Project Report
on

# WEB DEVELOPMENT

Submitted

In Partial Fulfillment of

**MASTER OF COMPUTER APPLICATIONS (MCA)**

**Submitted by:**

Name: Sneha Sidhu
Roll No: 24/SCA/BCA(AI/ML)/065

**Under the Supervision of:**

**Dr. Sakshi Gupta**

(Professor, SCA)

**School of Computer Applications**
**Manav Rachna International Institute of Research and Studies**
**(DEEMED TO BE UNIVERSITY)**
Sector-43, Aravalli Hills
Faridabad – 121001

**June 2025**

## Declaration

I do hereby declare that this project work entitled **“WEB DEVELOPMENT”** submitted by me for the partial fulfillment of the requirement for the award of **BACHELOR OF COMPUTER APPLICATIONS** is a record of my own work. The report embodies the finding based on my study and observation and has not been submitted earlier for the award of any degree or diploma to any Institute or University.

**SIGNATURE**
Name: SNEHA SIDHU
Roll No: 24/SCA/BCA(AI/ML)/065
Date:11/7/25

# Certificate from the Guide

This is to certify that the project report entitled **“WEB PAGE DESIGN”** submitted in partial fulfillment of the degree of **BACHELOR OF COMPUTER APPLICATIONS** to Manav Rachna International Institute of Research and Studies, Faridabad is carried out by Ms. SNEHA SIDHU (Roll No), 24/SCA/BCA(AI/ML)/065 under my guidance.

**Signature of the Guide**

Name: Dr.Sakshi Gupta

Date: 11.7.25

**Head of Department**

Name: Dr. Suhail Javed Quraishi

Date:

# ACKNOWLEDGEMENT

I gratefully acknowledge for the assistance, cooperation, guidance and clarification provided by **Ms. Dr. Sakshi Gupta** during the development of web page. My extreme gratitude to **Dr. Raj Kumar, Associate Professor & TPO** who guided us throughout the project. Without his willing disposition, spirit accommodation, frankness, timely clarification and above all faith in us, this project could not have been completed in due time. His readiness to discuss all important matters at work deserves special attention of.

I would like to extend my sincere gratitude to **Prof. (Dr.) Suhail Javed Quraishi – HOD, Prof. (Dr.) Rashmi Agrawal – Associate Dean and Prof. (Dr.) Brijesh Kumar – Dean** for their valuable teachings and advice. I want to thank all the department faculty members for their cooperation and support. I want to thank non-teaching staff of the department for their cooperation and support.

This opportunity is a big milestone in my career development. I will strive to use gained skills and knowledge in the best possible way, and I will continue to work on their improvement, to attain desired career objectives. I hope to continue cooperation with all of you in the future.

**INDEX**

# Web Page Design and Development for Magsstore.com

## INTRODUCTION

Magsstore.com is an online magazine store that offers subscriptions to a wide variety of magazines including fashion, sports, business, health, and lifestyle. It allows users to subscribe to both print and digital versions of popular publications. The website's effectiveness largely depends on its design, ease of navigation, and ability to handle transactions smoothly.

Category of magazines are:

Animals & Pets
Art & Photography
Automotive
Business & Finance
Celebrity
Children
Collectibles
Comics & Puzzles
Computer & Electronics
Digital Magazines
Entertainment & Music
Ethnic
Family & Parenting
Fashion & Beauty
Food & Beverages
Guns & Weapons
Health & Fitness
Hobbies & Crafts
Home & Garden
Hunting & Fishing
International
Journals
Lifestyle & Culture
Local & Regional
Men's Interest
News & Politics
Outdoor

Photography & Video
Reading & Enrichment
Religion & Spirituality
Science & Nature
Sports & Recreation
Teen
Weddings & Bridal

Women's Interest
Under $10
Between $10 - $15
Bundle Offer

## Objective

The main objective of this project was to study, analyze, and enhance the design and functionality of Magsstore.com, an ecommerce website that specializes in magazine subscriptions. The goal was to create a modern, user-friendly interface that improves user experience, navigation, and mobile responsiveness while maintaining the brand's identity.

### Aims & Objectives:

Magsstore aims to empower students, freshers, and aspiring professionals by offering high-quality training, practical exposure, and career-building opportunities in the field of Information Technology. The organization is committed to bridging the gap between academic learning and industry expectations by providing hands-on experience through real-time projects and expert mentorship.

**Objectives:**

- **Skill Enhancement:**
  To develop technical skills in key areas such as web development, mobile application development, artificial intelligence, machine learning, data science, and cloud computing.
- **Provide Real-Time Project Experience:**
  To offer practical learning through live projects, enabling interns to gain real-world exposure and build confidence in applying their knowledge.

- **Offer Internship Opportunities:**
  To provide structured internship programs that prepare students and freshers for professional roles by enhancing their understanding of workplace expectations and responsibilities.
- **Mentorship by Industry Experts:**
  To connect learners with experienced professionals who can guide, mentor, and support them in their career journey.
- **Promote Innovation and Creativity:**
  To encourage learners to think creatively, solve real-world problems, and develop innovative digital solutions.
- **Improve Employability:**
  To increase job readiness by building strong portfolios and resumes through active participation in projects and training sessions.
- **Bridge the Academia-Industry Gap:**
  To align learning modules with current industry trends, ensuring that participants gain relevant and up-to-date knowledge and skills.
- **Foster a Tech-Savvy Community:**
  To build a collaborative environment where learners, mentors, and developers can share ideas, collaborate on projects, and grow together as a tech-driven community.
- **Support Career Growth:**
  To assist individuals in choosing the right career path by offering professional development support, interview preparation, and resume-building sessions.

## Manpower:

11-50 employees

Associated members are not publicly known

# SYSTEM STUDY

## Tools and Technologies Used

- **Frontend Development**
  - HTML5
  - CSS3
  - JavaScript
  - Bootstrap Framework
- **Backend Development**
  - PHP
  - MySQL (for database)
- **Design Tools**
  - Adobe XD / Figma (for wireframes and prototypes)
  - Canva (for basic graphic elements)
- **Other Tools**
  - Google Fonts
  - Font Awesome
  - Git (for version control)
  -

### 1. Key Web Pages Designed

### a. Homepage

- Featured carousel with trending magazine covers.
- Search bar for quick magazine lookup.
- Category-wise browsing (e.g., Health, Business, Kids, etc.).
- Promotional banners.

### b. Product Detail Page

- High-resolution magazine image.
- Subscription options (monthly, yearly).
- Description and reviews.
- “Add to Cart” and “Buy Now” buttons.

### c. Cart and Checkout Page

- Easy cart management.
- Secure checkout form.
- Payment gateway integration placeholders.

### d. User Dashboard

- View current and past subscriptions.
- Edit profile.

- Manage payment methods.

## 2. Design Highlights

- **Responsive Design:** Pages adapt smoothly to mobile, tablet, and desktop screens.
- **UI/UX Principles:** Clear call-to-action buttons, intuitive navigation, and consistent color palette.
- **SEO-Friendly Structure:** Clean URLs, semantic HTML, and fast-loading pages.
- **Security Features:** SSL for secure data transmission (conceptual implementation).

## 3. System Limitations

- No dynamic functionality or backend integration
- Content is hardcoded; cannot be updated without editing HTML.

