MANGA VERSE

A COMIC WEBSITE

CCDATRCL PROJECT BY:

Group Members:

Michael George E. Aquino

James Adrian B. Castro

Rafael Iñigo T. Lopez

Alvin G. Pableo

Oscar Cloud Vincent C. Yumang

INTRODUCTION:

MANGAVERSE BEGAN AS A SMALL COMIC WEBSITE DESIGNED TO PROVIDE READERS WITH AN INSIGHT INTO MODERN JAPANESE COMICS, KNOWN AS MANGA. MANGA IS VERY DIVERSE AND REALLY PACKED WITH GOOD LITERATURE. MANGA IS REALLY POPULAR NOWADAYS DUE TO ITS KNOWN ANIMATION COUNTERPART, WHICH IS THE ANIME. ANIME ADAPTATION GIVES LIFE TO ANY MANGA OR LIGHT NOVEL. IT GIVES READERS AN ASPECT OF EXCITEMENT WHEN THEY ARE PICKING UP A MANGA OR READING IT ONLINE. THIS WEBSITE IS DESIGNED TO CATER TO SUCH CRAVINGS TO READ COMICS AND BRING LIFE TO ANY KIND OF JAPANESE LITERATURE TO THEIR MOBILE PHONES.

BACKGROUND:

Recent decades have seen the consumption of digital media rise substantially, particularly as a result of online reading environments such as web novels, comics, and manga. Manga, a Japanese art form that weaves compelling storytelling and expressive images, has attracted a huge amount of global popularity because of its simplicity and ubiquity in a variety of genres. But, many available platforms do not have many useful and personalizing features, especially for web-based systems intended for students or casual readers, but they are growing in popularity.

PURPOSE:

MangaVerse is a full-stack web application that forms a digital manga library. I hope to address common problems found in the online manga browsing community: disorganized collections, lack of personalization, lack of customer feedback mechanisms, and a system that allows users to create accounts, log in securely, browse the manga titles, post reviews, and keep track of their reading history by using this system.

OBJECTIVES:

General Objective:

To design and develop a full-stack web application that serves as an online manga library for readers.

Specific Objectives:

- **Implement a backend using Node.js and Express.js connected to MongoDB.**
- **Create an interactive frontend with HTML, CSS, and JavaScript.**
- **Allow users to register, log in, and manage their reading activity.**
- **Integrate a sorting feature using both MongoDB sorting and custom frontend sorting algorithms (Bubble Sort, Quick Sort, Selection Sort).**
- **Enable users to leave reviews on manga titles.**

SYSTEM OVERVIEW:

System Concept:

MangaVerse is a web-based application that provides users with access to a digital manga library. It enables account creation, manga browsing, and review submission.

Users:

Regular Users – can create an account, browse manga, view details, and post reviews.

System Requirements:

Hardware:

- Minimum 4GB RAM
- Dual-core processor

Software:

- Node.js v18+ (much better if the latest version)
- MongoDB
- Any modern browser (Chrome, Edge, Firefox)
- NetBeans IDE, VS Code (for development and testing)

DATA STRUCTURES USED:

Data Structure	Purpose	Justification
Array	Stores the list of manga retrieved from the backend.	Arrays allow easy iteration, sorting, and filtering.
Object	Represents individual manga, user data, and reviews.	Provides flexibility for JSON-based API data.
Stack (via History Feature)	Tracks recently read manga.	Simulates last-read-first-display behavior.
Queue	Manages asynchronous fetch tasks.	Ensures smooth, ordered data retrieval.

ALGORITHM DESIGN:

System Algorithms Overview

The backend of MangaVerse applies a combination of data handling, authentication, and data organization algorithms. While it doesn't include traditional sorting algorithms like Bubble Sort directly in the backend, it utilizes MongoDB's query-based sorting, authentication verification, and history tracking algorithms through structured logic.

1.AUTHENTICATION ALGORITHM (LOGIN & REGISTRATION):

```lua

function registerUser(username, email, password):
    hashedPassword = bcrypt.hash(password)
    newUser = { username, email, password: hashedPassword }
    save newUser to MongoDB
    generate JWT token for session

function loginUser(email, password):
    user = findUserByEmail(email)
    if user exists and bcrypt.compare(password, user.password):
        generate JWT token
        return success
    else:
        return error ("Invalid credentials")
```

2. MANGA SORTING ALGORITHM

```js
Manga.find({ title: { $regex: search, $options: 'i' } })
     .sort({ title: sort === 'asc' ? 1 : -1 });
```

2. MANGA SORTING ALGORITHM

Type:	Algorithm	Description:
Indexed Sort	B-Tree Index Traversal	When a field (e.g, title) is indexed, MongoDB leverages its B-tree index structure to read data in sorted order. This method is extremely efficient, with logarithmic complexity O(log n), and does not require an additional in-memory sort.
Non-Indexed Sort	In-Memory Merge Sort (Top-K Merge Sort)	When no index exists, MongoDB uses an optimized merge sort variant internally to arrange results. This algorithm is stable, ensuring consistent order for identical fields, and is designed to handle large datasets by merging sorted batches in memory.

