CSS326 Project Proposal

Project name

Novel Nest

Group Members

1. Singha Junchan - 6622770350 : Project Manager (PM) & Product Owner (PO)
2. Chanon Sipiyarak - 6622770319 : Front-end Developer (UI/UX)
3. Kanade Areepoonsiri - 6622770442: Backend Developer (API, Database)
4. Johnny Shakespeare Ramseyer - 6622772448 : Backend Developer (API, Database)

(Should have clear role separation for each person to each task [head team of each section])

Project Description:

About What It Is:

Novel Nest is an interactive web platform designed for reading, writing, and sharing novels online. It serves as a digital library and creative hub where users can discover stories across genres, support their favorite authors, and build a personal reading journey. Writers can publish their work, engage with readers.

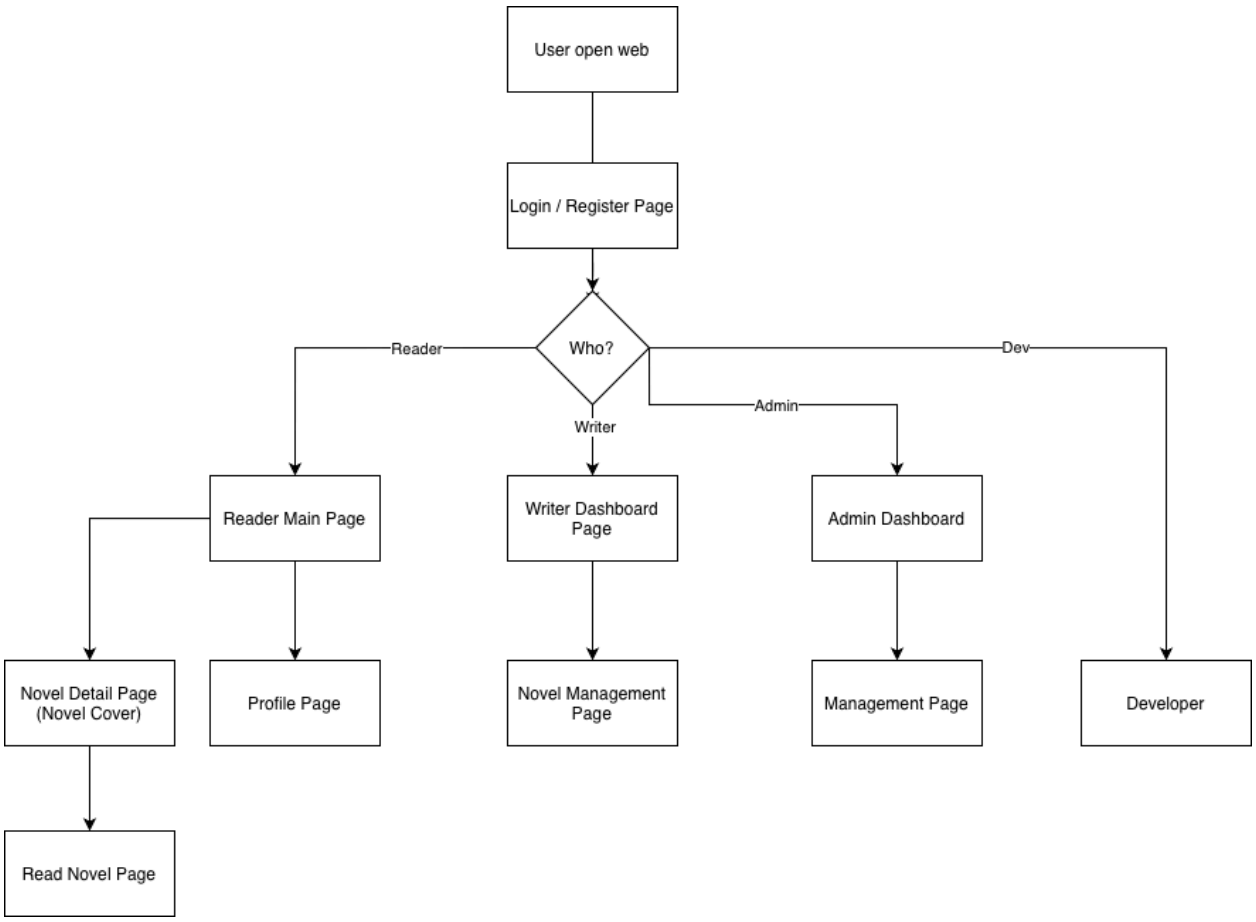
Who the Users Are:

The platform welcomes four types of users:

- **Readers**: who explore, and enjoy serialized novels.
- **Writers**: who create, publish, and manage their stories.
- **Admins**: who oversee system operations, content approval, and user management.
- **Developers**: who maintain and enhance platform features, security, and performance.

Each role is designed with tailored access and functions to create a smooth ecosystem where creativity and technology work together.

User Flows


User open web
Login / Register Page
Who?
Reader
Writer
Admin
Dev
Reader Main Page
Writer Dashboard Page
Admin Dashboard
Novel Detail Page (Novel Cover)
Profile Page
Novel Management Page
Management Page
Developer
Read Novel Page

User Flow Detail with Database Interaction

1. Core Flows (All Users)

Users begin by creating a secure account or logging in. The system authenticates them and directs them to the appropriate starting page based on their role (Reader, Writer, Admin, or Dev).

Flow	Page/Action	1. User Action	2. Backend Logic	3. Database Interaction (SQL)
1.0	**Login / Register Page (Sign Up)**	The user enters username, email, and password. Click "Sign Up."	Appends a new row to the 'Users' sheet with the user's info. Password is securely hashed. (stored in mysql)	**DB:** `INSERT` a new record into the `Users` table.
1.1	**Login / Register Page (Login)**	The user enters email/username and password. Click "Login."	Searches 'Users' sheet for matching credentials and verifies the hashed password. (stored in mysql)	**DB:** `FETCH` the user record from the `Users` table where credentials match.