## 4.User Interface Limitations

- Not responsive for all devices
- No animations, transitions, or accessibility features
- No dark mode or customization options

## 5. Audio Quality

- No sound or voice feedback in any project
- No media or interaction enhancements

## 6. Library Gaps

- Doesn’t use libraries like Bootstarp, Tailwind CSS, or react
- Lacks reusuable components or CSS frameworks

## 7. User-Friendly Interface

- Clear layout and menu
- Smooth navigation
- Mobile-first design
- Colour contrast and readable fonts
- Accessibilty support (keyboard navigation, screen reader friendly)

## 8.Design Highlights

- **Responsive Design:** Pages adapt smoothly to mobile, tablet, and desktop screens.
- **UI/UX Principles:** Clear call-to-action buttons, intuitive navigation, and consistent color palette.
- **SEO-Friendly Structure:** Clean URLs, semantic HTML, and fast-loading pages.
- **Security Features:** SSL for secure data transmission (conceptual implementation).

## 9. Challenges Faced

- Aligning design consistency across multiple pages.
- Ensuring cross-browser compatibility.
- Optimizing images for faster load time.
- Testing responsive behavior across devices.

## 10. Outcomes and Learnings

- Gained practical knowledge in web development using HTML, CSS, and JavaScript.
- Learned how to structure an eCommerce website effectively.
- Understood the importance of user experience and responsive design.

  Improved skills in debugging and browser testing.

# FEASIBILITY STUDY

## A: Technical Feasibility

Technical feasibility assesses whether the proposed system can be developed and implemented using the available technology, resources, and expertise. For the Spotify clone project, technical feasibility involves evaluating the programming languages, tools, and frameworks used to build the system.

### 1. Programming Language Used

- HTML- For structure of web pages
- CSS- For styling and layout
- JavaScript- For basic interactivity

### 2. Tools used

- Google Fonts
- Font Awesome
- Git (for version control)

### 3. Libraries & Framework

- None used in current versions
- All features implemented using vanilla HTML, CSS, and JavaScript

### 4. Performance

- Loading Time: Very fast (no heavy media or JavaScript)
- Compatibility: Works well on all modern browsers
- Stability: Stable for single-page applications
- Limitations: No dynamic data or asynchronous operation (no backend)

## B. Behavioural Feasibility

Behavioural feasibility evaluates how users will interact with the system, focusing on user satisfaction, ease of use, and the overall user experience.

### 1. User Interaction

- Buttons, links, and clickable elements are clearly visible and fuctional
- Basic but intuitive design with simple navigation

### 2. User Satisfaction

- Suitable for beginners and basic users
- Clear Structure of content and interaction
- Portfolio is personalized and easy to follow

### 3. Ad- free Experience

- Completely ad-free since it's self-developed
- No distractions or third-party interruptions

### 4. Ease of Use

- Clean layout with organized sections
- Buttons and links clearly labelled
- Minimalist design helps user focus on content

### 5. Behavioral challenges and Solutions

- Challenge: No form validation in contact or input fields
  Solution: Add required fields and JavaScript validation
- Challenge: Static and plan visuals
  Solution: Plan to add animations/transitions in next update
- Challenge: Basic interface might not appeal to advanced users
  Solution: Improve design with modern UI components and visuals

## C. Economic Feasibility

Economic feasibility analyzes the cost-effectiveness of the project, including development costs, potential savings, and revenue opportunities.

### 1. Development Cost

- 0 development cost
- No cost for tools, software, or hosting
- Self-coded with free resources

### 2. NO Potential Savings

- need to hire designers or developer
- need to buy templates or website builders
- Avoided subscription costs by using free platforms

### 3. Revenue Opportunities

- Portfolio site can attract freelance clients or job offers
- It can be enhanced and monetized through mobile apps or ads
- Landing page can be turned into a blog or monetized news platform

# PROJECT MONITORING SYSTEM

## A. Gantt chart

| Week | Task |
|---|---|
| Week 1 | Planning & Research, Market Study, Finalizing Objectives |
| Week 2 | UI/UX Design, Wireframing, Color & Layout Selection |
| Week 3 | HTML/CSS/JS Development, Content Writing, Functional Setup |
| Week 4 | Testing, Bug Fixing, Development, Presentation, Final report |

## Timeline Overview:

## A. Planning and Research

### 1. Project Scope and objectives

- The scope is to develop three static websites:
  A **Web** app for basic arithmetic operations
  A **Landing Page** highlighting current issue in India
  A **Portfolio Website** to showcase skills, resume and projects

- These projects are front-end only, built with HTML, CSS and JS

### Objectives:

- Built simple, functional, and visually clean web applications
- Improve front-end development skills and UI/UX understanding
- Create a personal brand through the portfolio website
- Present relevant social and academic topics in web format
- Ensure basic responsiveness and cross-device compatibility

## 2. Market Research

- **Global Digital Magazine Market Value**: Estimated over **$35 billion** (2024), expected to grow at a **CAGR of 5–7%** through 2030.
- Shift from print to **digital-first** or **digital-only** strategies.
- Growth driven by:
    - Increased mobile and tablet usage.
    - Subscription-based models (like Apple News+, Magzter).
    - Niche content (e.g., fashion, tech, business, health, culture).

Top Competitors / Magazine Website Examples

| Brand | Focus Area | Website |
|---|---|---|
| **National Geographic** | Science & Exploration | www.nationalgeographic.com |
| **Vogue** | Fashion & Lifestyle | www.vogue.com |
| **Wired** | Tech & Innovation | www.wired.com |
| **The Economist** | Business & Finance | www.economist.com |
| **TIME** | News & Culture | www.time.com |
| **Magzter** | Digital magazine platform | www.magzter.com |

## 3. User Requirements

- Identify the target audience and their needs.
- Gather feedback through surveys or interviews.

## B. Design Phase

## 1. UI/UX Design:

- Design intuitive and visually appealing interfaces for key pages (home, search, library, playlists, etc.).

- Ensure responsive design for compatibility across devices.

## 2. Wireframes and Mockups

- Create wireframes for each page to visualize the layout and structure.
- Develop detailed mockups to guide the development process.

## 3. Database and Architecture planning

### *1. High-Level Architecture*

```
pgsql
CopyEdit
Client (Web/Mobile)
     ↓
Frontend (React, Vue, etc.)
     ↓
API Layer (REST or GraphQL)
     ↓
Backend (Node.js / Django / Laravel)
     ↓
Database (SQL + optional NoSQL)
     ↓
CDN / Cache / Media Storage (e.g., Cloudinary, AWS S3)
```

---

***DATABASE DESIGN (Relational - e.g., PostgreSQL / MySQL)***

### *2. Core Tables*

#### *Users*

```
sql
CopyEdit
id (PK)
username
email
password_hash
role (admin, editor, subscriber)
profile_pic
created_at
```

#### *Articles*

```
sql
CopyEdit
id (PK)
title
slug
content (long text / rich HTML)
```

```
excerpt
cover_image_url
author_id (FK to Users)
published_at
status (draft/published/archived)
views_count
category_id (FK to Categories)
tags (JSON or join table)
```

### 🗂️ Categories

```
sql
CopyEdit
id (PK)
name
slug
description
```

### 🗂️ Tags

```
sql
CopyEdit
id (PK)
name
slug
```

### 🗂️ ArticleTags (Many-to-Many for Articles ↔ Tags)

```
sql
CopyEdit
article_id (FK)
tag_id (FK)
```

### 🗂️ Comments

```
sql
CopyEdit
id (PK)
article_id (FK)
user_id (FK)
comment_text
created_at
status (approved/pending)
```

### 🗂️ Subscriptions

```
sql
CopyEdit
id (PK)
user_id (FK)
plan_type (free/premium)
start_date
end_date
is_active
payment_status
```

