Department of Computer Science & Engineering

University of Moratuwa

NextBus

Spatio-temporal Forecasting of Bus Arrival Times to Downstream Stops

Final Project Report

Prepared By: Group 07

Galappaththi A.S. - 210172N
Gallage D. - 210173T
Gothatuwa G.W.H.D. - 210189X

Abstract

This project aims to enhance the reliability of public transportation systems by predicting bus arrival times for downstream stops. The inconsistency and unpredictability of bus arrivals pose challenges for daily commuters, impacting the convenience and efficiency of transit systems. The research addresses this issue by implementing advanced forecasting models that accurately estimate bus arrival times, thereby providing passengers with more reliable information and reducing overall wait times. Specifically, this project focuses on multi-route and crossroad scenarios, areas often overlooked in conventional single-route prediction models.

To address this challenge, the project utilizes spatio-temporal data from the New York City MTA bus network, capturing real-time details on bus locations, routes, and both scheduled and actual arrival times. Preprocessing steps include managing missing entries, standardizing timestamps, and engineering relevant features such as traffic patterns and time-based attributes. This cleaned and structured dataset forms the foundation for an advanced model, designed to handle the intricate temporal and spatial interactions crucial to accurate arrival time predictions.

The methodology combines a Transformer-based spatiotemporal model and Graph Neural Networks (GNNs), both recognized for their effectiveness in capturing complex data patterns. The Transformer's encoder-decoder architecture supports sequence-to-sequence forecasting, enabling accurate long-term predictions, while GNNs address spatial dependencies by modeling bus routes and stops as interconnected nodes and edges. This dual-model approach allows for a detailed understanding of both temporal trends and spatial relationships, resulting in reliable predictions within complex urban transit networks.

A major output of this project is a real-time prediction tool embedded in a user-friendly dashboard, which displays up-to-date bus arrival times for all stops. The dashboard operates efficiently with streaming data, ensuring passengers have access to timely updates on expected arrival times based on the model's forecasts. This tool not only visualizes predictions but also ensures accessibility, empowering users to make informed travel decisions.

The implications of this project are significant for public transit reliability and user satisfaction. By improving prediction accuracy and accessibility, the project can enhance passenger confidence in transit schedules and may encourage greater ridership. Overall, the project demonstrates the value of spatiotemporal modeling for public transportation and serves as a framework for future applications in other urban transit systems.

Table of Contents

1. Introduction

1.1 Background of the Public Transit Forecasting Problem

The reliability of public transportation systems is crucial for urban mobility, as it directly impacts the daily routines of millions of passengers. In large metropolitan areas, buses are a vital mode of transportation, yet they often experience unpredictable delays due to factors such as traffic congestion, weather conditions, and varying passenger volumes. These delays create challenges for commuters, who rely on precise arrival information to plan their journeys. Despite advancements in transit systems, the unpredictability of bus arrival times remains a significant issue, particularly in complex multi-route and crossroad scenarios. Traditional single-route prediction models do not fully capture the interdependencies within interconnected urban transit networks, leaving a gap in real-time predictive capabilities that this project aims to address.

1.2 Rationale Behind Developing a Spatio-Temporal Forecasting Model

The motivation for developing an advanced bus arrival prediction model stems from the increasing demand for reliable public transit systems that provide real-time, accurate information. Current models often fail to account for spatial and temporal factors across bus networks, limiting their effectiveness in delivering accurate multi-stop arrival predictions. To overcome these limitations, this project adopts a multi-faceted approach, leveraging spatio-temporal data to model and analyze bus arrival times more accurately. Advanced modeling techniques, including Graph Neural Networks (GNNs) and Transformers, offer powerful tools to address the unique challenges posed by urban bus transit systems. By implementing a model that can handle complex relationships within multi-route networks, the project aims to enhance both the reliability of predictions and the overall user experience for public transit passengers.

1.3 Significance and Objectives of the Forecasting System

The system's primary objective is to improve the accuracy of bus arrival predictions, contributing to a more dependable public transportation experience for passengers. Accurate prediction of bus arrival times offers significant benefits, including reduced wait times, enhanced transfer reliability, and increased passenger satisfaction. Reliable arrival information also has the potential to encourage higher public transit usage, thereby supporting sustainable urban mobility. This project aims to develop a system that accurately predicts bus arrival times in real-time across interconnected routes, providing actionable insights to both transit operators and commuters. By bridging the existing gap in forecasting capabilities, the system supports a future where public transit can more effectively compete with personal vehicles, aiding cities in reducing congestion and environmental impact.

1.4 Overview of the system

The developed system comprises three main components: data ingestion and preprocessing, model training using spatio-temporal methods, and real-time prediction display through a dashboard interface. The data ingestion module collects and preprocesses spatio-temporal data from New York City's MTA, addressing challenges such as missing values and feature engineering to create a robust dataset. The model training phase applies state-of-the-art techniques, including Transformer architectures and GNNs, to learn both spatial and temporal dependencies among bus stops, routes, and timing patterns. Finally, the dashboard interface displays real-time arrival predictions, ensuring that passengers and transit operators have access to reliable, continuously updated information. This system, therefore, combines advanced machine learning techniques with an accessible interface to create an integrated solution that meets the needs of a modern public transportation network.

2. Literature Review

Theoretical Aspects Considered

Our bus arrival prediction system builds on recent advancements in machine learning, especially in the fields of Graph Neural Networks (GNNs), Attention Mechanisms, and Transformers for spatiotemporal forecasting. We focus on improving both spatial and temporal prediction capabilities, surpassing traditional methods with a novel model architecture.

Graph Neural Networks (GNNs) have shown great potential in capturing complex spatial dependencies between nodes in traffic networks. Traditional methods such as CNNs and RNNs struggle to adapt to the irregular graph structure inherent in transportation data. GNNs, particularly Graph Convolutional Networks and Graph Attention Networks, offer a way to model the interconnected relationships across bus stops and routes (Jiang & Luo, 2022)grate both spatial and temporal dependencies, we also draw from recent work on Long-Range Transformers, specifically Spacetimeformer, which combines the power of GNNs with temporal attention for more comprehensive predictions (Grigsby et al., 2021) .

Related Systems

Several existing models inform our approach:

1. GNN for Traffic Forecasting: Jiang et al. (2022) show GNNs outperform CNNs and RNNs in capturing spatial-temporal dependencies in traffic but highlight the need for dynamic graphs to represent changing relationships—an area addressed in our Spacetimeformer adaptation.[2]
2. GBTTE (Graph Attention Network for Bus Travel Time Estimation): Rong et al. (2023) introduce a 3D cross-graph attention mechanism for improved accuracy, yet it doesn't fully account for evolving relationships over time, which our model incorporates.[3]

3. Dynamic Bus Arrival Prediction with ANN: Chien et al. (2002) propose ANN-based models for bus arrival prediction but face limitations in adapting to real-time traffic, a challenge our Spacetimeformer model overcomes with dynamic spatial-temporal relationships.[4]

4. ST-ResNet (Zhang et al., 2017): This model uses deep residual CNNs to predict citywide crowd flows, treating the city grid as an image. Residual blocks help it handle complex spatial features and avoid the vanishing gradient problem, allowing for more accurate forecasting in urban areas.[5]

Novelty and Improvements in our system

The system improves upon existing models by integrating Spacetimeformer with a dynamic GNN architecture that can model evolving spatial relationships between bus stops and routes. Unlike previous models that rely on predefined, static graphs, our approach captures dynamic relationships, making it more adaptable to real-time fluctuations in traffic and bus routes. The Spacetimeformer's dual-attention mechanism—combining both spatial and temporal attention—enhances predictive accuracy by simultaneously learning from geographic and temporal patterns, thus addressing a key limitation in earlier models.

