 

# 9530

**St.MOTHER THERESA ENGINEERING COLLEGE**

COMPUTER SCIENCE ENGINEERING

## **NM-ID**:B61DCBAF5F9AEADA4E54997D90371029

**REG NO**: 953023104047

**DATE**:22-09-2025

# Completed the project named as

# Phase 3

FRONT END TECHNOLOGY

**CHAT APPLICATION UI**

**SUBMITTED BY,**

KIRTHIC PRAKASH S

6382471983

**Phase 3 – MVP Implementation:**

1. **Project Setup :** 
   * **Development Environment**:
     + Install Node.js and a package manager (npm/yarn).
     + Use a frontend framework (React + Tailwind CSS) for fast UI building.
     + Set up version control with GitHub.
   * **Project Initialization**:
     + Create a project (npx create-react-app news-feed-ui).
     + Initialize GitHub repository with a .gitignore and README.
     + Configure .env for API keys (if integrating external news APIs like NewsAPI).
   * **Dependencies**:
     + UI: Tailwind CSS / Material UI. o Data fetching: Axios / Fetch API. o State management: Redux / Context API.
     + Routing: React Router.
     + Testing: Jest + React Testing Library.
2. **Core Features Implementation :** 
   * **User Interface (UI)**:
     + **Navbar**: Contains logo, search bar, and navigation links.
     + **Feed Page**: Displays a list of articles in a scrollable format (cards or tiles).
     + **News Card Component**: Shows image, title, short description, and source.
     + **Category Filters**: Buttons or tabs for categories like Tech, Sports, Politics.
     + **Search Functionality**: Users can search news by keywords.
     + **Article Details Page**: Opens a full article view with complete content.
   * **User Experience (UX)**:
     + Infinite scrolling or "Load More" button.
     + Highlight latest/breaking news.
     + Responsive design for mobile and desktop.
   * **Error Handling**:
     + Show placeholder text/images when no news is available.
     + Handle API errors (e.g., "Failed to fetch news").
3. **Data Storage (Local State / Database) :** 
   * **Local State**:
     + Manage articles, selected category, and search results. o Store UI preferences (dark/light mode, recent searches).
   * **Database Integration (optional for UI MVP)**:
     + Connect to a backend or use an external API (e.g., NewsAPI, Guardian API). o Store user preferences or bookmarked articles in a database (MongoDB, Firebase).
   * **Caching**:
     + Use localStorage or sessionStorage for recent searches or offline support.
4. **Testing Core Features** 
   * **Unit Tests**:
     + Test UI components (NewsCard, SearchBar).
     + Verify category filtering logic.
   * **Integration Tests**:
     + Test fetching data from the API and rendering in the feed.
     + Ensure search input updates the news feed correctly.
   * **End-to-End (E2E) Tests**:
     + Use Cypress/Playwright to simulate a user opening the app, filtering news, and reading an article.
   * **Bug Tracking**:
     + Track issues in GitHub (e.g., “Articles not rendering”, “Search results empty”).
5. **Version Control (GitHub) :** 
   * **Branching Strategy**:
     + main: stable version of the news feed app.
     + feature/news-feed, feature/article-details: separate branches.
   * **Commits**:
     + Use meaningful messages (feat: add search functionality, fix: API error handling).
   * **Pull Requests**:
     + Peer review before merging into main.
   * **Releases**:
     + v0.1 → UI skeleton (Navbar + Feed). o v0.2 → Working search + categories.
     + v1.0 → Fully functional MVP.
   * **CI/CD (Optional)**:
     + Automate builds and testing with GitHub Actions.
     + Deploy on Vercel/Netlify for preview links.