### 📚 NewsletterSubscribers

```
sql
CopyEdit
id (PK)
email
subscribed_at
```

### 📚 Media

```
sql
CopyEdit
id (PK)
file_url
uploaded_by (FK to Users)
uploaded_at
type (image, video, doc)
```

---

## ⚙ 3. Technology Stack

| Layer | Options |
|---|---|
| Frontend | React, Vue.js, or HTML/CSS/JS |
| Backend | Node.js + Express / Django / Laravel |
| API Layer | REST or GraphQL |
| Database | PostgreSQL / MySQL (relational) |
| Media Store | AWS S3 / Cloudinary |
| Cache/CDN | Redis / Cloudflare |
| Auth | JWT / OAuth2 |
| Deployment | Docker + NGINX + CI/CD (GitHub Actions, Vercel, etc.) |
| Admin Panel | Strapi / Custom dashboard / WordPress backend |

---

## 🔄 4. Workflow / User Roles

| Role | Access |
|---|---|
| **Admin** | Full control: users, content, payments, settings |

| ***Role*** | ***Access*** |
|---|---|
| ***Editor*** | *Add/edit/publish articles, moderate comments* |
| ***Writer*** | *Create/edit articles (needs approval)* |
| ***Reader*** | *View public articles, subscribe, comment* |
| ***Subscriber*** | *Access premium content, newsletters* |

---

*� 5. Scalability Tips*

- *Use **pagination and lazy loading** for article lists.*
- *Store **media files separately** from DB (S3/Cloudinary).*
- *Use **Redis or Memcached** for caching popular articles.*
- *Implement **rate limiting** for APIs.*
- *Use **background jobs (e.g., Celery, Bull)** for newsletter or image processing.*

---

## C. Development Phase

- **Front-End Development**
  - ➢ Develop the front-end using HTML, CSS, and JavaScript.
  - ➢ Implement features like the home page, search functionality

- **Back-End Development**
  - ➢ Set up the server and database using suitable technologies (e.g., Node.js, Express.js, MongoDB).
  - ➢ Implement user authentication, management, and streaming functionalities.

- **Integration**
  - ➢ Integrate front-end and back-end components to ensure seamless functionality.
  - ➢ Implement APIs for music data and user interactions.

- **Testing and Debugging**
  - ➢ Perform initial testing and debugging to fix any issues in the code.

## D. Testing Phase

- **Functional Testing:**
  - ➢ Test all features to ensure they work as expected

- **Usability Testing:**
  - ➢ Conduct usability tests with real users to gather feedback.
  - ➢ Identify any issues in the user experience and make improvements.

- **Performance Testing:**
  - ➢ Test the website's performance, including loading times and responsiveness.
  - ➢ Optimize code and resources for better performance**.**

- **Security Testing:**
  - ➢ Ensure user data and interactions are secure.
  - ➢ Implement security measures to protect against common vulnerabilities.

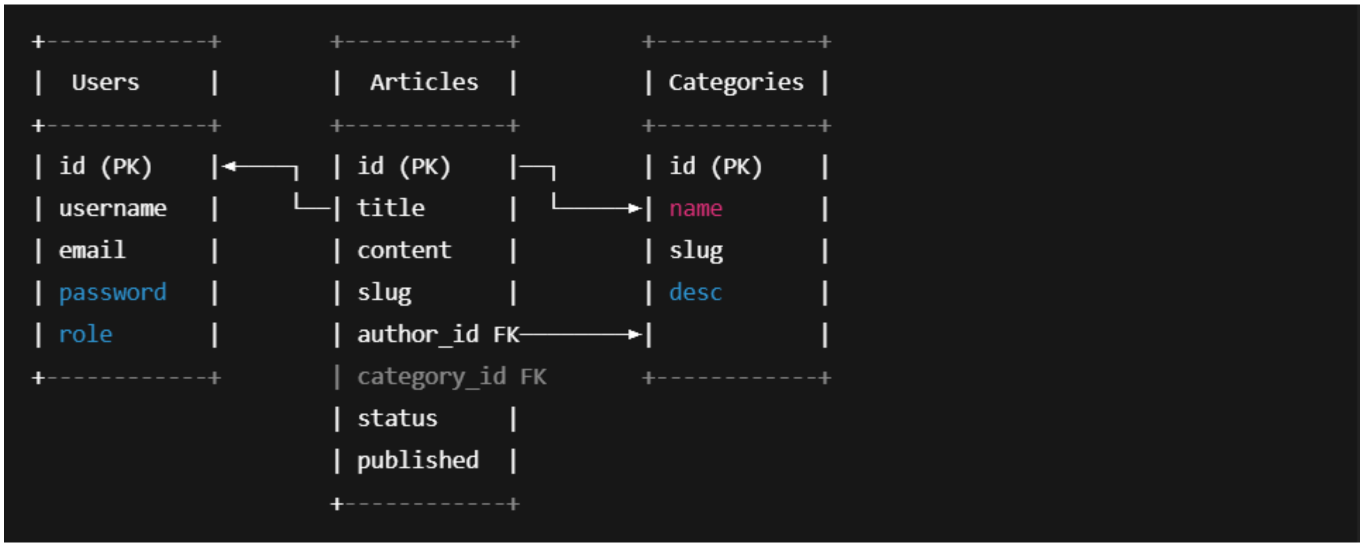
## E. Deployment Phase

- **Deployment Setup:**
  - ➢ Set up a hosting environment (e.g., AWS, Heroku).
  - ➢ Configure the server and database for production.

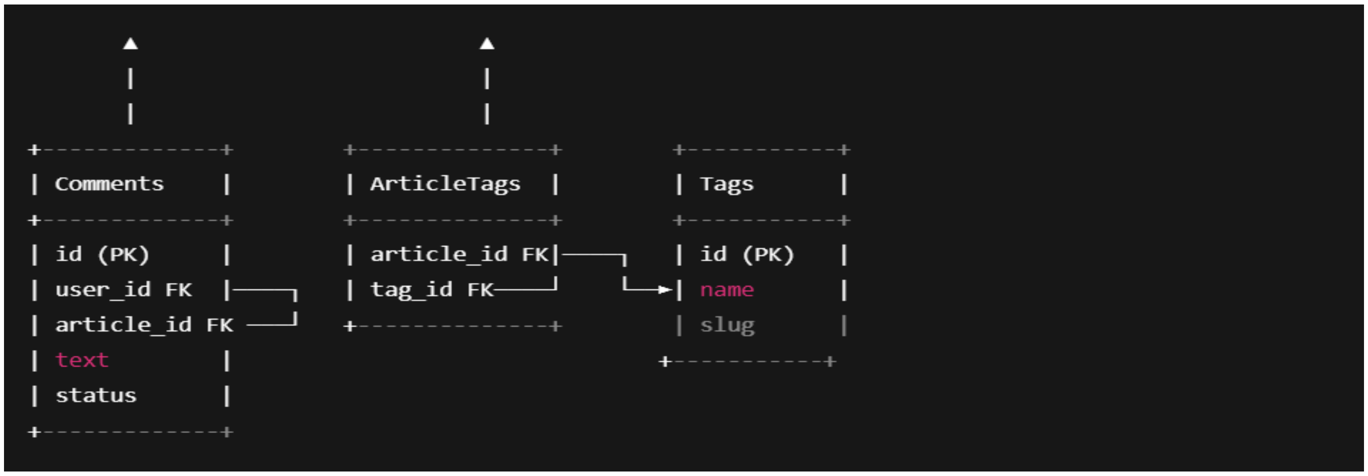
- **Deployment**
  - ➢ Deploy the website to the live environment.
  - ➢ Ensure all features are working correctly in the live setup’s

- **Monitoring and Maintenance**
  - ➢ Monitor the website for any issues or performance bottlenecks.
  - ➢ Gather user feedback and plan for future updates and improvements.