3. READING HISTORY ALGORITHM

```pgsql
function addHistory(userId, mangaTitle, chaptersRead):
    user = findById(userId)
    user.history.push({ mangaTitle, chaptersRead, date: now })
    save user
```

4. REVIEW SUBMISSION ALGORITHM

```
function addReview(mangaId, username, reviewText):
    manga = findById(mangaId)
    manga.reviews.push({ username, reviewText, date: now })
    save manga
```

IMPLEMENTATION

Programming Language and Tools:

- **Backend: Node.js (Express.js framework)**
- **Database: MongoDB (via Mongoose ORM)**
- **Authentication: JSON Web Token (JWT)**
- **Security: bcrypt.js for password hashing**
- **Configuration: dotenv for environment variable management**
- **CORS: For safe API communication between front-end and back-end**

KEY CODE SNIPPET (AUTHENTICATION):

```js
// User Registration
app.post('/api/register', async (req, res) => {
  const { username, email, password } = req.body;
  const hashedPassword = await bcrypt.hash(password, 10);
  const newUser = new User({ username, email, password: hashedPassword });
  await newUser.save();
  const token = jwt.sign({ id: newUser._id, username: newUser.username }, JWT_SECRET, { expiresIn:
  res.status(201).json({ token, user: { username: newUser.username, email: newUser.email } });
});
```

KEY CODE SNIPPET (READING HISTORY):

```
// Update History
app.post('/api/history', verifyToken, async (req, res) => {
  const { mangaTitle, chaptersRead } = req.body;
  const user = await User.findById(req.user.id);
  user.history.push({ mangaTitle, chaptersRead, date: new Date() });
  await user.save();
  res.status(201).json(user.history);
});
```

4. REVIEW SUBMISSION ALGORITHM

```
function addReview(mangaId, username, reviewText):
    manga = findById(mangaId)
    manga.reviews.push({ username, reviewText, date: now })
    save manga
```

KEY CODE SNIPPET (SORTING MANGA):

```js
app.get('/api/mangas', verifyToken, async (req, res) => {
  const { sort = 'asc', search = '' } = req.query;
  const mangas = await Manga.find({ title: { $regex: search, $options: 'i' } })
                            .sort({ title: sort === 'asc' ? 1 : -1 });
  res.json(mangas);
});
```

TESTING TABLE:

Test Case	Input	Expected Output	Actual Output	Result
Register User	username: “Leo”, email: “leo@test.com”, password: “12345”	Token returned & user saved	Success	✅
Login User	Valid email & password	JWT Token returned	Success	✅
Load Manga	/api/mangas?sort=asc	Sorted manga list	Sorted correctly	✅
Add Review	Valid manga ID & review text	Review added successfully	Works correctly	✅
Add History	mangaTitle: “Bleach”, chaptersRead: 5	History updated	Works correctly	✅

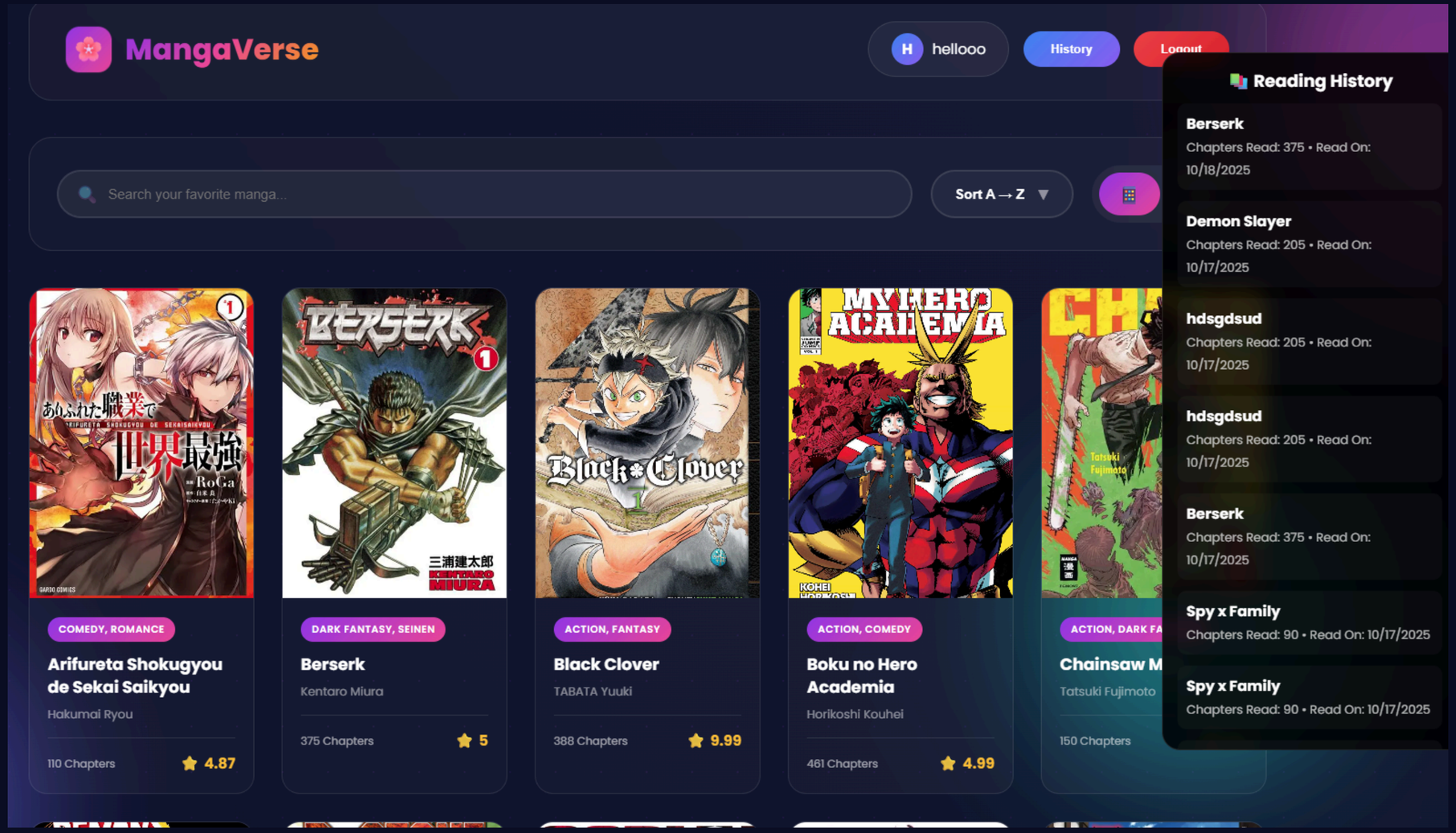
TESTING TABLE:

Load History	GET /api/history	List of read manga	Displays properly	✅
Invalid Token	Missing JWT	Access denied	Proper error message	✅

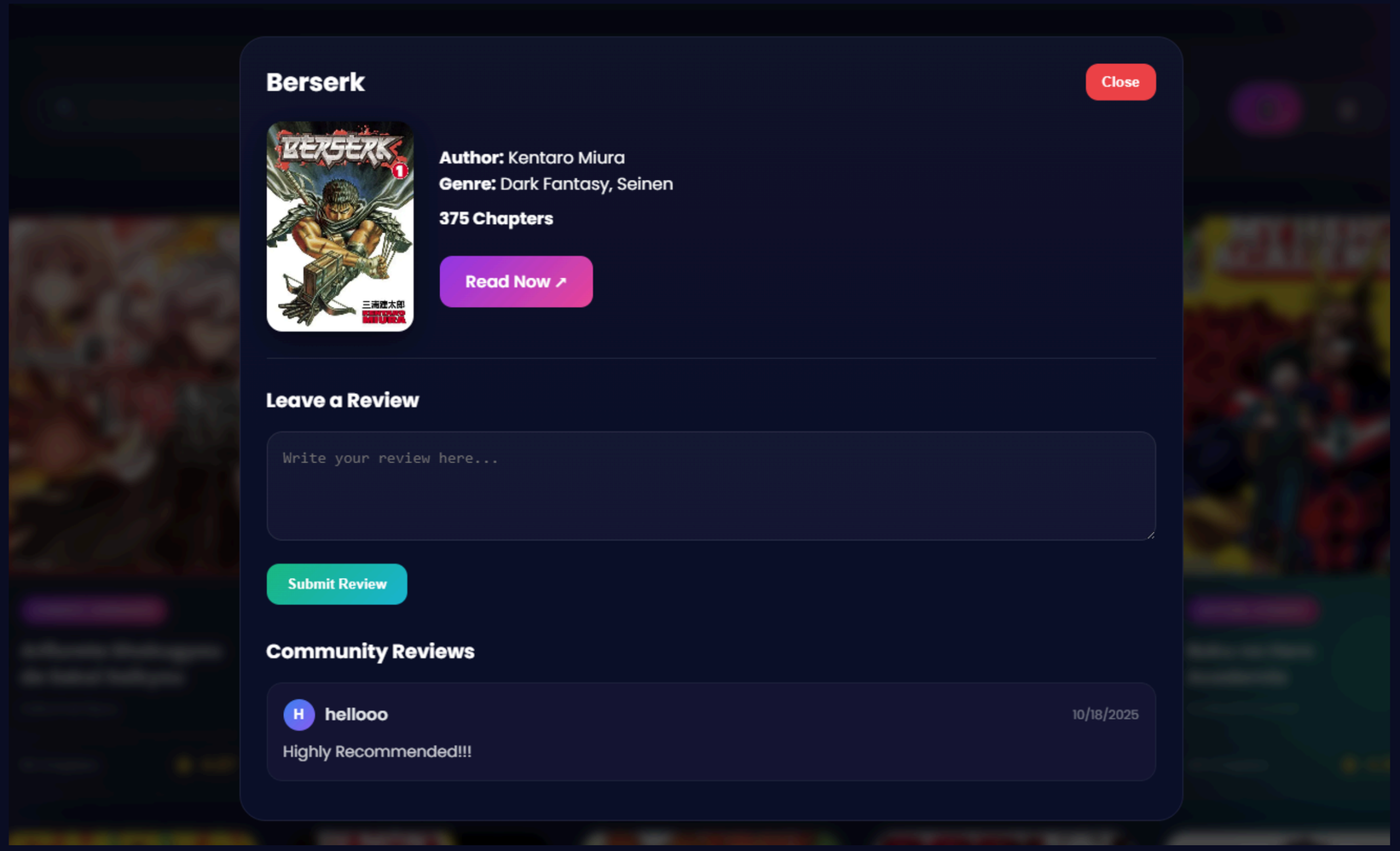
TEST RESULTS: (OUTPUT)

TEST RESULTS: (OUTPUT)