3. System Models

3.1 System Requirements

The Bus Arrival Time Prediction System is designed to improve passenger experience and enhance the efficiency of public transportation. The functional requirements include several core features: bus arrival time prediction, route planning, and real-time updates. The bus arrival time prediction feature employs advanced machine learning algorithms that analyze historical data, real-time traffic conditions, and user inputs to provide accurate predictions for passengers waiting at bus stops. The route planning functionality assists users in finding the most efficient journey by considering factors such as traffic patterns, estimated travel times, and transfer points, empowering passengers to make informed travel decisions. Additionally, real-time updates keep commuters informed about any changes to bus schedules or unexpected delays, thus enhancing the overall reliability of the service.

The non-functional requirements focus on reliability, scalability, and performance. The system is designed to maintain high reliability, ensuring that users receive timely and accurate information even during peak hours. Scalability is a critical aspect, allowing the system to grow in capacity and functionality as user demand increases, thus supporting an expanding network of bus routes and passengers. Performance metrics aim for minimal response times, providing a smooth user experience even under heavy loads. Moreover, robust security protocols are implemented to protect sensitive passenger data, and the user interface is designed for usability, ensuring it is intuitive for a diverse range of users, including those who may not be tech-savvy. Together, these requirements create a user-centric, efficient, and secure system that meets the demands of modern public transportation.

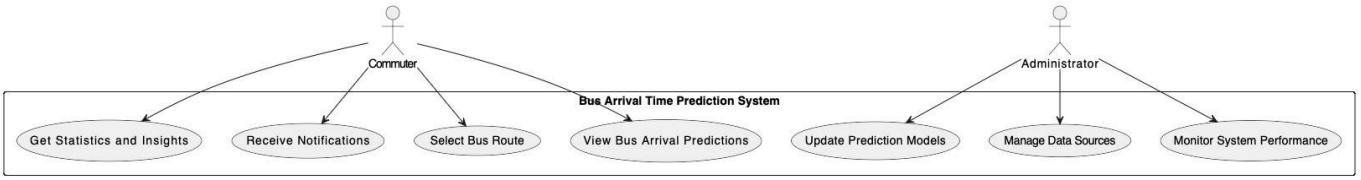


Fig. 1. Use Case Diagram for Bus Arrival Time Prediction System

Fig. 1. Use Case Diagram for Bus Arrival Time Prediction System illustrates the interactions between two primary actors in the Bus Arrival Time Prediction System: the Commuter and the Administrator. The Commuter can access key functionalities, including retrieving real-time bus arrival times, obtaining statistics and insights about historical arrival data, and receiving notifications regarding schedule changes or delays. This ensures that commuters can plan their journeys more effectively and stay informed. On the other hand, the Administrator is responsible for managing the bus schedule, allowing for updates and adjustments based on real-time data to maintain service efficiency. The use case diagram emphasizes the user-centric design of the system, showcasing how it addresses the needs of both commuters and administrators while facilitating improved operational efficiency in bus transit. This visual representation clarifies the functional landscape of the application, highlighting its dual focus on enhancing the commuter experience and ensuring smooth administrative oversight.

3.2 System Design

The Bus Arrival Time Prediction System is designed using a layered architecture that promotes modularity, maintainability, and scalability. At the presentation layer, the user interface—comprising web and mobile interface—enables both commuters and administrators to access real-time bus arrival information, view statistics, and receive notifications, ensuring ease of use for diverse users. The application layer contains the business logic, processing user requests, managing interactions between the presentation and data layers, and executing machine learning models for bus arrival time prediction, along with route planning and real-time updates. The data layer manages data storage and retrieval, integrating various sources such as GPS data and traffic management databases to provide accurate information. Additionally, the machine learning module within the application layer utilizes predictive algorithms trained on historical data to enhance arrival time accuracy. This layered architecture ensures a responsive, efficient, and scalable system that can adapt to growing user demands in urban transit.

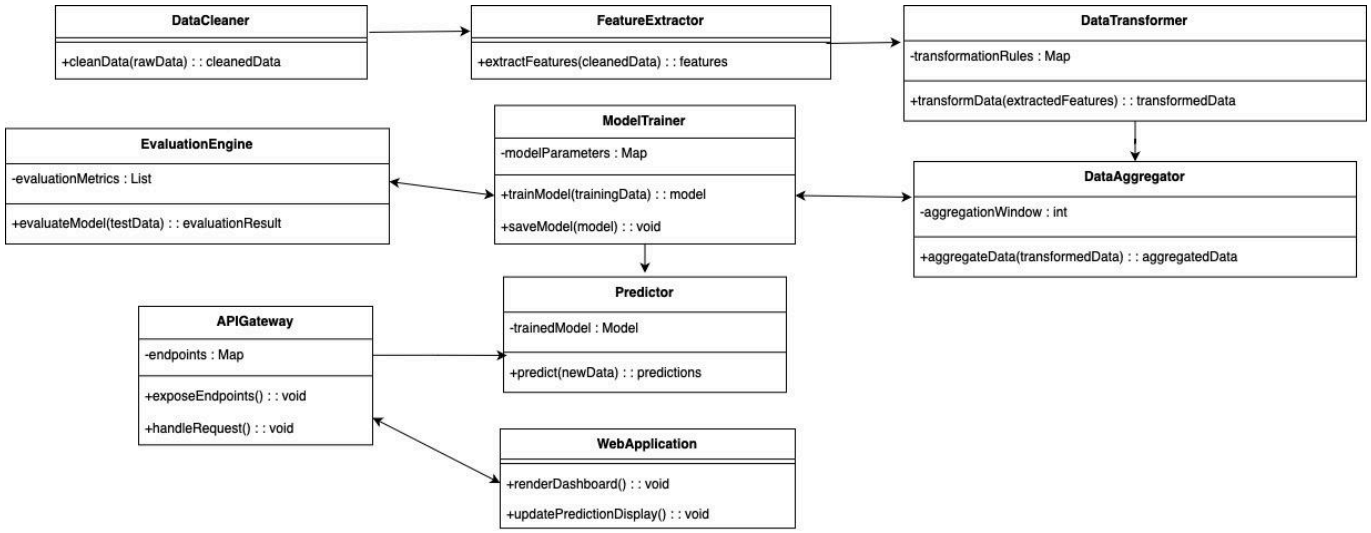


Fig.2. Class Diagram for Bus Arrival Time Prediction System

Fig.2. Class Diagram for Bus Arrival Time Prediction System outlines a system for predicting bus arrival times. It encompasses data cleaning, feature extraction, transformation, and aggregation to prepare data for model training. The trained model is evaluated to assess its performance. An API gateway exposes endpoints for receiving new data and generating predictions. These predictions are then integrated into a web application that displays a dashboard with the predicted bus arrival times, which are updated dynamically as new predictions become available.

