- **Website Development Roadmap for MiPress**
- **1. Website Structure**

The website will have a sidebar navigation similar to the provided design, structured into different sections for various user roles.

1. **User Menu (For all users)**
- **Home** – Dashboard showing recent activities
- **Manage Accounts** – User role management (if applicable)
- **Change Password** – Password reset functionality
- **Edit Profile** – Update personal details
- **Logout** – End session

2. **Submissions Menu (For authors)**
- **Submit Manuscript** – Form to upload a new submission
- **Display Submitted Manuscripts** – List of manuscripts submitted by the user
- **Display Co-Authored Manuscripts** – List manuscripts where the user is a co-author
-

3. **Reviewers Menu (For reviewers)**
- **Reviews** – List of assigned papers for review, categorized as 'Pending', 'Finished', or 'Declined'
-

4. **Academic Editor Menu (For editors & assistant editors)**
- **Editor Profile** – Manage personal details & preferences
- **Decisions** – List of manuscripts awaiting a decision
- **Pre-check Decisions** – Early-stage screening before full peer review
- **Special Issues** – Manage themed collections (if needed in future)

- **2. User Roles & Permissions**

Role	**Description & Permissions**
Author	Submit papers, suggest reviewers, track submissions, receive notifications on review decisions.
Reviewer	Review assigned papers, submit feedback
Assistant Editor	Manage initial submissions, send to Associate Editor, oversee revisions.
Associate Editor	Review submissions forwarded by Assistant Editor and decide next steps.
Editor-in-Chief	Make final acceptance/rejection decisions, oversee journal operations.
Admin	Manage users, journals, volumes, settings, and website maintenance.

- **3. Step-by-Step Website Process**

5. **A. Paper Submission Workflow (Author Side)**

1. User logs in as an **Author**.
2. Navigates to **"Submit Manuscript"**.
3. Fills in:

   - **Title, Abstract, Keywords**
   - **Upload File (PDF, DOCX, or LaTeX)**
   - **Journal Selection**
   - **Suggested Reviewers (Name, Email, Affiliation, Reason)**
4. Clicks "Submit" → Paper is stored in **Submissions Table** with status = 'Submitted'.
5. Author receives confirmation email.

---

6. **B. Editorial & Review Workflow (Editor & Reviewers Side)**
7. **Assistant Editor logs in**.
8. Reviews metadata & suggested reviewers.
9. **Forwards submission to Associate Editor**.
10. Associate Editor reviews the submission and either:
    - **Assigns reviewers** (either from suggested reviewers or selects manually), OR
    - **Declines the submission** if it does not meet journal standards.
11. If sent for review, reviewers receive an email invitation → Log in & access the manuscript.
12. Reviewers submit feedback → Stored in **Reviews Table**.
13. Associate Editor reviews feedback and sends the decision to **Editor-in-Chief**.
14. Editor-in-Chief makes the final decision:
    - **Accept** → Moves to **PublishedPapers Table**.
    - **Minor/Major Revisions** → Author is notified, allowed to submit revision.
    - **Reject** → Submission is closed.
15. Final notifications are sent to all parties.

---

16. **C. Continuous Publishing Workflow (After Acceptance)**

1. Accepted paper moves from **Submissions Table → PublishedPapers Table**.
2. Assigned to the **Current Volume of the Journal**.
3. DOI is generated and stored.
4. Paper is immediately available under the "Published Papers" section.

- **4. Database-Frontend Mapping**

Website Feature	Maps to Database Table(s)
User Registration/Login	Authors (also used for reviewers)
Submit Manuscript	Submissions
Suggest Reviewers	SuggestedReviewers
Assign Reviewers	Reviews
Track Submissions	Submissions, SubmissionAuthors
Editorial Decisions	Submissions, Reviews
Published Papers	PublishedPapers, Volumes

- **5. Technical Considerations**

17. **A. Backend Framework**

- **Recommended:** Django, Laravel, or Node.js with Express
- **Database:** MySQL or PostgreSQL
- **Authentication:** JWT-based login system

18. B. Frontend Framework

- **Recommended:** React.js or Vue.js
- **UI Components:** Bootstrap or TailwindCSS
- **Editor Dashboard:** Dynamic charts for tracking submissions

---

- **6. Additional Features for Future Consideration**

✅ **Automated DOI Generation** for published papers. ✅ **Email Notifications & Reminders** (Review assignments, submission updates, etc.). ✅ **Plagiarism Detection** integration. ✅ **Version Control for Revisions** (Allow authors to track past versions). ✅ **Analytics for Editors** (Review times, decision statistics, etc.).

