

Marvel Nexus

Web Developer: Alex McOmber

Table of contents

01.

Design

Covering the design and Idea behind this project.

02.

Frontend

Covering frontend development and features given and how they match diagrams that were original constructed

03.

Backend

Covering backend development and how it executes needed commands for given diagrams.

04.

Testing and Deployment

Covering the deployment of the application and the unit tests that were completed.



OI.

Design

Pitch

The image shows the classic Marvel logo, which consists of the word "MARVEL" in a bold, white, sans-serif font. The letters are contained within a red rectangular border that has a slight 3D effect, with the top and bottom bars being slightly thicker than the sides. The logo is centered within a white square, which is itself set against a light gray background.

Marvel Nexus is a web application for Marvel comic enthusiasts to explore comics and manage their comic collections. It offers detailed comic profiles, a personalized comic library and a user-friendly interface for browsing and tracking favorites. All you need in one application for any comic fan.

Requirements

Functional Requirements:

1. Search comics by title, series, or character.
2. Show a list of Users Comics collections
3. Show a list of Users Comics collections
4. Allow a user to add and delete comics to or from a collection
5. User authentication using OAuth.
6. Show and allow user profile to be edited
7. Save all user information and have session management

Non-Functional Requirements:

1. Performance: Fast response times for search queries (<2 seconds).
2. Scalability: Handle a growing database of comics.
3. Usability: Intuitive UI with responsive design.
4. Security: Secure user authentication and data storage.

Use Cases

1. Set up the database
2. Set up web server.
3. Initialize database
4. Create User
5. Log In
6. Save Comics
7. Explore Comics
8. Create Collection
9. Add to Collection
10. View Collections
11. Remove Comics from Collections
12. Remove Collections
13. Update User Account.
14. Log Out

Architectural Stack



Frontend: React Vite

Backend: Node.js, Express.js

Database: MongoDB for user, comic, and collection data.

API Integration: Marvel API (free to use with an account)

Authentication: OAuth 2.0 with google for user authentication.

Hosting: AWS frontend and backend.

Testing: Jest for backend Vitest for frontend



O2.

Frontend

Login

Please log in using your Google account:

Log in with Google

Profile



Name:

Alex Mcomber

Username:

alex.mcomber11

Email:

alex.mcomber11@gmail.com

Edit Profile

Profile



Name:

Alex

Username:

AlexM

Save Changes

Explore Comics

Spider-man

Search by Title

Search by Character

Search by Series

Showing 41–60



Explore Comics

Spider-man



Marvel Tales

Creators: Unknown

Description: (RE)INTRODUCING...SPIDER-BOY!The battle to save the Spider-Verse may be over, but spinning out of the restored Web of Life and Destiny returns the spectacular SPIDER-BOY, Peter Parker's stupendous sidekick! Wait, that can't be right - who IS this Spider-Boy, and what is his connection to the Amazing Spider-Man?!

Series: Spider-Man



Add to Collection:

- ☐ Hawkeye
- ☐ Black widow
- ☐ X-men
- ☐ Avengers

Confirm



Marvel Tales



Spider-Man #4



Beutejagd



Spider-Man #5

Collections

Enter title for new collection

Add Collection

Hawkeye

1 comic

Delete Collection

Black widow

2 comics

Delete Collection

X-men

0 comics

Delete Collection

Avengers


0 comics

Delete Collection

Back to Collections

Hawkeye

Delete Collection



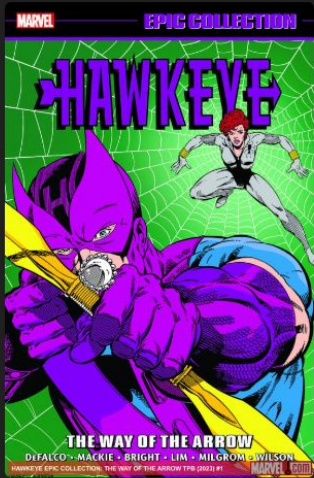
Hawkeye Epic Collection: The Way Of The Arrow (Trade Paperback)

Remove from Collection

Back to Collections

Hawkeye

Delete Collection



Hawkeye Epic Collection: The Way Of The Arrow (Trade Paperback)

Creators: Jeph York, Jeff Youngquist

Description: Collects Solo Avengers (1987) #1-20, Avengers Spotlight (1989) #21. The true origin of Hawkeye! Everybody knows that young Clint Barton joined a traveling carnival, where the Swordsman took him under his wing. But why did the legendary blade-wielder teach Clint archery? That's where Trick Shot comes in - the one man Hawkeye hoped he would never see again! But now he's back, and he's issued a death challenge to his one-time protégé! Meanwhile, Hawkeye gets hired by Silver Sable and goes up against some of the Marvel Universe's deadliest villains: the Red Skull, Doctor Octopus, the Abomination and...the Orb?! But Clint's marital difficulties with Mockingbird aren't helped when his old girlfriend the Black Widow calls for his aid! Plus: The solo spotlight turns to Clint's fellow Avengers - including Moon Knight, the Scarlet Witch, She-Hulk, the Black Panther and Namor!