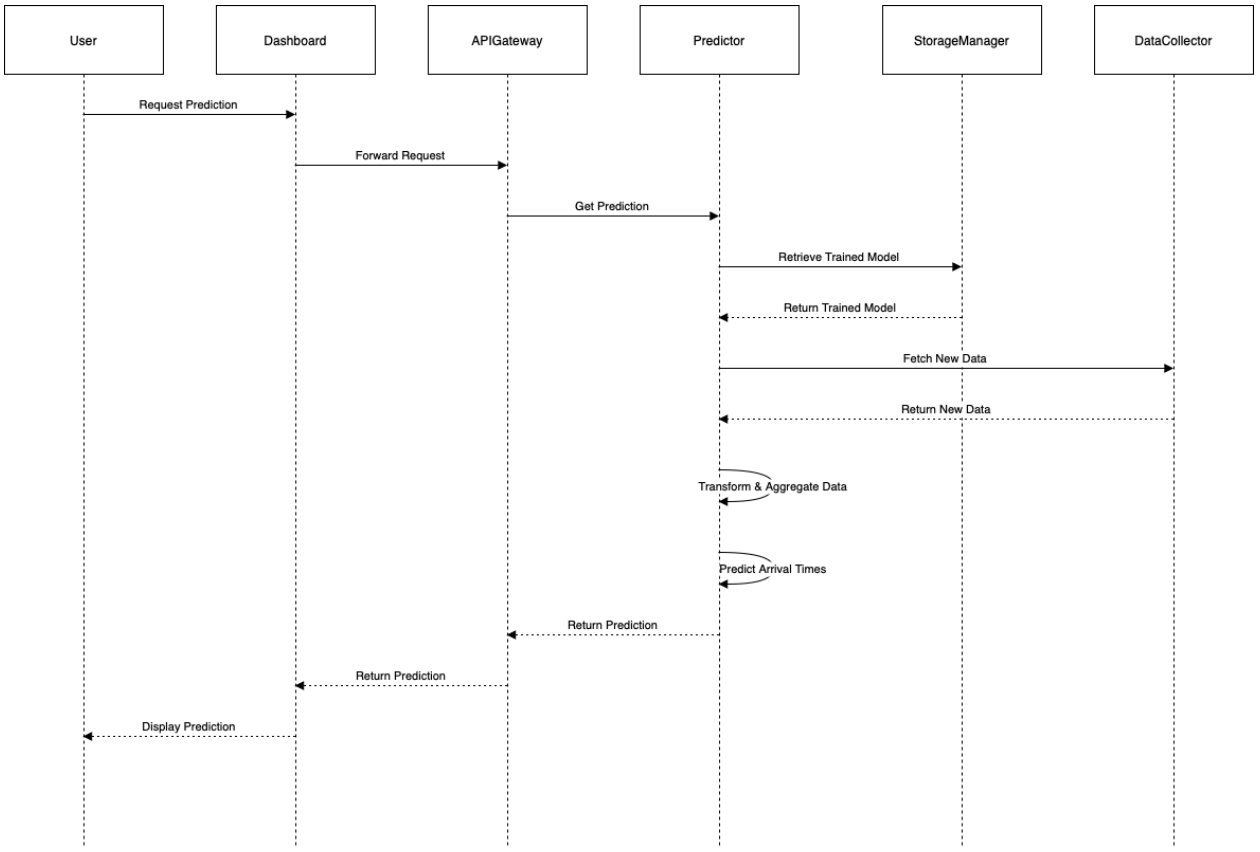


Fig.3. Sequence Diagram for Bus Arrival Time Prediction System

Fig.3. The sequence diagram illustrates the flow of interactions between the components involved in the bus arrival time prediction system. The user initiates the process by requesting a prediction from the dashboard. The dashboard forwards this request to the API gateway, which in turn requests a prediction from the Predictor. The Predictor retrieves the pre-trained model from the Storage Manager and fetches new data from the Data Collector. It then transforms and aggregates the new data, uses the trained model to predict the bus arrival times, and returns the prediction result to the API gateway. The API gateway then relays the prediction back to the dashboard, which displays it to the user. This sequence diagram effectively visualizes the collaborative effort of the components involved in providing accurate bus arrival time predictions.

3.3 Database Design

The database design for the Bus Arrival Time Prediction System relies on Firebase Storage and Firestore to manage and store essential data, including information on bus routes, bus stops, and hourly statistics. Firestore's NoSQL structure is suited to handle dynamic, real-time data updates, making it ideal for this system's requirements.

The Bus Route collection in Firestore contains details about each route, with fields such as destination_id (string identifier), direction (number, denoting route direction), name (string, for the route's name like "M60-SBS"), origin_id(string identifier), and stops_count (number indicating the total number of stops along the route). This structure allows for efficient retrieval of routes with associated metadata that is key for prediction models and route planning.

The Bus Stops collection includes individual stops on each route, with fields like latitude and longitude, and name (string, indicating the stop's name, such as "MORRIS AV/E 149 ST"). This setup provides spatial reference points for routes and enables accurate arrival predictions for each stop.

Lastly, the Hourly Stats collection holds statistical data on passenger volume by hour and day of the week for each route. Fields here include day_of_week (string, e.g., "Thursday"), hourly_counts (a JSON string mapping each hour to the passenger count), off_peak_hour (number, indicating the lowest traffic hour), peak_hour (number, representing the highest traffic hour), and route_name (string for the route, such as "Q55"). These stats are critical for analyzing patterns and optimizing arrival predictions based on typical passenger flow.

3.4 Machine Learning System Design

The Bus Arrival Time Prediction System's machine learning and data science components are designed to deliver accurate, real-time predictions by utilizing a blend of historical and live transit data. The core ML model is built to predict bus arrival times at each stop by analyzing various factors such as historical arrival times, route-specific traffic patterns, and live GPS data. This predictive capability leverages time-series forecasting and spatial modeling to capture both temporal patterns and the geographic dependencies across the bus network.

Key components include a data preprocessing pipeline, which cleans and normalizes data from multiple sources, including Firestore data on routes and stops and live data streams for real-time updates. This data

pipeline handles missing values, normalizes time series, encodes categorical data, and feature selection, ensuring the ML model receives standardized inputs for optimal performance.

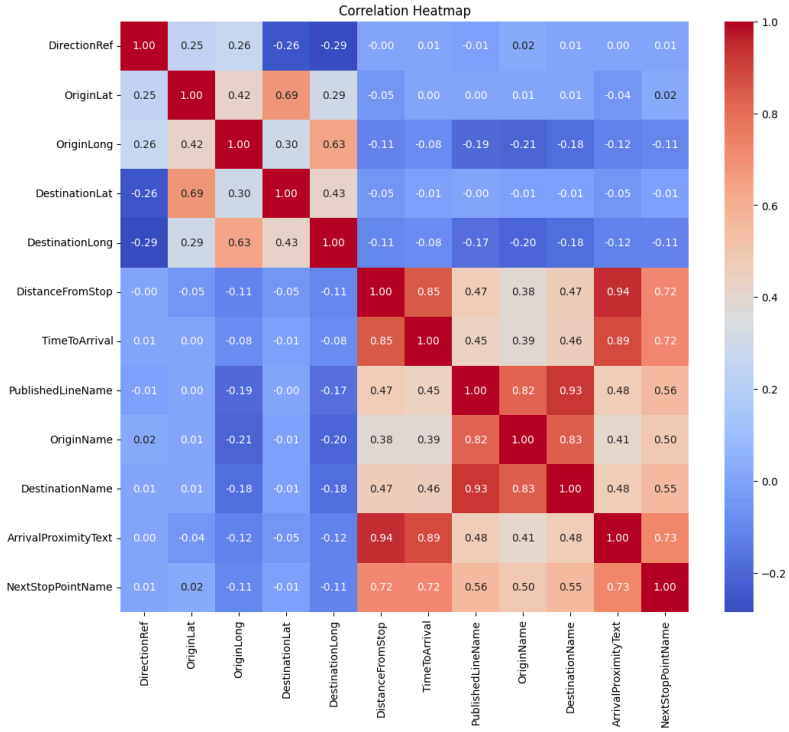


Fig. 4. Feature correlation heat map

The Bus Arrival Time Prediction System employs Spacetimeformer as the core model for machine learning and data science components. The Spacetimeformer model excels in capturing the intricate temporal dependencies and spatial relationships inherent in transit data. By incorporating spatial embeddings, the model encodes route-specific information, allowing it to differentiate between various bus lines and their unique patterns of movement. This capability is crucial in a dynamic urban environment where traffic conditions, passenger demand, and route modifications can significantly impact bus arrival times.

