MangaVerse: A Comic Website

Project Documentation

Submitted by:
Michael George E. Aquino
James Adrian B. Castro
Rafael Iñigo T. Lopez
Alvin G. Pableo
Oscar Cloud Vincent C. Yumang

Course: BS Computer Science
Subject: Data Structures and Algorithms
Institution: National University – Manila
Date: October 2025

Submitted to:
Ms. Marizkays P. Jamison

Github Link:
https://github.com/debug-phantom/CCDATRCL-PROJECT

I. 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.

A. 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.

MangaVerse is a site designed to be well-organized, user-friendly, and educational, providing users with the space and information to easily access, search, organize, and track manga reading habits. In the most current manga site, content delivery is most common, but data management, user interaction, and backend integration are important themes that are most important to understanding modern web systems.

Within the context of academia, this project has served as a bridge between web development theory and practical applications and integrated database management, user authentication, and algorithmic design into a cohesive system. Also highlighted here is how data structures and algorithms, such as arrays, stacks, and sorting, are essential to improving efficiency, organization, and overall system performance. The project was thus created both as entertainment and as a tool to show how professional-grade web systems operate using modular code architecture and full-stack implementation.

B. 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.

The purpose of this project is to demonstrate the practical application of computer science concepts, including data structures, database design, and sorting algorithms, in real-world development scenarios. On the backend, Node.js and MongoDB provide efficient data processing and secure user authentication. The frontend utilizes HTML, CSS, and JavaScript to deliver a visually appealing and responsive web interface. Additionally, the backend provides a visually pleasing and responsive frontend for HTML, CSS, and JavaScript.

MangaVerse serves both educational and practical purposes: it facilitates users' access to a virtual manga library, while providing a foundation for students and developers to learn how different technologies (frontend, backend, database) work together to create a coherent and secure online system.

II. 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.

III. 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 (to serve as a database)
- Any modern browser (Chrome, Edge, Firefox)
- NetBeans IDE, VS Code (for development and testing)

IV. 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.

V. 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):

Pseudocode:

```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")
```

Explanation:
This ensures secure access control by hashing user passwords before storage and verifying them during login. The use of JWT tokens allows authenticated communication between the frontend and backend.

2. **Manga Sorting Algorithm:**

When displaying manga collections, the system sorts manga titles alphabetically (A–Z or Z–A) depending on user selection.
This is achieved through MongoDB's built-in .sort() method.

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

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**

Pseudocode:

```
pgsql

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

Explanation:
When a user reads a manga, the system logs it with the manga title, chapters read, and timestamp. This acts like a stack (most recent entries appear first), tracking reading progression efficiently.

4. **Review Submission Algorithm**

Pseudocode:

```
css

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

Explanation:
Reviews are appended directly to a manga's document. This approach simplifies retrieval and avoids separate review collections, allowing quick user-to-manga association.

VI. 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

VI.1:
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 } });
});
```

```
gistration
/api/register', async (req, res) => {
username, email, password } = req.body;
shedPassword = await bcrypt.hash(password, 10);
wUser = new User({ username, email, password: hashedPassword });
wUser.save();
ken = jwt.sign({ id: newUser._id, username: newUser.username }, JWT_SECRET, { expiresIn: '1d' });
us(201).json({ token, user: { username: newUser.username, email: newUser.email } });
```

VI.2
Key Code Snippet (Reading History):

