



# Project Report:

## **Timelium** – Skill Timeline Builder

<https://timelium.vercel.app>

Team Member:  
Antor Hawlader  
ID: 222071024  
Dept: CSE  
9<sup>th</sup> Semester  
Date: 21-01-2026

Instructor:  
Md. Abdullah Ibn Noor,  
Lecturer,  
Department of CSE &  
CSIT  
SMUCT

## Abstract:

Timelium is a dynamic, full-stack web application designed to serve as a comprehensive Skill Timeline Builder and professional portfolio platform. Unlike traditional portfolio sites that are often niche-specific (e.g., GitHub for developers, Behance for designers), Timelium provides a universal platform where professionals from any background can showcase their skills, projects, experience, and education through a unique, shareable URL (<https://www.timelium.com/username>).

The system integrates modern social networking features, allowing users to follow others, like posts, and interact with a live feed. A distinguishing feature of Timelium is the integration of Timelium AI, a generative AI assistant that interacts with the user's live feed context to answer queries. Built using the MERN stack (MongoDB, Express/NestJS, React, Node.js), the project solves the problem of static resumes by creating an interactive, living history of a user's professional journey.

## CHAPTER 1: Introduction:


### 1.1 Background of the Project

In the current digital era, a static resume or CV is often insufficient to capture the full scope of a professional's capabilities. While platforms like LinkedIn exist, they are often viewed as rigid and overly corporate. Professionals need a space that acts as a personal brand headquarters—a "Timeline" that visualizes their growth, skills, and creative output in real-time. Timelium was developed to bridge this gap, offering a personalized hub that combines the utility of a CV with the engagement of social media.

### 1.2 Problem Statement

- **Fragmentation:** Professionals currently scatter their work across Drive links, social media, and static PDFs.
- **Lack of Interactivity:** Traditional portfolios do not allow visitors to interact (like/follow) or ask questions via AI.
- **Generic Templates:** Most portfolio builders do not cater to all professions equally.
- **Communication Barriers:** Finding direct "Hire Me" contact details (WhatsApp, Telegram) is often difficult on standard platforms.

### 1.3 Objectives of the Project

- **Primary Objective:** To develop a web-based platform where users can create a professional portfolio with a unique URL.
  - **Specific Objectives:**
    - To implement a secure authentication system (Login/Register).
    - To create a "Timeline" interface for Education, Experience, and Projects.
    - To integrate **Timelium AI** (Google Gemini) to assist users and visitors by answering questions based on the live feed context .
    - To enable direct "Hire Me" communication channels (WhatsApp, Messenger, Email).
- 

## 1.4 Scope of the Project

### The project covers:

- **User Module:** Profile creation, avatar upload, and skill management.
- **Social Module:** Posting updates with images, liking posts, and following users.
- **AI Module:** An integrated chatbot for user interaction.
- **Search:** Finding users by name, username, or skills.
- **Limitations:** Currently relies on internet connectivity; offline mode is not yet implemented.

## 1.5 Target Users

- Job Seekers and Freelancers.
- Students needing a digital portfolio.
- Recruiters looking for talent.
- Professionals wanting to document their career growth.

# CHAPTER 2: Literature Review / Related Work:

## 2.1 Existing Systems

- **LinkedIn:** The standard for professional networking.
- **Linktree:** Used for consolidating links but lacks depth (projects/timeline).
- **Personal Websites:** Highly customizable but require coding knowledge or expensive hosting.

## 2.2 Limitations of Existing Systems

- **LinkedIn:** Rigid structure; difficult to showcase creative flair; AI features are limited to premium recruiters.
- **Personal Websites:** Static; no "feed" or social interaction; no built-in AI assistant to guide visitors.

## 2.3 Proposed System Overview

Timelium improves on these by offering the structure of a resume (Education/Experience sections) combined with the engagement of a social feed. Uniquely, it includes a "Hire Me" dialog that consolidates direct messaging apps (WhatsApp, Telegram), simplifying the recruitment process.

# CHAPTER 3: Requirement Analysis:

## 3.1 Functional Requirements

- **Timeline Management:** CRUD (Create, Read, Update, Delete) operations for Skills, Projects, Experience, and Education .
- **Social Feed:** Users can create posts with images and text, and "Like" posts.
- **Timelium AI:** A chat interface that answers questions using the context of the posts currently visible on the feed.

