

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)

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, and earn rewards through the platform's coin-based system.

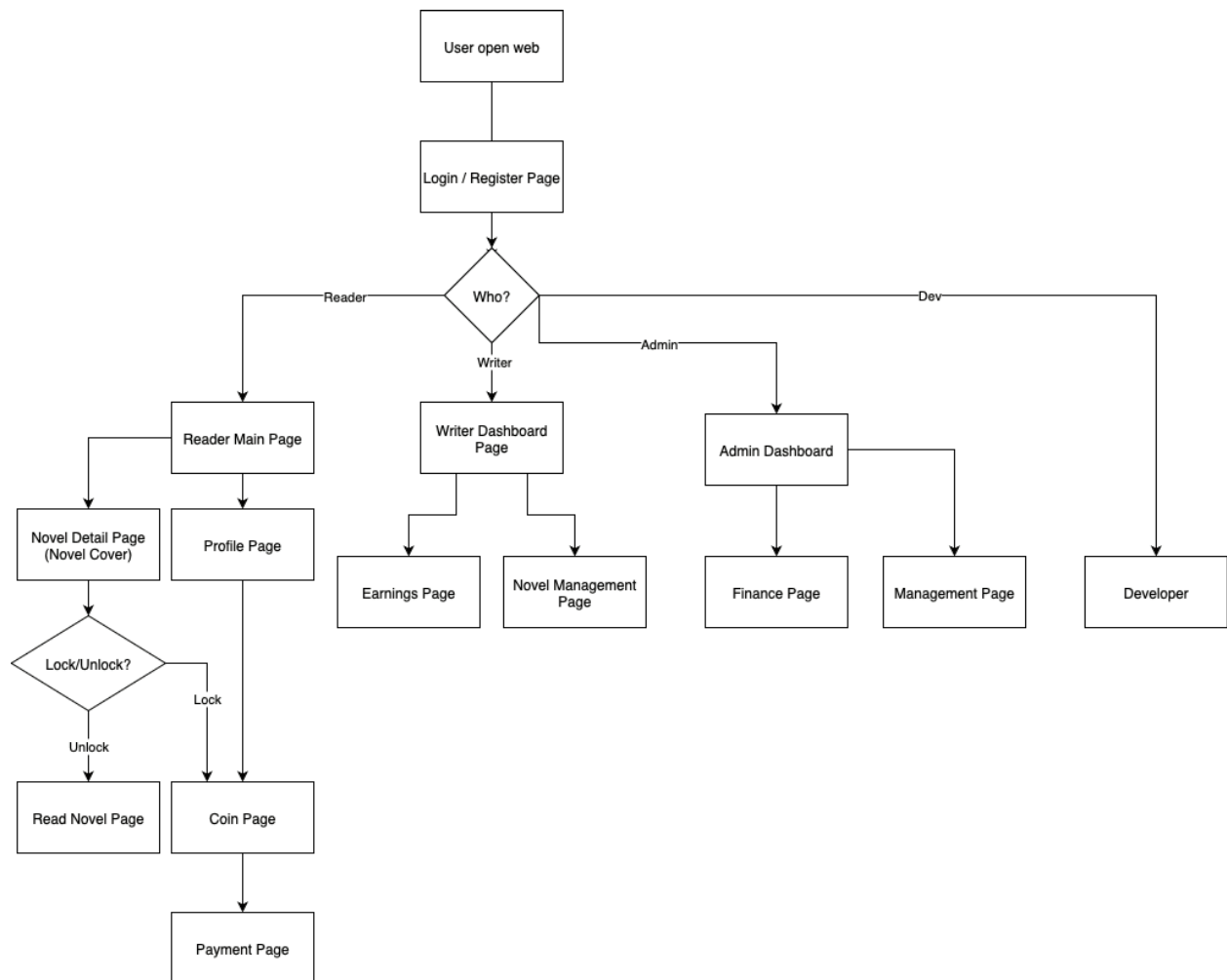
Who the Users Are:

The platform welcomes four types of users:

- **Readers:** who explore, purchase, 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 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
1.0	Login / Register Page (Sign Up)	The user enters username, email, and password. Click "Sign Up."	Validation: Checks if email or username already exists. Hashes the password. Assigns default role (Reader).	DB: INSERT new record into the Users table.
1.1	Login / Register Page (Login)	The user enters email/username and password. Click "Login."	Validation: Verifies user identity using the encrypted password. Fetches the user's role.	DB: FETCH user record from Users where credentials match.