```js
// 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);
});
```

VI.3
Key Code Snippet (Sorting Manga):

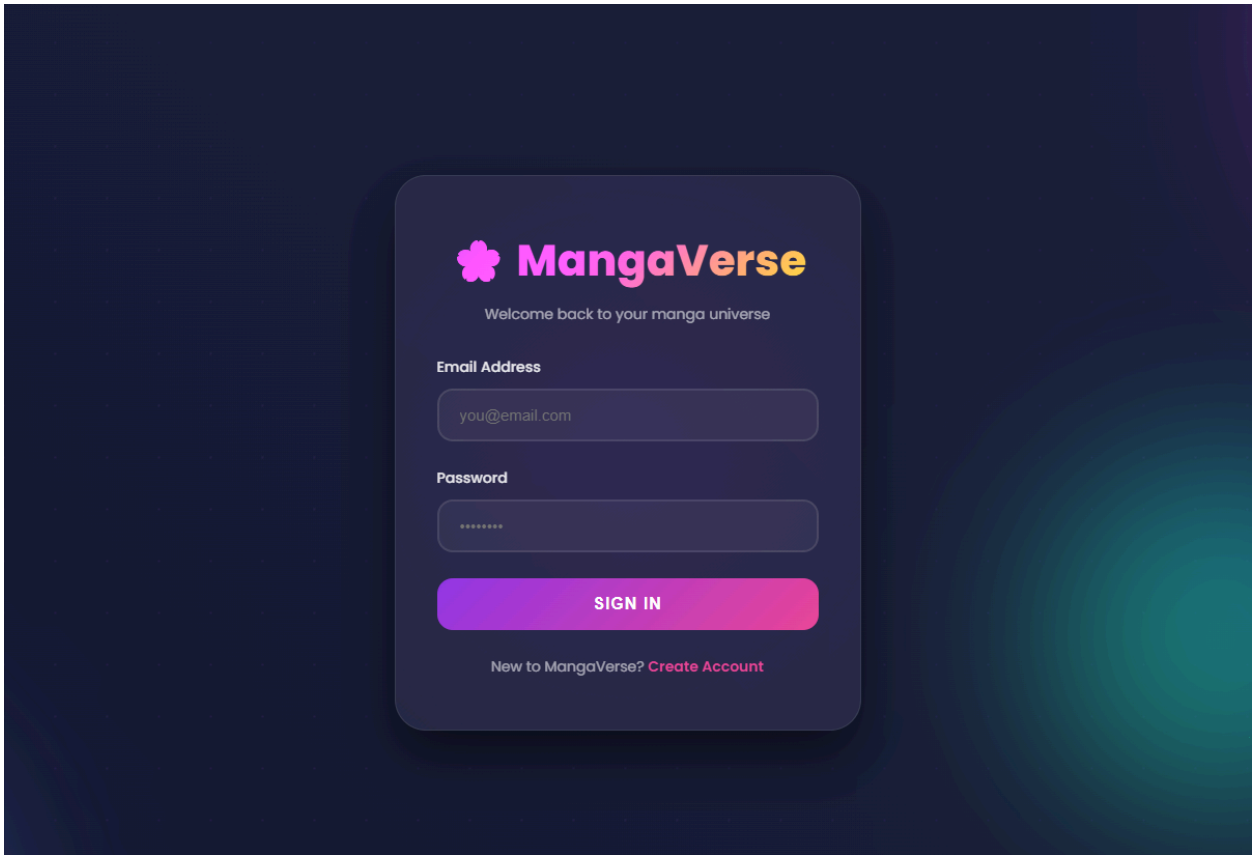
```js
app.get('/api/mangas', verifyToken, async (req, res) => {
  const { sort = 'asc', search = '' } = req.query;
  const mangas = await Manga.find({ title: { $regex: search, $options: 'i' } })
                            .sort({ title: sort === 'asc' ? 1 : -1 });
  res.json(mangas);
});
```

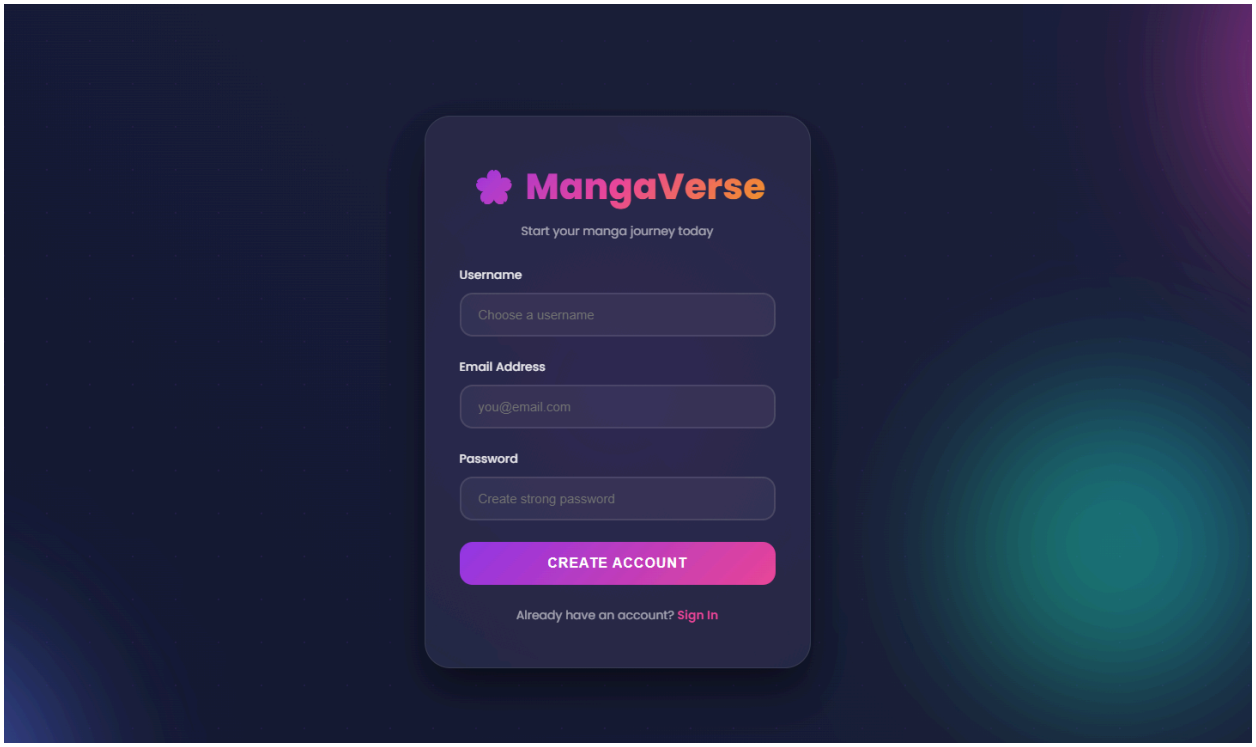
VII. Testing and Results

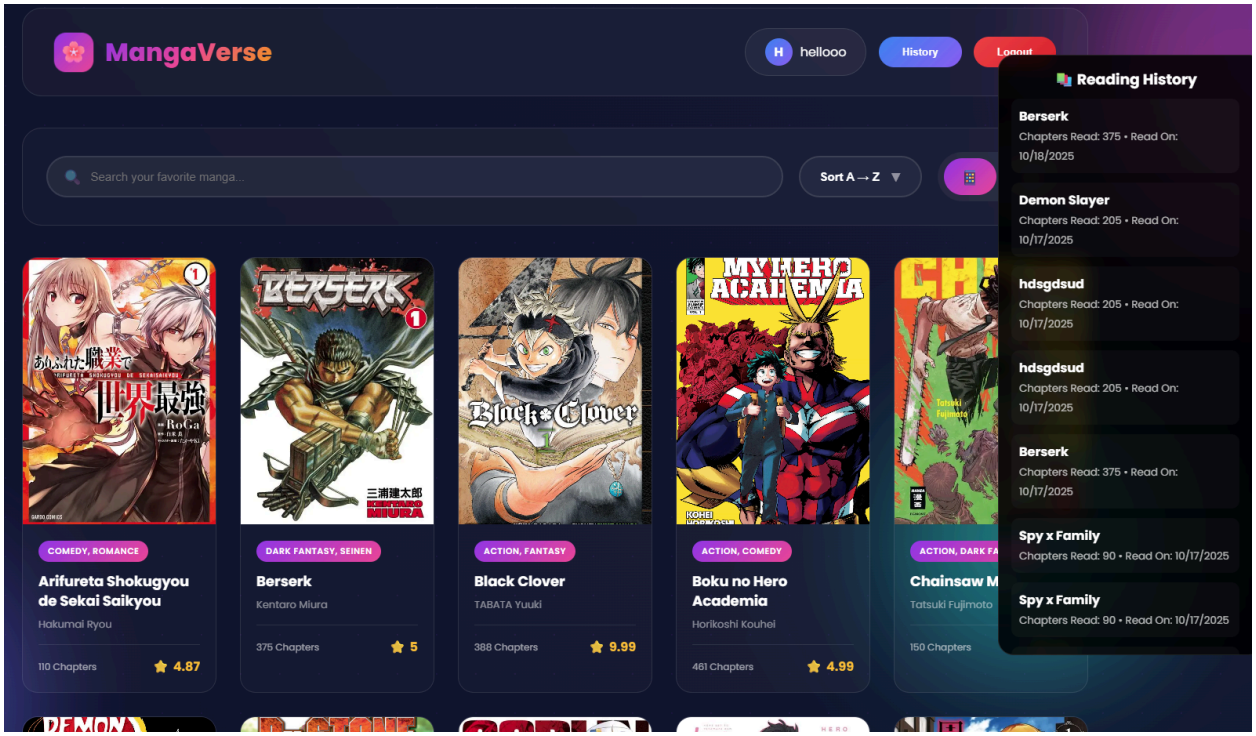
VII Table I. Test Table

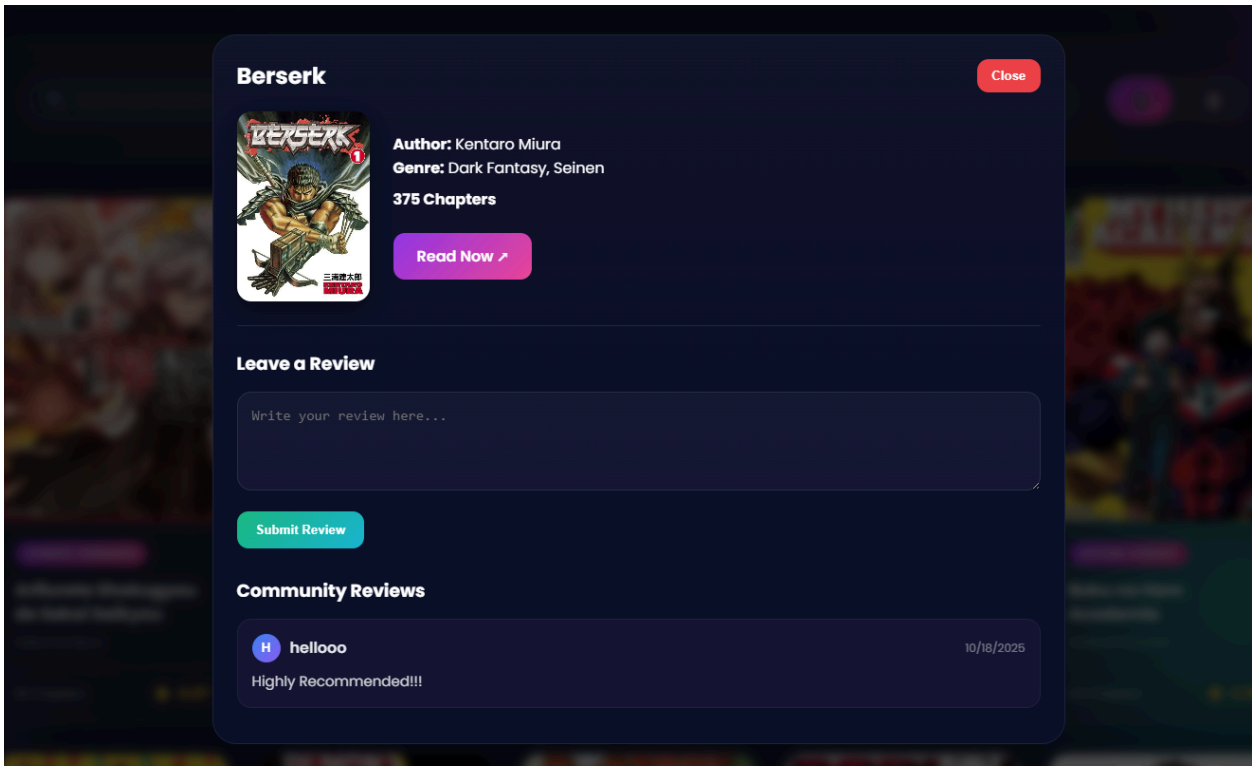
Test Case	Input	Expected Output	Actual Output	Result
Register User	username: “Leo”, email: “leo@test.com” , password: “12345”	Token returned & user saved	Success	✅
Login User	Valid email & password	JWT Token returned	Success	✅
Load Manga	/api/mangas?sort=asc	Sorted manga list	Sorted correctly	✅
Add Review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

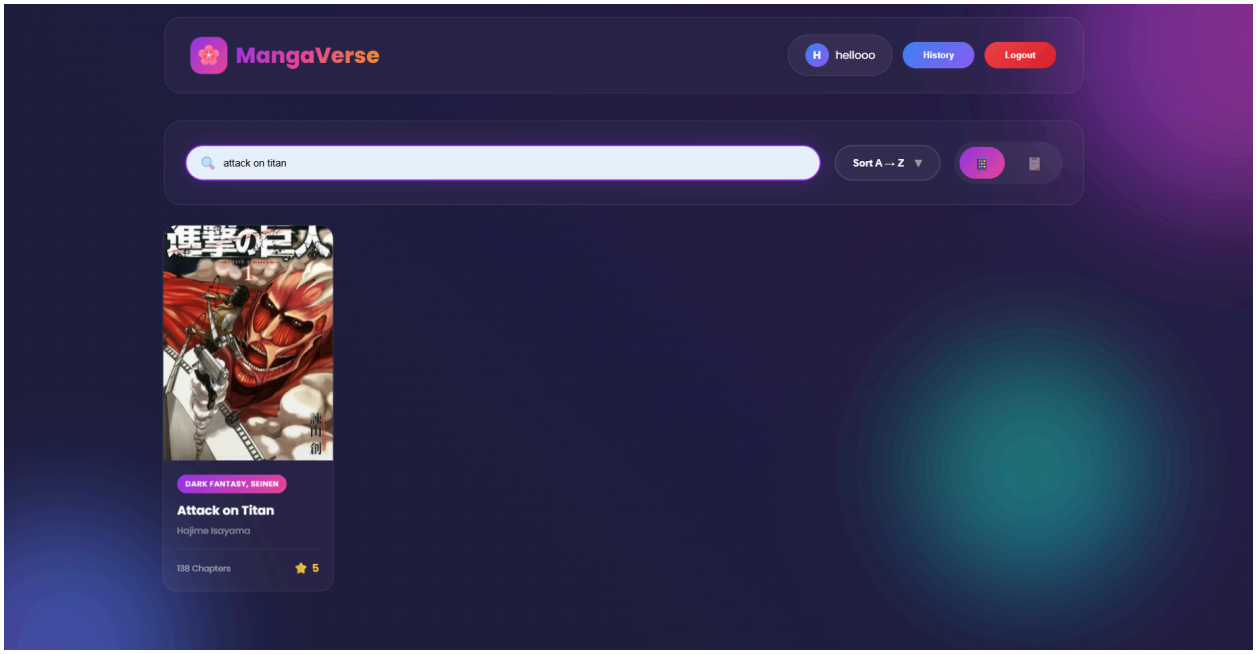
VII. Test Results (Screenshots of Output):

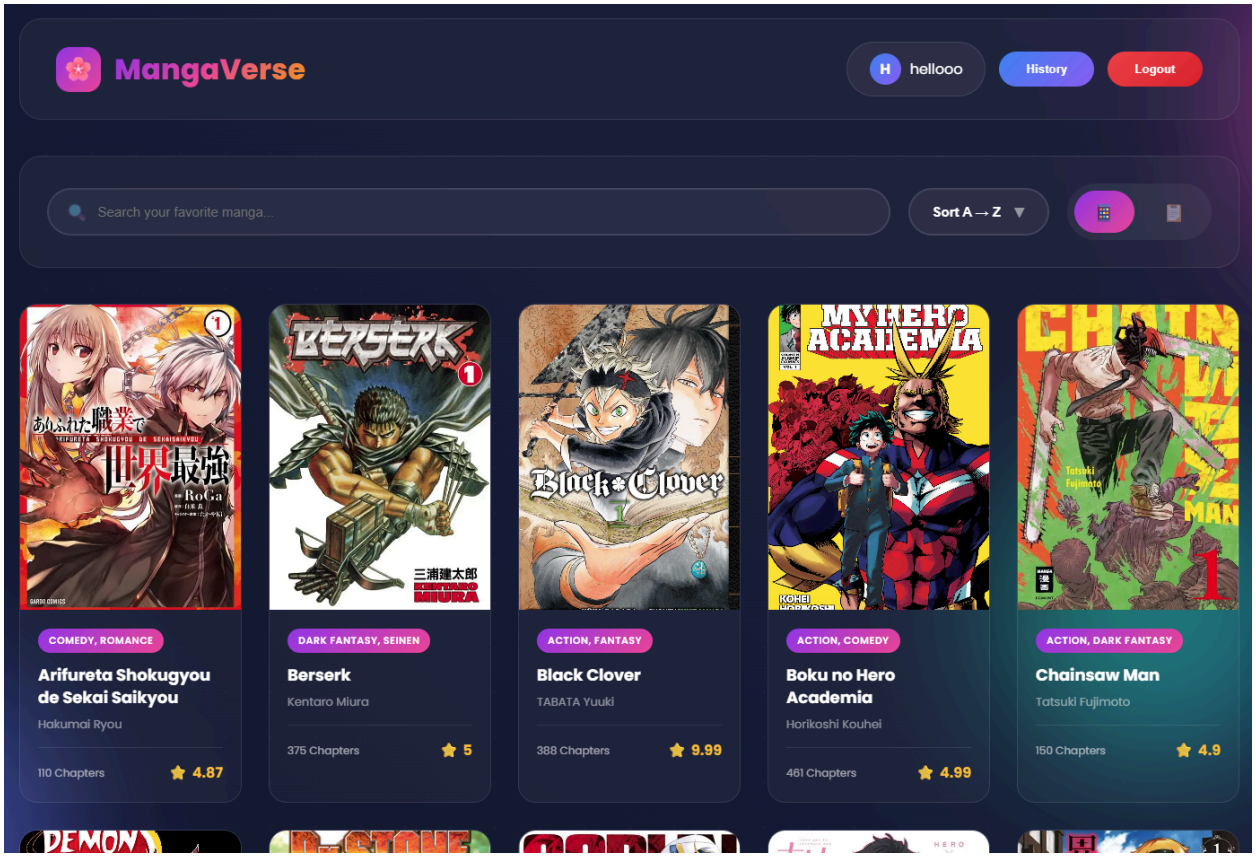





MangaVerse
H
hellooo
History
Logout
Search your favorite manga...
Sort A → Z
Reading History
Berserk
Chapters Read: 375 • Read On: 10/18/2025
Demon Slayer
Chapters Read: 205 • Read On: 10/17/2025
hdsgdsud
Chapters Read: 205 • Read On: 10/17/2025
hdsgdsud
Chapters Read: 205 • Read On: 10/17/2025
Berserk
Chapters Read: 375 • Read On: 10/17/2025
Spy x Family
Chapters Read: 90 • Read On: 10/17/2025
Spy x Family
Chapters Read: 90 • Read On: 10/17/2025
COMEDY, ROMANCE
Arifureta Shokugyou de Sekai Saikyou
Hakumai Ryou
110 Chapters
4.87
DARK FANTASY, SEINEN
Berserk
Kentaro Miura
375 Chapters
5
ACTION, FANTASY
Black Clover
TABATA Yuuki
388 Chapters
9.99
ACTION, COMEDY
Boku no Hero Academia
Horikoshi Kouhei
461 Chapters
4.99
ACTION, DARK FA
Chainsaw M
Tatsuki Fujimoto
150 Chapters


Berserk
Close
Author: Kentaro Miura
Genre: Dark Fantasy, Seinen
375 Chapters
Read Now ↗
Leave a Review
Write your review here...
Submit Review
Community Reviews