## 3.2 Non-Functional Requirements

- **Performance:** The frontend is built with Vite for rapid rendering.
- **Security:** Passwords are hashed using bcrypt. API endpoints are protected using JwtAuthGuard.
- **Usability:** The UI utilizes modern gradients and a dark-mode aesthetic for visual appeal.

## 3.3 Hardware Requirements

- **Server Side:** Minimum 1GB RAM, 1 vCPU (for Node.js runtime).
- **Client Side:** Any modern device with a web browser.

3.4 Software Requirements

- **OS:** Windows / Linux / macOS.
- **Runtime:** Node.js (v18+).
- **Database:** MongoDB Atlas (Cloud).
- **AI API:** Google Gemini API.
- **Image Hosting:** Cloudinary.

CHAPTER 4: System Design

4.1 System Architecture

Timelium follows a Client-Server Architecture.

- **Frontend:** React (SPA) communicating via Axios.
- **Backend:** NestJS (Node.js framework) exposing RESTful APIs.
- **Database:** MongoDB storing JSON-like documents.

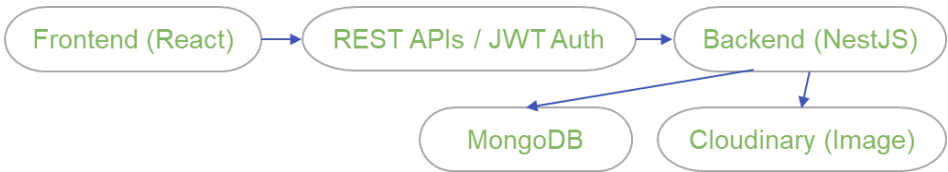
4.2 Use Case Diagram

- **Actor:** User/Visitor.
- **Use Cases:** Register, Login, View Profile (/:username), Add Project, Chat with AI, Search Users, Toggle Like.

4.3 Data Flow Description

1. When a user posts an image:
2. Client sends FormData (Text + File) to the server.
3. NestJS controller intercepts the file.
4. Service uploads the image to **Cloudinary** and gets a secure URL.
5. The URL and text are saved to **MongoDB**.

4.4 Data Flow Diagram



4.5 Database Design (Schema)

User Schema (user.schema.ts):

- fullname, username, email, password
- projects, experience, education: Arrays of objects

CHAPTER 5: Technology Stack

5.1 Frontend Technologies

- **Framework:** React.js (Vite).
- **Styling:** Tailwind CSS
- **UI Libraries:** Radix UI (Dialogs), Lucide React (Icons).

5.2 Backend Technologies

- **Framework:** NestJS (A progressive Node.js framework).
- **Language:** TypeScript.
- **API:** REST API.

POST	
ObjectId	_id
string	content
string	image_url
ObjectId	user_id
array_of_ObjectId	likes
string_array	location
date	createdAt

### 5.3 Database

- **Database:** MongoDB (NoSQL).
- **ODM:** Mongoose.

### 5.4 Authentication & Security

- **JWT:** JSON Web Tokens for stateless authentication.
- **Passport.js:** For handling authentication strategies.
- **Bcrypt:** For password hashing.

### 5.5 Tools & Platforms

- **AI:** Google GenAI (Gemini 2.5 Flash model).
- **Cloud Storage:** Cloudinary (for avatars and post images).
- **IDE:** VS Code.

## CHAPTER 6: Implementation

### 6.1 Project Folder Structure

The project is divided into two main directories:

- **server/src:** Contains the NestJS backend logic (Modules, Controllers, Services).
- **client/src:** Contains the React frontend logic (Components, Pages, Assets).

### 6.2 Frontend Implementation

The frontend is split into three main vertical sections ("Splits") for the home layout:

- **Split0:** Navigation, Search, and Trending topics.
- **Split1:** The main Feed where posts are displayed.
- **Split2:** Timelium AI chat interface and User Search results.

**Profile Page:** A dynamic component (Profile.tsx) that fetches data based on the URL parameter username and conditionally renders "Edit" buttons if the viewer is the profile owner.

### 6.3 Backend Implementation


- The backend is modularized using NestJS:
- **AuthModule:** Handles Login and Registration.
- **UserModule:** Manages profile updates, skills, and search.
- **PostsModule:** Handles creating posts, deleting, and liking.
- **GeminiModule:** Processes chat prompts by injecting the last 15 posts as context.

