CHAPTER IV: IMPLEMENTATION AND TESTING

4.1 Implementation and Overview

The Waterfall model is applied to our project as it follows a structured approach where development progresses through distinct and sequential phases. These phases include requirements analysis, system design, implementation, testing, deployment, and maintenance. Each phase must be fully completed before moving on to the next, ensuring a systematic and organized development process. This model emphasizes detailed documentation and a clear progression, making it easier to manage, track progress, and ensure deliverables are well-defined. It is particularly suitable for projects with fixed requirements, schedules, and budgets, where a thorough plan and predictable outcomes are essential. Although the Waterfall model is well-suited for smaller or straightforward projects due to its structured approach and emphasis on thorough documentation, it is less adaptable to changes and iterative feedback, which can be a limitation for more complex or dynamic project environments.

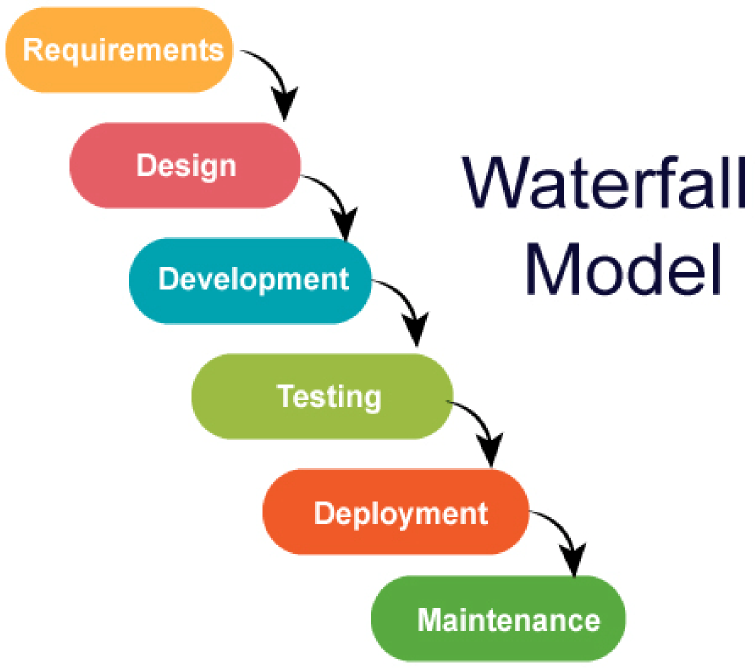

Figure 4.1: Waterfall Model

4.2 Tools Used

4.2.1 System Design Tools

- **Draw.io**

  Draw.io is a free, web-based diagramming tool that allows users to create flowcharts, UML diagrams, network diagrams, and other types of visual representations for planning and documenting systems. It offers a wide range of shapes, templates, and collaboration features to create professional diagrams.

  In our CMS, draw.io helps to visualize the architecture, data flow, and overall system design. It is used to map out the user roles, permissions, and the interaction between different entities.

- **Figma**

  Draw.io is a cloud-based design and prototyping that allows teams to collaborate in real-time on UI/UX design projects. It enables designers to create interactive prototypes, wireframes, and visual designs that can be shared and edited by team members simultaneously.

  In our CMS, Figma is used to design the user interface, create wireframes, and prototype user flows. It is used in our project to have a clear visual about the UI of our system and converting the design into the final product.

- **Canva**

  Canva is an online graphic design tool that allows users to create professional-quality designs easily with its drag-and-drop interface. It offers for a wide range of projects, including social media posts, presentations, posters, and more, making it accessible for both designers and non-designers.

  In our CMS, Canva is used to create logo and advertisement banner. Canva has helped to make visually appealing design banners.

4.2.1 Collaboration Tools

- **Git**

  Git is a distributed version control system that allows developers to track changes in code, collaborate with team members, and manage different versions of a project. It enables efficient branching, merging and rollback of changes.

In our CMS, it is used to manage and version control the codebase. It helped us developers to work on different features or bug fixes in isolated branches, and later merge them into main branch without disrupting the main code.

- **GitHub**

  GitHub is a web-based platform built on Git that provides hosting for version-controlled repositories, along with tools for collaboration, code review, and project management.

  In our CMS, GitHub serves as the central hub for hosting the codebase and managing collaborative among team members. It has helped us to track the bugs and feature requests through issues, and document the project using README files or wikis.

4.2.1 Front End Tools

- **HTML 5**

  HTML5 is the latest version of the language used to create web pages. It improves web development with better structure, multimedia support, interactivity, and offline features, making content management more efficient and accessible.

  In our content management system, HTML5 ensures the proper structure of the webpage, content organization, and seamless multimedia integration while improving the structure and scalability.

- **Tailwind CSS**

  Tailwind CSS is a CSS framework that makes building modern, responsive websites faster and easier. Instead of writing custom CSS, you use it pre-made utility classes directly in your HTML. This speed up development and makes customization simple, which is why many developers like using it.

  In our content management system, Tailwind CSS enables efficient and responsive design by providing pre-built utilities classes for layout, spacing, typography, and colors.

- **Framework: NextJs**

  NextJS is a React framework that simplifies the development of server-side rendered and static websites, providing features like routing, server-side rendering, and automatic code splitting for optimized performance.