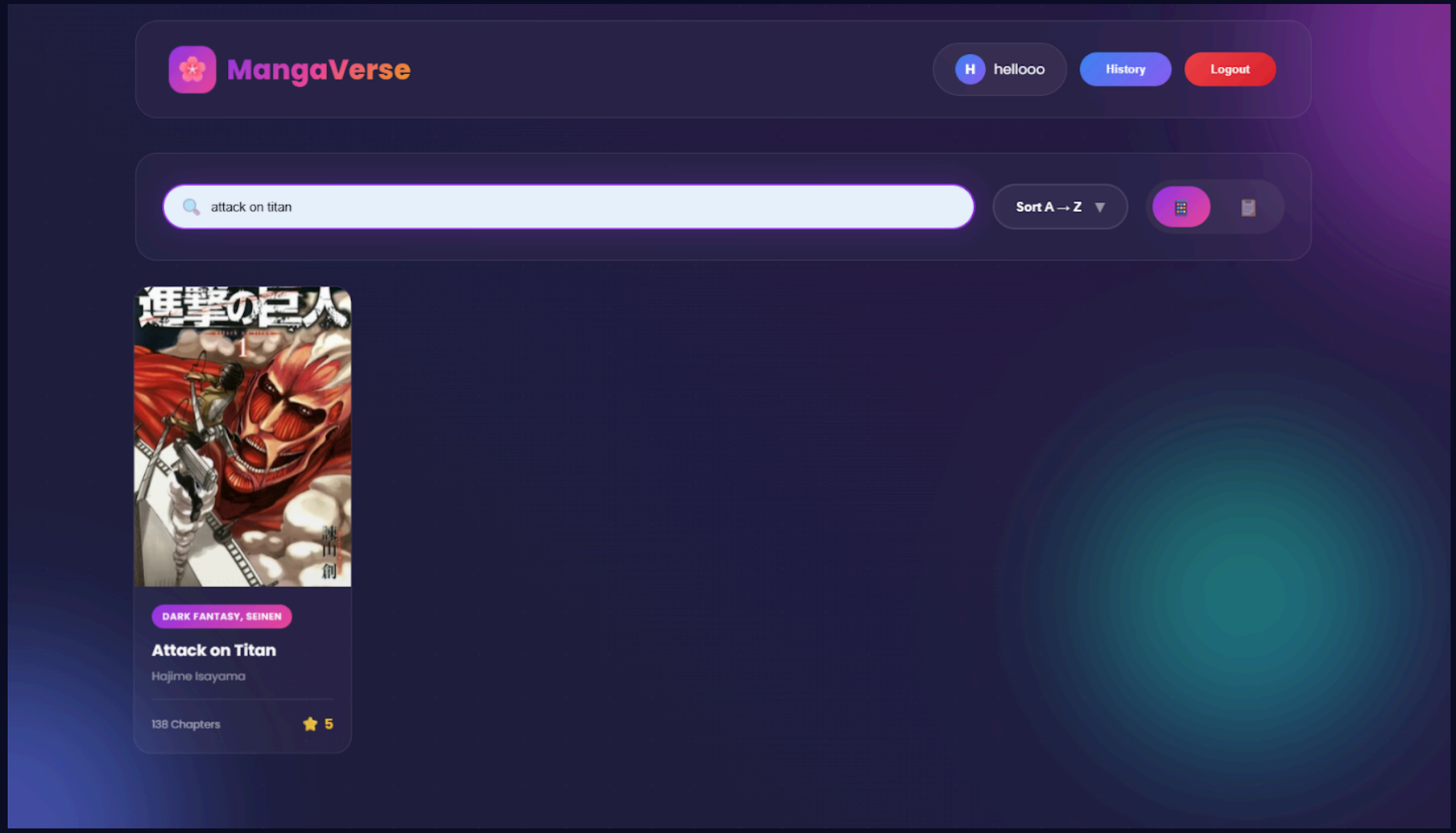
TEST RESULTS: (OUTPUT)



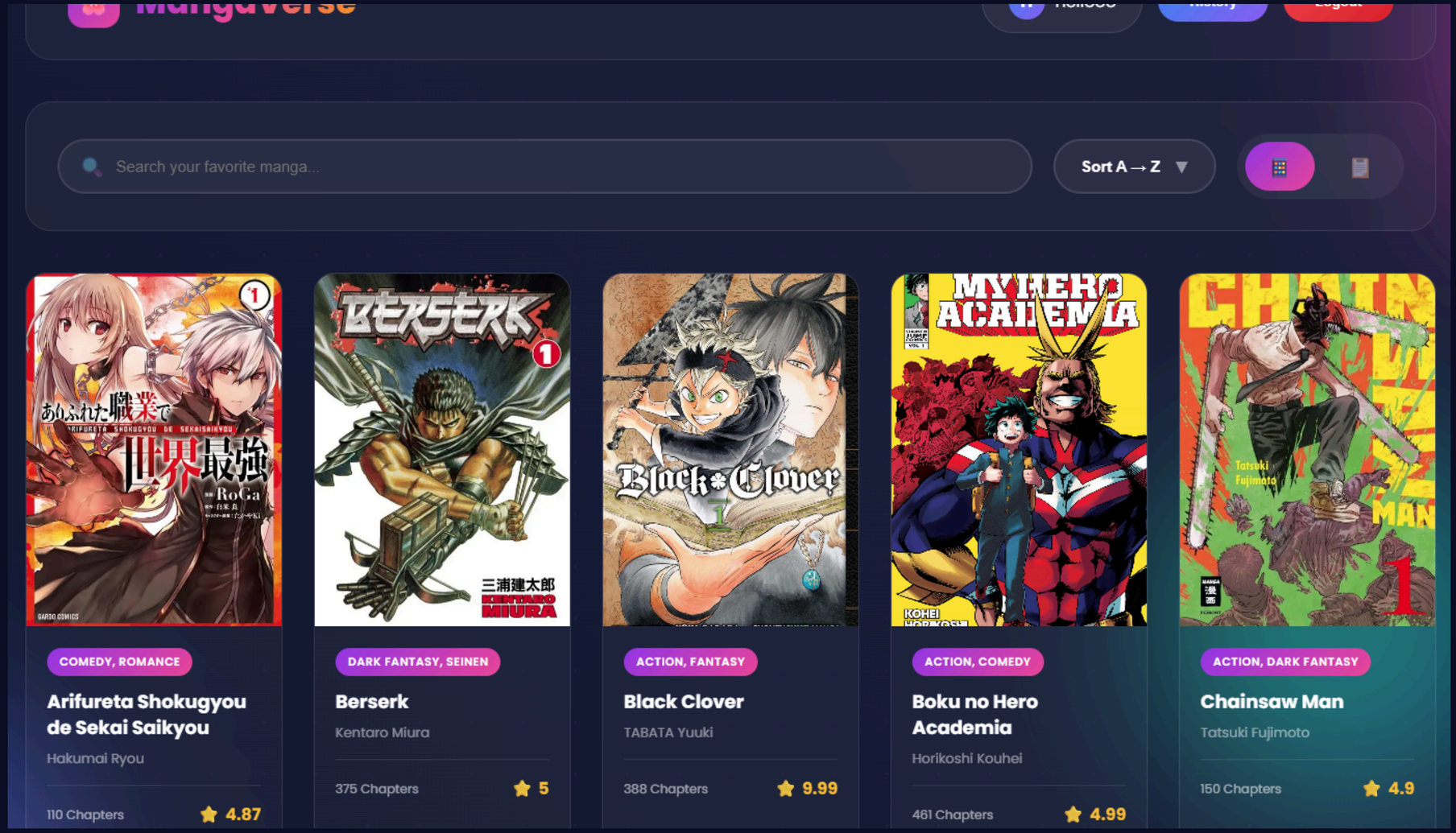
TEST RESULTS: (OUTPUT)