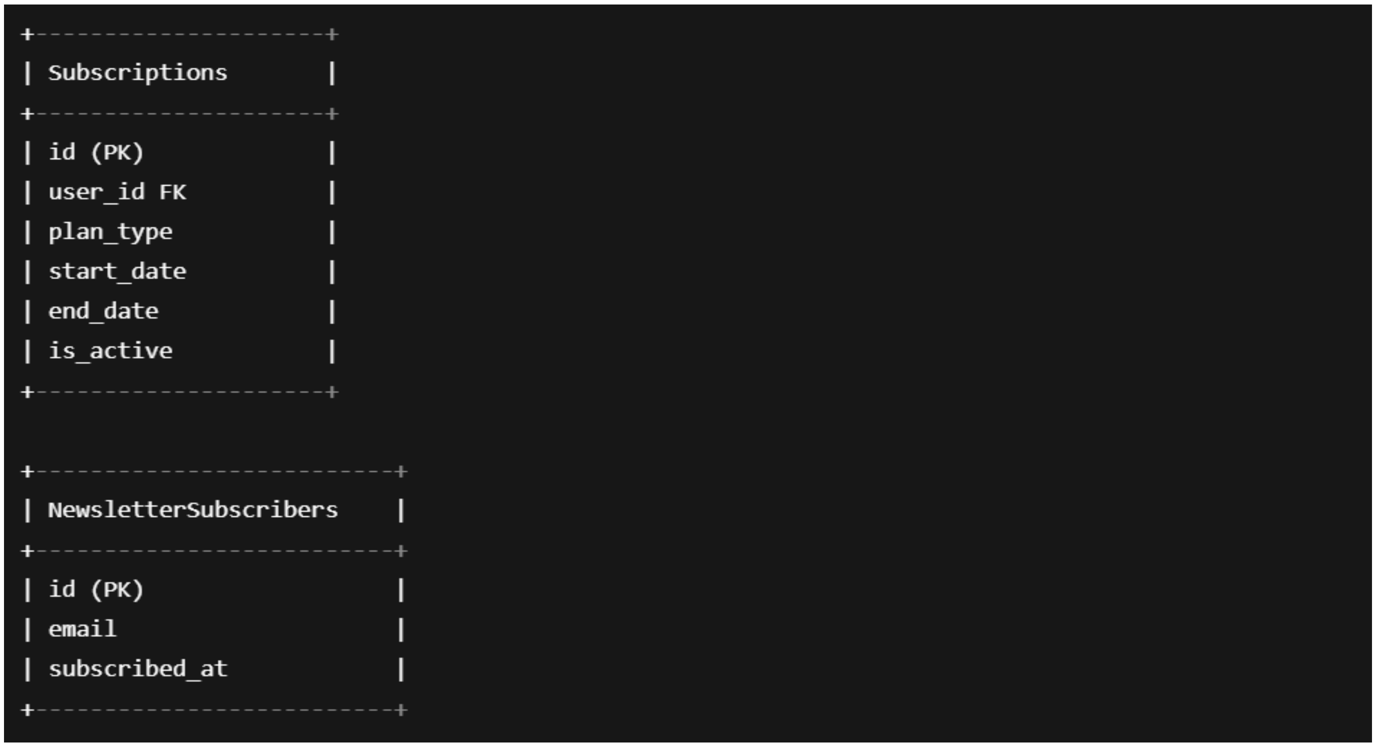
# ER Diagram: Magazine Website



```
+------------+        +------------+       +------------+
|  Users     |        |  Articles  |       | Categories |
+------------+        +------------+       +------------+
| id (PK)    |◄──┐    | id (PK)    |──┐    | id (PK)    |
| username   |   └──| title      |  └───►| name       |
| email      |        | content    |       | slug       |
| password   |        | slug       |       | desc       |
| role       |        | author_id FK──────►|            |
+------------+        | category_id FK     +------------+
                      | status     |
                      | published  |
                      +------------+
```



```
        ▲                       ▲
        |                       |
        |                       |
+-------------+      +-------------+      +-----------+
| Comments    |      | ArticleTags |      | Tags      |
+-------------+      +-------------+      +-----------+
| id (PK)     |      | article_id FK|───┐ | id (PK)   |
| user_id FK  |───┐  | tag_id FK───┘   └─►| name      |
| article_id FK ──┘  +-------------+      | slug      |
| text        |                          +-----------+
| status      |
+-------------+
```



```
+--------------------+
| Subscriptions      |
+--------------------+
| id (PK)            |
| user_id FK         |
| plan_type          |
| start_date         |
| end_date           |
| is_active          |
+--------------------+

+-------------------------+
| NewsletterSubscribers   |
+-------------------------+
| id (PK)                 |
| email                   |
| subscribed_at           |
+-------------------------+
| NewsletterSubscribers    |
+-------------------------+
| id (PK)                 |
| email                   |
| subscribed_at           |
+-------------------------+


+---------------+
| Media         |
+---------------+
| id (PK)       |
| file_url      |
| uploaded_by FK |
| type          |
| uploaded_at   |
+---------------+
```

Key Relationships:

- Users–Articles**: One-to-many (a user can write many articles).**
- Articles–Categories**: Many-to-one (each article belongs to one category).**
- Articles–Tags**: Many-to-many via `ArticleTags`.**
- Users–Comments–Articles**: Users comment on articles (many-to-many through comments).**
- Users–Subscriptions**: One-to-one or one-to-many depending on plans.**
- Users–Media**: Users can upload media.**

System Analysis: Magazine Website

---

## 1. 📌 Purpose of the System

To build a **user-friendly, scalable, and content-rich** magazine website that:

- Publishes categorized articles.
- Supports user subscriptions (free and premium).
- Allows user engagement (comments, likes, sharing).
- Provides content management for editors/admins.
- Generates revenue via ads, subscriptions, and e-commerce (if any).

---

## 2. 👥 Stakeholders

| Stakeholder | Role/Interest |
|---|---|
| Admin | System control, content approval |
| Editors/Writers | Create/edit/publish content |
| Readers | View articles, comment, share, subscribe |
| Advertisers | Promote through ads and banners |
| Developers | Build and maintain the website |
| Business Owners | Manage revenue, subscriptions, analytics |

---

## 3. ⚙ Functional Requirements

| Category | Functionality |
|---|---|
| 📝 Article Mgmt | Create, edit, delete, categorize articles |
| 👤 User Mgmt | Register, login, roles (reader/writer/admin) |

| Category | Functionality |
|---|---|
| 💬 Interaction | Comments, likes, shares |
| 💳 Subscriptions | Plan selection, payment, access control |
| 🔎 Search | Full-text search, filters by tags/category |
| 📧 Newsletter | Subscribe via email, send newsletters |
| 📈 Analytics | Track views, top articles, engagement |
| 🖼 Media Mgmt | Upload and embed images, videos, files |

---

## 4. 🔐 Non-Functional Requirements

| Type | Details |
|---|---|
| Performance | Fast loading, responsive design |
| Availability | 99.9% uptime, cloud-hosted |
| Scalability | Handle traffic spikes during new editions |
| Security | HTTPS, authentication, input validation |
| Maintainability | Modular, documented codebase |
| SEO | Optimized metadata, structured content |
| Accessibility | WCAG-compliant (for disabilities) |

---

## 5. 🏗 System Architecture

- **Frontend**: React / Vue / HTML-CSS-JS
- **Backend**: Node.js / Django / Laravel
- **Database**: PostgreSQL / MySQL
- **API**: REST / GraphQL
- **Media Storage**: AWS S3 / Cloudinary

- **Authentication**: JWT / OAuth2
- **CDN & Cache**: Cloudflare, Redis

---

## 6. 📤 Data Flow Diagram (DFD) – Level 1 (High Level)

```
css
CopyEdit
User --> [Login/Register] --> [Auth System]
     --> [Browse Articles] --> [Article DB]
     --> [Comment/Like] --> [Interaction DB]
     --> [Subscribe] --> [Payment Gateway +
Subscription DB]
Admin --> [Manage Content] --> [CMS / Admin Panel]
```

---

## 7. ⚠ System Constraints

- Limited internet bandwidth for some users → Optimize images, use lazy loading.
- Need for mobile support → Responsive or PWA needed.
- Budget constraints → Use open-source tools, scale later.
- Content moderation needed → Prevent spam comments or fake accounts.