2. Reader Flows

This flow is about discovery and engagement. A reader browses for novels, reads episodes, and if a chapter is locked, they are guided through a seamless process to buy coins and unlock it. 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, price, status). Calls the AI/Trending logic (via Procedure).	DB: Complex FETCH query on the Novels table, ordered by views or rating (for rankings) or linked to a Recommendation table (for AI).

2.1	Novel Detail Page (Cover)	The user loads the page and clicks "Wishlist" or "Follow Author." or "like"	State Change: <ul style="list-style-type: none"> - Calls the ToggleWishlist procedure to add or remove the novel from the user's list. - Calls the ToggleFollow procedure to follow or unfollow the author. 	DB: FETCH all data from Novel, Episodes, and Reviews tables. Interaction: INSERT or DELETE record from User_Follows. INSERT or DELETE record from User_Wishlist UPDATE views from novel
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 new record into the Reviews table. Trigger: Update the Novel.rating column.
2.3	Lock/Unlock Check (Purchase)	The user clicks a locked Episode.	Access Check: Check if the episode is_locked. If it is, check if the user has already purchased it. If not purchased, then proceed to the Coin Page	DB: FETCH from User_Purchased_Episodes where user_id and episode_id match. If a record is found, the episode is considered unlocked for that user. If no record is found, then proceed with the original logic (check coin balance, etc.).
2.4	Read Novel Page (Content)	User loads an unlocked episode. Posts a Comment or Reply.	Customization: Applies reading settings (font, color) from user profile. 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.	DB: FETCH Episode content. INSERT or update a record in the User_Reading_Progress table using the UpdateReadingProgress procedure.

2.5	Coin Page (Buy Coins)	The user views balance, selects a package, and clicks "Buy."	State Check: Temporarily locks the coin balance while awaiting payment confirmation.	DB: FETCH User coin balance and Transaction history.
2.6	Payment Page	The user completes the payment process via a third-party gateway.	Gateway Confirmation: Receives a success signal from PayPal/Stripe. Transaction: Executes the coin transfer (via Procedure).	DB: INSERT into the Transactions table. Update the Users coin balance.
2.7	Profile Page	Users load the page or update their bio/profile_picture.	Data Fetch: Retrieves all personalized data. Validation: Checks new input for length/format.	DB: FETCH User data, Reading_History, Wishlist, and Purchased_Novels. UPDATE Users table for profile changes.

3. Writer Flows

This is a streamlined content management and monetization journey. From a central dashboard, writers can easily create and manage their novels, add new episodes, and set a price for premium (locked) content. They can also track their earnings and interact with reader feedback.

Flow	Page/Action	1. User Action	2. Backend Logic	3. Database Interaction
3.0	Writer Dashboard	The writer loads the dashboard.	Aggregation: Calculates lifetime views, total coins_earned, and new Notifications.	DB: Complex FETCH and aggregate functions on Novels, Writer_Earnings, and Notifications tables, filtered by 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'.	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 enters episode_title, content, price, and sets is_locked. Click "Publish."	Validation: Checks if content exceeds a minimum length. Applies pricing rules.	DB: INSERT or UPDATE a record in the Episodes table. Trigger: Update the parent Novel's last_update timestamp.
3.3	Earnings Page (Withdrawal)	The writer views coin_summary and fills out the withdrawal request form.	Calculation: Verifies the minimum withdrawal threshold. Logs the request and moves the earnings to a pending state.	DB: FETCH data from Writer_Earnings and Payout_History. INSERT new record into Payout_Requests.