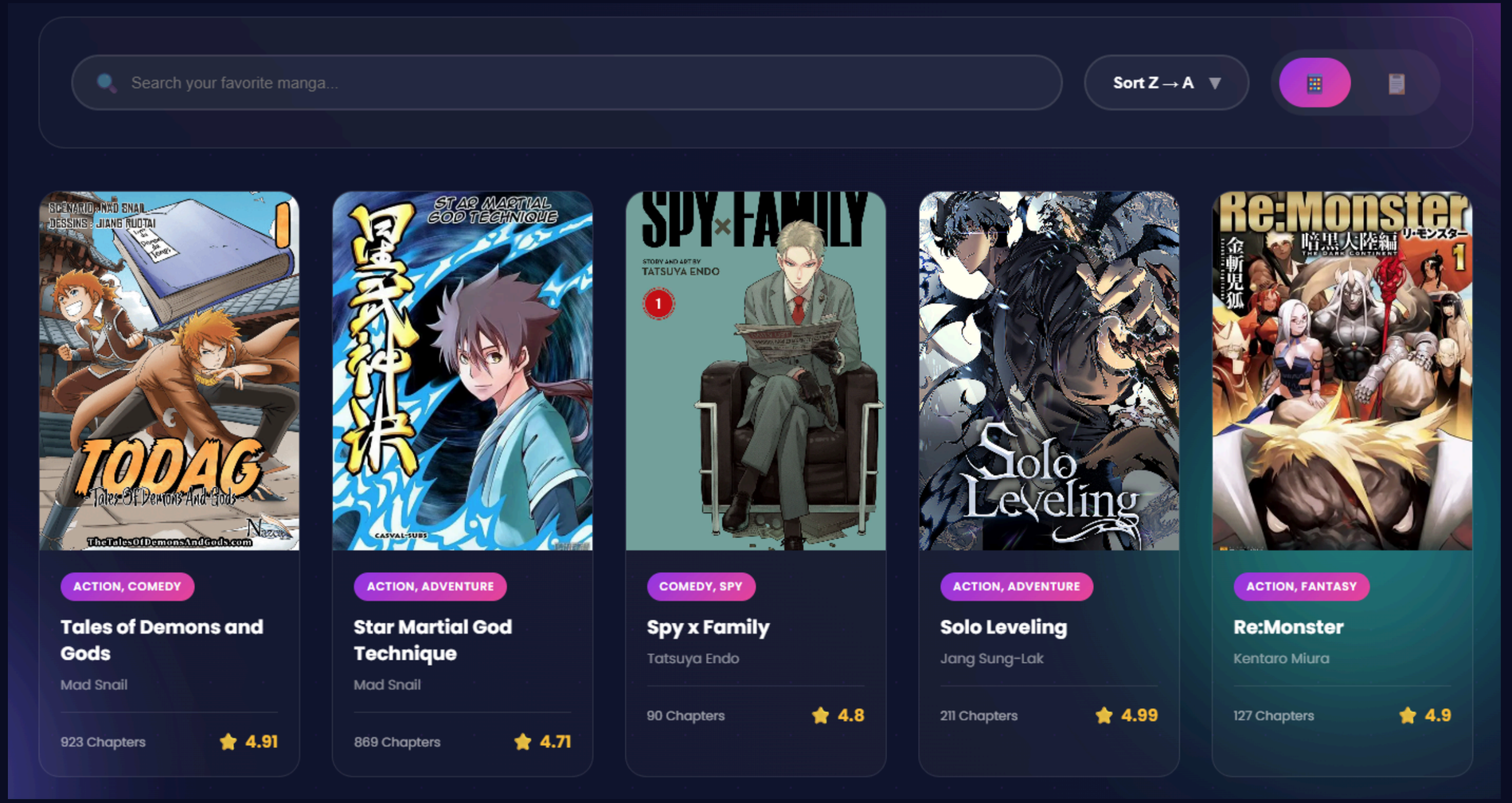
TEST RESULTS: (OUTPUT)



TEST RESULTS: (OUTPUT)



TEST RESULTS: (OUTPUT)



SOURCE CODE: (FRONT END)

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8" />
<meta name="viewport" content="width=device-width,initial-scale=1" />
<title>MangaVerse - Presentation Ready</title>
<style>
/* ---------- Full CSS (glowing UI) ---------- */
@import url('https://fonts.googleapis.com/css2?family=Poppins:wght@300;400;500;600;700;800&display=swap');
* { margin: 0; padding: 0; box-sizing: border-box; }
body {
  font-family: 'Poppins', sans-serif;
  background: #0a0e27;
  color: #fff;
  overflow-x: hidden;
  -webkit-font-smoothing:antialiased;
  -moz-osx-font-smoothing:grayscale;
}

/* Animated Background */
.bg-animation {
  position: fixed; top: 0; left: 0; width: 100%; height: 100%; z-index: -2;
  background: linear-gradient(45deg, #0a0e27, #1a1f3a, #2d1b4e);
  background-size: 400% 400%; animation: gradientFlow 15s ease infinite;
}
.bg-animation::before {
  content: ''; position: absolute; width: 200%; height: 200%; top: -50%; left: -50%;
  background: radial-gradient(circle, rgba(147,51,234,0.1) 1px, transparent 1px);
  background-size: 50px 50px; animation: moveGrid 20s linear infinite;
}
```

SOURCE CODE: (FRONT END)