The model processes multivariate time series data, where each input token corresponds to a specific variable's value at a designated time interval. This setup enables the model to learn interactions between space, time, and value information in a cohesive manner. For instance, factors such as the distance from the next stop, expected traffic conditions, and historical arrival times are all considered simultaneously. By doing so, the Spacetimeformer can identify patterns and correlations that might not be apparent when analyzing variables in isolation.

A key feature of the Spacetimeformer model is its spatiotemporal attention mechanism, which allows it to learn dynamic relationships among variables from the training data without relying on hardcoded graphs. This adaptability is essential for accurately forecasting bus arrivals in an environment where conditions are constantly changing.

The attention mechanism evaluates the importance of each input token, determining which variables and time frames should be emphasized based on their relevance to the current prediction. This flexibility facilitates the identification of critical factors that influence bus arrival times, such as delays due to road construction or increased passenger loads during peak hours.

To ensure the model remains effective over time, the Bus Arrival Time Prediction System incorporates a continuous model retraining framework. This approach allows the system to adapt to new data, trends, and anomalies, improving its predictive accuracy as conditions evolve. Additionally, a comprehensive model evaluation and monitoring framework is in place to track performance metrics and notify administrators of any significant accuracy drops. This proactive monitoring helps maintain a high level of service reliability and user satisfaction.

4. System Implementation

4.1 Implementation Procedure

The implementation process was carefully orchestrated to ensure that all system components—from data ingestion to user interaction—functioned cohesively and efficiently. Below are the major stages of the system's development:

- **Frontend Development**: We constructed the user interface using **React** and **Vite**, a combination that enabled faster loading times and a highly responsive experience for users. React's component-based architecture facilitated the development of reusable and efficient UI elements, while Vite streamlined the building process, minimizing loading lag for better user interaction. The frontend design emphasizes clarity and intuitiveness, allowing users to quickly access bus arrival data and insights with minimal clicks. Additionally, **Vitest** was integrated into the development pipeline to rigorously test each component, ensuring the UI remained reliable across various devices and scenarios. By implementing responsive design techniques, we also optimized the application for mobile use, making it accessible to passengers on the go.

- **Backend Configuration**: The backend was built using **FastAPI**, a modern, high-performance web framework that supported the system's need for real-time data exchange and processing. FastAPI's asynchronous capabilities allowed the backend to handle multiple data requests simultaneously, which was essential for providing accurate, up-to-date bus arrival predictions in real-time. Custom API endpoints were created to facilitate seamless communication between the frontend and backend. These endpoints allowed the frontend to request specific data—such as arrival predictions for a particular bus route—while enabling efficient and secure data exchange. The backend was configured to manage tasks like data fetching, processing, and storage, ensuring that users receive accurate information with minimal delay.

- **Data Processing Pipeline**: Data from the NYC MTA, which includes extensive bus location, route, and timing information, was streamed and stored in **Firebase Firestore**. To prepare the raw data for analysis and prediction, the dataset underwent a series of preprocessing steps:

    - **Handling Missing Values**: Given the scale of the data, we opted to remove rows with missing values, which preserved the dataset's overall statistical properties without compromising analysis quality.

    - **Outlier Detection and Removal**: Outliers in arrival times and other features were identified and filtered out to maintain data quality and model accuracy. These extreme values could otherwise skew predictions, so removing them improved overall model reliability.

    - **Date and Time Consistency**: We identified and resolved inconsistencies in time formats, such as converting '24:00' hours to '00:00' and adjusting dates accordingly. This step ensured accurate, uniform time-based data for the algorithms.

- **Feature Encoding and Normalization**: Key categorical features were encoded, and continuous variables were normalized to standardize data input for model training. This encoding and normalization step was critical for optimizing model performance, as it ensured uniformity in data representation.
- **Sampling**: A random sample of 500,000 records was selected from a total of 5 million records using stratified sampling, maintaining the statistical characteristics of the entire dataset without the need to process the full volume of data. This approach was efficient and prevented memory overload.

4.2 Materials

The project relied on a combination of data sources, tools, and technologies, each playing a unique role in the system's development:

- **Data Source**: The NYC MTA bus dataset was utilized as it offered a comprehensive stream of bus locations, schedules, and routes, updated every 10 minutes. This data allowed us to create a detailed predictive model, leveraging consistent updates for real-time predictions.
- **Frontend Stack**: The frontend was designed with **React** and **Vite** for creating a responsive, fast-loading user interface. **Vitest** was used for thorough testing, ensuring every component operated as expected and maintaining UI consistency across devices and use cases.
- **Backend Technologies**: **FastAPI** was chosen for its high-performance capabilities, allowing smooth communication with the frontend and efficient data processing. Firebase Firestore provided a flexible, scalable database solution, accommodating real-time updates in the dataset and making sure data was reliably stored and retrievable.
- **Testing and API Validation**: **Postman** was used to test API endpoints, ensuring that each endpoint performed correctly and returned accurate data, establishing the backend's reliability before connecting it to the frontend.
- **Deployment Platform**: The entire system was deployed on **Vercel**, enabling a scalable and responsive platform that supported both desktop and mobile access. Vercel's deployment capabilities allowed rapid updates to the application, streamlining maintenance and enabling real-time feedback.

4.3 The Algorithm

A combination of advanced algorithms was explored to determine the most effective model for predicting bus arrival times, focusing on their ability to handle spatiotemporal data—data that combines spatial and temporal elements to reflect changing patterns over time. Three models were evaluated for comparative analysis:

- **Graph Neural Network (GNN)**: GNN was applied to capture spatial relationships among different bus stops. The network mapped the spatial connectivity of stops and transit routes, attempting to model the relationships between stops in a way that could predict bus delays and arrivals. However, GNN struggled with temporal shifts, as it was less capable of adapting to dynamic temporal variations, which are crucial for accurate arrival predictions.
- **Convolutional Neural Network (CNN)**: CNN was tested for its strength in identifying spatial patterns in data. By focusing on spatial features across bus routes and stops, CNN could predict certain repetitive patterns in arrival times. However, the model lacked the flexibility needed to fully address time-dependent fluctuations, making it less effective in handling the real-time variability that occurs in public transit.
- **Spacetimeformer (Proposed Model)**: **Spacetimeformer** was selected as the primary model due to its state-of-the-art design for dynamic spatiotemporal forecasting. This model uniquely combines time-series prediction with spatial relationship modeling, capturing both aspects simultaneously. Spacetimeformer adapts to changing graph structures, making it ideal for scenarios like transit systems where stop-to-stop dynamics change based on time and traffic conditions.
    - **Hyperparameter Tuning**: To improve the model's predictive accuracy, time-series cross-validation was employed for hyperparameter tuning. This iterative process allowed us to identify optimal parameter values for Spacetimeformer, resulting in enhanced accuracy for bus arrival predictions. The model's performance was evaluated using **MAE**, **MSE**, and **RMSE** metrics, with Spacetimeformer outperforming GNN and CNN in each metric category, confirming its suitability for this task.