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.	Aggregation: Calculates overall site statistics (active_users, revenue_trends, etc.) for display.	DB: Aggregate COUNT and SUM queries on Users, Novels, and Transactions tables.
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.	DB: FETCH Reports data. UPDATE Novel status. DELETE abusive Comments.

4.3	Finance Page	Admin views coin purchases and handles a writer payout request.	Verification: Cross-checks the payout request against the recorded Writer_Earnings. Processes the transaction via a financial system.	DB: FETCH data from Transactions and Payout_Requests. Executes Procedure to record the payout and clear the pending request.
-----	--------------	---	--	--

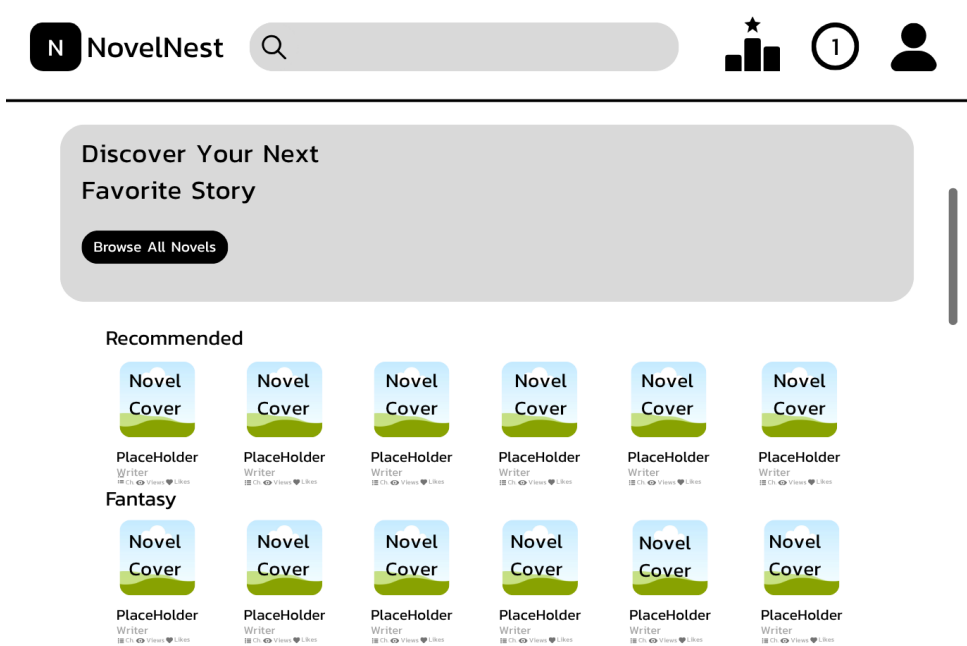
5. Developer Flows

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.

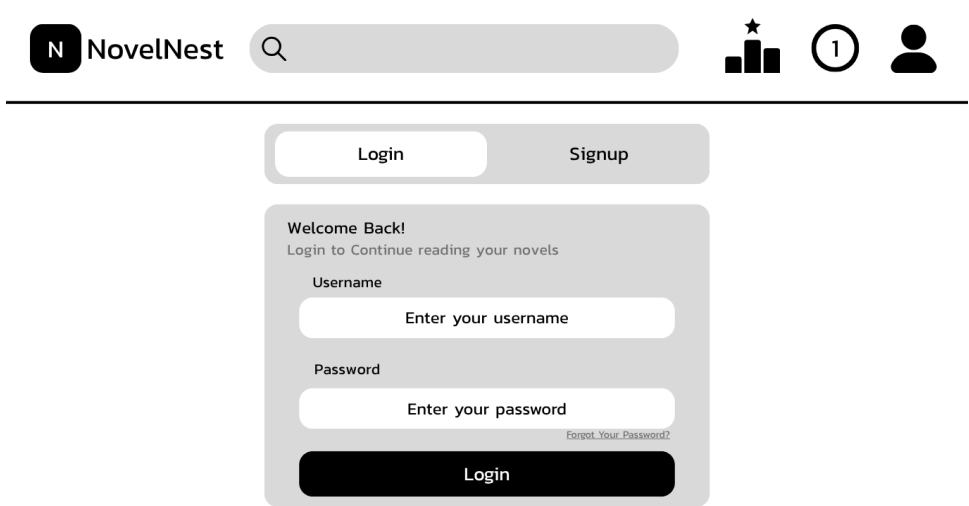
Flow	Page/Action	1. User Action	2. Backend Logic	3. Database Interaction
5.0	Developer Page	A user with the 'Developer' role logs in and is redirected to this page. Dev views system monitoring tabs.	Authorization Check: Verify the user's role is ' Developer '. If not, deny access. System Call: Accesses 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