H
hellooo
10/18/2025
Highly Recommended!!!


MangaVerse
H
hellooo
History
Logout
attack on titan
Sort A → Z
DARK FANTASY, SEINEN
Attack on Titan
Hajime Isayama
138 Chapters
5


MangaVerse
H
hellooo
History
Logout
Search your favorite manga...
Sort A → Z
COMEDY, ROMANCE
Arifureta Shokugyou de Sekai Saikyou
Hakumai Ryou
110 Chapters
4.87
DARK FANTASY, SEINEN
Berserk
Kentaro Miura
375 Chapters
5
ACTION, FANTASY
Black Clover
TABATA Yuuki
388 Chapters
9.99
ACTION, COMEDY
Boku no Hero Academia
Horikoshi Kouhei
461 Chapters
4.99
ACTION, DARK FANTASY
Chainsaw Man
Tatsuki Fujimoto
150 Chapters
4.9

VII. 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;
}
```

```
@keyframes gradientFlow { 0%{background-position:0% 50%}50%{background-position:100% 50%}100%{background-position:0% 50%}}
@keyframes moveGrid { 0%{transform:translate(0,0)}100%{transform:translate(50px,50px)}}

/* 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 rever
.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:1
.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), transpar
@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-color
@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; }

.form-group { margin-bottom:25px; }
.form-group label { display:block; margin-bottom:10px; color: rgba(255,255,255,0.9); font-weight:500; font-size:14px; }
.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
.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); }

.btn { width:100%; padding:16px; background: linear-gradient(135deg,#9333ea,#ec4899); color:white; border:none; border-radius:15px; font-size:16px; font-weight:700
.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
.btn:hover::before { left:100%; }
.btn:hover { transform:translateY(-3px); box-shadow:0 15px 35px rgba(147,51,234,0.5); }
.btn:active { transform:translateY(-1px); }

.toggle-auth { text-align:center; margin-top:25px; color: rgba(255,255,255,0.6); font-size:14px; }
.toggle-auth a { color:#ec4899; text-decoration:none; font-weight:600; transition:color .3s; }