4.4 Main Interface

The user interface was designed with simplicity, user-friendliness, and accessibility in mind, ensuring that users can easily access and understand the bus arrival predictions:

- **Dashboard Insights**: The main dashboard displays daily, weekly, and hourly bus arrival insights, allowing users to view peak hours, travel patterns, and any fluctuations in arrival times. The dashboard empowers users to make well-informed travel plans based on reliable arrival data.
- **Color-Coded Indicators**: To simplify navigation, color coding was implemented. Green indicates shorter wait times and better schedule adherence, while red highlights potential delays or longer wait times, allowing users to quickly gauge the status of buses and adjust their plans accordingly.

- **Responsive Layout**: The UI was designed with mobile accessibility in mind, offering a layout that adapts to smartphones, tablets, and desktops. Users can seamlessly access bus schedules and insights on the go, and can switch between light and dark modes to improve readability in different lighting environments.
- **Streamlined Navigation**: The UI design prioritizes clear and logical flow, with easily accessible tabs for viewing various insights and recommendations. This streamlined navigation makes it easy for users to move between different sections of the dashboard, reducing cognitive load and enhancing overall usability.
- **User-Friendly Suggestions**: To assist users with planning, the interface includes travel suggestions based on analysis of delay patterns and historical data. These recommendations help users optimize their routes, minimize wait times, and avoid potential delays.

Overall, the main interface combines an attractive, simple layout with efficient functionality, presenting users with essential transit data in an intuitive and visually appealing format. This approach encourages users to rely on the system for timely and accurate information, ultimately enhancing their transit experience and promoting smoother travel planning.

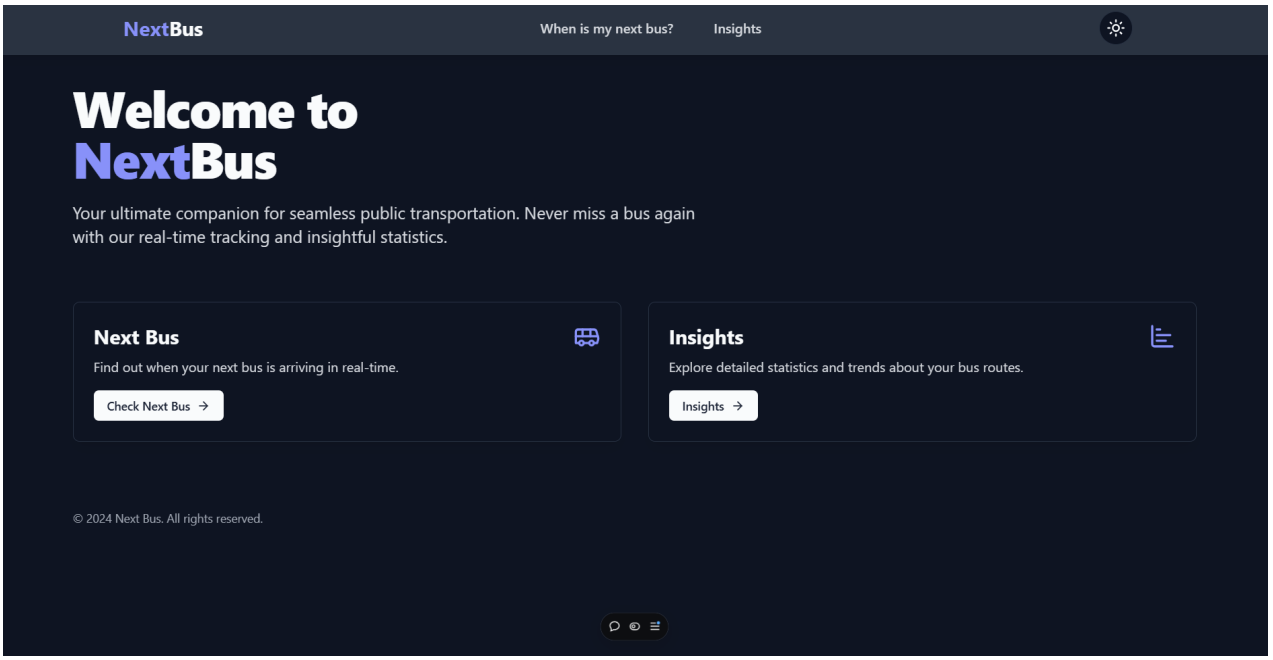


Fig. 5. Main user dashboard of the system

5. System Testing and Analysis

To ensure the robustness and reliability of the system, extensive testing and analysis were conducted across all key components, including the frontend, backend, database, and machine learning model. The testing process aimed to identify and resolve potential issues early, optimize performance under high loads, and validate the accuracy and efficiency of the predictive model. Each layer of the application was rigorously examined through unit testing, performance evaluations, and model validation techniques. This structured approach enabled the early identification of faults and ensured the system would perform reliably in real-world conditions.

5.1 Testing Approach

The testing approach for this project encompasses a comprehensive suite of techniques designed to ensure the robustness, functionality, and reliability of both the application components and the machine learning model. Frontend unit tests were conducted using **Vitest**, a testing framework for the frontend codebase, while **PyTest** was employed for backend unit testing. These frameworks enabled rigorous validation of individual functions and components to confirm they behave as expected in isolation. For code quality and consistency, **ESLint** and **Code Spell Checker** in Visual Studio Code were utilized, identifying syntax, style, and spelling issues early in the development process. Load testing was executed with **JMeter** to evaluate the system's behavior under varying levels of demand, while Firebase database performance was assessed to ensure optimal data handling. Additionally, cross-browser testing in Edge, Chrome, and Firefox verified consistent functionality and performance across multiple platforms.

5.2 Unit Testing, Results and Analysis of testing

Unit testing was critical in validating the integrity of both frontend and backend modules. **Vitest** allowed for modular testing of React components, assessing both UI behavior and logical functionalities, which was essential in isolating potential issues before integration. **PyTest** was similarly effective in testing backend modules, ensuring that each function behaved correctly under expected inputs and handling edge cases robustly. The test cases designed for these frameworks primarily targeted functionality and error handling, with particular emphasis on routes, request responses, and data transformations. Overall, the unit testing results confirmed the functionality of each module independently, with a high success rate in initial tests, which reduced the time needed for debugging during the integration phase.

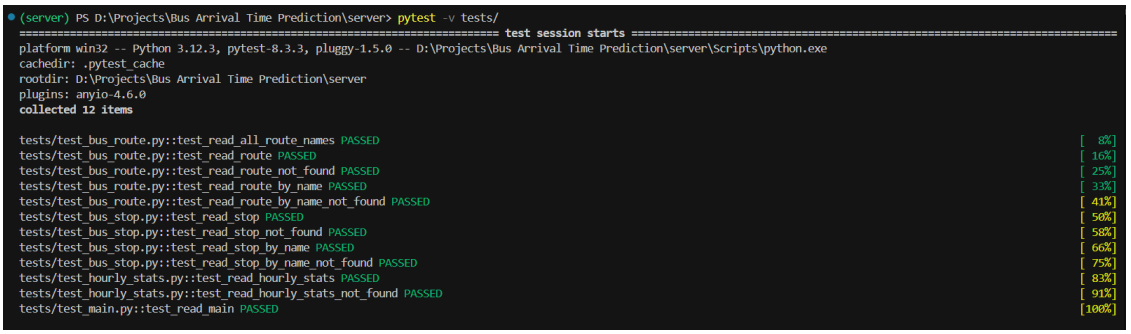


Fig. 6. PyTest log of the backend unit tests

5.3 Aspects related to Performance, Security, and Failures

Performance testing involved load assessments through **JMeter**, simulating high levels of concurrent user requests to measure the application's response times, throughput, and reliability under stress. These tests helped identify and optimize potential bottlenecks, particularly in API response times and data retrieval from **Firebase**. The results confirmed that the application could handle high loads with minimal delays, though some optimizations were made to the database queries to enhance efficiency. Cross-browser testing on **Edge**, **Chrome**, and **Firefox** provided assurance of consistent functionality and responsiveness. Security testing focused on safeguarding user data and preventing unauthorized access; thus, standard web security practices, including input validation and access control, were rigorously implemented. Failures identified through these tests were minimal, with adjustments promptly made to mitigate potential vulnerabilities and ensure resilience against unexpected issues.

