motivation in students for increased participation and excellence. Modern technology assistance with React, Node.js, and MongoDB depicts a system that is ready to scale, maintain, and remain highly responsive across devices.

Though the existing solution implants a sturdy base on which enhancements could be brought further into the fold—including analytics, automation, mobile support, and blockchain verifications—the project wherever it stands, exemplifies a solution to real-world problems in the educational sector through considerate system design and modern development methodologies.

24

**References**

1. **Cloudinary** (n.d.) *Cloud-based image and video management*. Available at: <https://cloudinary.com>
2. **MDN Web Docs** (n.d.) *Comprehensive web developer reference*. Available at: <https://developer.mozilla.org>
3. **MongoDB** (n.d.) *NoSQL database program*. Available at: <https://www.mongodb.com>
4. **Node.js** (n.d.) *JavaScript runtime built on Chrome's V8 engine*. Available at: <https://nodejs.org>
5. **React** (n.d.) *A JavaScript library for building user interfaces*. Available at: <https://reactjs.org>
6. **Redux Toolkit** (n.d.) *Standard way to write Redux logic*. Available at: <https://redux-toolkit.js.org>
7. **Tailwind CSS** (n.d.) *A utility-first CSS framework*. Available at: <https://tailwindcss.com>

25