.toggle-auth a:hover { color:#9333ea; }

.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

/* Header */
.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
@keyframes fadeIn { from{opacity:0} to{opacity:1} }

.logo { display:flex; align-items:center; gap:15px; }
.logo-icon { width:50px; height:50px; background: linear-gradient(135deg,#9333ea,#ec4899); border-radius:15px; display:flex; align-items:center; justify-content:ce
@keyframes pulse { 0%,100%{transform:scale(1)}50%{transform:scale(1.1)} }
.logo h1 { font-size:32px; font-weight:800; background:linear-gradient(135deg,#9333ea,#ec4899,#f59e0b); -webkit-background-clip:text; -webkit-text-fill-color:trans

.header-actions { display:flex; gap:15px; align-items:center; }
.user-badge { background: rgba(255,255,255,0.05); padding:12px 25px; border-radius:50px; border:1px solid rgba(255,255,255,0.1); display:flex; align-items:center;
.user-avatar { width:35px; height:35px; background:linear-gradient(135deg,#3b82f6,#8b5cf6); border-radius:50%; display:flex; align-items:center; justify-content:ce
.logout-btn { padding:12px 30px; background:linear-gradient(135deg,#ef4444,#dc2626); color:white; border:none; border-radius:50px; cursor:pointer; font-weight:600;
.logout-btn:hover { transform:translateY(-3px); box-shadow:0 10px 25px rgba(239,68,68,0.4); }

/* Controls */
.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-bottc
.search-box { flex:1; min-width:300px; position:relative; }
```

```
.search-icon { position:absolute; left:20px; top:50%; transform:translateY(-50%); color:rgba(255,255,255,0.4); font-size:20px; }
.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:#
.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); }
.sort-box { position:relative; }
.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
.sort-box select:focus { outline:none; border-color:#9333ea; box-shadow:0 0 30px rgba(147,51,234,0.3); }
.sort-box::after { content:'▼'; position:absolute; right:25px; top:50%; transform:translateY(-50%); pointer-events:none; color:rgba(255,255,255,0.5); }
