**Project Design Phase**

**Solution Architecture**

| Date | 12 April 2025 |
| --- | --- |
| Team ID | SWTID1744468652 |
| Project Name | i-movies: movie ticket booking system |
| Maximum Marks | 4 Marks |

**Solution Architecture:**

This architecture illustrates how the MERN Movie App platform functions across the User, Cloud, and Admin networks:

* **Frontend (UI):** Built with React, users interact through a responsive web interface for both the movie dashboard (admin) and movie listing (user) apps. The UI is styled with Tailwind CSS for a modern and clean look.
* **Backend (API):** Developed using Node.js and Express.js, it handles communication between the frontend, MongoDB database, and external services like Cloudinary and email (Nodemailer).
* **Authentication**: Managed via JSON Web Tokens (JWT), ensuring secure, role-based access for both admins and users.
* **Data Services**: Central MongoDB database stores movie details, genres, user information, and watchlists. Mongoose is used for schema modeling.
* **Image Management**: Cloudinary is integrated for storing and serving movie posters and related images.
* **Email Service:** Nodemailer is used for sending password reset emails and OTP confirmations to users.
* **Admin Dashboard:** Allows admins to manage movies and genres, view analytics, and moderate content through a dedicated dashboard interface.
* **User Features:** Users can browse movies, filter by genre/rating, search, manage their watchlist, and handle authentication (register, login, password reset).
* **API Communication:** Axios is used in the frontend for making HTTP requests to the backend.
* **Security & CORS:** CORS is configured to allow secure communication between frontend and backend hosted on different domains. Passwords are securely hashed using Bcrypt.

Color-coded legends in the architecture diagram distinguish UI components, backend logic, data storage, image hosting, authentication, and email services for clarity.

**Solution Architecture Diagram:**