2. Reader Flows

Readers browse for novels, read episodes which are linked from Google Drive. They can also interact by reviewing novels, commenting on episodes, and following their favorite authors.

Flow	Page/Action	1. User Action	2. Backend Logic	3. Database Interaction
2.0	**Reader Main Page**	The user loads the page, selects a Category, or filters by Tags.	**Query Building:** Constructs queries based on user filters (genre, tags, status). Calls the AI/Trending logic (via **Procedure**).	**DB:** Executes `FETCH` query on the `Novels` table, ordered by views or rating.
2.1	**Novel Detail Page (Cover)**	The user loads the page and clicks "Wishlist" or "Follow Author." or "like"	Append a new row to the 'Reviews' sheet. (stored in mysql) **Drive API:** Fetches cover image for display.	**DB:** `INSERT` a new record into the **User_Wishlist** or **User_Follows** table.

2.2	**Novel Detail Page (Review)**	Users enter a rating and comment. Click "Submit."	Validation: Ensures the user hasn't exceeded the review limit for this novel. Sanitizes the comment text.	**DB:** `INSERT` into **Reviews**; **Trigger** updates **Novels** average rating.
2.3	**Read Novel Page (Content)**	User opens an episode and able to read the episode's content	**Comment Logic:** If a reply, sets the parent_comment_id. **Bookmark Logic:** The system automatically calls the UpdateReadingProgress procedure to save this as the last-read chapter. **Sheets API:** Find the corresponding episode in the 'Episodes' sheet. **Drive API:** Use the URL from the sheet to fetch the content from the corresponding document in Google Drive.	**DB:** `FETCH` the `content_url` from the `Episodes` table. `INSERT` or `UPDATE` a record in the `User_Reading_Progress` table.
2.4	**Profile Page**	Users view or edit their bio/profile_picture.	**Data Fetch:** Retrieves all personalized data. **Validation:** Checks new input for length/format. Profile picture uploaded via Drive.	**DB:** FETCH User data: Reading_History and Wishlist. UPDATE Users table for profile changes.

3. Writer Flows

From a central dashboard, writers can easily create and manage their novels, and add new episodes. They can also track their interaction with reader feedback.

Flow	Page/Action	1. User Action	2. Backend Logic	3. Database Interaction
3.0	**Writer Dashboard**	The writer loads the dashboard, views analytics (total novels, episodes, likes).	**Aggregation:** total views, likes, and number of episodes per novel.	**DB:** FETCH and aggregate counts from **Novels** and **Episodes** where user_id = writer_id.
3.1	**Novel Management Page (Create Novel)**	Writer enters title, description, tags, and uploads cover_image. Click "Submit."	**Validation:** Checks required fields and image format. Sets status to 'Pending Approval' or 'Live'. **Drive API:** Upload the cover image to a designated folder in Google Drive. **Sheets API:** Append a new row to the 'Novels' sheet, including the Google Drive link for the cover image.	**DB:** INSERT new record into the Novels table with the user_id as the foreign key.
3.2	**Novel Management Page (Add/Edit Episode)**	The writer creates or updates an episode.	**Drive API:** Uploads text document to Drive and stores link. **Sheets API:** Updates 'Episodes' sheet.	**DB:** INSERT or UPDATE a record in the Episodes table. **Trigger:**Update the parent Novel's last_update timestamp.

4. Admin Flows

This is the platform oversight and moderation flow. Admins use their dashboard to monitor site-wide activity, manage users (verifying writers, suspending accounts), moderate content for safety, and handle financial tasks like approving writer payouts.

Flow	Page/Action	1. User Action	2. Backend Logic	3. Database Interaction
4.0	**Admin Dashboard**	Admin loads the dashboard, views statistics.	**Aggregation:** Calculates overall site statistics (active_users, novels, and reviews) for display.	**DB:** Aggregate COUNT and SUM queries on Users, Novels, and Reviews Table.
4.1	**Management Page (Users)**	Admin searches for a user and clicks "Suspend" or "Verify Writer."	**Permission Check:** Ensures the Admin has the right level for the action. Executes the status change.	**DB:** FETCH specific user details. UPDATE the Users table's role or status column.
4.2	**Management Page (Novels/Content)**	Admin reviews content (e.g., a reported comment). Click "Approve/Reject" or "Delete Spam."	**Moderation:** Logs the action and the Admin responsible via Sheets API.	**DB:** FETCH Reports data. UPDATE Novel status. DELETE comment form comments table.

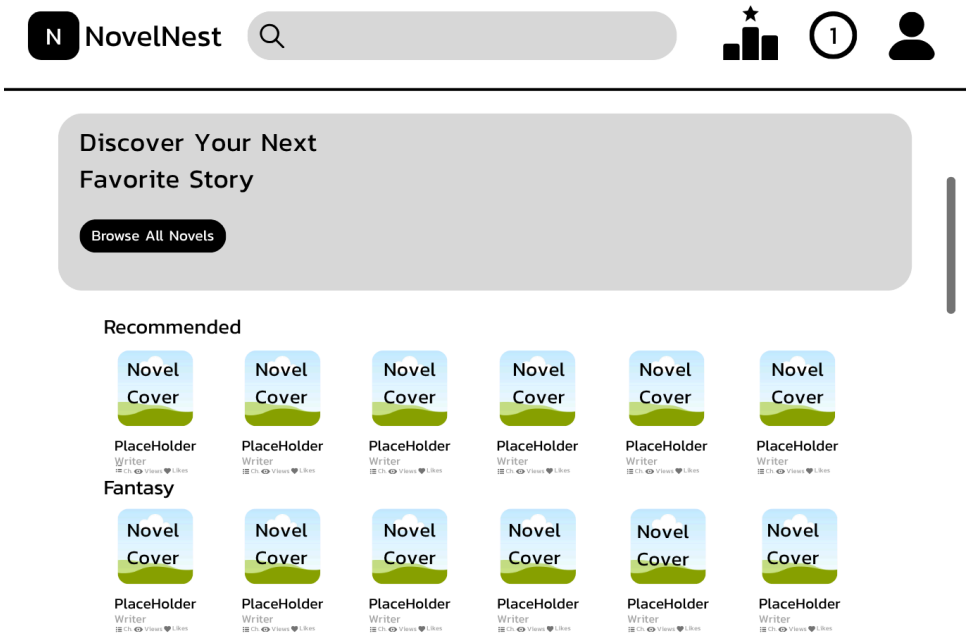
5. Developer Flows

This is a restricted, technical maintenance flow. A developer accesses a special system page to monitor server health, check security logs, and manage core system features, ensuring the platform remains stable and secure.