In our content management system, Next.js provides fast, scalable, and SEO friendly pages by doing server-side rendering and static generation. It helps in easy routing, dynamic content handling, and easy API integration.

- **Language: TypeScript**

  TypeScript is a programming language that adds static typing to JavaScript, helping developers catch errors and write more maintainable code.

  In our CMS, TypeScript enhances the development process by offering type safety, which reduces runtime errors and improves code clarity. It helps in managing complex data structures and ensures the data are handled correctly while passing from one component to another.

4.2.2 Back End Tools

- **Framework: ExpressJS**

  ExpressJS is a minimalist application framework for NodeJS, providing a robust set of features for building web servers and APIs quickly and efficiently. It simplifies the development process by offering powerful tools for handling HTTP requests, middleware integration, routing, and session management.

  In our CMS, due to its middleware system, Express allows us to add custom functionality like logging, error handling, and validation for our application.

- **Language: JavaScript**

  JavaScript is a versatile programming language used primarily for building interactive websites and web applications. It enables developers to create dynamic and interactive user experiences on websites and applications. Originally designed to run in web browsers, JavaScript is now widely used both on the client side (front-end) and server side (back-end, with platforms like Node.js).

  In our CMS, it helps in implementing dynamic features like form validation, and interactive UI elements.

- **Database: MongoDB**

  MongoDB is a popular NoSQL database that uses a document-oriented data model. It is designed for flexibility, scalability, and performance, making it suitable for a wide range of applications.

In our CMS, it helps in the storage and management of various types of data like users, contents, categories, comments, and logins. Also, it helps in creating flexible schema throughout the project.

- **Database Management Tools: MongoDB Compass**

  MongoDB Compass is GUI based platform for MongoDB to explore and manipulate data. Here, we can visualize the data stored in the database in a user-friendly manner.

  In our CMS, MongoDB compass helps by providing a user-friendly interface to visualize data, optimize queries, and manage collections and documents.

4.2.2 Testing Tools

- **Postman**

  Postman is a popular API testing tool ideal for validating Express.js projects. It allows developers to simulate HTTP requests (GET, POST, PUT, DELETE, etc.) to test routes, handle dynamic inputs like query parameters or headers, and inspect server responses. With features like scripting, environment management, and collections, Postman simplifies manual and automated testing. It ensures robust error handling and seamless integration with CI/CD workflows through its CLI tool, Newman.

  In our CMS, it is used to test and validate the RESTful APIs built with Express.js. It helps in ensuring that the API responses are accurate and efficient.

- **Jest**

  Jest is a powerful JavaScript testing framework widely used for testing Express.js projects. It provides a simple setup for writing unit and integration tests to validate API routes, middleware, and server logic. With features like test mocking, snapshot testing, and built-in assertions, Jest ensures reliable and maintainable tests. Its fast execution, detailed reports, and coverage tracking make it ideal for ensuring code quality and robustness in Express.js applications.

  In our CMS, it is used to write unit and integration tests for both frontend and backend components. It helps to maintain high code quality for improving the overall stability and reliability of the system.

4.3 Testing

Testing is the systematic process of evaluating a software application or system to identify and resolve defects, ensuring that it operates as intended. The primary objective of testing is to verify that the software complies with specified requirements, functions correctly, and provides a reliable and satisfactory user experience. This process involves executing the software or its components, using various test cases and scenarios, to detect errors or bugs.

4.3.1 Component Testing

Component testing is a type of software testing that focuses on verifying the functionality and behavior of individual components or modules in isolation to ensure they perform as expected.

Table 4.1: Component testing of Login.test.tsx

Test Case Id	Test Case	Test Description	Input Test Data	Expected Result	Actual Result	Remarks
T-01	Render Form Elements	Verify that the login form elements and links are rendered properly	N/A	All elements are present: username, password, login button, links.	All elements are present as expected.	Pass
T-02	Update Form State	Verify that the form state updates correctly on input changes.	Username: "testuser", Password: "password 123"	Form state updates with entered values.	Form state updated with values: "testuser" and "password1 23".	Pass
T-03	Handle Empty Submissi on	Verify that an empty form submission triggers an error notification.	Click the login button without entering any input.	Error notification displayed: "Enter valid credentials."	Error notification displayed: "Enter valid credentials."	Pass
T-04	Handle Successfu l Login	Verify successful login flow with correct	Username: "testuser", Password: "password	Notification: "Login successful", calls login	Notification : "Login successful", called login	Pass

		credentials.	123". Mock API returns valid token.	with token and user ID.	with token and user ID.	
T-05	Handle Validation Error	Verify that incorrect login credentials result in an error notification.	Username: "wronguser", Password: "wrongpass". Mock API response: { ok: false, status: 400, msg: "Invalid credentials" }.	Notification: "Invalid credentials".	Notification: "Invalid credentials".	Pass

