VIETNAM NATIONAL UNIVERSITY - HO CHI MINH CITY
UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

SPECIALIZED PROJECT

BUILD A MOBILE APP FOR DATA COLLECTION AND URBAN TRAFFIC ESTIMATION

Major: Computer Science

Council: Information System
Supervisors: Assoc. Prof. Tran Minh Quang
Ms. Eng. Bui Tien Duc
Secretary: Dr. Phan Trong Nhan
Students: Nguyen Minh Khoa - 2052538
Nguyen Nho Gia Phuc - 2052214

Ho Chi Minh City, 15th December, 2023

DECLARATION OF AUTHENTICITY

We would like to state that to project "Build a mobile app for data collection and urban traffic estimation "is our own work under the guidance of Assoc. Prof. Tran Minh Quang and MSc. Bui Tien Duc. Related knowledge, information and research works that we refer to will be clearly noted in the references section. We are responsible for the truthfulness of our topic.

Ho Chi Minh, December 2023

Students

Nguyen Nho Gia Phuc

Nguyen Minh Khoa

ACKNOWLEDGEMENT

In creating this undertaking, the guidance of our instructors, Associate Professor Tran Minh Quang and Master of Science Bui Tien Duc was indispensable. For this, we are extremely grateful.

Contents

List of Figures

List of Tables

1 INTRODUCTION

1.1 Motivation

Ho Chi Minh City is a thriving metropolis known for its vibrancy and economic importance, but it also has a persistent problem with traffic congestion. An astounding estimated cost of 6 billion USD per year [1] highlights the crippling effects of traffic congestion on Ho Chi Minh City, indicating not only the financial toll on businesses but also the pressing need to put into practice practical solutions for sustainable urban development. Within this context, the project's driving force is the necessity to improve the current mobile application aimed at facilitating better traffic conditions in Ho Chi Minh City. [2]

Figure 1: Traffic congestion at the Hang Xanh intersection at the end of 2022.

The current measures taken by the government to alleviate traffic congestion in Ho Chi Minh City have mostly failed, despite their concerted efforts. The rapid urbanization and increasing vehicular density have posed challenges for infrastructure projects and traffic management initiatives to keep up with. Moreover, the current deficiencies in the public transportation system have hindered its capacity to function as a viable alternative for a significant portion of the population. The insufficient effectiveness of these measures underscores the intricacy of the challenge, requiring a thorough reassessment and inventive strategy to attain significant and enduring outcomes.

1.2 Available solutions

In response to the pressing issue of traffic congestion in Ho Chi Minh City, various measures have been implemented to alleviate the strain on the urban transportation system. One notable initiative

involves ongoing investments in infrastructure development, with the construction of new roads and the expansion of existing thoroughfares aimed at accommodating the city's growing population and vehicular density. Additionally, the government has introduced and expanded public transportation services, such as buses and the metro system, as a means to encourage citizens to opt for more sustainable modes of commuting.

A different strategy for addressing traffic congestion involves the creation of specialized programs for the purpose of continuously monitoring the situation in real-time. These applications utilize technology to deliver real-time updates on traffic conditions, acknowledging the demand for immediate information. Through the collection and analysis of data from diverse sources, including GPS-enabled devices and surveillance cameras, these applications enable users to make well-informed decisions regarding their travels. An example is the TTGT HCMC application created by the Department of Transportation in Ho Chi Minh City. This application functions as a vital instrument, providing users with instant access to real-time updates on the status of routes and visual information through a network of 685 strategically positioned cameras throughout the city. These innovative technologies enhance the ability to effectively manage traffic congestion in urban environments by employing a dynamic and adaptable approach.

1.3 Problem statement

UTraffic [3] is a city traffic prediction system that relies on data gathered from the local population developed by Professor Tran Minh Quang, alongside his associates and students at Ho Chi Minh University of Technology. UTraffic's products aim to mitigate traffic congestion and enhance safety and convenience for users. Using big data analytics and machine learning, the solutions estimate and forecast traffic conditions accurately and quickly, warn users of traffic jams, support efficient routing that takes traffic conditions into account, and provide statistical information and forecasts to help managers make decisions. However, despite UTraffic's commendable objectives and the overarching focus on the mobile application and backend technologies, several challenges have surfaced in the system's current implementation that require careful consideration and resolution:

- In addition to impeding thorough data collecting and insights into traffic patterns in Ho Chi Minh City, the lack of an iOS version restricts our ability to interact with the sizable iPhone user base.
- In addition to relying on crowd-sourced data for efficient routing, our existing approaches to traffic congestion are not sufficiently diverse.