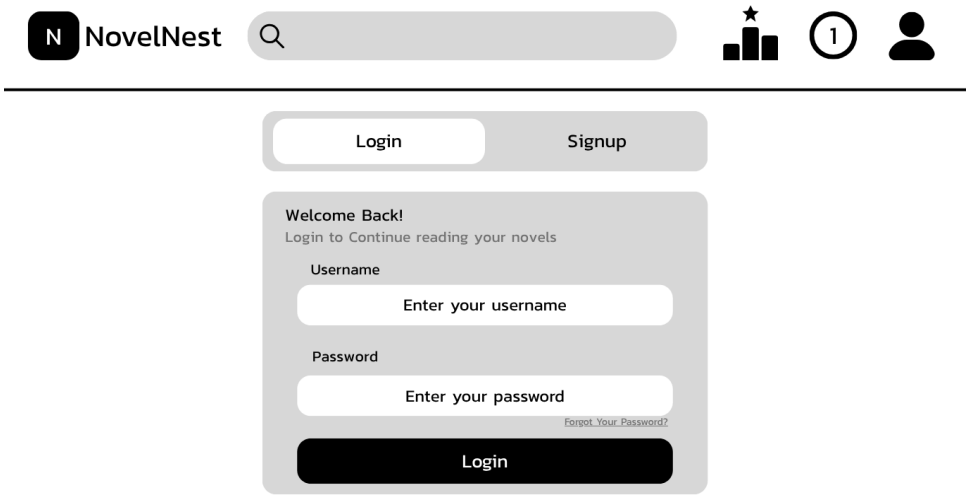
Flow	Page/Action	1. User Action	2. Backend Logic	3. Database Interaction
5.0	**Developer Page**	A user with the 'Developer' role logs in and access Dev views system monitoring tabs.	**Authorization Check:** Verify the user's role is **'Developer'**. If not, deny access. **System Call:** Accesses server logs and system metrics (OS, memory, security logs).	**DB (Minimal):**Primarily non-database interaction (file system, server APIs). May FETCH or UPDATE feature flags from a System_Config table.