---

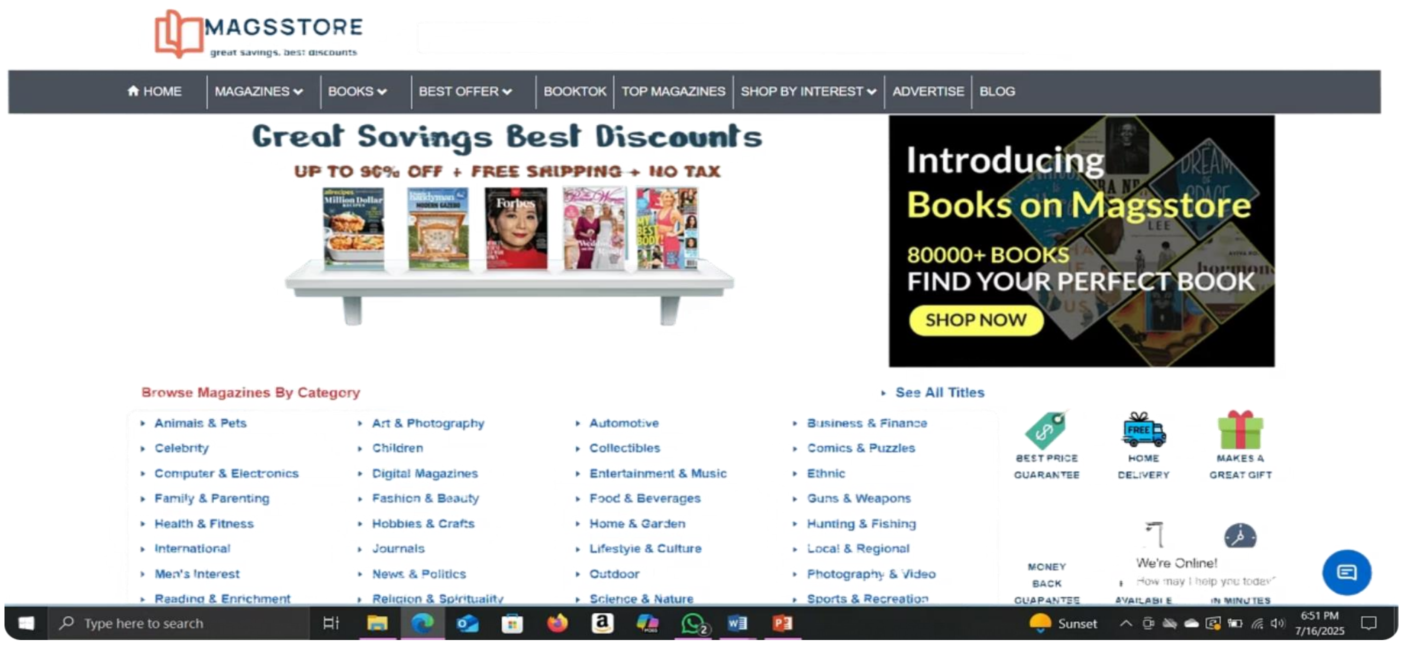
## 8. 📉 Risks

| Risk | Mitigation Strategy |
|---|---|
| Content theft (copy/paste) | Use anti-copy scripts, watermarks |
| Downtime during peak traffic | Use CDN, load balancing |
| Spam or malicious comments | CAPTCHA, comment moderation |
| Subscription fraud | Secure payments, validate transactions |
| Data loss | Regular backups, versioning |

---

## 9. ✅ Success Criteria

- Website loads in under 3 seconds.
- User base grows monthly.
- Stable article publishing with 99% uptime.
- Increased email subscribers and engagement.
- Revenue via ads/subscriptions meets targets.



Browse Magazines By Category

See All Titles

- Animals & Pets
- Celebrity
- Computer & Electronics
- Family & Parenting
- Health & Fitness
- International
- Men's Interest
- Reading & Enrichment
- Art & Photography
- Children
- Digital Magazines
- Fashion & Beauty
- Hobbies & Crafts
- Journals
- News & Politics
- Religion & Spirituality
- Automotive
- Collectibles
- Entertainment & Music
- Food & Beverages
- Home & Garden
- Lifestyle & Culture
- Outdoor
- Science & Nature
- Business & Finance
- Comics & Puzzles
- Ethnic
- Guns & Weapons
- Hunting & Fishing
- Local & Regional
- Photography & Video
- Sports & Recreation

System Design: Magazine Website

---

## 1. 🏗 High-Level Architecture

```
less
CopyEdit
[Client Browser]
       ↓
[Frontend (React/Vue/HTML)]
       ↓
[API Layer - REST or GraphQL]
       ↓
[Backend (Node.js / Django / Laravel)]
       ↓
+-------------------------------+
|  Relational DB (PostgreSQL)   |
|  Media Storage (AWS S3)       |
|  Cache (Redis)                |
|  Background Jobs (Bull/Celery)|
+-------------------------------+
       ↓
[Third-party services (Auth, Email, Payments)]
```

---

## 2. □ Core Components

| Component | Role |
|---|---|
| **Frontend** | UI for browsing, reading, interacting with content |
| **Backend API** | Business logic, routes, data processing |
| **Database** | Stores users, articles, categories, tags, comments, etc. |
| **Media Storage** | Store images, videos, downloadable content |
| **Authentication** | Manage login/signup, password hashing, sessions/tokens |
| **Authorization** | Role-based access control (Admin, Editor, Subscriber) |
| **Subscription System** | Manage premium access, plans, renewals |
| **Newsletter Engine** | Manages email campaigns, subscriptions |
| **Search Engine** | Optional: full-text search with ElasticSearch/Lucene |

| Component | Role |
|---|---|
| **Cache** | Cache popular articles, reduce DB load |
| **Admin Dashboard** | CMS for writers/editors/admins |

---

## 3. 📊 Data Flow Example: Viewing an Article

```
pgsql
CopyEdit
User → Frontend → API GET /articles/:slug
                        ↓
                 Backend Service
                        ↓
        Check Cache → Redis
             ↓        ↓
        [Cache Hit] [Cache Miss]
                        ↓
             Query Database → Articles, Tags, Author
                        ↓
           Compose Response + Store in Cache
                        ↓
                 Return to Frontend
                        ↓
                Render on Page
```

---

## 4. 💼 Suggested Folder Structure

```
bash
CopyEdit
/magazine-website
│
├── frontend/             # React/Vue app
│   └── components/
│   └── pages/
│
├── backend/
│   └── controllers/
│   └── models/
│   └── routes/
│   └── services/
│   └── utils/
│
├── database/
│   └── schema.sql
│   └── migrations/
```

```
│
├── media-storage/          # Linked to S3/Cloudinary
├── jobs/                   # Newsletter jobs, cleanup
tasks
├── tests/
└── .env
```

