

✓ Assignment Blueprint: Fullstack Integration & Deployment

🧠 Learning Objectives Breakdown

Objective	How to Achieve It
Connect frontend & backend	Use Axios or Fetch in React to call Express APIs. Test via Postman or browser tools.
Authentication Flows	Implement JWT-based login/logout with secure password hashing (bcrypt) on the backend.
Deploy Full Stack App	Use Vercel or Netlify for React, and Render or Heroku for Express backend. Set environment variables properly.
External APIs	Use TMDB API for movie data (register for an API key).

🧪 Hands-On Project Steps

🔗 1. Connect React Frontend to Express Backend

Backend (Express.js + MongoDB via Mongoose):

- Set up basic Express app
- Use `cors`, `body-parser`, and `dotenv`
- Example route:

```
// server/routes/movies.js
router.get('/search', async (req, res) => {
  const { query } = req.query;
  const response = await axios.get(`https://api.themoviedb.org/3/search/movie`, {
```

```
    params: { api_key: process.env.TMDB_API_KEY, query }
  });
  res.json(response.data);
});
```

Frontend (React):

```
const fetchMovies = async () => {
  const response = await axios.get('http://localhost:5000/api/movies/search?query=Inception');
  setMovies(response.data.results);
};
```

✅ Test full request/response loop

🔒 2. Implement Authentication in Fullstack App

Backend:

- Use `bcrypt` for password hashing
- Use `jsonwebtoken` for signing tokens
- Auth Routes: `/register`, `/login`, `/me`

```
const token = jwt.sign({ userId: user._id }, process.env.JWT_SECRET, { expiresIn: '1h' });
```

Frontend:

- On login, save JWT to localStorage
- Use Axios interceptor to include token in headers

```
axios.interceptors.request.use(config => {
  const token = localStorage.getItem('token');
  if (token) config.headers.Authorization = `Bearer ${token}`;
  return config;
});
```



Capstone Project: Movie Recommendation App



Core Features Breakdown

1. User Authentication

- Register/Login forms
- Hash passwords with `bcrypt`
- Secure JWT token handling in headers
- Protected routes via middleware

```
const verifyToken = (req, res, next) => {  
  const token = req.headers['authorization']?.split(' ')[1];  
  const decoded = jwt.verify(token, process.env.JWT_SECRET);  
  req.user = decoded;  
  next();  
};
```

2. Movie Discovery

- Connect to TMDB API
- Use React state to manage search and filters
- Add filters:
 - rating: `vote_average.gte`
 - release date: `primary_release_year`
 - genre: `with_genres`

3. User Features

- Save favorites → MongoDB collection: `favorites`
- Watchlists → User-specific lists in DB
- Rate/review: create `reviews` collection
- Profile → Edit username, avatar, etc.

4. Tech Requirements

Stack	Tech
Frontend	React, React Router, Context/Redux, Axios
Backend	Express.js, Node.js, MongoDB/Mongoose
External API	TMDB
Styling	TailwindCSS / Bootstrap / Material UI

Deployment Guide

Frontend (Vercel/Netlify)

- Connect GitHub repo
- Set `REACT_APP_API_URL` in Environment Variables
- Build script: `npm run build`

Backend (Render/Heroku)

- Deploy Express server
- Set `.env` variables: Mongo URI, JWT_SECRET, TMDB key
- Allow CORS from frontend origin
- Keep routes organized: `auth`, `movies`, `users`

CI/CD Pipeline

- Use GitHub Actions for auto-deployment on push
- Example workflow:

name: Deploy Backend

on:

push:

branches: [main]

jobs:

deploy:

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v2
- run: npm install && npm run build




Stretch Goals (Optional)

Feature	How to Do It
Social	Add followers field in user schema
Smart Rec Engine	Use TMDB's recommendations endpoint OR train a simple content-based model
Trailers	Use TMDB video endpoint
PWA	Add manifest + service workers via create-react-app built-in support



Summary Task List

 Task	Tools
React frontend with API calls	Axios, useEffect
Express backend with routes	Node.js, Mongoose

JWT Auth	<code>jsonwebtoken</code> , <code>bcrypt</code>
MongoDB DB setup	MongoDB Atlas
Movie discovery with TMDB	TMDB API
CI/CD & Deployment	Vercel + Render
State Management	Context API or Redux
UI Styling	Tailwind or Bootstrap



Suggested Timeline

Week	Focus
1	Backend setup + Auth
2	Frontend integration + UI
3	Movie API + filtering
4	Favorites + watchlist
5	Final deployment + testing
6	Stretch goals or polish