User Interface

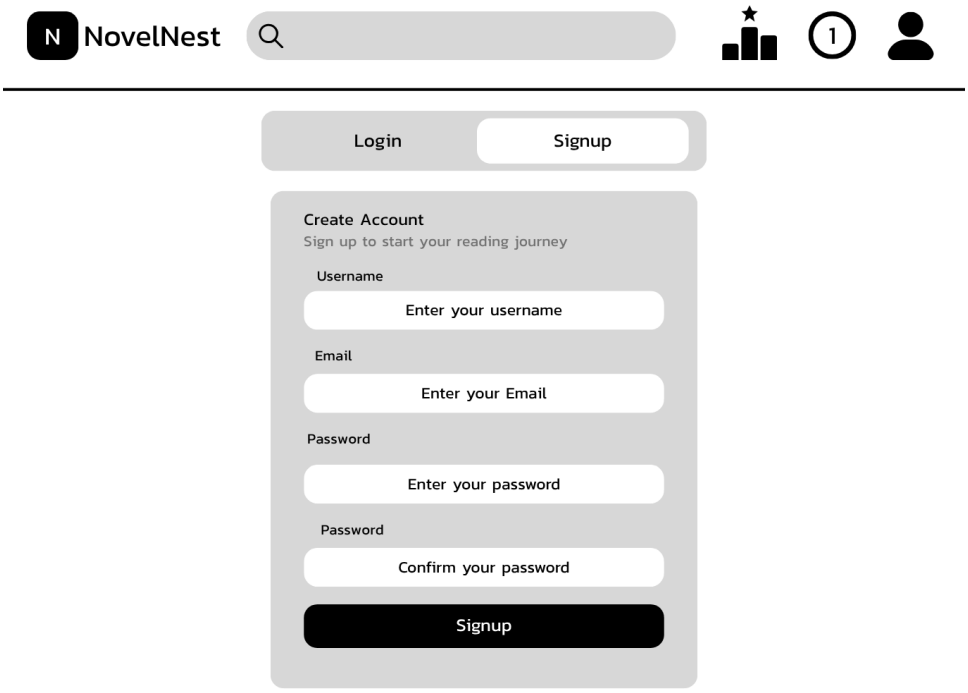
Main Page



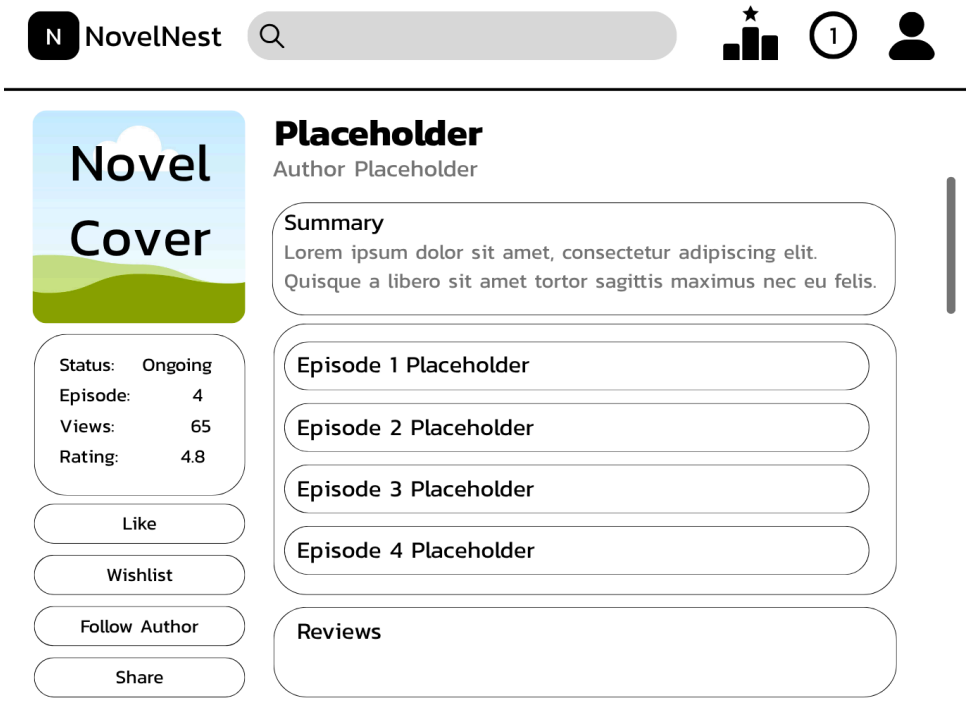
Login Page



Signup Page

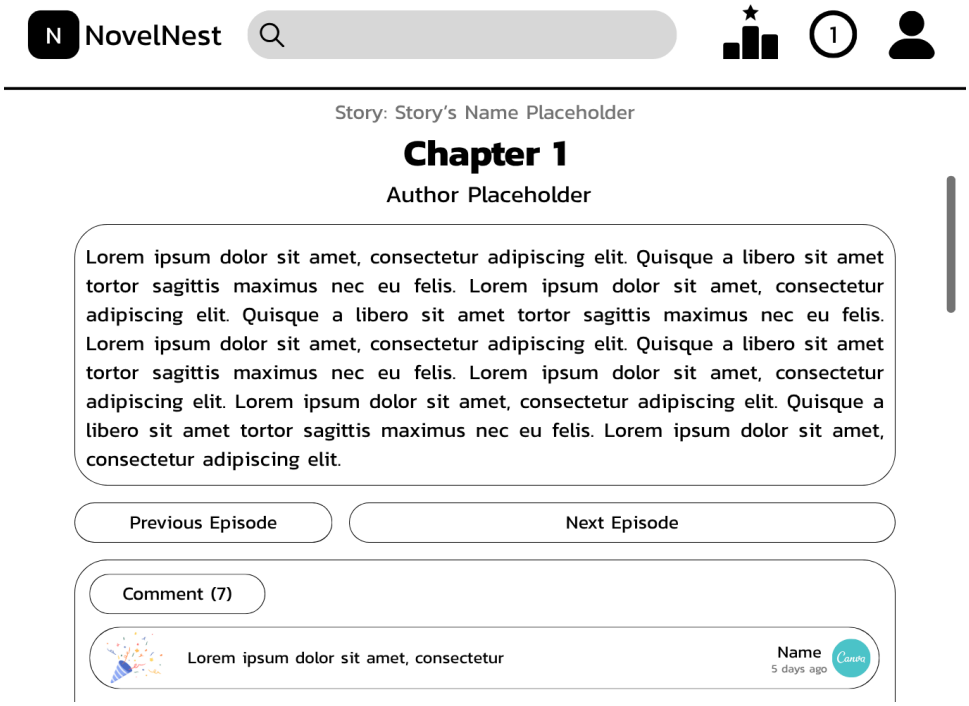


Novel Cover



Store novel in external drive and link to mysql.