---

## 5. 📋 Database Summary (Core Tables)

- `Users`: Handles login, roles
- `Articles`: Stores content
- `Categories`, `Tags`: For organizing
- `Comments`: User engagement
- `Subscriptions`: Access control
- `Media`: Image and video files
- `NewsletterSubscribers`: Email list

---

## 6. 🛠 Technology Stack

| Layer | Technology Options |
|---|---|
| Frontend | React, Next.js, Vue, Tailwind CSS |
| Backend | Node.js + Express / Django / Laravel |
| API | REST / GraphQL |
| Database | PostgreSQL / MySQL |
| Auth | JWT / OAuth2 / Firebase Auth |
| Payments | Stripe / Razorpay |
| Media Storage | AWS S3 / Cloudinary |
| Cache | Redis |
| Search | ElasticSearch (optional) |
| Deployment | Docker, NGINX, Vercel/Netlify for frontend |
| Monitoring | LogRocket, Sentry, Prometheus |

---

## 7. 🔐 Security Considerations

- HTTPS with TLS.
- Passwords hashed with bcrypt/scrypt.
- Input validation & sanitization (XSS, SQLi).
- Rate limiting and reCAPTCHA on forms.
- Role-based authorization for admins/editors.
- Audit logging for admin activity.

---

## 8. 📱 Performance & Scalability

| Area | Optimization Tactics |
|---|---|
| Images | Lazy load, CDN (Cloudflare, Imgix) |
| Articles | Paginate, cache with Redis |
| Search | Debounced frontend search, ElasticSearch |
| Newsletter | Use background queues (Celery, Bull) |
| Mobile users | PWA support, responsive design |

---

## 9. 📌 Future Scalability Options

- Microservices architecture
- Content Delivery Network (CDN) for faster content globally
- GraphQL federation for scaling API access
- Multilingual support (i18n)
- AI-generated article summaries or voice playback (TTS)

---

## ✅ 1. Purpose of System Testing

To ensure the entire magazine website — including frontend, backend, APIs, and databases — works as intended in an integrated environment and is ready for production.

---

### *▯ 2. Types of Testing and Their Objectives*

| Testing Type | Objective |
|---|---|
| **Functional Testing** | Ensure core functions like login, article view, and subscriptions work. |
| **UI/UX Testing** | Verify user interface, responsiveness, and usability. |
| **Performance Testing** | Check load time, scalability, and speed under high traffic. |
| **Security Testing** | Identify vulnerabilities like XSS, CSRF, SQLi. |
| **Compatibility Testing** | Confirm the website works across devices, browsers, and screen sizes. |
| **Database Testing** | Verify data integrity, relationships, and query performance. |
| **Regression Testing** | Ensure new features don't break existing ones. |
| **Integration Testing** | Test APIs, database, media storage, and payment systems together. |
| **Accessibility Testing** | Ensure WCAG compliance for disabled users. |
| **User Acceptance Testing (UAT)** | Final round of testing with real users or stakeholders. |

---

## 🛠 *3. Functional Test Cases (Examples)*

| Feature | Test Case Description | Expected Result |
|---|---|---|
| **Login** | Valid/Invalid credentials | Success/Error message shown |
| **View Article** | Click on article title | Full article loads with content |
| **Search** | Search by keyword/tag/category | Relevant results returned |
| **Comment System** | Add comment as logged-in user | Comment appears under article |
| **Subscription** | Purchase a plan with valid payment | Access to premium content |
| **Admin Panel** | Admin edits an article | Article updates reflect on frontend |

---

## 🚀 *4. Performance Testing Scenarios*

| Scenario | Tool | Target Metric |
|---|---|---|
| 1000 concurrent users | JMeter / Locust | Response < 3 sec; No server crash |
| Article page load time | Lighthouse / WebPageTest | Under 2 seconds |
| Media loading (images/videos) | Chrome DevTools | Use lazy load; Image < 200KB |

---

## 🔐 *5. Security Testing Checklist*

| Vulnerability | How to Test | Tool |
|---|---|---|
| SQL Injection | Input `' OR 1=1 --` in form fields | SQLMap, manual |
| XSS | Try `<script>alert(1)</script>` in comments | OWASP ZAP |

| Vulnerability | How to Test | Tool |
|---|---|---|
| CSRF | Check unauthorized form submissions | Postman, Burp Suite |
| Password Handling | Check for encryption (bcrypt/scrypt) | Code review |
| HTTPS/SSL | Verify certificate and secure protocols | Qualys SSL Labs |

---

### *▯ 6. Database Testing*

| Test Case | Expected Result |
|---|---|
| Foreign key integrity | No orphan records (e.g., article without author) |
| Article insert/update/delete | Reflected correctly on frontend |
| Subscriptions with expiry dates | Expired ones set to inactive |

---

### *🌍 7. Cross-Browser / Device Testing*

Test on:

- Browsers: Chrome, Firefox, Safari, Edge
- Devices: iPhone, Android, iPad, Desktop
- Tools: BrowserStack, LambdaTest, Chrome DevTools

---

### *▯▯ 8. User Acceptance Testing (UAT)*

Performed by stakeholders or end-users to verify:

- Navigation feels natural.
- Content is easy to access.
- Payments work smoothly.
- UI is clean, responsive, accessible.

Use feedback to fix minor bugs and polish UI.

---

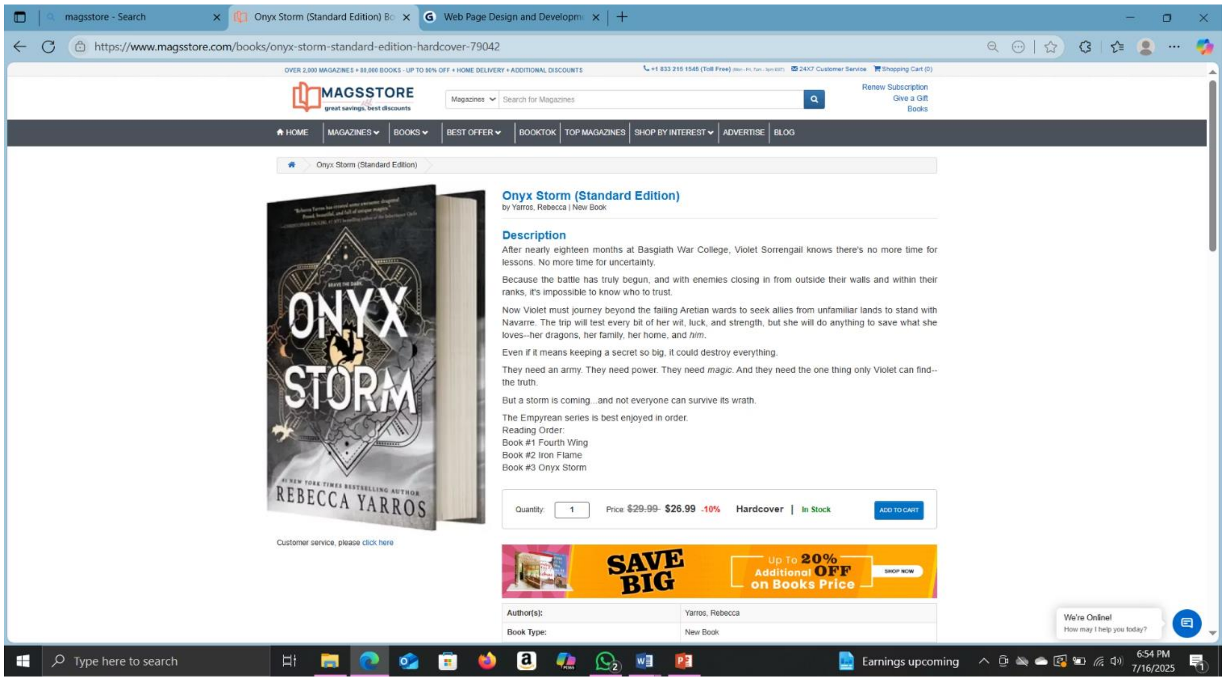
## 📈9. *Tools for System Testing*

| Purpose | Tools |
|---|---|
| Functional Testing | Selenium, Cypress |
| Load Testing | Apache JMeter, Locust |
| Security Testing | OWASP ZAP, Burp Suite |
| Accessibility Testing | Axe, Wave, Lighthouse |
| UI Testing | BrowserStack, Percy, Playwright |
| API Testing | Postman, Insomnia |