Table 4.2: Component testing of Register.test.tsx

Test Case Id	**Test Case**	**Test Description**	**Input Test Data**	**Expected Result**	**Actual Result**	**Remarks**
T-06	Render First Form	Verify that the first form of the registration component renders correctly.	N/A	Fields: Full Name, Email ID, Contact Number, Address are displayed.	Fields rendered as expected.	Pass
T-07	Switch to Second Form	Verify that clicking "Next" switches to the second form.	Click "Next" without filling the first form.	Fields: Username, Password, Confirm Password are displayed.	Fields rendered as expected.	Pass
T-08	Password Mismatch	Verify that mismatched passwords trigger an error notification.	Username: "testuser", Password: "password123", Confirm Password: "password456".	Notification: "Please match the password."	Notification displayed as expected.	Pass

T-09	Handle OTP Sending	Verify successful OTP sending notification after filling the forms correctly.	Full Name: "Test User", Email: "test@example.com", Contact: "1234567890", Address: "Test Address", Password: "123".	Notification: "OTP sent successfully."	Notification displayed as expected.	Pass
T-10	Successful Registration	Verify that successful registration navigates to the login page.	Same as T-04, with OTP: "123456".	Navigates to "/login".	Navigated as expected.	Pass
T-11	Handle API Errors	Verify that API errors are handled gracefully, and an error notification is shown.	Same as T-04, but with mocked API response: { ok: false, status: 400, msg: "API Error" }.	Notification: "API Error."	Notification displayed as expected.	Pass

Table 4.3: Component testing of ContentTable.test.tsx

Test Case Id	**Test Case**	**Test Description**	**Input Test Data**	**Expected Result**	**Actual Result**	**Remarks**
T-12	Handle API Error Gracefully	Verify that the component gracefully handles API errors when fetching content.	Simulate an API rejection (mockRejectedValue).	Content is not displayed, and no errors are thrown in the UI.	As expected.	Pass
T-13	Search Filtering	Verify that the content table filters results based on search input.	Input: "Test" and "NonExistent" in the search field.	Content "Test Content" is displayed for "Test" but not displayed for "NonExisten	As expected.	Pass

				t".		
T-14	Handle Delete Failure	Verify that the component handles failures when attempting to delete a content item.	Simulate delete API response with { ok: false } and msg: "Delete failed".	"Delete failed" notification is displayed, and the content remains in the list.	Delete failure handled correctly.	Pass
T-15	Pagination Update	Verify that pagination updates correctly when new data is fetched and pages are switched.	Use 15 content items and switch to page 2 using the pagination button.	Items on page 1 are not displayed after switching to page 2, and new items appear correctly.	Pagination works as expected.	Pass
T-16	Sorting by Creation Date	Verify that the content is sorted by creation date when fetched.	Provide items with creation dates 2024-01-02 and 2024-01-01.	The content with the newest date appears first in the list.	Content is sorted correctly.	Pass

Table 4.4: Component testing of CategoryTable.test.tsx

Test Case Id	**Test Case**	**Test Description**	**Input Test Data**	**Expected Result**	**Actual Result**	**Remarks**
T-17	Render Table Headers	Verify that the table headers are rendered correctly.	None	Headers "SN," "Title," "Created By," and "Action" are displayed.	Headers are displayed as expected.	Pass
T-18	Fetch and Display Categories	Verify that category data is fetched from the API and displayed correctly in the table.	Mock category data: [{ id: 1, title: "Test Category", user: { name: "Test User" } }]	The category title "Test Category" and creator name "Test User" are displayed in the table.	Category data is displayed correctly.	Pass

Table 4.5: Component testing of ShowPdf.test.tsx

Test Case Id	**Test Case**	**Test Description**	**Input Test Data**	**Expected Result**	**Actual Result**	**Remarks**
T-19	Render Required Elements	Verify that all required elements (title, description, user, etc.) are rendered correctly.	Valid mockProps object	Title, description, user name, thumbnail, profile picture, PDF viewer, and download link are displayed correctly.	All elements are displayed as expected.	Pass
T-20	Fetch Comments on Mount	Verify that the component fetches comments when mounted.	Valid mockProps.id	API call is made to fetch comments using contentId.	API call to fetch comments is verified.	Pass
T-21	Handle Fetch Error	Verify error handling when the fetch for comments fails.	Invalid API response (e.g., ok: false)	Error handling occurs without breaking the UI.	Fetch error is handled gracefully.	Pass
T-22	Default Profile Picture	Verify that the default profile picture is displayed when profilePic is null.	mockProps with profilePic: null	Default profile picture (/default.jpg) is displayed.	Default profile picture is displayed as expected.	Pass
T-23	Format Dates Correctly	Verify that the created and updated dates are formatted correctly.	Valid mockProps. createdAt and mockProps. updatedAt	Dates are displayed in MM/DD/YYYY format.	Dates are formatted and displayed correctly.	Pass

T-24	Render Advertisement Component	Verify that the ShowAdvertisement child component is rendered.	Valid mockProps	Advertisement component is rendered.	Advertisement component is displayed as expected.	Pass
T-25	Render Comment Component	Verify that the ShowComment child component is rendered.	Valid mockProps	Comment component is rendered.	Comment component is displayed as expected.	Pass

