Project Design Phase-II
Technology Stack (Architecture & Stack)

Date	13 April 2025
Team ID	SWTID1743509015
Project Name	iMovies – Online Movie Ticket Booking System
Maximum Marks	4 Marks

Technical Architecture Diagram

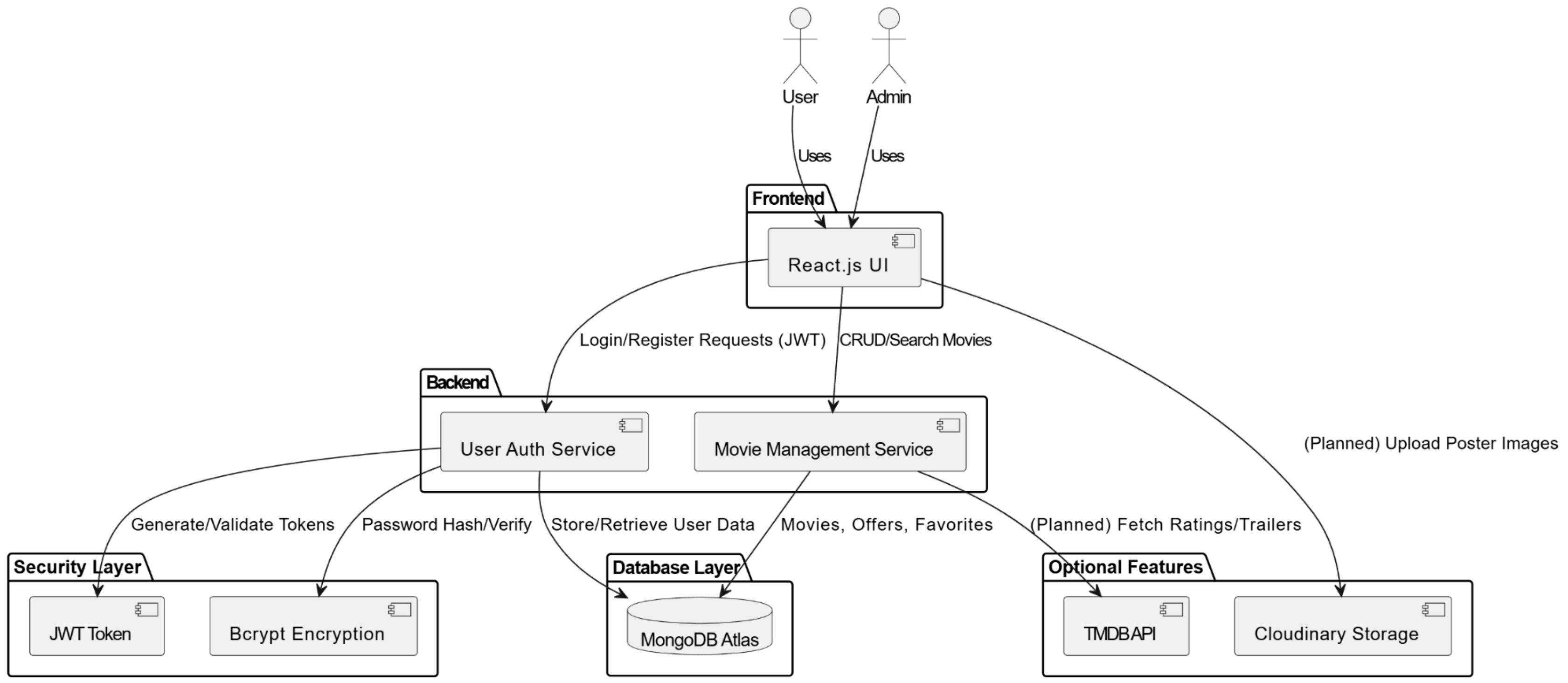

Table-1 : Components & Technologies:

S.No	Component	Description	Technology
1	User Interface	Web UI for user and admin to interact with the application	HTML, SCSS, JavaScript, React.js
2	Application Logic-1	Handles user registration, login, and JWT-based authentication	Node.js, Express.js, bcrypt, JWT
3	Application Logic-2	Handles movie CRUD operations, search, and categorization	Node.js, Express.js
4	Database	Stores user info, movie data, offers, favorites	MongoDB (NoSQL)
5	Cloud Database	MongoDB hosted on cloud to persist app data	MongoDB Atlas
6	File Storage	(Optional future) Storing poster images, user profile images	Local File System / Cloudinary (future)
7	External API-1	(Optional future) Fetching external movie ratings or trailers	TMDB API (planned integration)

Table-2: Application Characteristics:

S.No	Characteristics	Description	Technology
1	Open-Source Frameworks	The iMovies project uses open-source technologies to build a full-stack web application. ReactJS is used for building a responsive UI, NodeJS and ExpressJS handle the backend logic and APIs, while MongoDB stores and manages the data. Axios is used for handling HTTP requests between the client and server.	ReactJS (Frontend), NodeJS (Backend), ExpressJS (Backend), MongoDB (Database), Axios (HTTP requests)
2	Security Implementations	The application ensures secure access for both users and admins using JWT (JSON Web Token) for session management and Bcrypt for password encryption to prevent unauthorized access.	JWT Authentication, Bcrypt (Password Hashing)
3	Scalable Architecture	The project follows a scalable client-server architecture where the ReactJS frontend sends requests to a NodeJS/Express backend, making it easy to expand or modify features.	Client-Server Model
4	Availability	The application is deployed and functional on both frontend and backend, ensuring continuous access to users. While load balancing is not yet added, current deployment meets basic availability needs.	(Currently Implied through successful deployment)
5	Performance	API calls are optimized to reduce response times and improve overall user experience. Efficient routing and data fetching contribute to better app performance.	Optimized API Requests