```
/* Glowing Orbs */
.orb { position: fixed; border-radius: 50%; filter: blur(80px); opacity: 0.6; pointer-events: none; z-index: -1; }
.orb1 { width: 500px; height:500px; background: linear-gradient(45deg,#9333ea,#ec4899); top:-200px; right:-200px; animation: float 20s ease-in-out infinite; }
.orb2 { width: 400px; height:400px; background: linear-gradient(45deg,#3b82f6,#8b5cf6); bottom:-150px; left:-150px; animation: float 15s ease-in-out infinite reve
.orb3 { width: 350px; height:350px; background: linear-gradient(45deg,#10b981,#06b6d4); top:50%; right:10%; animation: float 18s ease-in-out infinite; }
@keyframes float { 0%,100%{transform:translate(0,0) rotate(0)}50%{transform:translate(50px,50px) rotate(180deg)} }

.container { max-width: 1400px; margin: 0 auto; padding: 20px; position: relative; z-index: 1; }

/* Auth Pages */
.auth-container { min-height:100vh; display:flex; justify-content:center; align-items:center; padding:20px; }
.auth-box { background: rgba(255,255,255,0.05); backdrop-filter: blur(20px); border:1px solid rgba(255,255,255,0.1); border-radius:30px; padding:50px 40px; width:
.auth-content { position: relative; z-index: 2; }
.auth-box::before { content:''; position:absolute; top:-50%; left:-50%; width:200%; height:200%; background: radial-gradient(circle, rgba(147,51,234,0.1), transpa
@keyframes rotate { 0%{transform:rotate(0)}100%{transform:rotate(360deg)} }
@keyframes slideUp { from{opacity:0;transform:translateY(50px)} to{opacity:1;transform:translateY(0)} }

.auth-logo { text-align:center; margin-bottom:30px; }
.auth-logo h1 { font-size:42px; font-weight:800; background: linear-gradient(135deg,#9333ea,#ec4899,#f59e0b); -webkit-background-clip:text; -webkit-text-fill-colo
@keyframes shimmer { 0%,100%{filter:brightness(1)}50%{filter:brightness(1.5)} }
.auth-logo p { color: rgba(255,255,255,0.6); font-size:14px; }

```

SOURCE CODE: (FRONT END)

```
56      .form-group { margin-bottom:25px; }
57      .form-group label { display:block; margin-bottom:10px; color: rgba(255,255,255,0.9); font-weight:500; font-size:14px; }
58      .form-group input { width:100%; padding:15px 20px; background: rgba(255,255,255,0.05); border:2px solid rgba(255,255,255,0.1); border-radius:15px; color:#fff; font
59      .form-group input:focus { outline:none; border-color:#9333ea; background:rgba(147,51,234,0.1); box-shadow:0 0 20px rgba(147,51,234,0.3); }
60
61      .btn { width:100%; padding:16px; background: linear-gradient(135deg,#9333ea,#ec4899); color:white; border:none; border-radius:15px; font-size:16px; font-weight:700
62      .btn::before { content:''; position:absolute; top:0; left:-100%; width:100%; height:100%; background:linear-gradient(90deg, transparent, rgba(255,255,255,0.3), tra
63      .btn:hover::before { left:100%; }
64      .btn:hover { transform:translateY(-3px); box-shadow:0 15px 35px rgba(147,51,234,0.5); }
65      .btn:active { transform:translateY(-1px); }
66
67      .toggle-auth { text-align:center; margin-top:25px; color: rgba(255,255,255,0.6); font-size:14px; }
68      .toggle-auth a { color:#ec4899; text-decoration:none; font-weight:600; transition:color .3s; }
69      .toggle-auth a:hover { color:#9333ea; }
70
71      .error-message { background: rgba(239,68,68,0.2); border:1px solid rgba(239,68,68,0.5); color:#fca5a5; padding:12px; border-radius:10px; margin-bottom:20px; displa
72
73      /* Header */
74      .header { background: rgba(255,255,255,0.03); backdrop-filter: blur(20px); border:1px solid rgba(255,255,255,0.1); padding:25px 40px; border-radius:25px; margin-bo
75      @keyframes fadeIn { from{opacity:0} to{opacity:1} }
76
77      .logo { display:flex; align-items:center; gap:15px; }
78      .logo-icon { width:50px; height:50px; background: linear-gradient(135deg,#9333ea,#ec4899); border-radius:15px; display:flex; align-items:center; justify-content:ce
79      @keyframes pulse { 0%,100%{transform:scale(1)}50%{transform:scale(1.1)} }
80      .logo h1 { font-size:32px; font-weight:800; background:linear-gradient(135deg,#9333ea,#ec4899,#f59e0b); -webkit-background-clip:text; -webkit-text-fill-color:trans
```

SOURCE CODE: (FRONT END)

```
85      .logout-btn { padding:12px 30px; background:linear-gradient(135deg,#ef4444,#dc2626); color:white; border:none; border-radius:50px; cursor:pointer; font-weight:600;
86      .logout-btn:hover { transform:translateY(-3px); box-shadow:0 10px 25px rgba(239,68,68,0.4); }
87
88      /* Controls */
89      .controls { background: rgba(255,255,255,0.03); backdrop-filter: blur(20px); border:1px solid rgba(255,255,255,0.1); padding:30px; border-radius:25px; margin-botto
90      .search-box { flex:1; min-width:300px; position:relative; }
91      .search-icon { position:absolute; left:20px; top:50%; transform:translateY(-50%); color:rgba(255,255,255,0.4); font-size:20px; }
92      .search-box input { width:100%; padding:16px 20px 16px 55px; background:rgba(255,255,255,0.05); border:2px solid rgba(255,255,255,0.1); border-radius:50px; color:#
93      .search-box input:focus { outline:none; border-color:#9333ea; background:rgba(147,51,234,0.1); box-shadow:0 0 30px rgba(147,51,234,0.3); }
94      .sort-box { position:relative; }
95      .sort-box select { padding:16px 50px 16px 25px; background:rgba(255,255,255,0.05); border:2px solid rgba(255,255,255,0.1); border-radius:50px; color:#fff; font-siz
96      .sort-box select:focus { outline:none; border-color:#9333ea; box-shadow:0 0 30px rgba(147,51,234,0.3); }
97      .sort-box::after { content:'▼'; position:absolute; right:25px; top:50%; transform:translateY(-50%); pointer-events:none; color:rgba(255,255,255,0.5); }
98      .view-toggle { display:flex; gap:10px; background:rgba(255,255,255,0.05); padding:8px; border-radius:50px; }
99      .view-btn { padding:10px 20px; background:transparent; border:none; border-radius:50px; color:rgba(255,255,255,0.5); cursor:pointer; transition:all .3s ease; font-
100     .view-btn.active { background: linear-gradient(135deg,#9333ea,#ec4899); color:white; }
```

SOURCE CODE: (FRONT END)