N

NovelNest

1

Login

Signup

Create Account

Sign up to start your reading journey

Username

Enter your username

Email

Enter your Email

Password

Enter your password

Password

Confirm your password

Signup

Novel Cover

N

NovelNest

1

Novel
Cover

Placeholder

Author Placeholder

Summary

Lorem ipsum dolor sit amet, consectetur adipiscing elit.
Quisque a libero sit amet tortor sagittis maximus nec eu felis.

Episode 1 Placeholder

Episode 2 Placeholder

Episode 3 Placeholder

Episode 4 Placeholder

Reviews

Status: Ongoing

Episode: 4

Views: 65

Rating: 4.8

Like

Wishlist

Follow Author

Share

Store novel in external drive and link to mysql.

Novel Content

N NovelNest

1

Story: Story's Name Placeholder

Chapter 1

Author Placeholder

>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque a libero sit amet tortor sagittis maximus nec eu felis. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque a libero sit amet tortor sagittis maximus nec eu felis. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque a libero sit amet tortor sagittis maximus nec eu felis. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque a libero sit amet tortor sagittis maximus nec eu felis. Lorem ipsum dolor sit amet, consectetur adipiscing elit.

Previous Episode

Next Episode

Comment (7)

>Lorem ipsum dolor sit amet, consectetur

Name

5 days ago

Store novel in external drive and link to mysql.

Profile Page

N NovelNest

1

Profile Picture

Profile

Name

Bio

>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Quisque a libero sit amet tortor sagittis maximus nec eu felis.

Badges

Reading History

Story's Name

19:47 10/5/2025

Story's Name

10:17 10/5/2025

Purchased Episodes

Story's Name

11:00 09/5/2025

Story's Name

19:47 07/5/2025

N NovelNest

1

Your Coin Balance

Use coins to unlock premium episodes

750

100 coins

50 THB

500 coins

250 THB

1000 coins

500 THB

Transaction History

+100 coins

09:37 10/6/2025

N NovelNest

1

Payment

Amount of Coins:

100 Coins

Amount Due:

50 THB

Total Amount:

850 coins

Payment Options

PayPal

true money wallet

PromptPay

stripe

MasterCard

Monthly Subscription

Receive 500 coins now

Receive 50 coins per day on log-in

for a total of 2000 coins per month

790 THB

Author Dashboard

N NovelNest

1

Author Dashboard

Total Novels

2

Total Episodes

47

Total Views

16,458

Coins Earned

2,010

Quick Actions

Create New Novel

Manage Novel

Your Novels

view all

Admin Dashboard

N NovelNest

1

Admin Dashboard

Total Users

6,879

Total Novels

385

Pending Reviews

9

Revenue

300k

Quick Actions

User Management

Novel Management

Reports

Admin Management

Announcement

Ads Management

User Management

View All Users

Manage Authors

Suspended Users

Novels

All Novels

Pending Approvals

Reports

User Reports

Author Reports

Finance

All Transactions

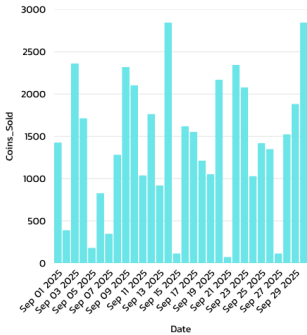


User	Date	Amount
2648	Sep 29 2025	500THB
1983	Sep 29 2025	250THB
317	Sep 29 2025	50THB