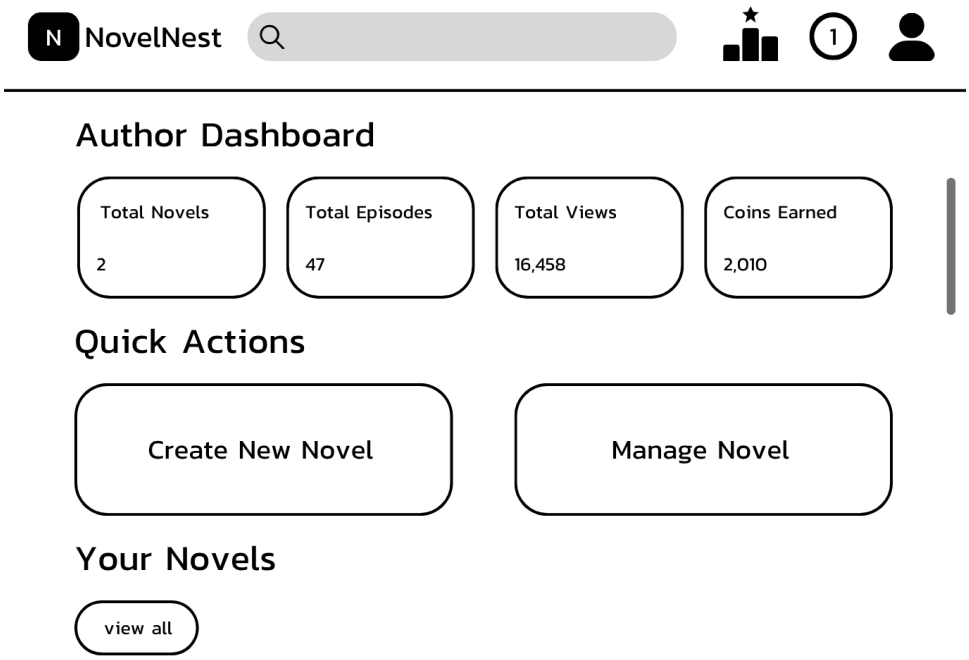
Novel Content



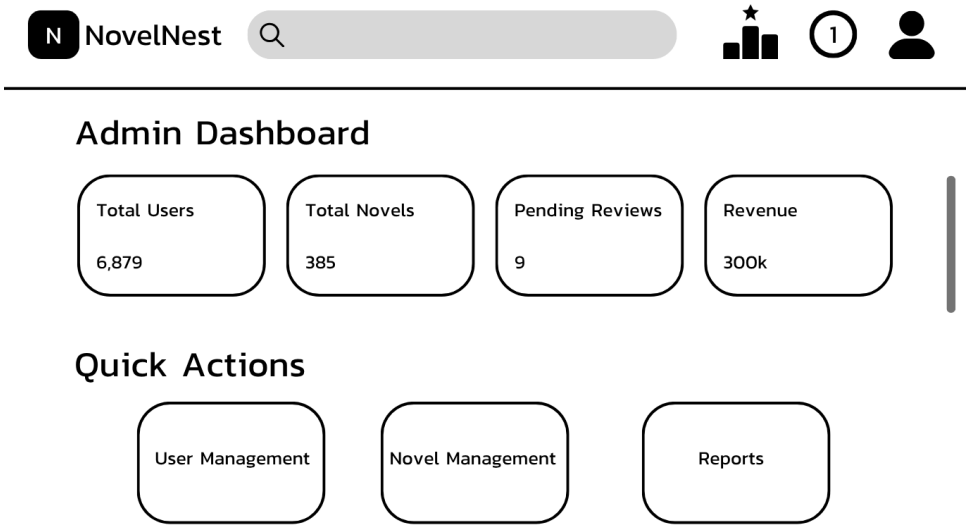
Store novel in external drive and link to mysql.

Profile Page

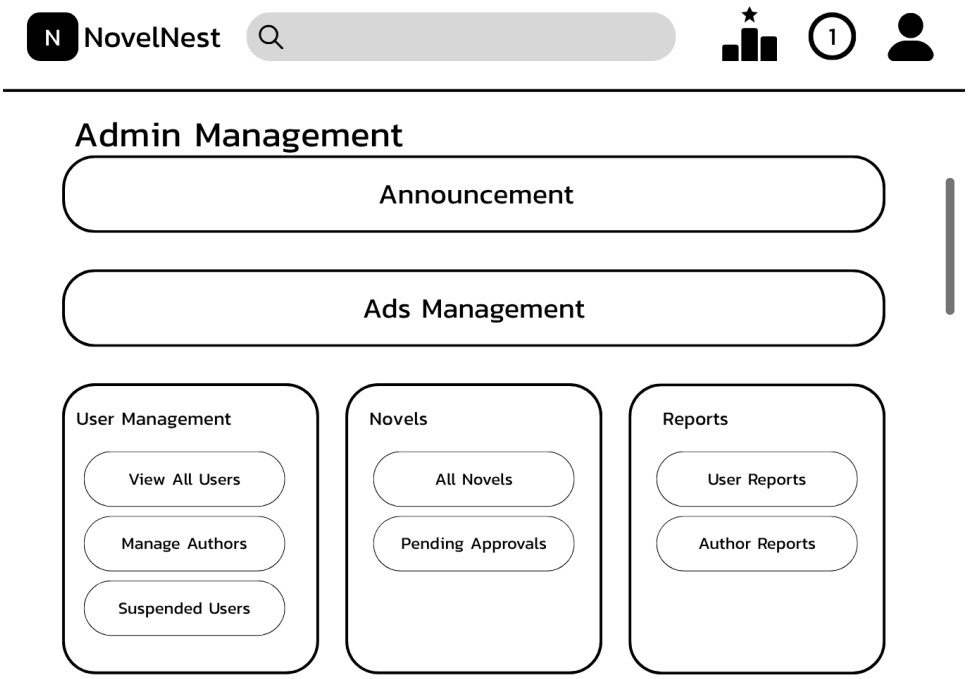
Author Dashboard



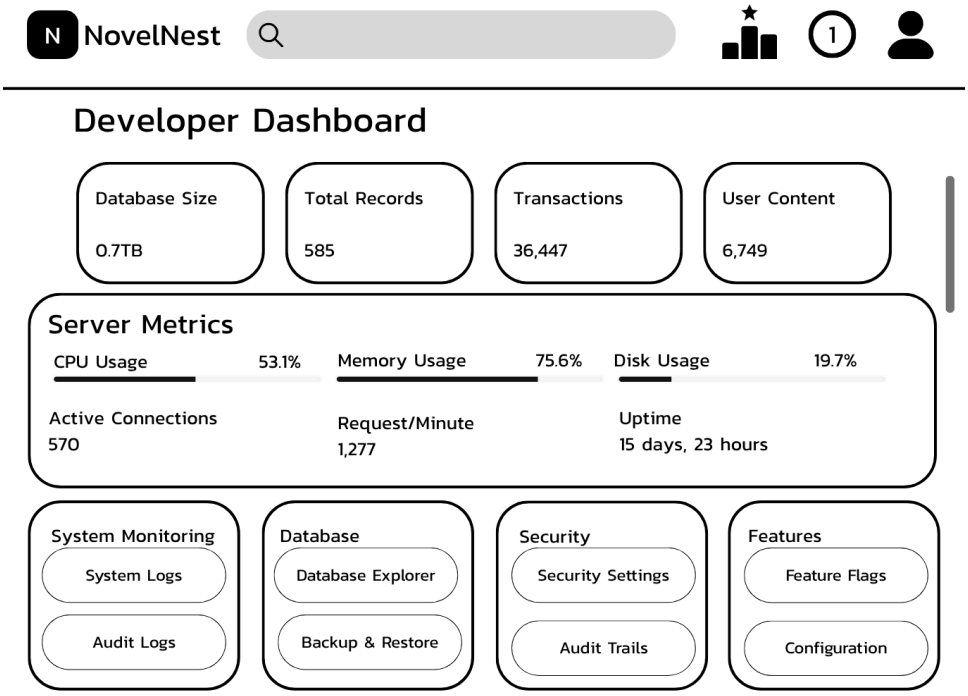
Admin Dashboard



Admin Management



Developer Dashboard



Database Design

Entities & Attributes

- **User Sheet:** Will contain columns like user_id, username, email, password, role, created_at, and bio. The profile_picture column will now store a URL to an image file in Google Drive.