```
/* Manga Grid */
.manga-grid { display:grid; grid-template-columns: repeat(auto-fill,minmax(240px,1fr)); gap:30px; margin-bottom:50px; animation: fadeIn 1s ease 0.4s both; }
.manga-card { background: rgba(255,255,255,0.03); backdrop-filter: blur(20px); border:1px solid rgba(255,255,255,0.1); border-radius:20px; overflow:hidden; cursor:
.manga-card::before { content:''; position:absolute; top:0; left:0; right:0; bottom:0; background: linear-gradient(135deg, rgba(147,51,234,0.2), rgba(236,72,153,0.
.manga-card:hover { transform: translateY(-15px) scale(1.05); border-color: rgba(147,51,234,0.5); box-shadow: 0 25px 50px rgba(147,51,234,0.4); }
.manga-card:hover::before { opacity:1; }
.manga-cover-wrapper { position:relative; overflow:hidden; height:340px; }
.manga-cover { width:100%; height:100%; object-fit:cover; transition:transform .6s ease; }
.manga-card:hover .manga-cover { transform:scale(1.15); }
.manga-overlay { position:absolute; top:0; left:0; right:0; bottom:0; background:linear-gradient(to top, rgba(0,0,0,0.9), transparent); opacity:0; transition:opaci
.manga-card:hover .manga-overlay { opacity:1; }
.quick-actions { position:absolute; bottom:15px; left:15px; right:15px; display:flex; gap:10px; opacity:0; transform:translateY(20px); transition:all .4s ease; }
.manga-card:hover .quick-actions { opacity:1; transform:translateY(0); }
.action-btn { flex:1; padding:10px; background:rgba(255,255,255,0.2); backdrop-filter: blur(10px); border:1px solid rgba(255,255,255,0.3); border-radius:10px; colo
.action-btn:hover { background:rgba(255,255,255,0.3); }

.manga-info { padding:20px; position:relative; z-index:1; }
.manga-genre { display:inline-block; padding:5px 12px; background: linear-gradient(135deg,#9333ea,#ec4899); border-radius:20px; font-size:11px; font-weight:600; ma
.manga-title { font-size:18px; font-weight:700; margin-bottom:8px; color:#fff; line-height:1.4; display:-webkit-box; -webkit-line-clamp:2; -webkit-box-orient:verti
.manga-author { font-size:13px; color:rgba(255,255,255,0.5); margin-bottom:15px; }
.manga-meta { display:flex; justify-content:space-between; align-items:center; padding-top:15px; border-top:1px solid rgba(255,255,255,0.1); }
.chapters { font-size:12px; color:rgba(255,255,255,0.6); }
.rating { display:flex; align-items:center; gap:5px; font-weight:700; color:#fbbf24; }
.star { font-size:16px; }
```

SOURCE CODE: (BACK END)

```
// MangaVerse Backend (with Links)

import express from 'express';
import mongoose from 'mongoose';
import cors from 'cors';
import path from 'path';
import jwt from 'jsonwebtoken';
import bcrypt from 'bcryptjs';
import { fileURLToPath } from 'url';
import dotenv from 'dotenv';

// --- INITIAL SETUP ---
dotenv.config();
const app = express();
const PORT = process.env.PORT || 3000;
const JWT_SECRET = process.env.JWT_SECRET || 'fallback_super_secret_key';
const __filename = fileURLToPath(import.meta.url);
const __dirname = path.dirname(__filename);

// --- MIDDLEWARE ---
app.use(cors());
app.use(express.json());

// --- MONGODB CONNECTION ---
mongoose.connect(process.env.MONGO_URI || 'mongodb://localhost:27017/mangaverse')
  .then(() => console.log('✅ MongoDB Connected Successfully'))
  .catch(err => console.error('❌ MongoDB Connection Error:', err));

```

SOURCE CODE: (BACK END)

```
29    // --- DATABASE SCHEMAS & MODELS ---
30    const userSchema = new mongoose.Schema({
31      username: { type: String, required: true },
32      email: { type: String, required: true, unique: true },
33      password: { type: String, required: true },
34      history: [{
35        mangaTitle: String,
36        chaptersRead: Number,
37        date: { type: Date, default: Date.now }
38      }]
39    });
40
41    const mangaSchema = new mongoose.Schema({
42      title: String,
43      author: String,
44      genre: String,
45      cover: String,
46      chapters: Number,
47      rating: Number,
48      // **NEW:** Added a field for an external link
49      link: { type: String, default: '#' },
50      reviews: [{
51        username: String,
52        review: String,
53        date: { type: Date, default: Date.now }
54      }]
55    });
```

SOURCE CODE: (BACK END)

```
41    const mangaSchema = new mongoose.Schema({
42      title: String,
43      author: String,
44      genre: String,
45      cover: String,
46      chapters: Number,
47      rating: Number,
48      // **NEW:** Added a field for an external link
49      link: { type: String, default: '#' },
50      reviews: [{
51        username: String,
52        review: String,
53        date: { type: Date, default: Date.now }
54      }]
55    });
56
57    const User = mongoose.model('User', userSchema);
58    const Manga = mongoose.model('Manga', mangaSchema);
```

SOURCE CODE: (BACK END)