1.4 Scope and objectives

The UTraffic app and backend, developed by previous teams, serve as the basis for our efforts to enhance and innovate Ho Chi Minh City's traffic circumstances. The objective of the project is to expand the number of users by creating an iOS version of the current Android application. The objective of this iOS expansion is to enhance the accessibility of the UTraffic application, thereby encouraging a broader range of users. Objectively, the introduction of supplementary bus-related services is intended to encourage the greater use of public transportation. These services will provide users with up-to-date information about bus routes, schedules, and other information, improving the attractiveness of sustainable commuting choices. At the same time, with the emergence of research groups with similar interest in the traffic field, making the UTraffic mobile app in particular and the whole UTraffic system

in general a place to showcase and deliver those new features will be one of our focus points. The project aims to make a substantial contribution to the improvement of urban mobility in Ho Chi Minh City through these collaborative efforts.

2 RELATED RESEARCH

2.1 Bus-Related Services

Previous groups' research has investigated traffic-related value-added services, such as nearby hospitals and automatic teller machines (cite that group here). When it comes to our project, one major candidate for us to continue this research direction is the addition of buses and public transport as a whole. Our research investigates a range of existing applications that provide services related to buses. Through a thorough analysis of established solutions within this field, our objective is to acquire insights that will guide the improvement of our application by finding a compromise between the current system's strengths and weaknesses.

The existing solutions in question are the package offered by FPT in the names of Go!Bus TPHCM and the Buyt TPHCM website, as well as BusMap by Phenikaa MaaS JSC.

2.1.1 The FPT Ecosystem

Besides Go!Bus TPHCM, FPT offers a wide set of choices for citizens to track traffic data, and these services are linked together. Most notably, the TTGT Tp Ho Chi Minh app allows users to see public cameras for traffic. We place particular emphasis on Go!Bus as it aligns with our project's focus, and Buyt TPHCM holds significance for data collection, as detailed in subsequent sections. These applications play pivotal roles in informing our research and development efforts.

2.1.1.1 Go!Bus Fundamentally, Go!Bus offers solely bus-related services like bus stop searching, bus route searching, and routing. The app also has some support for Grab when routing. The home screen for the app is as follows:

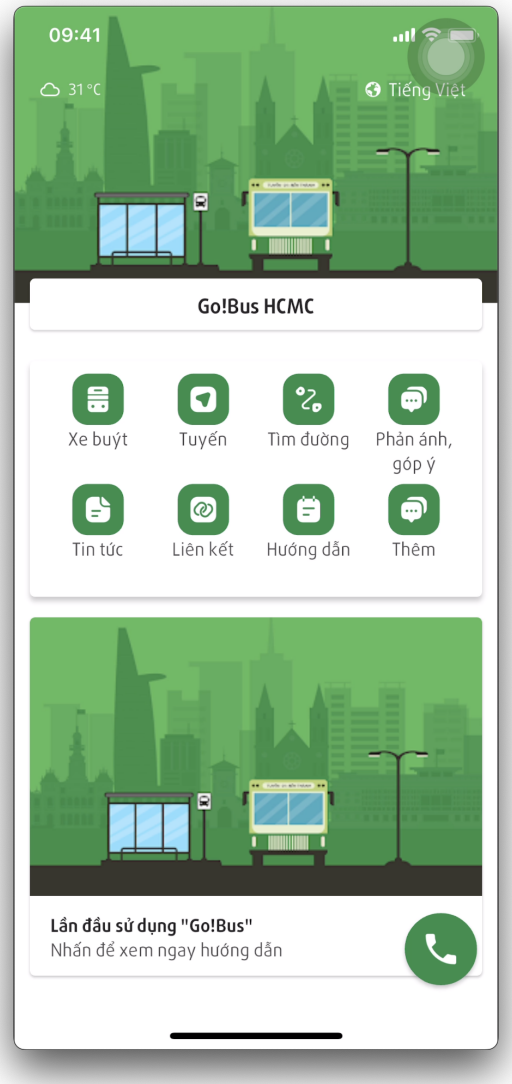
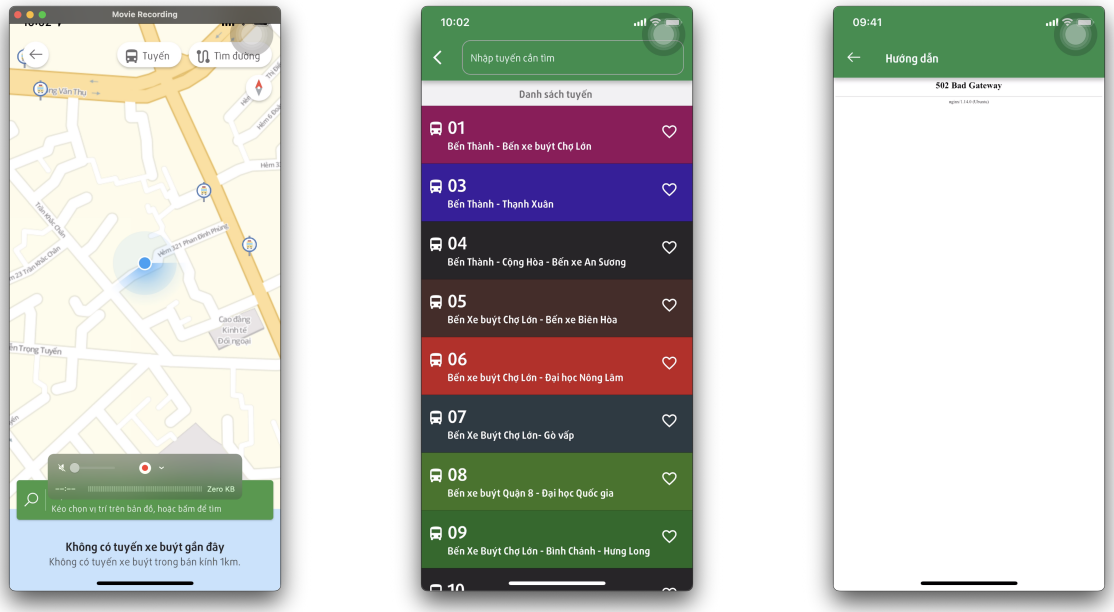