Series: Hawkeye Epic Collection: The Way Of The Arrow (2023)

Remove from Collection

Way Of The Arrow (Trade Paperback)



03.

Backend

Server Overview

```
// Middleware setup
app.use(cors({
  origin: `${process.env.FRONT_URL}`, // Frontend URL
  credentials: true, // Allow credentials (cookies) to be sent
  allowedHeaders: ['Content-Type', 'Authorization', 'X-Requested-With'],
}));

app.use(express.json({
  strict: true,
  verify: (req, res, buf) => {
    const contentType = req.headers['content-type'] || '';
    if (!contentType.includes('application/json')) return;

    try {
      JSON.parse(buf);
    } catch (err) {
      const preview = buf.toString().slice(0, 100);
      console.warn(`[JSON ERROR] From ${req.ip}: ${preview}`);
      throw new Error('Invalid JSON');
    }
  }
}));
```

- Made using javascript and node with help from express and CORS.
- The server's job is to communicate between the frontend and the API and database.
- The frontend sends a request to the backend for user data or stored collections data and the server requests that from MongoDB Atlas and returns it.
- The frontend sends a request for comic data and the backend sends a request for data from the API and returns it.

Server File Setup

```
// File upload setup
const multer = require('multer'); // For form data file uploads/update
const path = require('path');
const storage = multer.diskStorage({
  destination: function(req, file, cb){
    cb(null, 'uploads/');
  },
  filename: function(req, file, cb){
    cb(null, Date.now() + path.extname(file.originalname));
  }
});
const upload = multer({storage})
app.use('/uploads', express.static(path.join(__dirname, 'uploads')));

app.set("trust proxy", true); // or 1 for single-hop if needed

const isProduction = process.env.NODE_ENV === 'production';
const isTest = process.env.NODE_ENV === 'test';
```

Server Session Set up

```
// Session setup with MongoDB Atlas store
app.use(
  session({
    secret: process.env.SESSION_SECRET,
    resave: false,
    saveUninitialized: false,
    store: isTest
      ? new session.MemoryStore()
      : MongoStore.create({
          mongoUrl: process.env.MONGO_URL,
          collectionName: 'sessions',
          ttl: 14 * 24 * 60 * 60, // 14 days
        }),
    proxy: isProduction, // trust proxy only in production
    cookie: {
      maxAge: 1000 * 60 * 60 * 24, // 1 day
      httpOnly: true,
      secure: isProduction, // secure cookies only in production
      sameSite: isProduction ? 'none' : 'lax',
    },
  })
);
```

```
passport.serializeUser((user, done) => {
  done(null, user.googleId);
});

passport.deserializeUser(async (googleId, done) => {
  try {
    const user = await User.findOne({ googleId });
    if (!user) {
      return done(null, false);
    }
    done(null, user);
  } catch (err) {
    done(err, null);
  }
});

module.exports = passport;
```

Server OAuth Setup

```
// Passport initialization
app.use(passport.initialize());
app.use(passport.session());

// Connect to the database before starting the server
connectDatabase()
  .then(() => {
    //test route to make sure the backend is live on AWS
    app.get('/', (req, res) => {
      res.send('Backend is live!');
    });

    // Google OAuth routes
    app.get('/auth/google', passport.authenticate('google', { scope: ['profile', 'email'] }));
```

Passport

```
passport.use(  
  new GoogleStrategy(  
    {  
      clientID: process.env.GOOGLE_CLIENT_ID,  
      clientSecret: process.env.GOOGLE_CLIENT_SECRET,  
      callbackURL: `${process.env.BACKEND_URL}/auth/google/callback`,  
    },  
    async (accessToken, refreshToken, profile, done) => {  
      try {  
        // Check if the user already exists in the database  
        const existingUser = await User.findOne({ googleId: profile.id });  
        if (existingUser) {  
          return done(null, existingUser); // If the user exists, pass it to the session  
        }  
  
        // If the user doesn't exist, create a new user with Google profile information  
        const username = await generateUniqueUsername(profile.emails[0].value.split('@')[0]);  
        const newUser = new User({  
          googleId: profile.id,  
          username: username,  
          name: profile.displayName,  
          email: profile.emails[0].value,  
          profilePic: profile.photos[0].value,  
          collections: [],  
        });  
  
        await newUser.save();  
      } catch (error) {  
        return done(error, null);  
      }  
    })  
  )  
);
```

Database

```
const mongoose = require('mongoose');

async function connectDatabase() {
  if (process.env.NODE_ENV === 'test') {
    console.log("Skipping MongoDB Atlas connection in test mode");
    return;
  }

  try {
    await mongoose.connect(process.env.MONGO_URL);
    console.log(`Connected to MongoDB Atlas: ${mongoose.connection.name}`);
  } catch (err) {
    console.error("MongoDB connection error:", err);

    if (process.env.NODE_ENV !== 'development') {
      process.exit(1);
    }
  }
}

async function disconnectDatabase() {
  if (process.env.NODE_ENV === 'test') {
    await mongoose.connection.dropDatabase(); // Cleanup test data
  }

  await mongoose.disconnect();
  console.log('MongoDB disconnected');
}

module.exports = { connectDatabase, disconnectDatabase };
```



04.