```
41    const mangaSchema = new mongoose.Schema({
42      title: String,
43      author: String,
44      genre: String,
45      cover: String,
46      chapters: Number,
47      rating: Number,
48      // **NEW:** Added a field for an external link
49      link: { type: String, default: '#' },
50      reviews: [{
51        username: String,
52        review: String,
53        date: { type: Date, default: Date.now }
54      }]
55    });
56
57    const User = mongoose.model('User', userSchema);
58    const Manga = mongoose.model('Manga', mangaSchema);
```

SOURCE CODE: (BACK END)

```
60    // --- AUTH MIDDLEWARE ---
61    // ... (verifyToken function remains the same) ...
62    function verifyToken(req, res, next) {
63      const authHeader = req.headers['authorization'];
64      const token = authHeader && authHeader.split(' ')[1];
65      if (!token) {
66        return res.status(401).json({ message: 'Access denied. No token provided.' });
67      }
68      try {
69        const decoded = jwt.verify(token, JWT_SECRET);
70        req.user = decoded;
71        next();
72      } catch (err) {
73        res.status(400).json({ message: 'Invalid token.' });
74      }
75    }
76
```

SOURCE CODE: (BACK END)

```
79    app.post('/api/register', async (req, res) => {
80      try {
81        const { username, email, password } = req.body;
82        const hashedPassword = await bcrypt.hash(password, 10);
83        const newUser = new User({ username, email, password: hashedPassword });
84        await newUser.save();
85        const token = jwt.sign({ id: newUser._id, username: newUser.username }, JWT_SECRET, { expiresIn: '1d' });
86        res.status(201).json({ token, user: { username: newUser.username, email: newUser.email } });
87      } catch (err) {
88        res.status(400).json({ message: 'Registration failed. The email might already be in use.' });
89      }
90    });
91
92    app.post('/api/login', async (req, res) => {
93      try {
94        const { email, password } = req.body;
95        const user = await User.findOne({ email });
96        if (!user || !await bcrypt.compare(password, user.password)) {
97            return res.status(400).json({ message: 'Invalid credentials.' });
98        }
99        const token = jwt.sign({ id: user._id, username: user.username }, JWT_SECRET, { expiresIn: '1d' });
100       res.json({ token, user: { username: user.username, email: user.email } });
101     } catch (err) {
102       res.status(500).json({ message: 'Login failed due to a server error.' });
103     }
104   });
105
```

CONCLUSION:

The MangaVerse system demonstrated both the potential of the front-end and back-end technologies to create a dynamic, interactive, and user-friendly manga browsing platform. The front-end delivers visually striking effects and animation, and responsive design by using HTML, CSS, and JavaScript. Both Node.js and Express.js back-end make accessing data effortless; MongoDB is a flexible, powerful NoSQL database for users, manga, reviews, and reading histories; and MongoDB is flexible and efficient, providing users, manga data, reviews, and reading histories with high data accuracy.

CONCLUSION:

The greatest success of this project is the security built into the JWT system for authentication on JWT, which safeguards users' sessions in a way that reflects the simplicity of the interface. It also combines MongoDB's powerful sorting algorithms, B-tree index traversal, and in-memory merge sort to display manga results in ascending or descending order. It is the combination of speed and scale that allows a powerful application to manage multiple users, large datasets without an increase in performance loss.

RECOMMENDATION:

While MangaVerse has achieved its core goals, several areas can be improved for improved functionality and scalability.

- Add Manga Upload and Management System: Add an admin panel to allow authorized users to upload, edit, or remove manga entries dynamically without needing a static seeding solution.

- Integration and infinite scroll: In addition to pagination and infinite scroll, to increase performance, a possible scroll or infinite scroll can be added to the pagination and infinite scroll settings to speed up the loading of manga entries in large datasets.

- Improve Security: Add password hashing to the system using bcrypt and rate limiting to improve data protection.

- Involve cloud storage: Store manga cover images and data on a cloud service like AWS S3, Cloudinary, or Firebase Storage, and be able to scale and access all the data for global access.

RECOMMENDATION:

To run a mobile version of MangaVerse on mobile devices with React Native or Flutter to provide a cross-platform reading experience.

Recommendation System: Use a simple AI or machine learning model that suggests manga based on user preferences or reading history to suggest manga for user-defined preferences.

- Improve Review System: Add the ability to submit or comment on reviews and to display the average rating in each manga.
- Use cloud MongoDB (Atlas) for remote deployment and multi-user access.
- Incorporate real-time features (e.g., live reviews, reader statistics) using WebSockets.

RECOMMENDATION:

In conclusion, MangaVerse successfully lays the groundwork for an advanced, user-centric manga platform that merges aesthetic design with powerful functionality. The project not only serves as a strong example of practical web development but also opens pathways for future enhancement using modern technologies such as machine learning, cloud computing, and progressive web app (PWA) integration.

REFERENCES:

1. GeeksforGeeks. (n.d.). Sorting Algorithms in JavaScript. Retrieved from https://www.geeksforgeeks.org/sorting-algorithms/
2. MongoDB Documentation. (n.d.). Sorting Query Results. Retrieved from https://www.mongodb.com/docs/manual/reference/method/cursor.sort/
3. W3Schools. (n.d.). Node.js Tutorial. Retrieved from https://www.w3schools.com/nodejs/
4. MangaHere. (n.d.). MangaHere Official Website. Retrieved from https://www.mangahere.cc/
5. Mozilla Developer Network (MDN). (n.d.). JavaScript Array Methods. Retrieved from https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array
6. FreeCodeCamp. (2021). Understanding MongoDB for Beginners. Retrieved from https://www.freecodecamp.org/news/learn-mongodb-a4ce205e7739/
7. CSS-Tricks. (n.d.). Creating Animated Backgrounds in CSS. Retrieved from https://css-tricks.com/
8. Stack Overflow. (n.d.). Implementing Sorting Algorithms in JavaScript. Retrieved from https://stackoverflow.com/
9. Tutorialspoint. (n.d.). Express.js Overview. Retrieved from https://www.tutorialspoint.com/expressjs/index.htm