Figure 2: The home screen for Go!Bus.

Nevertheless, during our testing of these features, none of them functioned, or had inconsistent uptimes. Here are a few examples of these occurrences.



(a) No data for nearby routes.

(b) The stops do not show anything when pressed.

(c) Embedded web view with server error.

Figure 3: Go!Bus functionalities problems

It also portrayed some mobile app bad practices, such as embedding an excessive number of web views into the user interface. Apple, in particular, is not fond of this approach and publicly stated that apps using this will not be accepted in the future. This is one of the points that we will take into consideration when developing our iOS app.

2.1.1.2 Buyt TPHCM website Buyt TPHCM website is an online resource designed to provide comprehensive information and tools for navigating the public bus system in Ho Chi Minh City. This website aims to enhance the accessibility and user experience of public transportation within the city by offering the following key functionalities:

- **Detailed route information:** The website provides users with comprehensive information on all bus routes in Ho Chi Minh City, including individual stop locations, schedules, and route maps.
- **Real-time bus arrival predictions:** To promote efficient travel planning, buyttphcm.com.vn offers real-time bus arrival predictions for all routes, allowing users to make informed decisions about their travel time.
- **Convenient tools and resources:** This website provides access to the Go!Bus mobile application, enabling users to easily purchase tickets and plan their bus journeys on the go. Additionally, users can find helpful information about the Public Transport Management Center and other relevant resources.
- **Up-to-date news and information:** buyttphcm.com.vn keeps users informed of the latest developments in the Ho Chi Minh City bus system, including temporary route changes, service disruptions, and new initiatives.

With its regularly updated resources, Buyt TP HCM will play a crucial role in our data collection process for bus stops and routes in the later stages of our project.

2.1.2 BusMap

BusMap rose as a prime solution for city navigation using buses. It also expanded into other regions and countries like Thailand and is overall doing well.

The fundamental bus services like searching and routing are executed well on BusMap, bar some UI lag that could be due to the excessive inclusion of map views. Users can see the bus locations that are coming their way with an estimated time of arrival. There are also other services offered by BusMap, such as JobMap, MotelMap, and driver license support. However, sections like "Blogs" and "News" might be unnecessary for most users.