Withdrawal Request

User	Date	Amount
Kailing	Sep 29 2025	500THB

Revenue Analytics





Developer Dashboard

Database Size
0.7TB

Total Records
585

Transactions
36,447

User Content
6,749

Server Metrics

CPU Usage53.1%

Memory Usage75.6%

Disk Usage19.7%

Active Connections570

Request/Minute1,277

Uptime15 days, 23 hours

System Monitoring

System Logs

Audit Logs

Database

Database Explorer

Backup & Restore

Security

Security Settings

Audit Trails

Features

Feature Flags

Configuration

Database Design

Entities & Attributes

- **User:**
 - 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, communication, and password resets.
 - password: The user's encrypted password.
 - profile_picture: A link to the user's avatar.
 - 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: (store novel in external drive and link into mysql)**
 - 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 link to the novel's cover art.
 - 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: (store episode in external drive and link to mysql)**
 - 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: The text of the episode.
 - is_locked: A boolean indicating if the episode requires payment to read.
 - price: The cost in coins to unlock the episode.
 - 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.

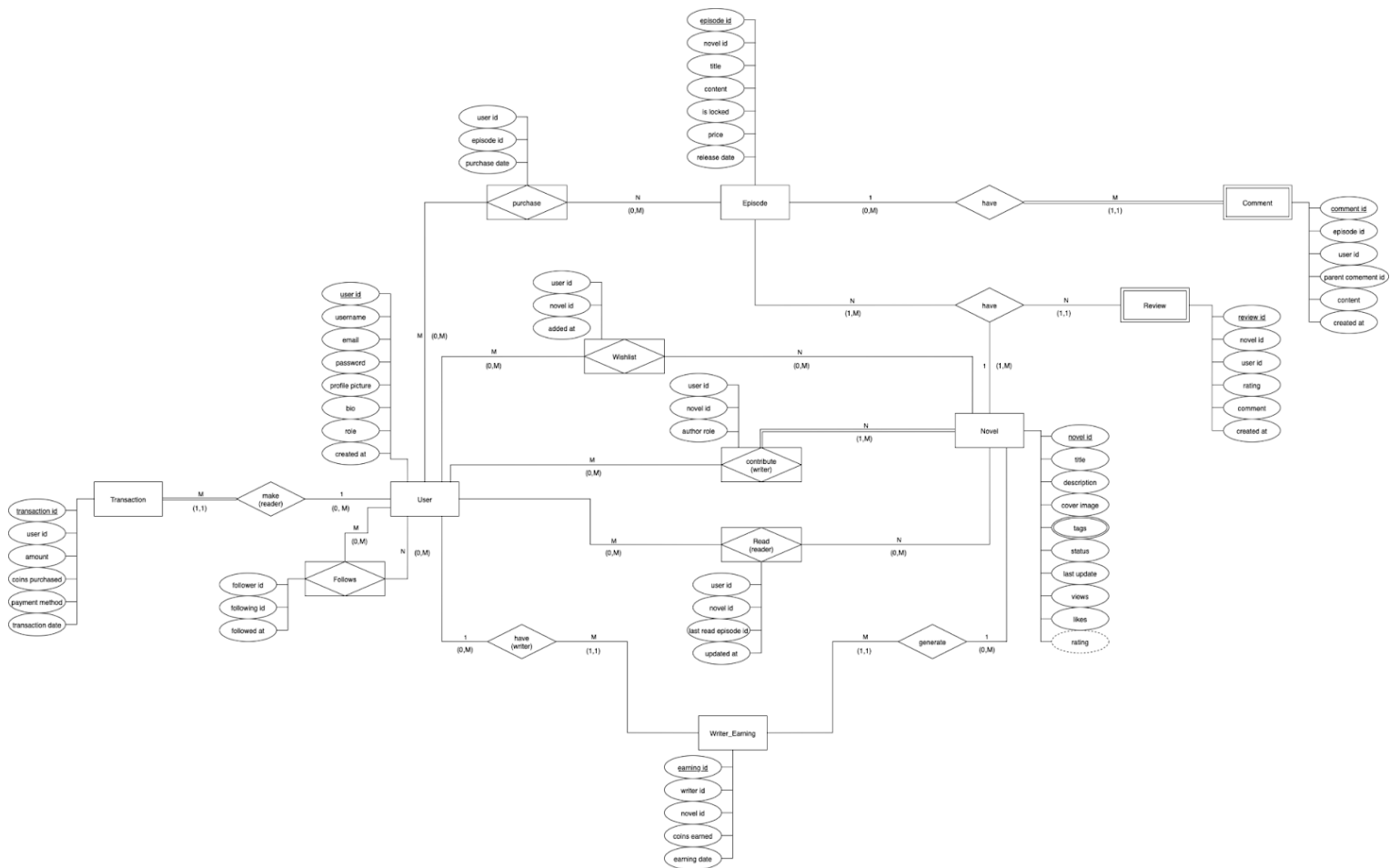
- user_id (Foreign Key): Links the review to the reader who wrote it.
- rating: The star rating given by the reader.
- comment: The text of the review.
- created_at: The timestamp of when the review was posted.
- **Comment: (if its too big, store in external drive and link to mysql)**
 - comment_id (Primary 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.
- **Transaction: (delete)**
 - transaction_id (Primary Key): A unique identifier for each transaction.
 - user_id (Foreign Key): Links the transaction to a user.
 - amount: The monetary value of the transaction.
 - coins_purchased: The number of coins bought.
 - payment_method: The method used for payment (e.g., PayPal, Stripe).