.view-toggle { display:flex; gap:10px; background:rgba(255,255,255,0.05); padding:8px; border-radius:50px; }
.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-
.view-btn.active { background: linear-gradient(135deg,#9333ea,#ec4899); color:white; }

/* 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; }

/* Loading */
.loading { text-align:center; padding:100px 20px; }
.loader { width:60px; height:60px; border:4px solid rgba(255,255,255,0.1); border-top-color:#9333ea; border-radius:50%; animation:spin 1s linear infinite; margin:0
@keyframes spin { to { transform:rotate(360deg) } }
.loading-text { font-size:18px; color:rgba(255,255,255,0.7) }

.hidden { display:none !important; }

/* Modal + reviews */
.modal-backdrop { position:fixed; inset:0; background:rgba(0,0,0,0.7); backdrop-filter: blur(5px); display:flex; align-items:center; justify-content:center; z-inde
.modal { width:90%; max-width:900px; background: linear-gradient(180deg, #10142c, #0c102d); border:1px solid rgba(255,255,255,0.08); border-radius:25px; padding:25
.modal .manga-header { display:flex; gap:25px; align-items:center; margin-bottom:20px; }
.modal .manga-cover { width:140px; height:200px; object-fit:cover; border-radius:15px; box-shadow: 0 10px 30px rgba(0,0,0,0.5); }

.review-item {
  background: rgba(255,255,255,0.03);
  padding: 15px;
  border-radius: 15px;
  margin-bottom: 12px;
  border: 1px solid rgba(255,255,255,0.05);
}
.review-header {
  display: flex;
  align-items: center;
```

```
    gap: 10px;
    margin-bottom: 8px;
  }
  .review-avatar {
    width: 30px;
    height: 30px;
    border-radius: 50%;
    background: linear-gradient(135deg,#3b82f6,#8b5cf6);
    display: flex;
    align-items: center;
    justify-content: center;
    font-weight: 600;
    font-size: 14px;
  }
  .review-author {
    font-weight: 600;
    color: #e5e7eb;
  }
  .review-date {
    font-size: 12px;
    color: rgba(255,255,255,0.4);
    margin-left: auto;
  }
  .review-body {
    font-size: 15px;
    line-height: 1.6;
    color: rgba(255,255,255,0.8);
  }