- **Novel Sheet:** Will have columns such as novel_id, title, description, tags, status, last_update, views, likes, and rating. The cover_image column will hold a Google Drive URL.

- **Episode Sheet:** Will include episode_id, novel_id, title, and release_date. The content column will store a Google Drive link to a document containing the episode's text.

- **User**: (User metadata is stored in the SQL database; the profile picture is an image file linked from Google Drive)
    - user_id (Primary Key): A unique identifier for each user.
    - username: The user's chosen name for logging in and displaying.
    - email: The user's email address, used for login and communication.
    - password: The user's encrypted password.
    - **profile_picture: A URL link to the user's avatar image stored in Google Drive.**
    - bio: A short description of the user. (For writer only)
    - role: The user's role (e.g., Reader, Writer, Admin, Developer).
    - created_at: The timestamp of when the user account was created.
- **Novel:** (Novel metadata is stored in the SQL database; the cover image is a file linked from Google Drive.)
    - novel_id (Primary Key): A unique identifier for each novel.
    - title: The title of the novel.
    - description: A summary of the novel's plot.
    - **cover_image: A URL link to the novel's cover art stored in Google Drive.**
    - tags: Keywords that describe the novel's themes or elements.
    - status: The current status of the novel (e.g., Ongoing, Completed).
    - last_update: The timestamp of the most recent update.
    - views: The total number of times the novel has been viewed.
    - likes: The total number of likes the novel has received.
    - rating: The average reader rating.
- **Episode**: (Episode metadata is stored in the SQL database; the episode content is a document linked from Google Drive.)
    - episode_id (Primary Key): A unique identifier for each episode.
    - novel_id (Foreign Key): Links the episode to its novel.

    - title: The title of the episode.
    - **content: A URL link to the document file containing the episode's text, stored in Google Drive**
    - release_date: The date the episode was published.
- **Review**:
    - review_id (Primary Key): A unique identifier for each review.
    - novel_id (Foreign Key): Links the review to a novel.
    - user_id (Foreign Key): Links the review to the reader who wrote it.
    - rating: The star rating given by the reader.
    - comment: The text of the review.
    - created_at: The timestamp of when the review was posted.
- Comment:
    - comment_id (Primary Key): A unique identifier for each comment.
    - episode_id (Foreign Key): Links the comment to an episode.
    - user_id (Foreign Key): Links the comment to the user who posted it.
    - parent_comment_id (Foreign Key): For threaded replies, this links a comment to its parent.
    - content: The text of the comment.
    - created_at: The timestamp of when the comment was posted.
- User_Wishlist:
    - user_id (Foreign Key): Links to the user who created the wishlist item.
    - novel_id (Foreign Key): Links to the novel that was wishlisted.
    - added_at: The timestamp of when the novel was added to the wishlist.

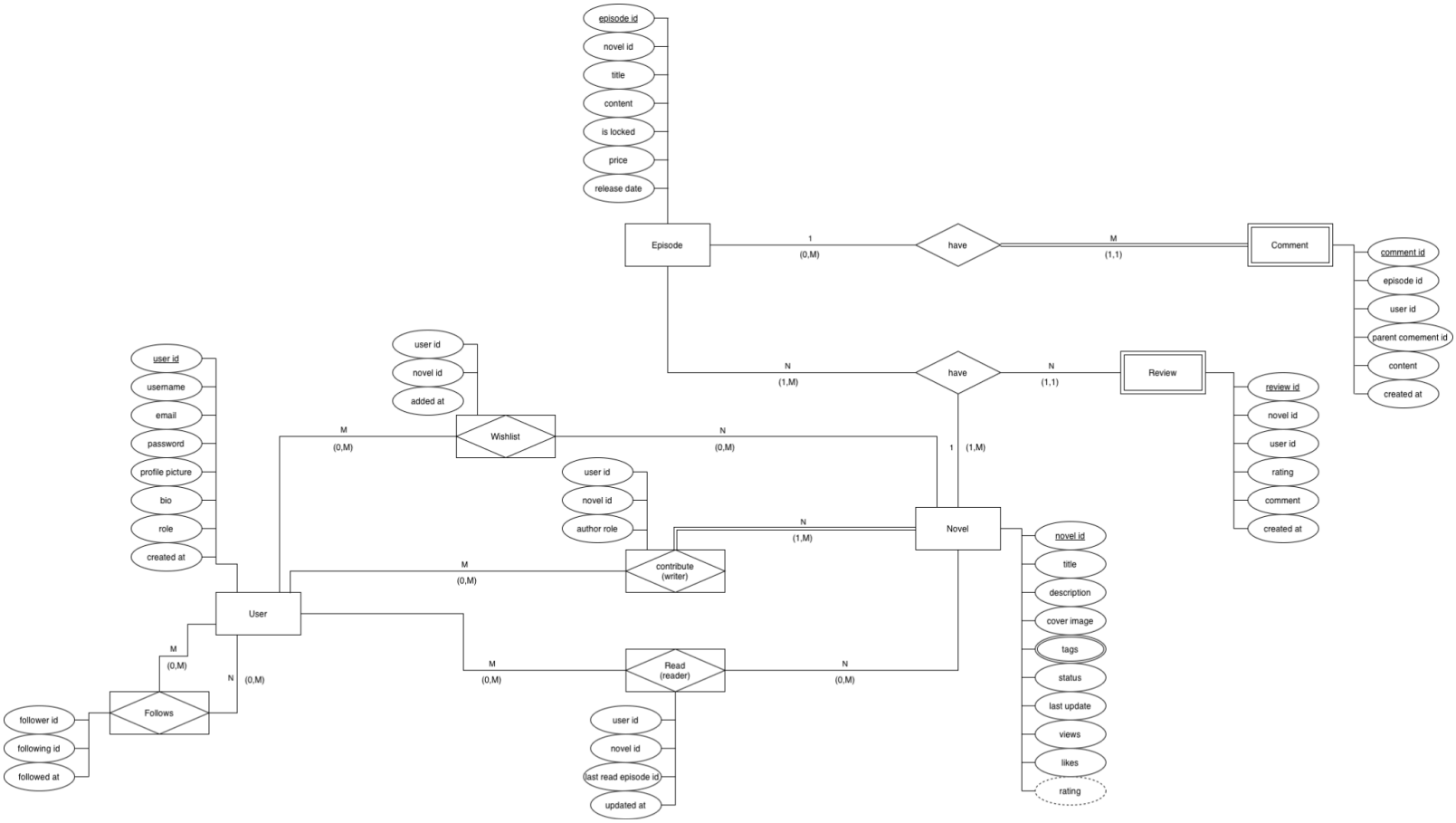
- **User_Follows**:
    - follower_id (Foreign Key): Links to the user who is doing the following.
    - following_id (Foreign Key): Links to the user (author) who is being followed.
    - followed_at: The timestamp of when the follow action occurred.
- **User_Reading_Progress**:
    - user_id (Foreign Key): Links to the user whose progress is being tracked.
    - novel_id (Foreign Key): Links to the novel being read.
    - last_read_episode_id (Foreign Key): Links to the most recently read episode (the bookmark).
    - updated_at: The timestamp of when the progress was last updated.
- **Novel_Authors**
    - **user_id (Foreign Key): Links to the Users table (the author).**
    - **novel_id (Foreign Key): Links to the Novels table.**
    - Author_role : A text field to describe their contribution (e.g., 'Primary Author', 'Illustrator').