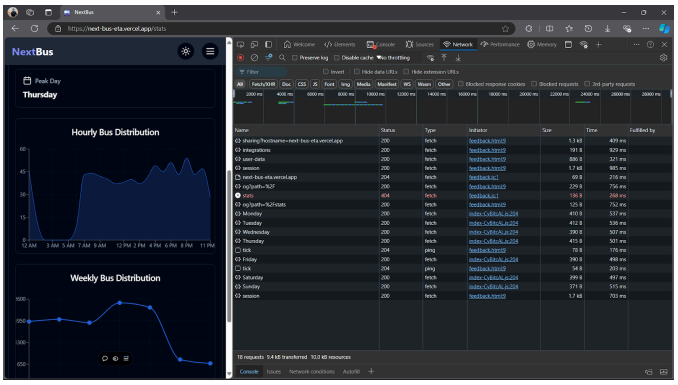


Fig. 7. Browser performance details

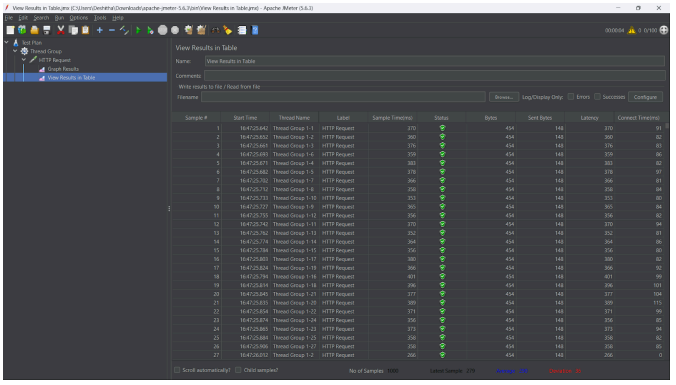


Fig. 8. Load testing logs from JMeter

5.4 Evaluation of ML Models

The machine learning models were evaluated using **Time Series Cross-Validation**, an approach well-suited for spatiotemporal data forecasting. This method involves training and testing the model over sequential splits of historical data, allowing for more reliable performance metrics that account for temporal dependencies. Key evaluation metrics included Mean Absolute Error (MAE), and Root Mean Squared Error (RMSE), providing insights into the model's predictive accuracy. Through iterative tuning and evaluation, the model demonstrated substantial accuracy, with minimal overfitting and generalization errors. This evaluation approach was critical in assessing the model's reliability in real-world scenarios, ensuring it could provide accurate and timely predictions essential for enhancing user experience in the public transit application.

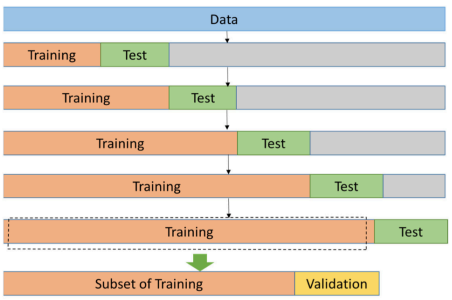


Fig. 9. Dataset splitting for the time series cross validation

To assess the effectiveness of our proposed Spacetimeformer model, we compared its results with existing forecasting methods. Table 1. below summarizes the performance differences:

Model	MAE	MSE	RMSE
GNN	0.1889	0.0630	0.2510
CNN	0.1978	0.0911	0.3019
Spacetimeformer	0.1187	0.0364	0.1907

Table 1. Final model comparison table

In comparison, Spacetimeformer consistently achieved lower MAE and RMSE values than the other methods, indicating superior predictive accuracy and robustness for spatiotemporal forecasting. This improvement highlights the model's ability to capture complex spatial and temporal dependencies, which are crucial for accurate public transportation predictions.

6. Conclusion and Future Work

6.1 Conclusion

This project has effectively demonstrated the application of spatiotemporal forecasting models to improve public transportation by accurately predicting bus arrival times. Utilizing data from the NYC MTA, we developed a comprehensive system that minimizes passenger wait times and enhances the reliability of transit services. Our comparative analysis highlighted Spacetimeformer as the most effective model, achieving superior performance compared to Graph Neural Networks (GNN) and Convolutional Neural Networks (CNN), owing to its capability to dynamically process spatiotemporal data.

The results indicate that Spacetimeformer successfully captures both spatial and temporal relationships, which is evident in its lower Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) in comparison to other models. Moreover, we created a user-friendly interface that visually conveys bus arrival information and trends, significantly increasing the system's practical value. This project underscores the potential of predictive technologies in public transportation, offering a scalable solution for cities seeking to enhance transit scheduling and improve user satisfaction.

6.2 Future Work

Building on our current success, several avenues for future development are proposed to further refine the model and extend its application. Future efforts will concentrate on the following areas:

- **Creation of a Local Dataset**: While this project utilized NYC MTA data, our ultimate objective is to implement these predictive models for transit systems in Sri Lanka. Partnering with local transit authorities to gather a comprehensive dataset focused on Sri Lankan bus routes, stops, and

schedules would enable the model to generate more accurate predictions relevant to local conditions.

- **Incorporation of Environmental and Traffic Data**: To account for additional factors influencing bus schedules, future models could integrate data on weather conditions, road situations, and real-time traffic updates. Including these external variables would enhance the model's adaptability to real-world scenarios, improving both accuracy and reliability in diverse conditions.

- **Development of a Mobile Application**: To enhance accessibility for users, we plan to create a mobile app that delivers bus arrival predictions and transit insights directly to passengers' smartphones. This app will feature real-time notifications for bus arrivals and delays, route-specific alerts, and customizable travel preferences. Designed for mobile devices, the app will allow users to quickly and reliably access arrival information on the move, increasing the system's practical utility.

- **Expansion of Algorithmic Models**: Although Spacetimeformer has proven effective, further exploration of emerging models such as Temporal Graph Networks (TGNs) or ensemble models that leverage the strengths of multiple approaches could yield additional improvements. These alternative models may better capture complex data relationships, offering enhanced predictive capabilities and flexibility.

- **Advanced Hyperparameter Tuning**: To achieve higher accuracy levels, future efforts could focus on implementing automated tuning techniques, such as Bayesian Optimization or Grid Search, over a broader and more precise range of parameters. This would facilitate more refined configurations of the Spacetimeformer model, potentially reducing error rates and improving prediction accuracy.

- **User Interface and Experience Enhancements**: To ensure continuous improvement, we intend to refine the dashboard based on user feedback, incorporating features like detailed transit analytics, personalized travel recommendations, and interactive maps. Such enhancements would allow users to engage more fully with the system, tailoring it to meet a wider variety of travel needs and preferences.

- **Real-Time Model Adaptation**: Given the dynamic nature of transit systems, with schedules frequently adjusted due to various factors, implementing a real-time learning loop could enable the model to continuously adapt to new data, refining its predictions as patterns change. This capability would help maintain the system's effectiveness in response to ongoing variations in traffic or operational changes.

