**Title Page**

**Project Report on**

**SOCIAL MEDIA FEED ALGORITHM**

**Submitted By:**  
**SUMANT SUNDRAM**  
**UID:** 24BCA10578  
**Semester:** 2  
**Course:** BCA  
**Subject:** DATA STRUCTURES USING C   
**Project Guide:** PARWAN SIR

**Submitted To:**  
Mr. Parwan

**Acknowledgement**

I would like to express my sincere gratitude to my subject teacher, **[Insert Teacher's Name]**, for their guidance, encouragement, and valuable insights throughout the development of this report on **"Social Media Feed Algorithm in Data Structures"**.

I also thank my classmates and friends for their support and constructive feedback. Special thanks to the online resources and documentation that provided useful references for understanding complex concepts in a simplified manner.

Lastly, I acknowledge the contributions of various developers and engineers whose real-world implementations of feed algorithms have inspired this study.

**Abstract**

In the digital age, social media platforms have become central to information dissemination and user engagement. One of the key components behind the effectiveness of these platforms is the feed algorithm, which determines the order and relevance of content presented to users. This report explores the data structures and algorithmic principles that power social media feed systems. It highlights the role of arrays, heaps, graphs, hash maps, and queues in handling large-scale data, personalizing content, and ensuring performance efficiency. By analyzing how user behavior, content relevance, and platform goals influence feed ranking, this study provides insights into the intelligent design of content delivery mechanisms. The report also touches upon the challenges of scalability, fairness, and real-time processing. Overall, it offers a foundational understanding of how data structures are utilized to optimize user experience on social media platforms.

**Introduction**

Social media platforms have revolutionized the way people connect, communicate, and consume information. One of the most important features that keeps users engaged is the **content feed**—a dynamic list of posts, updates, or media shown based on a user's preferences and interactions. Rather than displaying content chronologically, modern social media platforms use sophisticated algorithms to determine what appears in a user’s feed and in what order.

These **feed algorithms** are built on fundamental **data structures** and **algorithms** that help manage vast amounts of data efficiently and deliver personalized content in real-time. Structures like arrays, heaps, graphs, and hash maps are essential for storing, sorting, retrieving, and ranking content. This report aims to explore how these data structures are applied in feed algorithms and how they contribute to making social media platforms fast, responsive, and user-friendly.

**Problem Statement**

With millions of users generating and interacting with content every second, social media platforms face the challenge of delivering personalized, relevant, and timely content to each user. The core problem lies in designing an algorithm that can efficiently sort and prioritize massive volumes of posts based on user behavior, preferences, and content relevance, while ensuring scalability and low latency.

This report aims to examine how fundamental data structures can be effectively used to address this challenge. It focuses on understanding how arrays, heaps, graphs, hash maps, and queues contribute to building feed algorithms that power personalized social media experiences.

**Flow Chart**

Source Code

import geocoder

import requests

import folium

# -----------------------------

# 1. Configuration & Constants

# -----------------------------

ORS\_API\_KEY = "YOUR\_OPENROUTESERVICE\_API\_KEY" # ← Replace with your API key

ORS\_ENDPOINT = "https://api.openrouteservice.org/v2/directions/driving-car"

# -----------------------------

# 2. Get Current Location

# -----------------------------

# Use IP-based geolocation (approximate)

geo = geocoder.ip('me')

current\_lat, current\_lng = geo.latlng

# -----------------------------

# 3. Define Destination

# -----------------------------

# Example: New Delhi (latitude, longitude)

destination = (28.6139, 77.2090)

# -----------------------------

# 4. Request Route from ORS

# -----------------------------

headers = {

"Authorization": ORS\_API\_KEY,

"Content-Type": "application/json"

}

body = {

"coordinates": [

[current\_lng, current\_lat], # Note: ORS expects [lng, lat]

[destination[1], destination[0]]

]

}

resp = requests.post(ORS\_ENDPOINT, json=body, headers=headers)

resp.raise\_for\_status()

# Extract the coordinate list for the route

route\_coords = resp.json()["features"][0]["geometry"]["coordinates"]

# Convert from [lng, lat] to (lat, lng) for plotting

route\_latlng = [(lat, lng) for lng, lat in route\_coords]

# -----------------------------

# 5. Build the Map

# -----------------------------

m = folium.Map(location=(current\_lat, current\_lng), zoom\_start=13)

# Marker: Current Location

folium.Marker(

location=(current\_lat, current\_lng),

popup="Current Location",

icon=folium.Icon(color="green", icon="user", prefix="fa")

).add\_to(m)

# Marker: Destination

folium.Marker(

location=destination,

popup="Destination",

icon=folium.Icon(color="red", icon="flag", prefix="fa")

).add\_to(m)

# Draw Route Polyline

folium.PolyLine(

locations=route\_latlng,

weight=6,

opacity=0.7

).add\_to(m)

# -----------------------------

# 6. Save & Output

# -----------------------------

output\_file = "gps\_navigation\_map.html"

m.save(output\_file)

print(f"Map with route saved to {output\_file}")

**Future Enhancement**

While the basic GPS navigation system described in this report provides essential functionality, there are several potential improvements and enhancements that could be made to increase its accuracy, usability, and feature set. Some possible future enhancements are:

**1. Real-Time Traffic Data Integration**

* **Current Limitation:** The system currently provides a static route based on geographic coordinates. It does not account for real-time traffic conditions such as congestion, roadblocks, or accidents.
* **Enhancement:** Integrating real-time traffic data (from APIs like Google Maps or OpenStreetMap) could allow the system to dynamically update routes, offering alternative paths based on current traffic conditions and helping to avoid delays.