---

## ✅ 10. Final Sign-Off Criteria

- All **critical and high-severity bugs** are resolved.
- **Functional & non-functional** test cases pass.
- Site works across major browsers and devices.
- No **security vulnerabilities** remain open.
- UAT completed with positive feedback.


magsstore - Search
Onyx Storm (Standard Edition) Bo
Web Page Design and Developm
https://www.magsstore.com/books/onyx-storm-standard-edition-hardcover-79042
OVER 2,000 MAGAZINES • 80,000 BOOKS - UP TO 90% OFF • HOME DELIVERY • ADDITIONAL DISCOUNTS
+1 833 215 1545 (Toll Free)
24X7 Customer Service
Shopping Cart (0)
MAGSSTORE
great savings, best discounts
Magazines
Search for Magazines
Renew Subscription
Give a Gift
Books
HOME
MAGAZINES
BOOKS
BEST OFFER
BOOKTOK
TOP MAGAZINES
SHOP BY INTEREST
ADVERTISE
BLOG
Onyx Storm (Standard Edition)
ONYX STORM
REBECCA YARROS
Quantity: 1
ADD TO CART
SAVE BIG
Up To 20% Additional OFF on Books Price
SHOP NOW
We're Online!
How may I help you today?
Type here to search
Earnings upcoming
6:54 PM
7/16/2025

Onyx Storm (Standard Edition)
by Yarros, Rebecca | New Book

Description

After nearly eighteen months at Basgiath War College, Violet Sorrengail knows there's no more time for lessons. No more time for uncertainty.

Because the battle has truly begun, and with enemies closing in from outside their walls and within their ranks, it's impossible to know who to trust.

Now Violet must journey beyond the failing Aretian wards to seek allies from unfamiliar lands to stand with Navarre. The trip will test every bit of her wit, luck, and strength, but she will do anything to save what she loves--her dragons, her family, her home, and *him*.

Even if it means keeping a secret so big, it could destroy everything.

They need an army. They need power. They need *magic*. And they need the one thing only Violet can find--the truth.

But a storm is coming...and not everyone can survive its wrath.

The Empyrean series is best enjoyed in order.
Reading Order:
Book #1 Fourth Wing
Book #2 Iron Flame
Book #3 Onyx Storm

Price: ~~$29.99~~ $26.99 -10% Hardcover | In Stock

Customer service, please click here

| | |
|---|---|
| Author(s): | Yarros, Rebecca |
| Book Type: | New Book |

# System Implementation Plan: Magazine Website

---

## 1. □ Implementation Overview

System implementation is the phase where the magazine website, after being developed and tested, is deployed to a live environment. It involves:

- Hosting setup
- Database deployment
- Domain connection
- Content population
- Admin and user training
- Go-live and post-launch support

---

## 2. 🚛 Implementation Steps

---

### 🔹 A. Hosting & Deployment Setup

| Step | Description | Tools / Services |
|---|---|---|
| Choose hosting platform | Cloud or shared hosting | AWS, DigitalOcean, Vercel, Netlify |
| Set up server | Web server, runtime, SSL | NGINX, Apache, Node.js, Docker |
| Setup CI/CD pipeline | Automate deployment | GitHub Actions, Jenkins, GitLab CI |
| Install SSL certificate | HTTPS for secure access | Let's Encrypt, Cloudflare |

---

### 🔹 B. Frontend Deployment

| Step | Description | Tools |
|---|---|---|
| Build static files | Compile frontend code | React/Vue CLI |
| Upload to hosting/CDN | Deploy frontend | Netlify, Vercel, S3 |
| Test responsiveness | Ensure mobile compatibility | Chrome DevTools, BrowserStack |

---

## 🔹 C. Backend & Database Setup

| Step | Description | Tools / Tech |
|---|---|---|
| Setup database | Create schema and seed initial data | PostgreSQL/MySQL |
| Migrate data | Move existing content if any | SQL scripts, CSV import |
| Deploy backend | API server, admin panel | Node.js, Django, Laravel |
| Configure environment | Store secrets, keys, DB info | `.env`, AWS Parameter Store |

---

## 🔹 D. Third-Party Integrations

| Feature | Integration Needed | Services |
|---|---|---|
| Payments | Subscription plans | Stripe, Razorpay |
| Newsletter | Mailing list integration | Mailchimp, Sendinblue |
| Media Storage | Store images, videos | Cloudinary, AWS S3 |
| Analytics | Traffic and user behavior | Google Analytics, Hotjar |
| SEO Monitoring | Search visibility | Google Search Console, Ahrefs |

---

## 🔹 E. Security Configuration

| Action | Tool / Tech Used |
|---|---|
| HTTPS setup | Let's Encrypt, Cloudflare |
| Auth & authorization | JWT, OAuth2 |
| Password hashing | Bcrypt/Scrypt |
| Rate limiting | Express-rate-limit, NGINX |
| WAF & firewall | Cloudflare, AWS WAF |

---

## 3. 👥 Training & Documentation

| Type | Audience | Content |
|---|---|---|
| Admin training | Admins/Editors | How to publish, moderate, manage users |
| User guide | General users | Navigating the site, subscribing |
| Technical docs | DevOps/Developers | Deployment, config, and scaling |

Deliverables:

- User Manual (PDF/Web)
- Admin Panel Walkthrough
- FAQ or Help Section on Website

---

## 4. ✅ Go-Live Checklist

| Item | Status |
|---|---|
| ✅ Final testing on production | Completed |
| ✅ DNS configured & domain pointing | Completed |
| ✅ CDN & SSL active | Completed |
| ✅ Backup plan enabled | Completed |
| ✅ Monitoring & logging enabled | Completed |
| ✅ Admin accounts created | Completed |
| ✅ Content preloaded or published | Completed |

---

## 5. 📈 Post-Launch Activities

| Task | Description |
|---|---|
| Bug monitoring | Use tools like Sentry, LogRocket |
| Analytics tracking | GA4, Hotjar setup |
| Feedback collection | User surveys, contact form |

| Task | Description |
|---|---|
| SEO verification | Google/Bing Webmaster Tools |
| Daily backups | DB and media content |
| Update cycle planning | New features and content |

---

## 6. 📄 Documentation (Optional but Recommended)

- `README.md` – Setup instructions
- `config.yaml` – Server & environment settings
- `deployment.md` – Steps for CI/CD or manual deployment
- `admin_guide.pdf` – For content team or editors
- `security_policy.md` – Auth & data protection rules

## Documentation

Documentation included detailed reports of planning, system analysis, design, testing, and deployment. It covered project objectives, tools used, code structure, test cases, user feedback, and improvements. This helps future developers understand, maintain, and enhance the projects efficiently.