In summary, this project lays a solid foundation for predicting bus arrival times in public transit systems, providing significant benefits in terms of convenience, reliability, and efficiency. By incorporating additional data sources, enhancing model capabilities, and improving user accessibility—particularly through the development of a mobile application—this system can serve as a crucial tool for smart transit planning in urban areas, ultimately delivering a more connected and user-friendly public transportation experience.

References

[1] Grigsby, J., Wang, Z., Nguyen, N., & Qi, Y. (2021). Long-Range Transformers for Dynamic Spatiotemporal Forecasting

[2] Jiang, W., & Luo, J. (2022). Graph neural network for traffic forecasting: A survey. Expert Systems with Applications, 207, 117921.

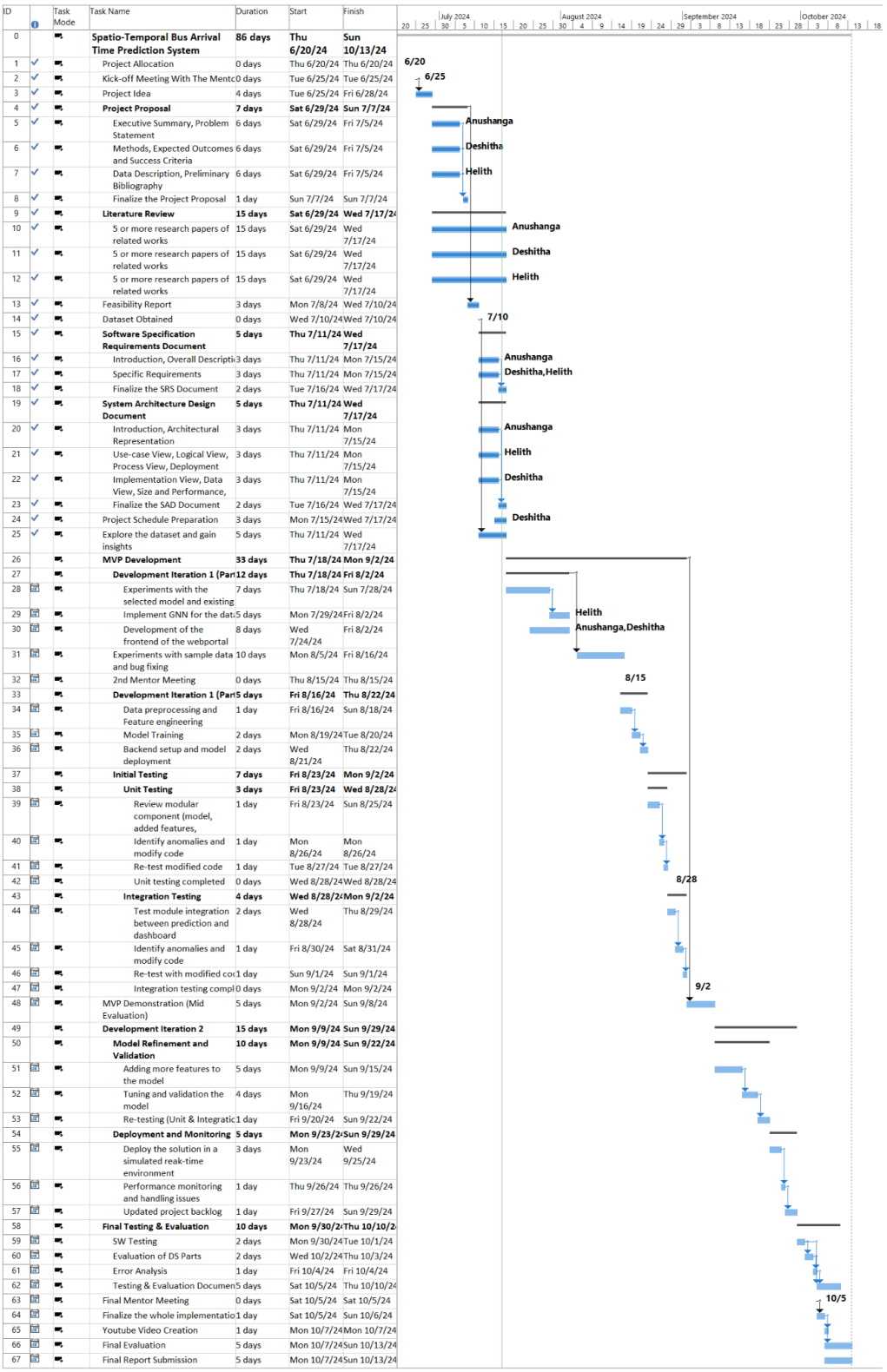
[3] Jiang, W., Luo, J., He, M., & Gu, W. (2023). Graph neural network for traffic forecasting: The research progress. ISPRS International Journal of Geo-Information, 12(3), 100.

[4] Dynamic Bus Arrival Time Prediction with Artificial Neural Networks 123 Steven I-Jy Chien, M.ASCE ; Yuqing Ding ; and Chienhung Wei

[5] Zhang, Junbo & Zheng, Yu & Qi, Dekang. (2016). Deep Spatio-Temporal Residual Networks for Citywide Crowd Flows Prediction. Proceedings of the AAAI Conference on Artificial Intelligence. 31. 10.1609/aaai.v31i1.10735.

Appendix A: Project Schedule

The following Gantt chart provides a detailed timeline of the project schedule, including key phases, tasks, and milestones. This schedule was created to ensure timely completion of each phase, from initial research and data collection to model evaluation and report writing.