Table 4.6: Component testing of ShowPost.test.tsx

Test Case Id	**Test Case**	**Test Description**	**Input Test Data**	**Expected Result**	**Actual Result**	**Remarks**
T-26	Render Component with Data	Verify that the component renders all content data correctly.	Valid mockContentData	Component displays the title, description, user info, and other details from mockContentData.	All content is rendered correctly.	Pass
T-27	Fetch Content Data on Mount	Verify that the API call is made to fetch content data on component mount.	Valid API endpoint and id in URL	API call to api/content/post/123 is made successfully.	API call verified.	Pass
T-28	Speech Synthesis Controls	Verify that the Speak button initializes the speech synthesis functionality.	Valid SpeechSynthesis mock and button click simulation	SpeechSynthesisUtterance is called and initializes the speech functionality.	Speech synthesis functionality works as expected.	Pass
T-29	Render Advertisement Component	Verify that the ShowAdvertisement	Valid mockContentData	Advertisement component is displayed	Advertisement component is displayed	Pass

		child component is rendered.		correctly.	as expected.	
T-30	Render Comment Component	Verify that the ShowComment child component is rendered.	Valid mockContentData	Comment component is displayed correctly.	Comment component is displayed as expected.	Pass
T-31	Format Dates Correctly	Verify that created and updated dates are formatted and displayed correctly.	Valid mockContentData.created_at	Dates are displayed in the correct format (e.g., MM/DD/YYYY).	Dates are formatted and displayed correctly.	Pass
T-32	Handle Download Functionality	Verify that the download button is rendered and functional.	Valid mockContentData.location	Download button is displayed with a title attribute and works as expected when clicked.	Download button is displayed as expected.	Pass

Table 4.7: Component testing of ShowVideo.test.tsx

Test Case Id	**Test Case**	**Test Description**	**Input Test Data**	**Expected Result**	**Actual Result**	**Remarks**
T-33	Render Component Elements	Verify that all required elements (title, description, user info, etc.) are rendered correctly.	Valid mockProps	Component displays all required elements, including user info, title, description, and media-related elements.	Pass	N/A
T-34	Fetch Comments on Mount	Verify that comments are fetched when the component	Valid mockProps.id	API call to api/comment/comments?contentId=123 is made	Pass	N/A

		mounts.		successfully.		
T-35	Handle Fetch Errors	Verify that the component handles comment-fetching errors gracefully.	Simulate API failure (ok: false)	Component gracefully handles error (e.g., shows no comments or an error message).	Pass	N/A
T-36	Default Profile Picture	Verify that the default profile picture is displayed when profilePic is null.	mockProps. profilePic: null	Component renders the default profile picture (/default.jpg).	Pass	N/A
T-37	Format Dates Correctly	Verify that created and updated dates are formatted and displayed correctly.	Valid mockProps. createdAt and mockProps. updatedAt	Dates are displayed in the correct format (e.g., MM/DD/Y YYY).	Pass	N/A
T-38	Render Advertiseme nt Component	Verify that the ShowAdvert isement child component is rendered.	Valid mockProps	Advertiseme nt component is displayed correctly.	Pass	N/A
T-39	Render Comment Component	Verify that the ShowComm ent child component is rendered.	Valid mockProps	Comment component is displayed correctly.	Pass	N/A
T-40	Render Video Iframe Source	Verify that the iframe for video playback is rendered with the correct source URL.	Valid mockProps.l ocation	Iframe source URL matches ${process.e nv.NEXT_P UBLIC_BA CKEND_A PI}${mockP rops.locatio n}.	Pass	N/A

T-41	Render Download Link	Verify that the download link for the video is rendered with the correct href.	Valid mockProps.location	Download link href matches ${process.env.NEXT_PUBLIC_BACKEND_API}${mockProps.location}.	Pass	N/A

Table 4.8: Component testing of Logout.test.tsx

Test Case Id	**Test Case**	**Test Description**	**Input Test Data**	**Expected Result**	**Actual Result**	**Remarks**
T-42	Render Nothing When Closed	Verify that the component does not render anything when isOpen is false.	isOpen: false	Modal content is not present in the DOM.	Pass	N/A
T-43	Render Modal When Open	Verify that the modal content is rendered when isOpen is true.	isOpen: true	Modal displays confirmation message and Yes and Cancel buttons.	Pass	N/A
T-44	Call onClose on Cancel	Verify that onClose is called when the Cancel button is clicked.	isOpen: true + Click Cancel button	onClose callback function is invoked.	Pass	N/A
T-45	Call onClose on X Button	Verify that onClose is called when the close (X) button is clicked.	isOpen: true + Click close button	(Currently commented out) Expect onClose callback to be invoked.	N/A	Test commented.
T-46	Call Logout on Yes	Verify that the logout function is called when	isOpen: true + Click Yes button	Logout function is invoked successfully.	Pass	N/A

		the Yes button is clicked.				
T-47	Handle Logout Errors Gracefully	Verify that errors during the logout process are logged without breaking the application.	Simulated logout rejection (mock error)	Console logs error message (e.g., "Logout failed").	N/A	Test commented.

4.3.2 API Testing

API testing is the process of validating Application Programming Interfaces (APIs) to ensure they function correctly, reliably, and securely by checking responses, performance, and error handling for various requests.