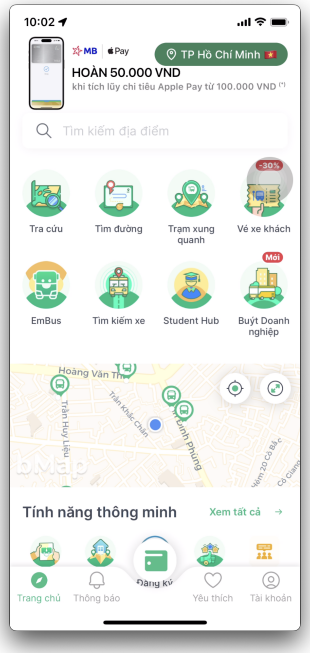


Figure 4: The home screen for BusMap

2.1.3 Conclusion

Both of these apps share a common feature of being equipped with an abundance of supplementary services. While Go!Bus TPHCM is compatible with Grab, BusMap has diversified its partnerships to include other businesses. One apparent limitation of UTraffic is the lack of access to real-time data, such as the current locations of buses and the duration until the next journey. Nevertheless, with sufficient effort and development, our application can excel as a dependable routing and search engine. Ultimately, the primary function of UTraffic is to process congestion data; therefore, we must be careful not to delve too thoroughly into other domains.

2.2 UTraffic's system analysis and design

2.2.1 Current code base structure

2.2.2 Implemented APIs

During our research to improve the performance and coherence of the application, we have developed new API endpoints. They are tailored to work serve two major purposes when it comes to buses, which are the bus routes and the bus stops. The endpoints' description are as follows:

1. API endpoints related to bus stops:
    - **Get nearby bus stops**: This endpoint takes in a latitude value, a longitude value, and a radius value in meters. It then returns a list of bus stops that are within the radius of the given location and in the form of GeoJSON.
    - **Get nearby disability friendly bus stops**: This endpoint takes the same query paramaters like the nearby bus stops endpoint, but it returns a list of disability friendly bus stops instead.
    - **Get the details of a bus stop**: This endpoint takes in a bus stop ID and returns the details of that bus stop.
2. API endpoints related to bus routes:
    - **Get all bus routes**: This endpoint simply returns all the bus routes in the database to the requesting client. This helps in building a list view for browsing purposes.
    - **Get the details of a bus route**: This endpoint takes in a bus route ID and returns the details of that bus route.
    - **Get the bus stops along a bus route's path**: This endpoint takes in a bus route ID and the leg, either **forward** or **return**. The endpoint then returns the bus stops along that bus route's path, preserving the order of the stops. The response is in the form of GeoJSON.
    - **Get the path of a bus route**: This endpoint takes in a bus route ID, and the desired leg and then returns the path of that bus route in the form of GeoJSON.

The details of the API endpoints as well as the deployment efforts that we have done are discussed further in Section **??**.

We also made changes to the currently deployed API endpoints. At the moment, the endpoint `/traffic-status/get-status-v2` receives the query paramaters which are the northeast and southwest bounds of the client's visible map and the zoom level in order to return the appropriate street segments. This endpoint is currently used by the Android app as well as the web version of UTraffic. We have implemented another endpoint, `/traffic-status/get-status-v3`, to handle the same functionality but with GeoJSON as the response. This is done by following the GeoJSON's standard format of being a FeatureCollection of Features, where each Feature is a Point with the coordinates of the traffic status. The following is the comparison between the two format when taking a node as an example:

```
{
  "_id": 62391,
  "polyline": {
    "coordinates": [
      [
        106.6509813,
        10.7442391
      ],
      [
        106.6514738,
        10.7445197
      ]
    ],
    "_id": "657a650dbf0f55d99d183678",
    "type": "LineString"
  },
  "priority": 2,
  "segment": 62391,
  "source": "base-data",
  "velocity": 30,
  "color": "#ffff00",
  "street": {
    "name": "Võ Văn Kiệt",
    "type": "trunk"
  }
}
```

(a) The current response format.