ID	Task Mode	Task Name	Duration	Start	Finish
0		**Spatio-Temporal Bus Arrival Time Prediction System**	**86 days**	**Thu 6/20/24**	**Sun 10/13/24**
1		Project Allocation	0 days	Thu 6/20/24	Thu 6/20/24
2		Kick-off Meeting With The Mentc	0 days	Tue 6/25/24	Tue 6/25/24
3		Project Idea	4 days	Tue 6/25/24	Fri 6/28/24
4		**Project Proposal**	**7 days**	**Sat 6/29/24**	**Sun 7/7/24**
5		Executive Summary, Problem Statement	6 days	Sat 6/29/24	Fri 7/5/24
6		Methods, Expected Outcomes and Success Criteria	6 days	Sat 6/29/24	Fri 7/5/24
7		Data Description, Preliminary Bibliography	6 days	Sat 6/29/24	Fri 7/5/24
8		Finalize the Project Proposal	1 day	Sun 7/7/24	Sun 7/7/24
9		**Literature Review**	**15 days**	**Sat 6/29/24**	**Wed 7/17/24**
10		5 or more research papers of related works	15 days	Sat 6/29/24	Wed 7/17/24
11		5 or more research papers of related works	15 days	Sat 6/29/24	Wed 7/17/24
12		5 or more research papers of related works	15 days	Sat 6/29/24	Wed 7/17/24
13		Feasibility Report	3 days	Mon 7/8/24	Wed 7/10/24
14		Dataset Obtained	0 days	Wed 7/10/24	Wed 7/10/24
15		**Software Specification Requirements Document**	**5 days**	**Thu 7/11/24**	**Wed 7/17/24**
16		Introduction, Overall Descripti	3 days	Thu 7/11/24	Mon 7/15/24
17		Specific Requirements	3 days	Thu 7/11/24	Mon 7/15/24
18		Finalize the SRS Document	2 days	Tue 7/16/24	Wed 7/17/24
19		**System Architecture Design Document**	**5 days**	**Thu 7/11/24**	**Wed 7/17/24**
20		Introduction, Architectural Representation	3 days	Thu 7/11/24	Mon 7/15/24
21		Use-case View, Logical View, Process View, Deployment	3 days	Thu 7/11/24	Mon 7/15/24
22		Implementation View, Data View, Size and Performance,	3 days	Thu 7/11/24	Mon 7/15/24
23		Finalize the SAD Document	2 days	Tue 7/16/24	Wed 7/17/24
24		Project Schedule Preparation	3 days	Mon 7/15/24	Wed 7/17/24
25		Explore the dataset and gain insights	5 days	Thu 7/11/24	Wed 7/17/24
26		**MVP Development**	**33 days**	**Thu 7/18/24**	**Mon 9/2/24**
27		**Development Iteration 1 (Par**	**12 days**	**Thu 7/18/24**	**Fri 8/2/24**
28		Experiments with the selected model and existing	7 days	Thu 7/18/24	Sun 7/28/24
29		Implement GNN for the dat	5 days	Mon 7/29/24	Fri 8/2/24
30		Development of the frontend of the webportal	8 days	Wed 7/24/24	Fri 8/2/24
31		Experiments with sample data and bug fixing	10 days	Mon 8/5/24	Fri 8/16/24
32		2nd Mentor Meeting	0 days	Thu 8/15/24	Thu 8/15/24
33		**Development Iteration 1 (Par**	**5 days**	**Fri 8/16/24**	**Thu 8/22/24**
34		Data preprocessing and Feature engineering	1 day	Fri 8/16/24	Sun 8/18/24
35		Model Training	2 days	Mon 8/19/24	Tue 8/20/24
36		Backend setup and model deployment	2 days	Wed 8/21/24	Thu 8/22/24
37		**Initial Testing**	**7 days**	**Fri 8/23/24**	**Mon 9/2/24**
38		**Unit Testing**	**3 days**	**Fri 8/23/24**	**Wed 8/28/24**
39		Review modular component (model, added features,	1 day	Fri 8/23/24	Sun 8/25/24
40		Identify anomalies and modify code	1 day	Mon 8/26/24	Mon 8/26/24
41		Re-test modified code	1 day	Tue 8/27/24	Tue 8/27/24
42		Unit testing completed	0 days	Wed 8/28/24	Wed 8/28/24
43		**Integration Testing**	**4 days**	**Wed 8/28/24**	**Mon 9/2/24**
44		Test module integration between prediction and dashboard	2 days	Wed 8/28/24	Thu 8/29/24
45		Identify anomalies and modify code	1 day	Fri 8/30/24	Sat 8/31/24
46		Re-test with modified cor	1 day	Sun 9/1/24	Sun 9/1/24
47		Integration testing compl	0 days	Mon 9/2/24	Mon 9/2/24
48		MVP Demonstration (Mid Evaluation)	5 days	Mon 9/2/24	Sun 9/8/24
49		**Development Iteration 2**	**15 days**	**Mon 9/9/24**	**Sun 9/29/24**
50		**Model Refinement and Validation**	**10 days**	**Mon 9/9/24**	**Sun 9/22/24**
51		Adding more features to the model	5 days	Mon 9/9/24	Sun 9/15/24
52		Tuning and validation the model	4 days	Mon 9/16/24	Thu 9/19/24
53		Re-testing (Unit & Integratic	1 day	Fri 9/20/24	Sun 9/22/24
54		**Deployment and Monitoring**	**5 days**	**Mon 9/23/24**	**Sun 9/29/24**
55		Deploy the solution in a simulated reak-time environment	3 days	Mon 9/23/24	Wed 9/25/24
56		Performance monitoring and handling issues	1 day	Thu 9/26/24	Thu 9/26/24
57		Updated project backlog	1 day	Fri 9/27/24	Sun 9/29/24
58		**Final Testing & Evaluation**	**10 days**	**Mon 9/30/24**	**Thu 10/10/24**
59		SW Testing	2 days	Mon 9/30/24	Tue 10/1/24
60		Evaluation of DS Parts	2 days	Wed 10/2/24	Thu 10/3/24
61		Error Analysis	1 day	Fri 10/4/24	Fri 10/4/24
62		Testing & Evaluation Documen	5 days	Sat 10/5/24	Thu 10/10/24
63		Final Mentor Meeting	0 days	Sat 10/5/24	Sat 10/5/24
64		Finalize the whole implementatio	1 day	Sat 10/5/24	Sun 10/6/24
65		Youtube Video Creation	1 day	Mon 10/7/24	Mon 10/7/24
66		Final Evaluation	5 days	Mon 10/7/24	Sun 10/13/24
67		Final Report Submission	5 days	Mon 10/7/24	Sun 10/13/24

Appendix B: Data Dictionary

The dataset is available for public use on Kaggle and can be accessed here: New York City Bus Data on Kaggle.

This dataset comprises data from NYC MTA buses, capturing real-time bus location, route details, and scheduled versus actual arrival times. Each entry reflects a snapshot of bus status at roughly 10-minute intervals, offering insights into traffic patterns and congestion.

Column Name	Description	Example
RecordedAtTime	The timestamp when the data was recorded.	2017-06-01 00:03:34
DirectionRef	Numeric code representing the bus's direction on the route.	0
PublishedLineName	The official route number or name of the bus line.	B8
OriginName	Name of the starting location of the bus route.	4 AV/95 ST
OriginLat	Latitude of the starting location of the bus route.	40.616104
OriginLong	Longitude of the starting location of the bus route.	-74.031143
DestinationName	Name of the destination location of the bus route.	BROWNSVILLE ROCKAWAY AV
DestinationLat	Latitude of the destination location of the bus route.	40.656048
DestinationLong	Longitude of the destination location of the route.	-73.907379
VehicleRef	Unique identifier for the bus.	NYCT_430
VehicleLocation.Latitude	Real-time latitude location of the bus.	40.63517
VehicleLocation.Longitude	Real-time longitude location of the bus.	-73.960803
NextStopPointName	Name of the next bus stop on the route.	FOSTER AV/E 18 ST
ArrivalProximityText	Proximity indicator of the bus to the next stop (e.g., "approaching," "at stop").	approaching
DistanceFromStop	Distance remaining from the bus to the next stop (in meters).	76
ExpectedArrivalTime	Estimated arrival time at the next stop.	2017-06-01 00:03:59
ScheduledArrivalTime	Scheduled arrival time at the next stop according to the timetable.	24:06:14

This dictionary offers a detailed reference for interpreting the dataset, clarifying each field's significance and typical values for analysis or application use.

Appendix C: User Guide

This User Guide provides an overview of how to effectively use and interact with the system developed in this project. It includes detailed instructions on accessing features, navigating the dashboard, and understanding prediction outputs. The guide is intended to ensure that users can maximize the benefits of the system and utilize its features to improve their transit experience.

Accessing the Application:

Visit the app at Next Bus ETA. The landing page offers an overview with direct links to the main sections: **NextBus** and **Insights**.

Navigating the Header:

At the top, you'll find:

- **Dark/Light Mode Toggle**: Switch between themes for comfortable viewing.
- **Navigation Links**: Quickly access **NextBus** and **Insights** sections.

Using the "NextBus" Section:

This section provides arrival times for buses based on your selected route and stop.

- **Selecting a Destination**: Choose your route and destination stop to view ETA information.
- **Arrival Times**: Color-coded for easy interpretation - **Green** for short wait times or on-time arrivals, **Red** for longer waits or delays.

Exploring the "Insights" Section:

The **Insights** section provides data on bus patterns and tips based on past trends.

- **Trends**: View daily and weekly patterns, helping you choose the best travel times.
- **Recommendations**: Find smart suggestions like "On Sundays, there are NO buses planned," or insights on high-frequency days.

Mobile-Friendly Design:

The app is optimized for mobile, offering a seamless experience across devices.