Department: Computer



**Project**

**Report**

**Cloud**

**Base**

**Find everything**

**from the internet**

**En no : 2128020601101**

**Vaibhav Senta**

|  |  |  |
| --- | --- | --- |
| NO | Topic Name | Page No |
| 1 | Abstract |  |
|  | Introduction |  |
|  | System Design |  |
|  | Implementation Workflow |  |
|  | Challenges Faced |  |

Abstract

Cloud Base is a Node.js-based web application designed to provide a platform for users to upload, manage, and view movies. It includes functionalities like user authentication with JWT, movie management, and account management. The application uses MongoDB for storing user and movie data, Multer for handling file uploads, and JWT for managing user sessions securely. The website is user-friendly, enabling users to interact easily with the content, upload movies, and manage their profiles.

Introduction

Cloud Base is built to simplify movie management for users, allowing them to upload, view, and manage movies in a secure and organized way. The core functionalities of the website include:

* User Signup & Login: Allow users to create accounts, log in using JWT, and securely authenticate their sessions.
* Movie Upload: Users can upload movies with metadata such as title, genre, release date, etc., along with movie files and posters.
* Account Management: Users can edit their profiles and delete their accounts when needed.

This report explains the system design, implementation process, libraries used, challenges faced, and the future scope of the project.

System Design

## Architecture

The Cloud Base system follows a client-server architecture:

* Client Side: The client side is built using HTML, CSS, and EJS for rendering dynamic pages (like the homepage, movie upload forms, etc.). The front-end interacts with the server via HTTP requests.
* Server Side: The back-end is developed using Node.js with the Express framework, handling routing and managing HTTP requests.
* Database: MongoDB stores data related to users and movies, including metadata and file paths.
* Authentication: The system uses JWT (JSON Web Token) for securely managing user authentication. JWT tokens are generated during login and used for verifying the user in subsequent requests.

## Database Design

1. User Collection:
   1. Fields: email, password (hashed), username, first name, last name, phone number, status (active/deleted), createdAt.
   2. Stores all user-related information, with password encryption via \*\*bcryptjs\*\*.

Movie Collection:

  - \*\*Fields\*\*: title, description, release date, duration, genre, cast, poster path, movie file path, available audio tracks, subtitle tracks, etc.

  - Stores movie metadata along with the file paths for movie posters and movie files uploaded by users.

## Flowcharts

- \*\*User Signup Flow\*\*:

  - \*\*Step 1\*\*: User enters details (email, password, etc.) on the signup page.

  - \*\*Step 2\*\*: Server validates inputs (checks email format, password strength).

  - \*\*Step 3\*\*: If valid, the password is hashed, and user data is stored in MongoDB.

  - \*\*Step 4\*\*: Confirmation response is sent to the user.

- \*\*Movie Upload Flow\*\*:

  - \*\*Step 1\*\*: User selects a movie to upload and fills in the movie metadata (title, genre, description).

  - \*\*Step 2\*\*: Server checks if the movie already exists in the database (prevents duplicates).

  - \*\*Step 3\*\*: If the movie is new, the server stores metadata and uses \*\*Multer\*\* to upload movie files and poster images to the server.

  - \*\*Step 4\*\*: The movie data is stored in MongoDB, including file paths, and the server responds with a success message.

---