### 6.4 AI Integration

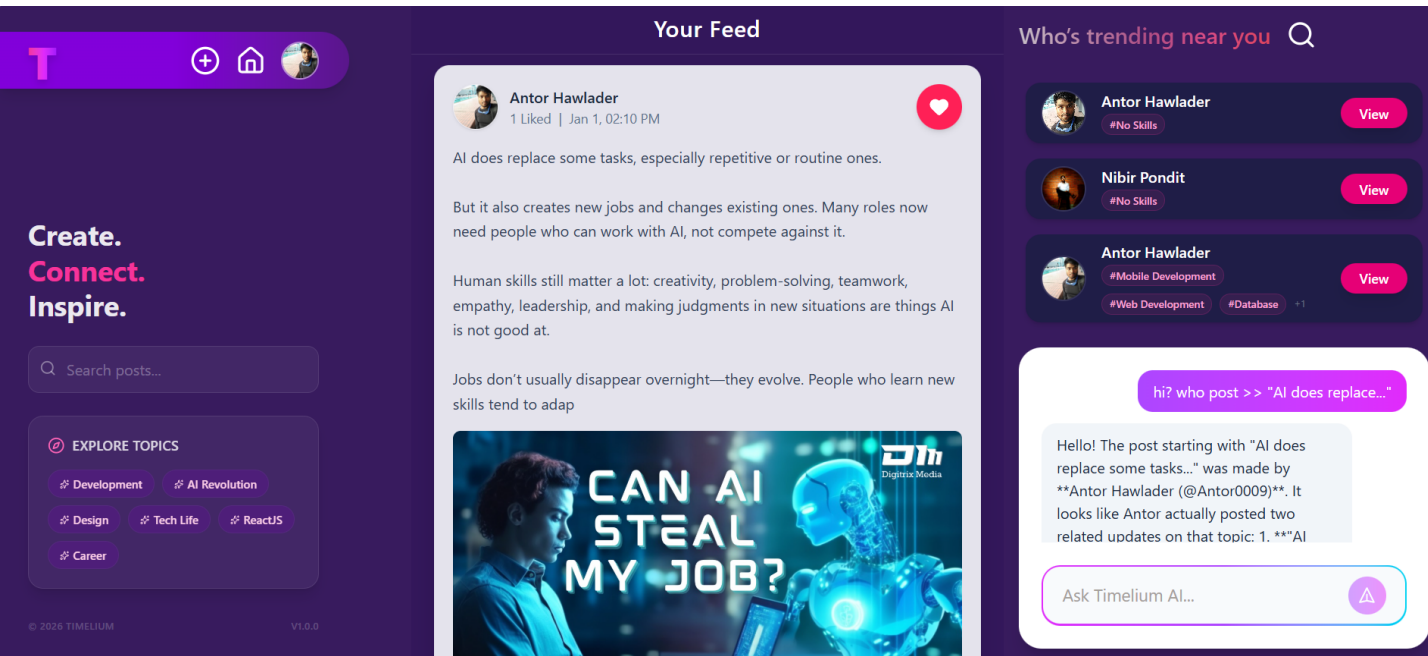
The GeminiService constructs a system instruction that includes the "Current Feed Context." This allows the AI to answer questions like "Who posted about React recently?" by analyzing the text and metadata of the latest posts .

## CHAPTER 8: Results & Screenshots

### 8.1 Performance Evaluation

- **Response Time:** The use of NestJS ensures fast API responses.
  - **Search Efficiency:** The backend implements regex-based search for users and skills, allowing for case-insensitive querying.
- 

## 8.1 Output Screens



## CHAPTER 9: Advantages & Limitations

### 9.1 Advantages

- **Context-Aware AI:** Unlike generic chatbots, Timelium AI knows what is happening on the user's feed.
- **Direct Communication:** The "Hire Me" feature reduces friction between recruiters and candidates.

### 9.2 Limitations

- **AI Rate Limits:** The Google Gemini API has usage limits which may affect the chat feature under heavy load.

## CHAPTER 10: Conclusion & Future Work

### 10.1 Conclusion

Timelium successfully implements a modern, social-media-infused portfolio builder. It empowers users to present themselves professionally while maintaining the dynamic nature of a social network. The integration of AI for feed interaction and the structured "Timeline" approach makes it a robust tool for personal branding in the digital age.

### 10.2 Future Enhancements

To further improve Timelium, the following features are planned:

- **Custom Email Gateway:** Allowing non-registered visitors to send emails directly to users through the platform without exposing the user's actual email address.
- **Messenger Integration:** Real-time chat between users within the platform (WebSockets).
- **Graph Analysis:** Visual analytics showing profile visits, post reach, and skill growth over time.
- **Mobile App:** Developing a React Native version for iOS and Android.