Table 4.9: API Testing

Endpoint	**Method and isToken Needed**	**Body**	**Response & Status**
/api/auth/sendOTP	POST, no	{"email" : "shresthanewar678@gmail.com"}	Shows message "Name is required" (500)
/api/auth/register	POST, no	{ "name": "Jghghhggh", "username": "johndoe1", "password": "SecureP@ssw0rd!", "email": "shresthanewar67@gmail.com",	Upon valid OTP and unique username and email shows "Registration successful" (200) or else show "Invalid OTP" (400)

		"address": "123 Main Street, Apt 4B, Springfield, USA", "phone_number": "1231234", "otp": "227973" }	

/api/auth/login	POST, no	{
		"username": "admin",
		"password": "admin"
		}
			Shows message “Credentials error” (400) on incorrect username and password

/api/auth/login	POST, no	{
		"username": "test",
		"password": "test123"
		}
			Shows message “Login successful” (200) on correct username and password

/api/user/	GET, yes	nil	Shows the list of users in the system with their details except password
/api/user/role?role=viewer	GET, yes	nil	Shows the list of viewers if user is admin or editor
/api/user/role?role=author	GET, yes	nil	Shows the list of authors if user is admin or editor
/api/user/role?role=editor	GET, yes	nil	Shows the list of editors if user is admin or editor

/api/user/role?role=admin	GET, yes	nil	Shows the list of admin if user is admin or editor
/api/user/66b610084996992b3d5db30c	GET, yes	nil	Shows the details of the user with id “66b610084996992b3d5db30c” (200)
/api/user/66b610084996992b3d5db30d	GET, yes	nil	Shows “No users found” if ID doesn’t match (404)
/api/user/66b610084996992b3d5db30c	PUT, yes	{ "name" : "Sophia Shrestha", "username" : "johndoe131", "phone_number" : "24234234234" }	Edit the detail of user with username “johndoe131” (200) if username is found.
/api/user/66b610944996992b3d5db318	DELETE, yes	nil	Delete the user with id “66b610944996992b3d5db318” from the database (200)
/api/user/change-password/66b6101c4996992b3d5db310	PUT, yes	{ "old_password":"password123", "new_password": "password12"	Edit the password of the user with id “66b6101c4996992b3d5db310”

		}	to the new password (200)
/api/user/change-email/66b6101c4996992b3d5db310	PUT, yes	{ "new_email" : "shresthanewar678@gmail.com" }	Edit the email of the user with id “66b6101c4996992b3d5db310” to the new email (200)
/api/user/approve-author/66b6101c4996992b3d5db310?approve=true	PUT, yes	nil	Approve the user as author if user with id “66b6101c4996992b3d5db310” is a viewer (200) or else show “User is not a viewer”
/api/user/change-to-editor/66b6101c4996992b3d5db310	PUT, yes	nil	Change the user role of user with id “66b6101c4996992b3d5db310” to editor.
/api/user/promote-admin/668ba90852a9bfcfcef20b7f	PUT, yes	nil	Change the user role of user with id “668ba90852a9bfcfcef20b7f” to admin (200)
/api/user/upload-profile-picture/66b610084996992b3d5db30c	PUT, yes	form-data profile-pic : /C:/Users/MANOJ/Pictur	Uploads profile picture for the user with id “66b610084996

		es/Screenshots/Experienc e at Freelancer.com.png	992b3d5db30c" (200)
/api/content/add/post	POST, yes		
/api/content/add/pdf	POST, yes	form-data title: PDF sample description: This is a pdf pdf: /C:/Users/MANOJ/Downloads/SmartDustbin.pdf thumbnail: /C:/Users/MANOJ/Downloads/people-woman-yoga-meditation.jpg category_id: 66cdd525db61a9ffa4f48401 user_id: 66b610084996992b3d5db30c content_type: pdf	Adds pdf detail to the database and stores pdf in the server (200) with a message "PDF added successfully"
/api/content/add/video	POST, yes	form-data title: Video sample description: This is a video pdf: /C:/Users/MANOJ/Downloads/SmartDustbin.mp4	Adds video detail to the database and stores video in the server (200) with a message "Video added successfully"

		thumbnail: /C:/Users/MANOJ/Downloads/people-woman-yoga-meditation.jpg category_id: 66cdd525db61a9ffa4f48401 user_id: 66b610084996992b3d5db30c content_type: pdf	

/api/content/edit/post/66963e7d41fd5196a57e5a14	PUT, yes	{
		"title": "Sample PDF Title",
		"description": "This is a sample description for the PDF content.",
		"blog" : "<h1>THis is Hellosadsfadsfdf</h1>"
		}
			Edit the detail for the post with id “66963e7d41fd5196a57e5a14” (200)