Relationship Sets

- **One-to-Many**:
    - A Novel can have multiple Episodes.
    - A Novel can have multiple Reviews.
    - An Episode can have multiple Comments.
    - An Episode can have multiple Reviews.

- **Many-to-Many**:
    - **User-Novel (Authorship):** A User (Writer) can contribute to multiple Novels, and a Novel can have multiple authors.
        - This is managed by the **Novel_Authors** table.
    - **User-Novel (Wishlist):** A User can add multiple Novels to their wishlist, and a Novel can be in the wishlists of multiple users.
        - This is managed by the **User_Wishlist** table.
    - **User-Novel (Reading Progress):** A User has a unique reading progress for each Novel they read.
        - This is managed by the **User_Reading_Progress** table.
    - **User-User (Follows):** A User can follow multiple authors, and an author (User) can be followed by multiple users.
        - This is managed by the **User_Follows** table.

ER Diagram



Trigger, Procedure

Triggers

- **update_novel_rating**: This will be an **onEdit()** function in Google Apps Script that is automatically triggered whenever a new review is added to the "Reviews" sheet. The function will then read the relevant ratings, calculate the new average, and update the corresponding cell in the "Novels" sheet.
- **log_user_deletions**: This can be a manually triggered function or part of a user management function in your script that copies a user's data to an archive sheet before deleting it from the main "Users" sheet.

Can use (ledger table) to store log

Procedures become Google Apps Script Functions:

- **GetTrendingNovels(time_period)**: This will be a custom function in your script that queries the "Novels" sheet based on views, likes, and recent reviews.
- **ToggleWishlist(user_id, novel_id)**: This function will search the "User_Wishlist" sheet for a matching user_id and novel_id pair and either add or remove the row.
- **ToggleWishlist(user_id, novel_id)**: Adds or removes a novel from a user's wishlist by inserting or deleting a record in the User_Wishlist table.

(need to separate into two procedure)

- **ToggleFollow(current_user_id, author_user_id)**: Follows or unfollows an author by inserting or deleting a record in the User_Follows table.
- **UpdateReadingProgress(user_id, novel_id, episode_id)**: Creates or updates a record in User_Reading_Progress to save the user's last read chapter. (optional, if want to keep)

Physical Design for MySQL (need to fix physical table for linking to google drive)

1. Users

Stores information for all registered accounts, regardless of role.

Attribute Name	Data Type	Description	Key	Nullable	Unit / Format	Constraints	Example Value
user_id	Integer	Unique identifier for each user.	Primary Key	No	Integer	Must be unique and positive	101
username	String	The user's public display name.		No	Text	Must be unique	'alice_writer'
email	String	User's email for login and notifications.		No	Email Format	Must be unique	'alice@example.com'
password	String	The user's encrypted (hashed) password.		No	Encrypted String	Non-empty string	'<hashed_password_string>'
profile_picture	String (URL)	URL to the user's avatar image stored in Google Drive.		Yes	URL		'https://drive.google.com/uc?id=...'
bio	String	A short user-written biography.		Yes	Text		'An aspiring fantasy author.'
role	String	Defines the user's permissions on the site.		No	ENUM	('Reader', 'Writer', 'Admin', 'Dev')	'Writer'
created_at	DateTime	Timestamp of when the account was created.		No	Timestamp	Default: CURRENT_TIMESTAMP	'2025-10-06 09:00:00'

2. Novels

Stores the main information about each novel.

Attribute Name	Data Type	Description	Key	Nullable	Unit / Format	Constraints	Example Value
novel_id	Integer	Unique identifier for each novel.	Primary Key	No	Integer	Must be unique and positive	2001
title	String	The official title of the novel.		No	Text	Non-empty string	'The Crystal Key'
description	String	A summary or synopsis of the novel's plot.		Yes	Text		'A story about a lost artifact...'
cover_image	String	URL to the novel's cover artwork stored in Google Drive.		Yes	URL		'https://drive.google.com/uc?id=[file_id]'
tags	String	Keywords describing the novel's genre/themes.		Yes	JSON Array	Stored as a JSON array of strings	'["fantasy", "magic"]'
status	String	The current publication status of the novel.		No	ENUM	('Ongoing', 'Completed', 'Hiatus')	'Ongoing'

last_update	DateTime	Timestamp of the last time an episode was added.		Yes	Timestamp		'2025-10-05 18:00:00'
views	Integer	A counter for the total number of views.		No	Integer	Default: 0, Must be positive	15023
likes	Integer	A counter for the total number of likes.		No	Integer	Default: 0, Must be positive	1200
rating	Decimal	The calculated average rating from user reviews.		Yes	Decimal(3,2)	Range: 0.00-5.00	4.75

3. Episodes

Stores the content for each individual chapter or episode of a novel.

