The methodology employed in this project is structured to systematically address the design, development, and implementation of a comprehensive hostel navigation system for the Federal University of Technology Akure (FUTA). The approach integrates geospatial data handling, web-based technologies, real-time user interaction, and system validation to ensure the development of a reliable and user-friendly application. The following sections provide an extensive overview of the methodology.

1. Data Collection and Preparation

1.2. **Data Cleaning and Processing**

The raw data collected from GPS required cleaning and processing to ensure its suitability for database integration and map visualization. This involved the following processes:

1. Preparation of the GPS data and names iin an excel sheet. The table was structured and the names were cross checked. Duplicates were also removed.
2. The excel sheet was then saved as a csv file and imported to ArcGIS desktop for further processing.
3. The csv file was used to create a layer by importing the csv file to ArcGIS as xy data.
4. The created layer was then exported as a feature class
5. The shapefile was reprojected to the EPSG: 4326 (WGS 1984) coordinate system as the web mapping package to be used does not accept none other than the WGS 198 or web Mercator. The result was exported as a shapefile.

A PostgreSQL database named ‘FutaHostels’ was created using the PG Admin app on a windws PC

A postGIS extension was also added so as to facilitate the management of spatial data

The postGIS bundle app for PostgreSQL was launched and connected to the created database

The shapefile created from aArcMap was then imported to the database using the bundle app.

The database was assessed on PG Admin to check if it’s in a good condition.

THE WEB MMAPPING PROCESS

The data was read to the web map application through a JQuery.ajax function on the javascript. This is used to execute a python file in the web server (wampserver)

This python file connects to the database through the psycopg python package. This facilitates the connection to the database and executes queries on the database.

The Python code then process the result from the query and converts it to a geojson data format.

For example; during the retrieval of all hostels during the initialization of the map, the ajax function is used to execute a python function that connects to and query the database to return all features (hostels) as a geojson. The geojson data is then used by leaflet to ceate a visualization

For the inital query, this query text was used

‘’’SELECT name, ST\_AsGeoJSON(geom) as geom

FROM hostels;’’’

For querying the database when the search button is pressed, the following query was applied;

‘’’SELECT Name, ST\_AsGeoJSON(geom) as geom FROM hostels

WHERE Name = %s''', (name,)’’’

Where name is the name that the user enters on the search bar.

The routing was carried out using the open route service (ORS) api. The ORS api uses a road network dataset from the open street map (OSM) as the network dataset used for the routing. The RS requires connecting with an API key and this was gotten after signing up on the website. After the ORS connection in the python script for routing, a route was created, passing some necessary inputs.

The resulting route was also converted to a geojson data, read through ajax and a layer would be created from it.

**Web Development and Integration**

The user interface of the hostel navigation system was developed using HTML, CSS, and JavaScript, ensuring a responsive and user-friendly design. The interface includes a map display, search bar, current location button and route calculation button. CSS was used to style the elements, providing a visually appealing and consistent design across different devices and screen sizes.

Leaflet.js, a popular open-source JavaScript library for interactive maps, was chosen to render the campus map and display hostel locations. Leaflet.js's lightweight nature and extensive plugin ecosystem made it ideal for this project. The map was initialized with multiple base layers (e.g., Google Streets, Google Hybrid, and Google Satellite), allowing users to switch between different views depending on their preference.

The data containing hostel locations was integrated into the map using Leaflet.js's native support for GeoJSON layers. The GeoJSON data is provided from the ajax function that communicates with a server to retrieve data from the database. The hostels were represented by points. Popups were added to the markers to display information about each hostel when clicked.

**Implementing Search, Suggestion Features and location tracking**

The search functionality was a key component of the hostel navigation system, allowing users to find specific hostels by name. This was implemented using Python Common Gateway Interface (CGI) scripts, which interacted with the PostgreSQL database to retrieve the relevant hostel data. The `search.py` script was designed to handle both exact and partial matches, ensuring flexibility in the search process.

To enhance the search experience, real-time search suggestions were implemented using AJAX for asynchronous data retrieval and the `fuzzywuzzy` Python library for fuzzy string matching. As users typed in the search bar, AJAX requests were sent to the server to retrieve potential matches, which were then displayed as suggestions in a dropdown list. This approach significantly improved the usability of the search feature, particularly for users unsure of exact hostel names.

Once a user selected a search result, the corresponding hostel's location was highlighted on the map, and the map view was automatically adjusted to center on the hostel. This was achieved by passing the selected hostel's coordinates from the CGI script to the Leaflet.js map via AJAX, ensuring a seamless and interactive experience.

location tracking was implemented to allow users to determine their current position on the map. This feature utilized the browser's Geolocation API, which retrieves the user's current coordinates. The API was integrated into the web application using JavaScript, with fallback mechanisms in place for browsers that do not support geolocation.

Once the user's location was retrieved, it was displayed on the map with a distinct marker, different from those used for hostel locations. A radius circle was drawn around the marker to indicate the accuracy of the location data. The map was automatically centered on the user's location through a custom function, providing a clear view of their position relative to nearby hostels.

**Route Planning with OpenRouteService (ORS)**

Route planning was an essential feature of the hostel navigation system, enabling users to find the shortest path to take to an hostel. The OpenRouteService (ORS) API was used for this purpose, as it provides robust routing capabilities. The ORS client was initialized in the `route.py` Python script, with the project's API key securely stored and managed.

The route planning algorithm was implemented in Python, utilizing the ORS API to calculate the best route to navigate to hostels. It takes as input the coordinated of the start and end points, which will be supplied by the search bar or the location buton. The script sends the geocoded coordinates to the ORS API, which returns the route data.

The route data returned by the ORS API was displayed on the map using Leaflet.js. A polyline was drawn on the map to represent the route, with pre-existing markers (during search) indicating the start and end points

**System Testing and Validation**

To validate the system's usability and effectiveness, a series of user testing sessions were conducted. Selected users, including students of FUTA, were invited to use the system and provide feedback on its functionality, ease of use, and performance. This feedback was analyzed to identify any issues or areas for improvement, which were then addressed in subsequent iterations of the development process.