/api/content/edit/pdf/669640684c469b87f9c9540b	PUT, yes	form-data	Edit the detail for the pdf with id “669640684c469b87f9c9540b” (200)
/api/content/edit/video/669641fe4c469b87f9c9540e	PUT, yes	form-data	Edit the detail for the video with id “669641fe4c469

			b87f9c9540e” (200)
/api/content/delete/6697137d9b4aa6c594cbc530	DELETE, yes	nil	Delete the content(post, pdf, video) with id “6697137d9b4aa6c594cbc530” (200) and remove content from server
/api/content/approve/6698e1e6d3046b6250a317c7	PUT, yes	nil	Approves content with id “6698e1e6d3046b6250a317c7” (200)
/api/content/post/66963f0b41fd5196a57e5a1a	GET, yes	nil	Get the detail of the post with id “66963f0b41fd5196a57e5a1a” (200)
/api/content/pdf/6697132b9b4aa6c594cbc52d	GET, yes	nil	Get the detail of the pdf with id “6697132b9b4aa6c594cbc52d” (200)
/api/content/video/669641fe4c469b87f9c9540e	GET, yes	nil	Get the detail of video with id “669641fe4c469b87f9c9540e” (200)

/api/content/	GET, yes	nil	Get the list of contents with details (200)
/api/category/add	POST, yes	{ "title" : "Green", "user_id": "66b610084996992b3d5db30c" }	Adds the title called green to the database (200)
/api/category/edit/66b6162193b17c060c4d4f8a	PUT, yes	{ "title": "Communication System", "user_id": "66b610084996992b3d5db30c" }	Edits the category detail with id "66b6162193b17c060c4d4f8a "(200)
/api/category/delete/66b6162193b17c060c4d4f8a	DELETE, yes	nil	Delete the category with id "66b6162193b17c060c4d4f8a" (200)
/api/category/66b6160993b17c060c4d4f85	GET, yes	nil	Get detail of category with id "66b6160993b17c060c4d4f85"
/api/category	GET, no	nil	Get the list of category detail (200)
/api/user/countuser	GET, yes	nil	Get the number of individual users (admin, editor, author

			and viewer in the database) (200
/api/user/add	POST, yes	{ "name": "Jane Smith", "username": "editor", "address": "456 Elm St, Metropolis", "password": "editor123", "email": "editor@example.com", "role": "editor", "phone_number": "9845671234" }	Adds user to the database (200)

4.3.3 System Testing

System testing is a type of software testing that evaluates the complete and integrated system to ensure it meets specified requirements, functioning as intended in real-world scenarios.

Table 4.10: System Testing of Admin

Test Case Id	**Test Case**	**Test Description**	**Input Test Data**	**Expected Result**	**Actual Result**	**Remarks**
T-01	Login to System	Enter username and password	admin, admin123	Log into admin dashboard	Logged into admin dashboard	Pass
T-02	Add Editor	Enter name, address, contact number, role as editor, email, username password,	Editor, damauli, 9800765443, editor, editor25@gmail.com, editor25, editor123,	Add new editor to the system	Added new editor to the system	Pass

		new password	editor123			
T-03	Add Author	Enter name, address, contact number, role as author, email, username password, new password	Author, damauli, 9800765443, author, author25@gmail.com, author25, author123, author123	Add new author to the system	Added new author to the system	Pass
T-04	Edit Editor	Enter name, username, address and contact	Editor, editor25, damauli, 9810234555	Update editor	Updated editor successfully	Pass
T-05	Edit Author	Enter name, username, address and contact	Author, author25, damauli, 9810234555	Update author	Updated author successfully	Pass
T-06	Edit Viewer	Enter name, username, address and contact	Viewer, viewer25, damauli, 9807657743	Update viewer	Updated viewer successfully	Pass
T-07	Delete Editor	Click to the delete icon in the editor		Delete the editor	Editor deleted	Pass
T-08	Delete Author	Click to the delete icon in the author		Delete the author	Author deleted	Pass
T-09	Delete Viewer	Click to the delete icon in the viewer		Delete the viewer	Viewer deleted	Pass
T-10	Add PDF	Enter title, description, author, PDF, thumbnail, category		PDF added successfully	PDF added successfully	Pass

T-11	Add Video	Enter title, description, author, video, thumbnail, category		Video added successfully	Video added successfull y	Pass
T-12	Add Post using Speech to Text	Enter title, description,a uthor, thumbnail, category, speech to text post		Post added successfully	Post added successfull y	pass
T-13	Add Post using text editor	Enter title, description, thumbnail, category, write post		Post added successfully	Post added successfull y	pass
T-14	Edit PDF	Edit title, description,a uthor, PDF, thumbnail, category		PDF edited successfully	PDF edited successfull y	pass
T-15	Edit Video	Edit title, description, author, thumbnail, category		Video edited successfully	Video edited successfull y	pass
T-16	Edit Post	Edit title, description, author, thumbnail, category, post		Post edited successfully	Post is edited while typing but not edited while providing voice command	Fail
T-17	Delete PDF	Click to the delete icon in the PDF content		PDF deleted successfully	PDF deleted successfull y	pass

T-18	Delete Video	Click to the delete icon in the video content		Video deleted successfully	Video deleted successfull y	pass
T-19	Delete Post	Click to the delete icon in the post content		Post deleted successfully	Post deleted successfull y	pass
T-20	Add Category	Enter category name, choose author		Category added successfully	Category added successfull y	Pass
T-21	Edit Category	Enter category name, choose author		Category edited successfully	Category edited successfull y	Pass
T-22	Delete Category	Click to the delete icon in the category		Category deleted successfully	Category deleted successfull y	Pass
T-23	Accept Viewer as Author					
T-24	Reject Viewer as Author	Click to the reject button in the request		Reject viewer as author	Rejected viewer as author	Pass
T-25	Accept author's content					
T-26	Reject author's content	Click to the reject button in the request		Reject content request	Rejected content request	pass
T-27	Notify admin of pending requests	Click to the notification in the sidebar		Display pending requests	Display pending requests	Pass