Attribute Name	Data Type	Description	Key	Nullable	Unit / Format	Constraints	Example Value
episode_id	Integer	Unique identifier for each episode.	Primary Key	No	Integer	Must be unique and positive	3010
novel_id	Integer	Links the episode to its parent novel.	Foreign Key	No	Integer	Must exist in Novels table	2001

title	String	The title of the chapter.		No	Text	Non-empty string	'Chapter 1: The Hidden Village'
content	String	A URL link to the Google Document file containing the full text of the episode.		No	URL	Must be a valid Google Drive link	'https://docs.google.com/document/d/[doc_id]/view'
release_date	DateTime	The date and time the episode was published.		No	Timestamp	Default: CURRENT_TIMESTAMP	'2025-09-15 10:00:00'

4. Reviews

Stores user-submitted ratings and reviews for novels.

Attribute Name	Data Type	Description	Key	Nullable	Unit / Format	Constraints	Example Value
review_id	Integer	Unique identifier for each review.	Primary Key	No	Integer	Must be unique and positive	4001
novel_id	Integer	Links the review to a specific novel.	Foreign Key	No	Integer	Must exist in Novels table	2001
user_id	Integer	Links the review to the user who wrote it.	Foreign Key	No	Integer	Must exist in Users table	105

rating	Integer	The star rating given by the user.		No	Integer	Range: 1-5	5
comment	String	The user's written review text.		Yes	Text		'I couldn't put this down!'
created_at	DateTime	Timestamp of when the review was submitted.		No	Timestamp	Default: CURRENT_TIMESTAMP	'2025-10-01 12:34:56'

5. Comments

Stores user comments on specific episodes.

Attribute Name	Data Type	Description	Key	Nullable	Unit / Format	Constraints	Example Value
comment_id	Integer	Unique identifier for each comment.	Primary Key	No	Integer	Must be unique and positive	5001
episode_id	Integer	Links the comment to a specific episode.	Foreign Key	No	Integer	Must exist in Episodes table	3010
user_id	Integer	Links the comment to the user who wrote it.	Foreign Key	No	Integer	Must exist in Users table	108

parent_comment_id	Integer	Links a reply to its parent comment.	Foreign Key	Yes	Integer	Must exist in Comments table	5001
content	String	The text content of the comment.		No	Text		'What a cliffhanger!'
created_at	DateTime	Timestamp of when the comment was posted.		No	Timestamp	Default: CURRENT_TIMESTAMP	'2025-09-20 20:05:10'

6. Novel_Authors (Bridge Table)

Manages the many-to-many relationship between writers and novels (for co-authorship).

Attribute Name	Data Type	Description	Key	Nullable	Unit / Format	Constraints	Example Value
user_id	Integer	The ID of the author.	Primary, FK	No	Integer	Must exist in Users table	101
novel_id	Integer	The ID of the co-authored novel.	Primary, FK	No	Integer	Must exist in Novels table	2001
author_role	String	The specific role of the author on the project.		No	Text	Default: 'Author'	'Primary Author'

7. User_Wishlist (Bridge Table)

Manages the many-to-many relationship for user wishlists.

Attribute Name	Data Type	Description	Key	Nullable	Unit / Format	Constraints	Example Value
user_id	Integer	The ID of the user.	Primary, FK	No	Integer	Must exist in Users table	108
novel_id	Integer	The ID of the wishlisted novel.	Primary, FK	No	Integer	Must exist in Novels table	2005
added_at	DateTime	Timestamp of when the item was added.		No	Timestamp	Default: CURRENT_TIMESTAMP	'2025-10-06 08:30:00'

8. User_Follows (Bridge Table)

Manages the "following" relationship between users (readers) and authors.

Attribute Name	Data Type	Description	Key	Null able	Unit / Format	Constraints	Example Value
follower_id	Integer	The user who is doing the following.	Primary, FK	No	Integer	Must exist in Users table	105
following_id	Integer	The author who is being followed.	Primary, FK	No	Integer	Must exist in Users table	101
followed_at	DateTime	Timestamp of when the follow action occurred.		No	Timestamp	Default: CURRENT_TIMESTAMP	'2025-09-30 19:45:00'

9. User_Reading_Progress (Bridge Table)

Acts as a bookmark system, saving a user's progress in a novel.

Attribute Name	Data Type	Description	Key	Nullable	Unit / Format	Constraints	Example Value
user_id	Integer	The ID of the user.	Primary, FK	No	Integer	Must exist in Users table	108
novel_id	Integer	The ID of the novel being read.	Primary, FK	No	Integer	Must exist in Novels table	2001
last_read_episode_id	Integer	The ID of the last episode the user read.	Foreign Key	No	Integer	Must exist in Episodes table	3015
updated_at	DateTime	Timestamp of when the progress was last updated.		No	Timestamp	Updates on every change	'2025-10-06 09:15:20'

Timeline

Weeks	Month	Date	Plan
Week 8	Oct	7	Project Proposal Submission
Week 9		14	Backend Foundation & Setup
Week 10		21	Core Content API Development
Week 11		28	Front-end Development (Reader View)
Week 12	Nov	4	Reader Interaction Features
Week 13		11	Writer & Admin Dashboards

Week 14		18	Final Testing & Presentation Prep
Week 15		25	Plo
Week 16	Dec	2	No Lab (Final Exam)