 - transaction_date: The timestamp of the transaction.
- **Writer_Earning: (delete)**
 - earning_id (Primary Key): A unique identifier for each earning record.
 - writer_id (Foreign Key): Links the earnings to a writer.
 - novel_id (Foreign Key): The novel that generated the earnings.
 - coins_earned: The number of coins earned.
 - earning_date: The date the earnings were recorded.
- **User_Purchased_Episodes: (delete)**
 - user_id (Foreign Key): Links to the user who made the purchase.
 - episode_id (Foreign Key): Links to the episode that was purchased.
 - purchase_date: The timestamp of when the purchase occurred.
- **User_Wishlist: (optional, if too much work -> can delete)**
 - 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 (TEXT, e.g., 'Primary Author', 'Illustrator', 'Translator'): An optional field to describe their contribution.

Relationship Sets

- **One-to-Many:**
 - A Novel can have multiple Episodes.
 - A Novel can have multiple Reviews.
 - An Episode can have multiple Comments.
 - A User (Reader) can make multiple Transactions.
 - An Episode can have multiple Reviews.
 - A User (Writer) can have multiple Writer_Earning records.
 - A Novel can generate multiple Writer_Earning records (one for each author for each period).
- **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-Episode (Purchases):** A User can purchase multiple locked Episodes, and an Episode can be purchased by many Users.
 - This is managed by the **User_Purchased_Episodes** 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**: After a new Review is submitted, this trigger can automatically recalculate the average rating for the corresponding Novel.
- **update_last_novel_activity**: When a new Episode is added to a Novel, this trigger can update the last_update timestamp for that Novel. (delete)
- **log_user_deletions**: If a user's account is deleted, this trigger can archive their basic information in a separate table for historical purposes before deletion.