This roadmap provides a **clear step-by-step guide** for the website developer to follow, ensuring a structured and well-integrated publishing system. 🚀

Optimized Database Tables, Attributes, and Relationships

1. Authors Table (Stores all registered users: authors & reviewers)

```
CREATE TABLE Authors (
   AuthorID INT PRIMARY KEY AUTO_INCREMENT,
   Username VARCHAR(50) UNIQUE NOT NULL,
   Email VARCHAR(255) UNIQUE NOT NULL,
   Name VARCHAR(255) NOT NULL,
   Affiliation VARCHAR(255) NULL,
   Country VARCHAR(100) NULL,
   ORCID_ID VARCHAR(50) UNIQUE NULL
);
```

- **One-to-Many** → Authors submit multiple papers (Submissions).
- **One-to-Many** → Authors can be assigned as **reviewers** (Reviews).
- **One-to-Many** → Authors can be **editors** (JournalEditors).

2. Journals Table (Manages different journals)

```
CREATE TABLE Journals (
   JournalID INT PRIMARY KEY AUTO_INCREMENT,
   Name VARCHAR(255) UNIQUE NOT NULL,
   ISSN VARCHAR(50) UNIQUE NOT NULL,
   EditorInChiefID INT,
   Description TEXT NULL,
   FOREIGN KEY (EditorInChiefID) REFERENCES Authors(AuthorID) ON DELETE SET NULL
);
```

▯ **One-to-Many** → A journal has multiple **volumes** (Volumes).
▯ **One-to-Many** → A journal has multiple **submissions** (Submissions).
▯ **One-to-One** → Each journal has **one Editor-in-Chief**.

3. Volumes Table (Handles journal publishing volumes)

```
CREATE TABLE Volumes (
   VolumeID INT PRIMARY KEY AUTO_INCREMENT,
   JournalID INT NOT NULL,
   Year INT NOT NULL,
   StartDate DATE NOT NULL,
   EndDate DATE NULL,  -- NULL means volume is still active
   FOREIGN KEY (JournalID) REFERENCES Journals(JournalID) ON DELETE CASCADE,
   UNIQUE (JournalID, Year)  -- Ensures one volume per year per journal
);
```

One-to-Many → A volume contains multiple **published papers** (PublishedPapers).

4. Submissions Table (Tracks manuscript submissions)

```
CREATE TABLE Submissions (
    SubmissionID INT PRIMARY KEY AUTO_INCREMENT,
    Title VARCHAR(255) NOT NULL,
    Abstract TEXT NOT NULL,
    FilePath VARCHAR(255) NOT NULL,
    Keywords TEXT NULL,
    Status ENUM('Submitted', 'Under Review', 'Accepted', 'Rejected', 'Revision Requested') NOT
NULL DEFAULT 'Submitted',
    SubmissionDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    RevisedDate TIMESTAMP NULL,
    AcceptedDate TIMESTAMP NULL,
    JournalID INT NOT NULL,
    CorrespondingAuthorID INT NOT NULL,
    FOREIGN KEY (JournalID) REFERENCES Journals(JournalID) ON DELETE CASCADE,
    FOREIGN KEY (CorrespondingAuthorID) REFERENCES Authors(AuthorID) ON DELETE CASCADE
);
```

□ **One-to-Many** → A submission can have multiple **authors** (SubmissionAuthors).
□ **One-to-Many** → A submission can have multiple **reviews** (Reviews).
□ **One-to-Many** → Authors can suggest **reviewers** (SuggestedReviewers).