```

```
@media (max-width:768px) {
  .manga-cover-wrapper { height: 280px; }
  .manga-grid { grid-template-columns: repeat(auto-fill,minmax(180px,1fr)); gap:20px; }
  .modal { width:95%; padding: 15px; }
  .modal .manga-header { flex-direction: column; align-items: flex-start; }
}
</style>
</head>
<body>
  <div class="bg-animation"></div>
  <div class="orb orb1"></div>
  <div class="orb orb2"></div>
  <div class="orb orb3"></div>

  <div id="loginPage" class="auth-container">
    <div class="auth-box">
      <div class="auth-content">
        <div class="auth-logo">
          <h1>🌸 MangaVerse</h1>
          <p>Welcome back to your manga universe</p>
        </div>
        <div id="loginError" class="error-message"></div>
        <form id="loginForm">
          <div class="form-group">
            <label>Email Address</label>
            <input type="email" id="loginEmail" required placeholder="you@email.com">
          </div>
          <div class="form-group">
            <label>Password</label>
            <input type="password" id="loginPassword" required placeholder="••••••••">
          </div>
          <button type="submit" class="btn">Sign In</button>
        </form>
        <div class="toggle-auth">New to MangaVerse? <a href="#" id="showRegister">Create Account</a></div>
      </div>
    </div>
  </div>

  <div id="registerPage" class="auth-container hidden">
    <div class="auth-box">
      <div class="auth-content">
        <div class="auth-logo">
          <h1>🌸 MangaVerse</h1>
          <p>Start your manga journey today</p>
        </div>
        <div id="registerError" class="error-message"></div>
        <form id="registerForm">
          <div class="form-group"><label>Username</label><input type="text" id="registerUsername" required placeholder="Choose a username"></div>
          <div class="form-group"><label>Email Address</label><input type="email" id="registerEmail" required placeholder="you@email.com"></div>
          <div class="form-group"><label>Password</label><input type="password" id="registerPassword" required placeholder="Create strong password"></div>
          <button type="submit" class="btn">Create Account</button>
        </form>
        <div class="toggle-auth">Already have an account? <a href="#" id="showLogin">Sign In</a></div>
      </div>
    </div>
  </div>

  <div id="mainPage" class="container hidden">
    <div class="header">
      <div class="logo">
```

```
          <div class="logo-icon">🌸</div>
          <h1>MangaVerse</h1>
        </div>
        <div class="header-actions">
          <div class="user-badge">
            <div class="user-avatar" id="userAvatar"></div>
            <span id="userName"></span>
          </div>
          <button class="logout-btn" id="historyBtn" style="background:linear-gradient(135deg,#3b82f6,#8b5cf6);">History</button>
          <button class="logout-btn" id="logoutBtn">Logout</button>
        </div>
      </div>

      <div class="controls">
        <div class="search-box">
          <span class="search-icon">🔍</span>
          <input type="text" id="searchInput" placeholder="Search your favorite manga...">
        </div>
        <div class="sort-box">
          <select id="sortSelect"><option value="asc">Sort A → Z</option><option value="desc">Sort Z → A</option></select>
        </div>
        <div class="view-toggle">
          <button class="view-btn active" data-view="grid">▦</button>
          <button class="view-btn" data-view="list">☰</button>
        </div>
      </div>

      <div id="mangaGrid" class="manga-grid"></div>

      <div id="loading" class="loading hidden">
```

```
        <div class="loader"></div>
        <div class="loading-text">Loading your manga collection...</div>
      </div>
    </div>

  <script>
  /* ---------- Full Frontend JS (with Links) ---------- */

  const API_URL = 'http://localhost:3000/api';
  let token = localStorage.getItem('token');
  let currentUser = JSON.parse(localStorage.getItem('user') || '{}');
  let allMangas = [];

  if (token) {
    showMainPage();
  }
  document.getElementById('showRegister').addEventListener('click', (e) => { e.preventDefault(); document.getElementById('loginPage').classList.add('hidden'); docume
  document.getElementById('showLogin').addEventListener('click', (e) => { e.preventDefault(); document.getElementById('registerPage').classList.add('hidden'); docume
  document.getElementById('loginForm').addEventListener('submit', async (e) => {
    e.preventDefault();
    const email = document.getElementById('loginEmail').value.trim();
    const password = document.getElementById('loginPassword').value;
    try {
      const res = await fetch(`${API_URL}/login`, {
        method: 'POST', headers: {'Content-Type':'application/json'}, body: JSON.stringify({ email, password })
      });
      const data = await res.json();
      if (!res.ok) return showError('loginError', data.message || 'Login failed');
      localStorage.setItem('token', data.token);
      localStorage.setItem('user', JSON.stringify(data.user));
```

```
      token = data.token;
      currentUser = data.user;
      showMainPage();
    } catch (err) {
      showError('loginError','Connection error. Please try again.');
    }
  });
  document.getElementById('registerForm').addEventListener('submit', async (e) => {
    e.preventDefault();
    const username = document.getElementById('registerUsername').value.trim();
    const email = document.getElementById('registerEmail').value.trim();
    const password = document.getElementById('registerPassword').value;
    try {
      const res = await fetch(`${API_URL}/register`, {
        method: 'POST', headers: {'Content-Type':'application/json'}, body: JSON.stringify({ username, email, password })
      });
      const data = await res.json();
      if (!res.ok) return showError('registerError', data.message || 'Registration failed.');
      localStorage.setItem('token', data.token);
      localStorage.setItem('user', JSON.stringify(data.user));
      token = data.token;
      currentUser = data.user;
      showMainPage();
    } catch (err) {
      showError('registerError','Connection error. Please try again.');
    }
  });
  document.getElementById('logoutBtn').addEventListener('click', () => {
    localStorage.removeItem('token');
    localStorage.removeItem('user');
    token = null;
    currentUser = {};
    allMangas = [];
    const panel = document.getElementById('historyPanel');
    if (panel) panel.remove();
    document.getElementById('mainPage').classList.add('hidden');
    document.getElementById('loginPage').classList.remove('hidden');
  });
  function showMainPage() {
    document.getElementById('loginPage').classList.add('hidden');
    document.getElementById('registerPage').classList.add('hidden');
    document.getElementById('mainPage').classList.remove('hidden');
    document.getElementById('userName').textContent = currentUser.username || 'User';
    document.getElementById('userAvatar').textContent = (currentUser.username && currentUser.username.charAt(0).toUpperCase()) || 'U';
    loadMangas();
  }
  async function loadMangas() {
    const sort = document.getElementById('sortSelect').value;
    const search = document.getElementById('searchInput').value;
    const loading = document.getElementById('loading');
    const grid = document.getElementById('mangaGrid');
    loading.classList.remove('hidden');
    grid.innerHTML = '';
    try {
      const res = await fetch(`${API_URL}/mangas?sort=${sort}&search=${encodeURIComponent(search)}`, {
        headers: { 'Authorization': `Bearer ${token}` }
      });
      if (!res.ok) throw new Error('Failed to fetch mangas');
      const mangas = await res.json();
      allMangas = mangas;
```

```
        loading.classList.add('hidden');
        renderMangaGrid();
      } catch (err) {
        loading.classList.add('hidden');
        grid.innerHTML = `<p style="text-align:center;color:#f87171;">Failed to load mangas. Please refresh.</p>`;
      }
    }
    function renderMangaGrid() {
        const grid = document.getElementById('mangaGrid');
        grid.innerHTML = allMangas.map(m => `
          <div class="manga-card" data-id="${m._id}">
            <div class="manga-cover-wrapper">
              <img class="manga-cover" src="${m.cover || ''}" alt="${escapeHtml(m.title)}" onerror="this.src='https.placehold.co/240x340/0a0e27/ffffff?text=No+Image'">
              <div class="manga-overlay"></div>
              <div class="quick-actions">
                <button class="action-btn read-btn" data-id="${m._id}">Details & Reviews</button>
              </div>
            </div>
            <div class="manga-info">
