Project Documentation

INSIGHT STREAM: Navigate the News Landscape

1.Introduction

- Project Title: INSIGHT STREAM: Navigate the News Landscape
- Team ID: NM2025TMID35585
- Team Leader: AJITHA.J & rjaprnrjaprn@gmail.com
- Team Members:

-SANTHIYA.A & sandy.mails321@gmail.com

-KANIMOZHI.S & skkani1107@gmail.com

-INDHUMATHI.M & venthank902@gmail.com

-DEVADHARSHINI.S & dharshselva06@gamil.com

2. Project Overview

- PURPOSE:

An Insight stream serves as a dynamic platform for sharing expertise, fostering engagement, and empowering viewers with valuable knowledge and perspectives across diverse topics.

1. Knowledge Sharing

2. Audience Engagement

3. Expert Perspectives

4. Informed Decision-making

5. Learning Platform

- FEATURES:

-Real- time data insights

-Data visualization

-Customizable dashboards

3. Architecture

- Modular Design: Enables flexible content delivery.

- Data Pipeline: Processes insights from various sources.
- Content Management: Organizes curated insightful content.

4. Setup Instructions

- Prerequisites:

-Technical Infrastructure

-Software Tools

-Team Expertise

-Budget Allocation

- Installation Steps:

#Server/Cloud Configuration

#Install Required Software

#Database Setup

5. Folder Structure

insight- stream/

```
├— app/
|  ├— controllers/
|  ├— models/
|  ├— services/
|  ├— utils/
|  └— views/
├— assets/
|  ├— css/
|  ├— js/
|  ├— images/
|  └— fonts/
```

6. Running the Application

- Frontend:

  - User Interface (UI) for users to interact with Insight Stream.

  - Display insights, content curated by the platform.

  - Handle user interactions (like comments, shares).

- Backend:

  - Serve data/insights to the frontend.

  - Manage content curation, storage.

  - Handle analytics, user data processing.

- Access:

  #Start Node.js server node server.js

  #Server listens on port (e.g.,3000)

  #http://localhost:3000

7. API Documentation

- User:

1. GET /api/users/{id}: Fetch user details by ID.

   - Response: User profile data (name, email...).

2. POST /api/users: Create a new user.

   - Request Body: { "name": "string", "email": "string" }.

   - Response: Created user details.

3. PUT /api/users/{id}: Update user details.

   - Request Body: { "name": "newName" }.

4. DELETE /api/users/{id}: Delete a user.

- Projects:

1. GET /api/projects: List projects.

   - Response: Array of projects with IDs, names.

2. GET /api/projects/{id}: Fetch project details by ID.

- Response: Project data (name, description...).

3. POST /api/projects: Create a project.

- Request Body: { "name": "string", "description": "string" }.

4. PUT /api/projects/{id}: Update project details.

- Chats:

1. GET /api/chats/{projectId}: Get chats for a project.

- Response: Array of chat messages.

2. POST /api/chats: Send a chat message.

- Request Body: { "projectId": "id", "userId": "id", "message": "text" }.

3. GET /api/chats/{id}/messages: Fetch messages in a chat thread.

8. Authentication

Example (Node.js/Express)

```
// Login route
app.post('/login', (req, res) => {
 const { username, password } = req.body;
 // Verify credentials
 if (validCredentials) {
  const token = generateToken(user);
  res.json({ token });
 } else {
  res.status(401).send('Unauthorized');
 }
});

// Protected route
```

```
app.get('/api/user', authenticateToken, (req, res) => {


 res.json(req.user);

});
```

9.User Interface

1. *Dashboard*: Overview of insights.

2. *Content Feed*: Shows curated content.

3. *Profile*: User settings.

4. *Search*: Find insights/projects.

5. *Interactions*: Comments, likes

10. Testing

Manual testing involves human testers interacting with the application to identify issues.

Tools:Jest,Cypress,Mocha