5. SubmissionAuthors Table (Handles multi-author submissions)

```
CREATE TABLE SubmissionAuthors (
    SubmissionAuthorID INT PRIMARY KEY AUTO_INCREMENT,
    SubmissionID INT NOT NULL,
    AuthorID INT NULL,  -- NULL for unregistered authors
    Name VARCHAR(255) NOT NULL,
    Email VARCHAR(255) NOT NULL,
    Affiliation VARCHAR(255) NULL,
    IsCorresponding BOOLEAN DEFAULT FALSE,
    AuthorOrder INT NOT NULL,
    FOREIGN KEY (SubmissionID) REFERENCES Submissions(SubmissionID) ON DELETE CASCADE,
    FOREIGN KEY (AuthorID) REFERENCES Authors(AuthorID) ON DELETE SET NULL
);
```

Many-to-One → Each author is linked to **one submission**.

6. SuggestedReviewers Table (Stores author-suggested reviewers)

```
CREATE TABLE SuggestedReviewers (
    SuggestedReviewerID INT PRIMARY KEY AUTO_INCREMENT,
    SubmissionID INT NOT NULL,
    Name VARCHAR(255) NOT NULL,
    Email VARCHAR(255) NOT NULL,
    Affiliation VARCHAR(255) NULL,
    Reason TEXT NULL,
    FOREIGN KEY (SubmissionID) REFERENCES Submissions(SubmissionID) ON DELETE CASCADE
);
```

One-to-Many → A submission can have multiple suggested reviewers.

7. JournalEditors Table (Handles editorial roles)

```
CREATE TABLE JournalEditors (
    JournalEditorID INT PRIMARY KEY AUTO_INCREMENT,
    JournalID INT NOT NULL,
    AuthorID INT NOT NULL,
    Role ENUM('Reviewer', Associate Editor') NOT NULL,
    FOREIGN KEY (JournalID) REFERENCES Journals(JournalID) ON DELETE CASCADE,
    FOREIGN KEY (AuthorID) REFERENCES Authors(AuthorID) ON DELETE CASCADE
);
```

Many-to-One → Each journal has multiple **editors**.

8. Reviews Table (Tracks peer review process)

```
CREATE TABLE Reviews (
    ReviewID INT PRIMARY KEY AUTO_INCREMENT,
    SubmissionID INT NOT NULL,
    ReviewerID INT NOT NULL,  -- Reviewer is an author
    Comments TEXT NULL,
    Recommendation ENUM('Accept', 'Minor Revision', 'Major Revision', 'Reject') NOT NULL,
    ReviewDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (SubmissionID) REFERENCES Submissions(SubmissionID) ON DELETE CASCADE,
    FOREIGN KEY (ReviewerID) REFERENCES Authors(AuthorID) ON DELETE CASCADE
);
```

- **One-to-Many** → A submission can have multiple reviews.

9. PublishedPapers Table (Tracks accepted & published papers)

```
CREATE TABLE PublishedPapers (
    PaperID INT PRIMARY KEY AUTO_INCREMENT,
    SubmissionID INT NOT NULL,
    JournalID INT NOT NULL,
    VolumeID INT NOT NULL,
    DOI VARCHAR(100) UNIQUE NULL,
    PublishedDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FilePath VARCHAR(500) NOT NULL,
    FOREIGN KEY (SubmissionID) REFERENCES Submissions(SubmissionID) ON DELETE CASCADE,
    FOREIGN KEY (JournalID) REFERENCES Journals(JournalID) ON DELETE CASCADE,
    FOREIGN KEY (VolumeID) REFERENCES Volumes(VolumeID) ON DELETE CASCADE
);
```

One-to-Many → A volume contains multiple **published papers**.

10. EditorialBoardInvitations Table (Tracks editorial board recruitment)

```
CREATE TABLE EditorialBoardInvitations (
   InvitationID INT PRIMARY KEY AUTO_INCREMENT,
   Name VARCHAR(255) NOT NULL,
   Affiliation VARCHAR(255) NOT NULL,
   Country VARCHAR(100) NOT NULL,
   Email VARCHAR(255) UNIQUE NOT NULL,
   ORCID_ID VARCHAR(50) UNIQUE NULL,
   ResearchInterests TEXT NULL,
   LinkedInProfile VARCHAR(255) NULL,
   PreferredRole ENUM('Editor-in-Chief', 'Assistant Editor', 'Reviewer Only') NOT NULL,
   MotivationStatement TEXT NULL,
   CVFilePath VARCHAR(500) NULL,
   Status ENUM('Pending', 'Accepted', 'Rejected') DEFAULT 'Pending',
   InvitationDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

One-to-Many → Admins can invite multiple researchers