              <span class="manga-genre">${escapeHtml(m.genre)}</span>
              <h3 class="manga-title">${escapeHtml(m.title)}</h3>
              <p class="manga-author">${escapeHtml(m.author)}</p>
              <div class="manga-meta">
                <span class="chapters">${m.chapters} Chapters</span>
                <span class="rating"><span class="star">⭐</span>${m.rating}</span>
              </div>
            </div>
          </div>`).join('');

        document.querySelectorAll('.read-btn').forEach(b => {
          b.addEventListener('click', (e) => {
            e.stopPropagation();
            const mangaId = e.target.getAttribute('data-id');
            openReadModal(mangaId);
          });
        });
    }
    function renderReviews(reviews, containerEl) {
        if (!reviews || reviews.length === 0) {
            containerEl.innerHTML = '<p style="color: rgba(255,255,255,0.5); text-align: center; padding: 20px 0;">No reviews yet. Be the first to leave one!</p>';
            return;
        }

        containerEl.innerHTML = reviews.map(r => `
            <div class="review-item">
                <div class="review-header">
                    <div class="review-avatar">${escapeHtml(r.username.charAt(0).toUpperCase())}</div>
                    <span class="review-author">${escapeHtml(r.username)}</span>
                    <span class="review-date">${new Date(r.date).toLocaleDateString()}</span>
                </div>
                <p class="review-body">${escapeHtml(r.review)}</p>
            </div>
        `).reverse().join('');
    }
    async function openReadModal(mangaId) {
      try {
        const res = await fetch(`${API_URL}/mangas/${mangaId}`, {
            headers: { 'Authorization': `Bearer ${token}` }
        });
        if (!res.ok) throw new Error('Could not fetch manga details.');
```

```
421         const manga = await res.json();
422
423         fetch(`${API_URL}/history`, {
424             method: 'POST',
425             headers: { 'Content-Type': 'application/json', 'Authorization': `Bearer ${token}` },
426             body: JSON.stringify({ mangaTitle: manga.title, chaptersRead: manga.chapters })
427         });
428
429         const backdrop = document.createElement('div');
430         backdrop.className = 'modal-backdrop';
431         const modal = document.createElement('div');
432         modal.className = 'modal';
433
434         modal.innerHTML = `
435           <div style="display:flex;justify-content:space-between;align-items:center;margin-bottom:20px;">
436             <h2 style="margin:0">${escapeHtml(manga.title)}</h2>
437             <div><button id="closeModal" style="background:#ef4444;border:none;padding:10px 15px;color:white;border-radius:12px;cursor:pointer;font-weight:600;">Close<
438           </div>
439           <div class="manga-header">
440             <img class="manga-cover" src="${manga.cover || ''}" alt="${escapeHtml(manga.title)}" onerror="this.src='https.placehold.co/140x200/0a0e27/ffffff?text=No+Im
441             <div style="flex:1;">
442               <p><strong>Author:</strong> ${escapeHtml(manga.author)}</p>
443               <p><strong>Genre:</strong> ${escapeHtml(manga.genre)}</p>
444               <p style="margin:8px 0;"><strong>${manga.chapters} Chapters</strong></p>
445               <a href="${escapeHtml(manga.link)}" target="_blank" rel="noopener noreferrer" style="display:inline-block; margin-top:15px; padding:12px 25px; background
446                 Read Now ↗
447               </a>
448             </div>
449           </div>
450           <hr style="border:none;border-top:1px solid rgba(255,255,255,0.1);margin:25px 0;" />
```

```
451
452           <div id="reviewFormWrapper" style="margin-bottom: 30px;">
453             <h3 style="margin:0 0 15px">Leave a Review</h3>
454             <textarea id="reviewComment" rows="4" style="width:100%;padding:15px;border-radius:15px;background:rgba(255,255,255,0.03);border:1px solid rgba(255,255,255
455             <div style="display:flex;gap:10px;margin-top:15px;">
456               <button id="submitReviewBtn" style="padding:12px 20px;border-radius:12px;border:none;background:linear-gradient(135deg,#10b981,#06b6d4);color:white;cursc
457             </div>
458           </div>
459
460           <div id="reviewsContainer">
461               <h3 style="margin:0 0 15px;">Community Reviews</h3>
462               <div id="reviewsList"></div>
463           </div>
464         `;
465
466         backdrop.appendChild(modal);
467         document.body.appendChild(backdrop);
468
469         const reviewsListEl = modal.querySelector('#reviewsList');
470         renderReviews(manga.reviews, reviewsListEl);
471
472         modal.querySelector('#closeModal').addEventListener('click', () => backdrop.remove());
473
474         modal.querySelector('#submitReviewBtn').addEventListener('click', async () => {
475           const reviewText = modal.querySelector('#reviewComment').value.trim();
476           if (!reviewText) return alert('Please write a review before submitting.');
477
478           try {
479             const postRes = await fetch(`${API_URL}/mangas/${mangaId}/reviews`, {
480               method: 'POST',
```

```
            headers: { 'Content-Type': 'application/json', 'Authorization': `Bearer ${token}` },
            body: JSON.stringify({ review: reviewText })
          });
          const result = await postRes.json();
          if (!postRes.ok) return alert(result.message || 'Failed to submit review');

          const updatedMangaRes = await fetch(`${API_URL}/mangas/${mangaId}`, { headers: { 'Authorization': `Bearer ${token}` } });
          const updatedManga = await updatedMangaRes.json();

          const mangaIndex = allMangas.findIndex(m => m._id === mangaId);
          if (mangaIndex !== -1) {
              allMangas[mangaIndex] = updatedManga;
          }

          renderReviews(updatedManga.reviews, reviewsListEl);
          modal.querySelector('#reviewComment').value = '';
          alert('Review submitted successfully!');

        } catch (err) {
          alert('Failed to submit review due to a connection error.');
        }
      });
    } catch(err) {
        alert('Error: Could not open manga details.');
        console.error(err);
    }
  }
  document.getElementById('historyBtn').addEventListener('click', async () => {
    const existing = document.getElementById('historyPanel');
    if (existing) { existing.remove(); return; }
```

```
    const panel = document.createElement('div');
    panel.id = 'historyPanel';
    panel.style.cssText = 'position:fixed; top:80px; right:20px; width:340px; max-height:80vh; overflow-y:auto; background:rgba(0,0,0,0.85); backdrop-filter:blur(10
    panel.innerHTML = `<h3 style="margin-bottom:10px;text-align:center;">📜 Reading History</h3><div id="historyContent" style="min-height:80px;">Loading…</div><di
    document.body.appendChild(panel);
    document.getElementById('closeHistory').addEventListener('click', () => panel.remove());
    try {
      const res = await fetch(`${API_URL}/history`, { headers: { 'Authorization': `Bearer ${token}` } });
      if (!res.ok) throw new Error('Failed to fetch history');
      const history = await res.json();
      const content = panel.querySelector('#historyContent');
      content.innerHTML = history.length ? history.map(h => `
        <div style="margin-bottom:12px;padding:10px;background:rgba(255,255,255,0.03);border-radius:10px;">
          <strong>${escapeHtml(h.mangaTitle)}</strong><br>
          <small style="color:rgba(255,255,255,0.7);">Chapters Read: ${h.chaptersRead} • Read On: ${new Date(h.date).toLocaleDateString()}</small>
        </div>
      `).reverse().join('') : '<p style="color:#aaa;">No history yet. Read a manga to start!</p>';
    } catch (err) {
      const content = panel.querySelector('#historyContent');
      if (content) content.innerHTML = '<p style="color:#f87171;">Failed to load history.</p>';
    }
  });
  document.getElementById('sortSelect').addEventListener('change', loadMangas);
  document.getElementById('searchInput').addEventListener('input', () => { clearTimeout(window.searchTimer); window.searchTimer = setTimeout(loadMangas, 500); });
  function showError(elementId, message) {
    const el = document.getElementById(elementId);
    if (!el) return alert(message);
    el.textContent = message; el.style.display = 'block';
    setTimeout(() => { el.style.display = 'none'; }, 4000);
  }
```

```
    function escapeHtml(s) {
      if (!s) return '';
      return String(s).replace(/&/g,'&amp;').replace(/</g,'&lt;').replace(/>/g,'&gt;').replace(/"/g,'&quot;').replace(/'/g,'&#39;');
    }

    </script>
    </body>
    </html>
```