T-28	Add comment on content	Enter comment and hit post button	Very helpful Content	Display the comment on the comment	Display the comment on the comment	Pass
T-29	Delete comment on content	Click on comment in the side bar and click delete icon on comment		Delete the comment	Comment deleted	Pass
T-30	Reply to comment	Click on reply text on the comment of the content	Thank You	Reply to the comment successfully	Replied to the comment successfull y	Pass
T-31	View Content	Click on the content in the sidebar		Display content table	Displayed content table	Pass
T-32	View Users	Click on the user in the sidebar		Display user table successfully	Displayed user table successfull y	Pass
T-33	View Category	Click on the Category in the sidebar		Display category table successfully	Displayed category table successfull y	Pass
T-34	View Comment s	Click on the comment in the sidebar		Displayed comment table successfully	Displayed comment table successfull y	Pass
T-35	Logout	Click on logout in the sidebar and click yes		Logout of the admin page successfully	Logged out of the admin page successfull y	Pass

Table 4.11: System Testing of Editor

Test Case Id	**Test Case**	**Test Description**	**Input Test Data**	**Expected Result**	**Actual Result**	**Remarks**
T-01	Login to System	Enter username and password	editor, editor123	Log into editor dashboard	Logged into editor dashboard	Pass
T-02	Add Author	Enter name, address, contact number, role as author, email, username password, new password	Author, damauli, 9800765443, author, author25@gmail.com, author25, author123, author123	Add new author to the system	Added new author to the system	Pass
T-03	Edit Author	Enter name, username, address and contact	Editor, author25, damauli, 9810234555	Update author	Updated author successfully	Pass
T-04	Edit Viewer	Enter name, username, address and contact	viewer, viewer25, damauli, 9810234555	Update viewer	Updated viewer successfully	Pass
T-05	Delete Author	Click to the delete icon in the author		Delete the author	Author deleted	Pass
T-06	Delete Viewer	Click to the delete icon in the viewer		Delete the viewer	viewer deleted	Pass
T-07	Add PDF	Enter title, description, PDF, thumbnail, category		PDF added successfully	PDF added successfully	Pass
T-08	Add Video	Enter title, description,		Video added successfully	Video added	Pass

		video, thumbnail, category			successfull y	
T-09	Add Post using Speech to Text	Enter title, description, thumbnail, category, speech to text post		Post added successfully	Post added successfull y	pass
T-10	Add Post using text editor	Enter title, description, thumbnail, category, write post		Post added successfully	Post added successfull y	pass
T-11	Edit PDF	Edit title, description, PDF, thumbnail, category		PDF edited successfully	PDF edited successfull y	pass
T-12	Edit Video	Edit title, description, thumbnail, category		Video edited successfully	Video edited successfull y	pass
T-13	Edit Post	Edit title, description, thumbnail, category, post		Post edited successfully	Post edited successfull y	pass
T-14	Delete PDF	Click to the delete icon in the PDF content		PDF deleted successfully	PDF deleted successfull y	pass
T-15	Delete Video	Click to the delete icon in the video content		Video deleted successfully	Video deleted successfull y	pass
T-16	Delete Post	Click to the delete icon in the post content		Post deleted successfully	Post deleted successfull y	pass

T-17	Add Category	Enter category name, choose author		Category added successfully	Category added successfull y	Pass
T-18	Edit Category	Enter category name, choose author		Category edited successfully	Category edited successfull y	Pass
T-19	Delete Category	Click to the delete icon in the category		Category deleted successfully	Category deleted successfull y	Pass
T-20	Accept Viewer as Author	Click to the accept button in the request		Accept viewer as author	Accepted viewer as author	Pass
T-21	Reject Viewer as Author	Click to the reject button in the request		Reject viewer as author	Rejected viewer as author	Pass
T-22	Accept author’s content	Click to the accept button in the request		Accept author’s content	Accepted author’s content	Pass
T-23	Reject author’s content	Click to the reject button in the request		Reject content request	Rejected content request	pass
T-24	Notify editor of pending requests	Click to the notification in the sidebar		Display pending requests	Display pending requests	Pass
T-25	Add comment on content	Enter comment and hit post button	Very helpful Content	Display the comment on the comment	Display the comment on the comment	Pass
T-26	Delete comment on content	Click on comment in the side bar and click		Delete the comment	Comment is deleted	Pass

		delete icon on comment				
T-27	Reply to comment	Click on reply text on the comment of the content	Thank You	Reply to the comment successfully	Replied to the comment successfull y	Pass
T-28	View Content	Click on the content in the sidebar		Display content table	Displayed content table	Pass
T-29	View Users	Click on the user in the sidebar		Display user table	Displayed user table	Pass
T-30	View Category	Click on the Category in the sidebar		Display category table	Displayed category table	Pass
T-31	View Comment s	Click on the comment in the sidebar		Displayed comment table	Displayed comment table	Pass
T-32	Logout	Click on logout in the sidebar and click yes		Logout of the editor page	Logged out of the editor page	Pass