## Components

The web application clone application consists of the following key components:

- **Front-end**: HTML, CSS, JavaScript
- **Back-end**: Node.js, Express.js
- **Database**: MySQL/PostgreSQL
- **External APIs**: Music metadata services, authentication providers

## Technology Stack

- **Front-end**: HTML5, CSS3, JavaScript (ES6+)
- **Back-end:** Node.js, Express.js
- **Database**: MySQL/PostgreSQL
- **Development Tools**: Visual Studio Code, Git/GitHub
- **Hosting Platform**: AWS/Azure/GCP

- ## System Requirements

## Hardware Requirements

- **Server:** Dual-core processor, 4GB RAM, SSD storage
- **Network:** Stable internet connection

## Software Requirements

- **Operating System:** Linux/Windows Server
- **Web Server**: Apache/Nginx
- **Database:** MySQL/PostgreSQL
- **Programming Languages:** HTML, CSS, JavaScript (Node.js)
- **Additional Software:** Libraries, APIs, SDKs

### Text editor

- **Visual Studio Code**: Integrated development environment for coding, debugging, and version control.

### Version Control

- **Git**: Distributed version control system for tracking changes, managing codebase, and facilitating collaboration.

### Hosting Platform

- **AWS/Azure/GCP:** Cloud-based hosting platforms for scalable deployment, resource management, and performance optimization.

### System Design File/Data Design

- **HTML Files:** Structured web pages for user interface elements.
- **CSS Files**: Stylesheets for visual design and layout.
- **JavaScript Files:** Client-side scripting for interactive features and user actions.
- **Image and Media Files:** Media assets for displaying album covers, artist images, and user avatars.
- **Other Files:** Configuration files, documentation, and third-party libraries.

### Database Storage

- Utilization of MySQL/PostgreSQL for structured data storage.
- File storage for media content (e.g., music files, images).
- Cloud storage options for scalable data management.

- **Implementation**

### Development Environment Setup

- Installation and configuration of necessary software components (Node.js, Express.js, databases, etc.).
- Setup of development environment in Visual Studio Code.

- Integration with version control system (Git/GitHub) for collaborative development.

### Coding Standards

- Adherence to coding conventions and best practices for HTML, CSS, JavaScript, and Node.js.
- Documentation of coding standards to ensure consistency and maintainability of codebase.

### Version Control Workflow

- Branching strategy for feature development, bug fixes, and release management.
- Commit practices, pull request reviews, and merge strategies to maintain code quality and integrity.

- **Testing**

### Test Strategy

• Definition of test objectives, scope, and methodologies (unit testing, integration testing, system testing).

• Selection of testing tools and frameworks (Mocha, Chai, Jest) for automated testing.

### Test Cases and Results

• Development and execution of test cases covering functional, performance, and security aspects.

• Analysis of test results, identification of defects, and prioritization for resolution.

### Performance Testing

• Load testing to evaluate application performance under expected and peak loads.

• Optimization measures to enhance scalability, responsiveness, and resource utilization.

**Security Testing**

• Vulnerability assessments and penetration testing to identify and mitigate security risks.

• Implementation of encryption, authentication, and access control mechanisms.

- ## Deployment

### Deployment Environment

• Configuration of production environment settings (AWS/Azure/GCP).

• Setup of web servers (Apache/Nginx) and database servers (MySQL/PostgreSQL) for deployment.

### Deployment Strategy

• Continuous integration and deployment pipelines (CI/CD) for automated build, test, and deployment processes.

• Rollout strategies (blue-green deployment, canary releases) for minimizing downtime and ensuring application availability.

### Configuration Management

• Management of configuration files, environment variables, and application settings across different deployment environments.

• Monitoring and logging setup for performance monitoring, error tracking, and troubleshooting.

- ## User Guide

### System Access

• User registration and authentication mechanisms.

• Account management (password reset, profile settings) for registered users.

### Features Overview

• Overview of core features (search songs, playlist management, user preferences).

• Usage instructions for accessing and utilizing music streaming functionalities.

### Usage Instructions

• Step-by-step guides for common user tasks (creating playlists, liking songs, exploring recommendations).

• Troubleshooting tips and FAQs for resolving common issues.

- **Maintenance and Support**

### Monitoring and Maintenance Procedures

• Implementation of monitoring tools for real-time performance metrics and system health checks.

• Scheduled maintenance windows and updates for ensuring system reliability and security.

### Bug Tracking and Resolution

• Bug reporting and tracking using issue management tools (Jira, Trello). • Prioritization, assignment, and resolution of reported issues to maintain application quality.

### Backup and Recovery

• Implementation of backup strategies (regular database backups, file system snapshots) for data protection and disaster recovery.

• Procedures for restoring data and recovering from system failures or data breaches.

# SCOPE OF THE PROJECT

The scope of this project includes the design and development of three static web applications: a Landing Page on current issues, and a Personal Portfolio website. Each project is built using HTML, CSS, and JavaScript to demonstrate front-end skills. The focus is on clean UI design, responsive layouts, and functional features. The project serves educational purposes, showcasing beginner-level development, user interaction, and personal branding in a web environment.

## 1. Functional Scope

- Provide informative content on social topics in the landing page.
- Showcase skills, projects, and resume in the portfolio.
- Enable download of resume and navigation across website sections.

## 2. Technical Scope

- Built using HTML, CSS, and JavaScript (no backend).
- Internal CSS and inline JavaScript for interactivity .
- Deployed using static hosting platforms like GitHub Pages or Netlify.
- Developed using VS Code with optional Live Server extension.

## 3. User Experience (UX) Scope

- Simple, intuitive interface with clean layout.
- Responsive design for desktop, tablet, and mobile users.
- Easy navigation with clear headings and well-structured content.
- Downloadable resume and easy contact visibility for convenience.

## 4. Security and Compliance Scope

- No personal data collected—minimizing data risk.
- Avoided external scripts or risky code (e.g., minimized `eval ()` use).
- Static project; no user authentication or backend forms involved.
- Adheres to basic accessibility guidelines (readable text, clear contrast).

## 5. Scalability and Performance Scope

- Lightweight and fast-loading design for performance.
- Can be scaled by adding more projects, blog sections
- Compatible with future enhancements using external CSS/JS libraries.
- Optimized for cross-browser functionality.

## 6. Feature Enhancement Scope

- Future scope includes adding:
  - Contact forms with validation
  - Dark/Light mode toggle
  - Project filter/search functionality
  - More interactive animations or transitions

## 7. Content Management scope

- Currently managed through manual editing of HTML.
- Future enhancement could include integration with CMS or JSON data.
- Easy to update text, links, and project descriptions in code.

## 8. Analytics and Insights Scope

- Currently no analytics tools integrated.
- Future scope to add:
  - Google Analytics for tracking visitors
  - Resume download counter
  - Button interaction tracking

## 9. Community and Interactive Scope

- No social sharing features currently included.
- Can add:
  - Social media icons in the portfolio
  - Comment/discussion section in the landing page
  - Contact form for user feedback or job inquiries

# BIBLIOGRAPHY

Designing and developing a web interface for Magsstore.com has provided valuable experience in eCommerce development. The project enhanced both my technical skills and design thinking, while also helping me understand how real-world businesses function online. This knowledge will serve as a solid foundation for more advanced projects in the future.

- https://www.magsstore.com
- W3Schools – HTML/CSS/JS Documentation
- Bootstrap Documentation
- Figma.com for wire framing