Source Code:
(Backend)

```
// 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));

// --- DATABASE SCHEMAS & MODELS ---
const userSchema = new mongoose.Schema({
  username: { type: String, required: true },
  email: { type: String, required: true, unique: true },
  password: { type: String, required: true },
  history: [{
    mangaTitle: String,
    chaptersRead: Number,
    date: { type: Date, default: Date.now }
  }]
});

const mangaSchema = new mongoose.Schema({
  title: String,
  author: String,
  genre: String,
  cover: String,
  chapters: Number,
  rating: Number,
  // **NEW:** Added a field for an external link
  link: { type: String, default: '#' },
  reviews: [{
    username: String,
    review: String,
    date: { type: Date, default: Date.now }
  }]
});
```

```
56
57      const User = mongoose.model('User', userSchema);
58      const Manga = mongoose.model('Manga', mangaSchema);
59
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
77      // --- API ROUTES ---
78      // ... (register, login routes remain the same) ...
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
106
107     // MODIFIED: Seed data now includes a 'link' for each manga
108     app.post('/api/mangas/seed', async (req, res) => {
109         try {
110             await Manga.deleteMany({});
111             const sampleMangas = [
112                 { title: "Attack on Titan", author: "Hajime Isayama", chapters: 139, rating: 4.8, genre: "Action, Fantasy", cover: "https://placehold.co/240x340/a32a28
113                 { title: "Bleach", author: "Tite Kubo", chapters: 686, rating: 4.5, genre: "Shonen", cover: "https://placehold.co/240x340/1a1a1a/ffffff?text=Bleach", l
```

```
            { title: "One Piece", author: "Eiichiro Oda", chapters: 1100, rating: 4.9, genre: "Adventure", cover: "https://placehold.co/240x340/0077c2/ffffff?text=
            { title: "Naruto", author: "Masashi Kishimoto", chapters: 700, rating: 4.7, genre: "Action, Ninja", cover: "https://placehold.co/240x340/ff7f00/ffffff?
            { title: "Jujutsu Kaisen", author: "Gege Akutami", chapters: 230, rating: 4.9, genre: "Supernatural", cover: "https://placehold.co/240x340/7b2cbf/fffff
        ];
        const result = await Manga.insertMany(sampleMangas);
        res.status(201).json({ message: `${result.length} manga titles were successfully added.` });
    } catch (err) {
        res.status(500).json({ message: 'Error seeding manga data.' });
    }
});

// ... (GET /api/mangas, GET /api/mangas/:id, reviews, and history routes remain the same) ...
app.get('/api/mangas', verifyToken, async (req, res) => {
  try {
    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);
  } catch (err) {
    res.status(500).json({ message: 'Error loading mangas.' });
  }
});
```

```
app.get('/api/mangas/:id', verifyToken, async (req, res) => {
    try {
        const manga = await Manga.findById(req.params.id);
        if (!manga) return res.status(404).json({ message: 'Manga not found.' });
        res.json(manga);
    } catch (err) {
        res.status(500).json({ message: 'Failed to load manga details.' });
    }
});

app.post('/api/mangas/:id/reviews', verifyToken, async (req, res) => {
  try {
    const { review } = req.body;
    const manga = await Manga.findById(req.params.id);
    if (!manga) return res.status(404).json({ message: 'Manga not found.' });
    manga.reviews.push({ username: req.user.username, review });
    await manga.save();
    res.status(201).json({ message: 'Review added successfully.' });
  } catch (err) {
    res.status(500).json({ message: 'Failed to add review.' });
  }
});

app.post('/api/history', verifyToken, async (req, res) => {
  try {
    const { mangaTitle, chaptersRead } = req.body;
    const user = await User.findById(req.user.id);
    if (!user) return res.status(404).json({ message: 'User not found.' });
    user.history.push({ mangaTitle, chaptersRead, date: new Date() });
```

```
166         await user.save();
167         res.status(201).json(user.history);
168       } catch (err) {
169         res.status(500).json({ message: 'Failed to update history.' });
170       }
171     });
172
173     app.get('/api/history', verifyToken, async (req, res) => {
174       try {
175         const user = await User.findById(req.user.id);
176         if (!user) return res.status(404).json({ message: 'User not found.' });
177         res.json(user.history);
178       } catch (err) {
179         res.status(500).json({ message: 'Failed to load history.' });
180       }
181     });
182
183
184     // --- SERVE FRONTEND ---
185     app.use(express.static(path.join(__dirname, 'public')));
186     app.get('*', (req, res) => {
187       res.sendFile(path.join(__dirname, 'public', 'Manga_FrontEnd_Revised.html'));
188     });
189
190     // --- START SERVER ---
191     app.listen(PORT, () => console.log(`🚀 Server is running at http://localhost:${PORT}`));
192
```

VIII. Conclusion and Recommendations

A. 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, 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.

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.

In addition, users can spend more time engaging and capturing information by including user reviews and reading history features, and thereby further enhance the experience. These features mimic the personalized experience of a real-world manga platform.

The project shows how a small team can design and implement a fully functioning full-stack web application that connects design, functionality, and performance. This document also highlights the importance of API integration, database management, and synchronous communication between client and server in modern web applications.

B. Recommendations:

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.

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.

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.

IX. 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