Deployment & Testing

S3 & Elastic Beanstalk

S3

Purpose: Hosts the React Vite frontend as a static website.

Setup: Uploaded the built frontend (dist/) to an S3 bucket.

Configured the bucket for static website hosting.

Set bucket policy to allow public read access for website assets.

Key Advantage: Fast and scalable static content delivery.

Elastic Beanstalk

Purpose: Hosts the Node.js/Express backend for API routing, database interaction, and user authentication.

Setup: Deployed using Elastic Beanstalk environment with Node.js platform.

Backend connects to MongoDB Atlas and handles secure sessions with Google OAuth2. Backend makes calls to the API.

Configured environment variables for API keys and DB URLs.

Cloudfront & Route 53

Cloudfront (CDN)

Purpose: Accelerates both frontend and backend content delivery globally.

Enforces HTTPS and handles caching for improved performance.

Benefit: Reduced latency and enhanced security.

Route 53

Purpose: Routes traffic to S3 and Elastic Beanstalk via custom domains.

Setup: marvel-nexus.com for the frontend (S3 via CloudFront).
marvel-nexus-backend.click for the backend (Elastic Beanstalk via CloudFront).

Configured A and CNAME records to point to CloudFront distributions.

Enabled SSL/TLS certificates via AWS Certificate Manager (ACM).

Dev Flow

Frontend Build

Frontend build → S3 → CloudFront → Route 53 → User.

Backend Build

Backend (Express app) → Elastic Beanstalk → CloudFront → Route 53 → User.

Frontend Tests: Vitest

```
✓ src/tests/comicDetail.test.jsx (3)
✓ src/tests/login.test.jsx (2)
✓ src/tests/comicCard.test.jsx (4)
✓ src/tests/collectionCard.test.jsx (4)
✓ src/tests/collection_gallery.test.jsx (4)
✓ src/tests/explore.test.jsx (7)
✓ src/tests/collection.test.jsx (4)
✓ src/tests/profile.test.jsx (4)
✓ src/tests/navbar.test.jsx (2)

Test Files  9 passed (9)
  Tests    34 passed (34)
Start at    19:54:36
Duration    2.33s (transform 432ms, setup 0ms,

PASS  Waiting for file changes...
       press h to show help, press q to quit
```

Passport (Mock)

```
if (isTest) {  
  passport.use(  
    new MockStrategy(async (accessToken, refreshToken, profile, done) => {  
      try {  
        const mockProfile = {  
          id: "test123",  
          displayName: "Test User",  
          emails: [{ value: "testuser@example.com" }],  
          photos: [{ value: "https://example.com/test-profile-pic.png" }]  
        };  
  
        return done(null, mockProfile);  
      } catch (err) {  
        console.error('Error in Mock Strategy:', err.message);  
        return done(err, null);  
      }  
    })  
  );  
}
```

Google Mock

```
const passport = require('passport');
const util = require('util');

function MockStrategy(options, verify) {
  this.name = 'google';
  this.verify = verify;
}

MockStrategy.prototype.authenticate = function (req) {
  const fakeProfile = {
    id: '1234567890',
    displayName: 'Test User',
    emails: [{ value: 'testuser@example.com' }],
    photos: [{ value: 'https://example.com/avatar.png' }],
  };

  const self = this;
  function done(err, user) {
    if (err) {
      return self.error(err);
    }
    self.success(user);
  }

  this.verify(null, null, fakeProfile, done);
};

util.inherits(MockStrategy, passport.Strategy);
```

Backend Tests: Jest

```
PASS tests/server.test.js
  • Console

console.log
  Skipping MongoDB Atlas connection in test mode

    at connectDatabase (db.js:5:13)

console.log
  Running on 8080

    at Server.<anonymous> (server.js:368:36)

console.log
  In-memory MongoDB connected

    at Object.<anonymous> (tests/server.test.js:20:11)

console.log
  MongoDB memory server stopped

    at Object.<anonymous> (tests/server.test.js:27:11)

(node:20404) [MONGODB DRIVER] Warning: useUrlParser is a deprecated option: useUrlParser ha
(Use `node --trace-warnings ...` to show where the warning was created)
(node:20404) [MONGODB DRIVER] Warning: useUnifiedTopology is a deprecated option: useUnifiedTopol
PASS tests/model.test.js
PASS tests/userRoute.test.js

Test Suites: 3 passed, 3 total
Tests:      22 passed, 22 total
Snapshots:  0 total
Time:       2.28 s, estimated 3 s
Ran all test suites.
```



Thanks!

Do you have any questions?

Alex.McOmber11@gmail.com

+1 801-2230-2205

<https://marvel-nexus.com>