Procedures

- **PurchaseEpisode(reader_id, episode_id):** Handles the transaction of a reader purchasing a locked episode. It checks the reader's coin balance, deducts the price, records the purchase in the User_Purchased_Episodes table, and grants access. (delete)
- **ProcessWriterPayout(writer_id):** This procedure would calculate a writer's earnings and handle the process of a withdrawal request, moving earnings from a pending state to paid. (delete)
- **GetTrendingNovels(time_period):** This procedure would query the database to retrieve a list of novels that are currently trending based on views, likes, and new reviews within a specified time frame (e.g., daily, weekly, monthly). (keep this two)
- **GenerateReaderRecommendations(reader_id):** This procedure could analyze a reader's history and preferences to suggest new novels they might enjoy. (keep this two)
- **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.
- **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

1. Users

Stores information for all registered accounts, regardless of role.

Attribute Name	Data Type	Description	Key	Null able	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 to the user's avatar image.		Yes	URL		'https://.../img/101.jpg'
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.		Yes	URL		'https://.../covers/2001.jpg'
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	Null able	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	The full text content of the episode.		No	Text (Long)		'The sun rose over the mountains...'
is_locked	Boolean	Determines if the episode requires purchase.		No	Boolean	Default: 0 (false)	1 (true)
price	Integer	The cost in virtual coins to unlock.		Yes	Integer	Must be positive	10
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. Transactions

Logs all real-money transactions for coin purchases.

Attribute Name	Data Type	Description	Key	Nullable	Unit / Format	Constraints	Example Value
----------------	-----------	-------------	-----	----------	---------------	-------------	---------------

transaction_id	Integer	Unique identifier for each transaction.	Primary Key	No	Integer	Must be unique and positive	6001
user_id	Integer	Links the transaction to the purchasing user.	Foreign Key	No	Integer	Must exist in Users table	105
amount	Decimal	The real monetary value of the transaction.		No	Decimal(10,2)	Must be positive	9.99
coins_purchased	Integer	The number of virtual coins bought.		No	Integer	Must be positive	1000
payment_method	String	The payment gateway used (e.g., Stripe, PayPal).		No	Text		'Stripe'
transaction_date	DateTime	Timestamp of when the transaction was completed.		No	Timestamp	Default: CURRENT_TIMESTAMP	'2025-10-03 11:22:33'

7. Writer_Earnings

Logs each instance of a writer earning coins from their novels.

Attribute Name	Data Type	Description	Key	Nullable	Unit / Format	Constraints	Example Value
earning_id	Integer	Unique identifier for each earning record.	Primary Key	No	Integer	Must be unique and positive	7001
writer_id	Integer	Links the earning to the writer who earned it.	Foreign Key	No	Integer	Must exist in Users table	101
novel_id	Integer	Links the earning to the novel that generated it.	Foreign Key	No	Integer	Must exist in Novels table	2001
coins_earned	Integer	The number of coins earned in this instance.		No	Integer	Must be positive	50
earning_date	DateTime	Timestamp of when the coins were earned.		No	Timestamp	Default: CURRENT_TIMESTAMP	'2025-10-04 00:00:00'

8. Novel_Authors (Bridge Table)

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

Attribute Name	Data Type	Description	Key	Null able	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'

9. User_Pu

rchased_Episodes (Bridge Table)

Tracks which users have purchased which locked episodes.

Attribute Name	Data Type	Description	Key	Null able	Unit / Format	Constraints	Example Value
user_id	Integer	The ID of the purchasing user.	Primary, FK	No	Integer	Must exist in Users table	105
episode_id	Integer	The ID of the purchased episode.	Primary, FK	No	Integer	Must exist in Episodes table	3011
purchase_date	Date Time	Timestamp of when the purchase occurred.		No	Timestamp	Default: CURRENT_TIMESTAMP	'2025-10-05 14:00:15'

10. 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'

11. User_Follows (Bridge Table)

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

Attribute Name	Data Type	Description	Key	Nullable	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'
-------------	----------	---	--	----	-----------	----------------------------	-----------------------

12. User_Reading_Progress (Bridge Table)

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

Attribute Name	Data Type	Description	Key	Null able	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	Project Submission & Presentation
Week 16	Dec	2	No Lab (Final Exam)