**2. Voice Navigation**

* **Current Limitation:** The system currently provides visual guidance through a map, but there is no audio or voice-based navigation.
* **Enhancement:** Adding voice guidance would improve the user experience, particularly for drivers, allowing them to follow turn-by-turn instructions without needing to look at the screen. This could be implemented using text-to-speech (TTS) libraries like Google Text-to-Speech or integration with navigation services like Google Maps.

**3. Offline Navigation Support**

* **Current Limitation:** The system requires an active internet connection to fetch maps, calculate routes, and retrieve traffic data.
* **Enhancement:** Implementing offline navigation support would enable users to access maps and routes even when they are in areas with poor or no internet connectivity. This could involve downloading map tiles and route data ahead of time for offline use.

**4. 3D Maps and Augmented Reality (AR) Integration**

* **Current Limitation:** The map is displayed in a 2D format, and navigation is basic without advanced visual aids.
* **Enhancement:** Integrating 3D maps and augmented reality (AR) features would enhance the navigation experience. For example, an AR overlay could show directions on the live camera feed of the user’s environment, making it easier to follow routes in complex, urban areas.

**5. Personalized Recommendations**

* **Current Limitation:** The system offers a generic route based only on start and destination coordinates.
* **Enhancement:** By collecting data on user preferences (e.g., preferred routes, historical destinations, favourite locations), the system could provide personalized recommendations for frequently visited locations or suggest alternate routes that align with user preferences.

**6. Integration with Smart Devices**

* **Current Limitation:** The system currently operates as a standalone application.
* **Enhancement:** Integration with wearable devices (like smartwatches) or in-car systems (such as Apple CarPlay or Android Auto) would allow for seamless navigation. These devices could alert the user with haptic feedback or audio cues, making navigation more hands-free.

**7. Multi-modal Navigation**

* **Current Limitation:** The system currently only provides driving directions.
* **Enhancement:** Expanding the system to offer multi-modal navigation could allow users to choose between walking, cycling, public transit, or driving, and provide the best route based on the chosen mode of transport. This would require integration with public transportation APIs and walking/cycling route algorithms.

**8. User-Reported Data**

* **Current Limitation:** The system is based on predefined data from the mapping service.
* **Enhancement:** Allowing users to report issues such as road closures, hazards, or accidents directly through the app could improve the accuracy and timeliness of the route recommendations. This would turn the system into a more interactive, community-driven navigation tool.

**9. Automated Driving Assistance**

* **Current Limitation:** The system provides basic navigation but does not assist with car control or autonomous driving.
* **Enhancement:** For future autonomous or semi-autonomous vehicles, the system could be expanded to include integration with the car's control system, such as automated lane guidance, speed adjustments based on road conditions, and parking assistance.

**10. Sustainability and Eco-Friendly Routing**

* **Current Limitation:** The system currently focuses on providing the fastest route, which may not always be the most environmentally friendly.
* **Enhancement:** The addition of "eco-friendly" routing options that prioritize fuel-efficient or low-emission paths could promote greener transportation. These routes could avoid congested areas or use roads with less traffic, improving both time efficiency and environmental impact.

**Conclusion**

In conclusion, the GPS navigation system represents a crucial advancement in modern navigation, providing users with accurate, real-time location data and route guidance for efficient travel. The system, as demonstrated in this report, leverages GPS satellite data, mapping services, and APIs to help users navigate from one location to another with ease.

Throughout this report, we explored the working principles of GPS technology, the core components of a GPS navigation system, and its practical applications. The system described uses basic geolocation and routing techniques, offering real-time navigation support and saving users valuable time and effort by suggesting optimized routes.

However, there is a significant potential for future development in this field. With enhancements like real-time traffic updates, voice navigation, offline capabilities, and integration with emerging technologies such as augmented reality and autonomous vehicles, the GPS navigation system could evolve into a far more comprehensive and intuitive tool. These improvements would not only enrich the user experience but also contribute to smarter, more sustainable transportation systems.

Overall, the GPS navigation system is a pivotal technology that continues to evolve, and its impact on daily travel, transportation, and logistics is undeniable. As we look toward future enhancements, the possibilities for improving navigation systems and further integrating them into everyday life are immense.

**Reference**

1. Wikipedia – “Global Positioning System.” *Wikipedia, The Free Encyclopaedia*.  
   <https://en.wikipedia.org/wiki/Global_Positioning_System>
2. Geocoder Python Library Documentation.  
   [https://geocoder.readthedocs.io](https://geocoder.readthedocs.io/)
3. Folium Python Library Documentation.  
   <https://python-visualization.github.io/folium>
4. Open Route Service API Documentation.  
   <https://openrouteservice.org/dev/#/home>
5. Geopy Python Client for Geocoding Documentation.  
   [https://geopy.readthedocs.io](https://geopy.readthedocs.io/)
6. Google Maps Platform – Maps JavaScript API Documentation.  
   <https://developers.google.com/maps/documentation/javascript>
7. Parkinson, B. W., Spilker, J. J. (Eds.). *Global Positioning System: Theory and Applications*, Volume I & II. American Institute of Aeronautics and Astronautics, 1996–1997.
8. Hofmann‐Wellinghoff, B., Lichten egger, H., Collins, J. *Global Positioning System: Theory and Practice*. Springer, 5th Edition, 2001.