Table 4.12: System Testing of Author

Test Case Id	**Test Case**	**Test Description**	**Input Test Data**	**Expected Result**	**Actual Result**	**Remarks**
T-01	Login to System	Enter username and password	author, author123	Log into author’s dashboard	Logged into author’s dashboard	Pass
T-02	Add PDF	Enter title, description, PDF, thumbnail, category		Send request to admin and editor to add PDF	Sent request to admin and editor to add PDF	Pass
T-03	Add Video	Enter title, description, video,		Send request to admin and editor to add	Sent request to admin and	Pass

		thumbnail, category		video	editor to add video	
T-04	Add Post using Speech to Text	Enter title, description, thumbnail, category, speech to text post		Send request to admin and editor to add post	Sent request to admin and editor to add post	pass
T-05	Add Post using text editor	Enter title, description, thumbnail, category, write post		Send request to admin and editor to add post	Sent request to admin and editor to add post	pass
T-06	Edit Own PDF	Edit title, description, PDF, thumbnail, category		PDF edited	PDF edited	pass
T-07	Edit Own Video	Edit title, description, thumbnail, category		Video edited	Video edited	pass
T-08	Edit Own Post	Edit title, description, thumbnail, category, post		Post edited	Post edited	pass
T-09	Delete Own PDF	Click to the delete icon in the PDF content		PDF deleted	PDF deleted	pass
T-10	Delete Own Video	Click to the delete icon in the video content		Video deleted	Video deleted	pass
T-11	Delete Own Post	Click to the delete icon in the post content		Post deleted	Post deleted	pass

T-12	Add Category	Enter category and click save button		Add category	Category added	Pass
T-13	Edit Own Category	Enter category and click save button		Edit category	Category edited	Pass
T-14	Delete Own Category	Click to the delete icon in the category		Delete category	Category deleted	Pass
T-15	Add comment on content	Enter comment and hit post button	Very helpful Content	Display the comment on the comment	Display the comment on the comment	Pass
T-16	Delete Own comment on content	Click on comment in the side bar and click delete icon on comment		Delete the comment	Comment is deleted	Pass
T-17	Reply to comment	Click on reply text on the comment of the content	Thank You	Reply to the comment successfully	Replied to the comment successfull y	Pass
T-18	View Own Content	Click on the content in the sidebar		Display content table successfully	Displayed content table successfull y	Pass
T-19	View Category	Click on the Category in the sidebar		Display category table successfully	Displayed category table successfull y	Pass
T-20	View Comment s	Navigate to the content		Display comment	Displayed comment	Pass

T-21	Logout	Click on logout in the sidebar and click yes		Logout of the author’s page successfully	Logged out of the author’s page successfull y	Pass

Table 4.13: System Testing of Registered Viewer

Test Case Id	**Test Case**	**Test Description**	**Input Test Data**	**Expected Result**	**Actual Result**	**Remarks**
T-01	Login to System	Enter username and password	author, author123	Log into author’s dashboard	Logged into author’s dashboard	Pass
T-02	Request to become Author	Click on Apply as Author button on the homepage		Send request as notification to admin and editor	Sent request as notification to admin and editor	Pass
T-03	Notify viewer in email if they are accepted to become author	Open the email that you provided while signing up into the system		Send notification of viewer accepted as author	Sent notification of viewer accepted as author	Pass
T-04	View Content	Navigate to the homepage and view content section		Display contents	Displayed contents	Pass
T-05	View Comment s on the content	Click on the post and view comment section		Display comment	Displayed comment	Pass
T-06	Add comment on	Enter comment and hit post	Very helpful Content	Display the comment on the comment	Display the comment on the	Pass

	content	button			comment	
T-07	Delete Own comment on content	Click on comment in the side bar and click delete icon on comment		Delete the comment	Comment is deleted	Pass
T-08	Reply to comment	Click on reply text on the comment of the content	Thank You	Reply to the comment successfully	Replied to the comment successfull y	Pass
T-09	View Category	Navigate to the homepage and view category section		Display category	Displayed category	Pass
T-10	Logout	Click on logout in the sidebar and click yes		Logout of the author’s page successfully	Logged out of the author’s page successfull y	Pass

Table 4.14: System Testing of Non-registered Viewer

Test Case Id	**Test Case**	**Test Description**	**Input Test Data**	**Expected Result**	**Actual Result**	**Remarks**
T-01	View Content	Navigate to the content section in the homepage		Display Content	Displayed Content	Pass
T-02	View Category	Navigate to the category section in the homepage		Display Category	Displayed Category	Pass

T-03	Cannot comment to the content	Enter comment and hit post button	Very helpful content	Show prompt to register	Showed prompt to register	Pass
T-04	Sign Up to the system	Enter full name, username, email, contact number, address password, confirm password and hit submit button and type otp and submit	Anita Shrestha, anita, anita@gmail.com, 9807654321, damauli, anita123, anita123,	Register as Viewer	Registered as viewer	Pass