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "type": "Feature",
      "geometry": {
        "type": "LineString",
        "coordinates": [
          [
            106.6509813,
            10.7442391
          ],
          [
            106.6514738,
            10.7445197
          ]
        ]
      },
      "properties": {
        "_id": 62391,
        "priority": 2,
        "segment": 62391,
        "source": "base-data",
        "velocity": 30,
        "color": "#ffff00",
        "street": {
          "name": "Võ Văn Kiệt",
          "type": "trunk"
        }
      }
    }
  ]
}
```

(b) The GeoJSON response format.

Figure 5: Comparison between the current response format and the GeoJSON format.

GeoJSON helps tremendously when implementing for the iOS app, as it is a standard format that is supported by Apple's MapKit [4]. This means that we can easily convert the response into a list of `MKAnnotation` and display them on the map without having to parse the original response to dedicated classes, saving development effort and adhering to Apple's best practices.

3 THEORETICAL BASIS

3.1 Model

3.1.1 Model-View-Controller (MVC)

MVC is an architectural design pattern for building applications that separates the application into three main logical components: the model, the view, and the controller. In the MVC model:

- **Model**: Represents the data and business logic of the application. It is the "unchanging essence" of the application and should be stable and long-lived.
- **View**: Represents the user interface of the application. There can be multiple views for different devices, such as a web interface, mobile app, or command line interface.
- **Controller**: Handles user input and updates the model and view accordingly. It acts as an intermediary between the model and the view, processing business logic and managing data flow.

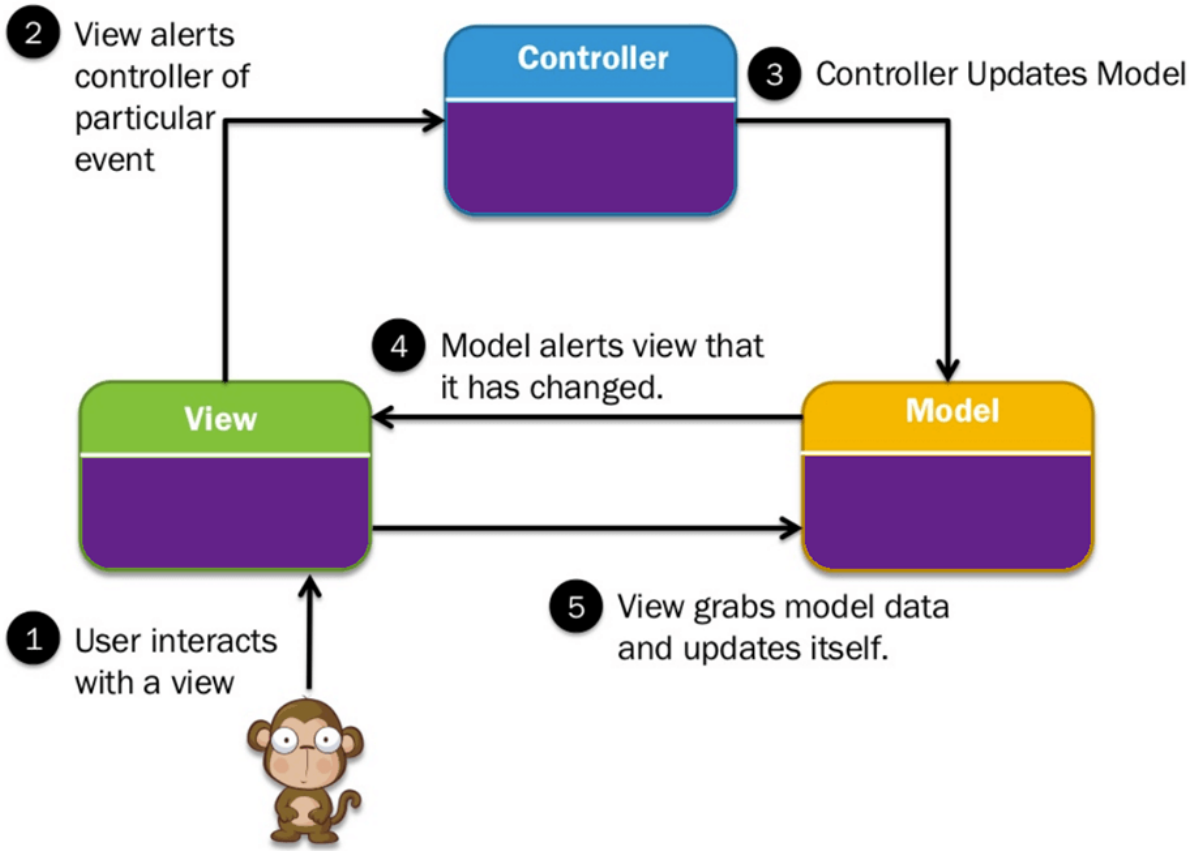


Figure 6: MVC architecture

3.1.2 Model-View-ViewModel (MVVM)

MVVM architecture offers two-way data binding between view and view-model. The view-model makes use of observer pattern to make changes in the view-model. The components of MVVM are:

- **Model**: Holds the data and business logic. The model is independant of UI or user interaction.
- **View**: Represents how the user interacts with the application.

- **View-model**: Acts as a bridge between the view and the model. It provides data from the model to the view and handles user interaction with the view.

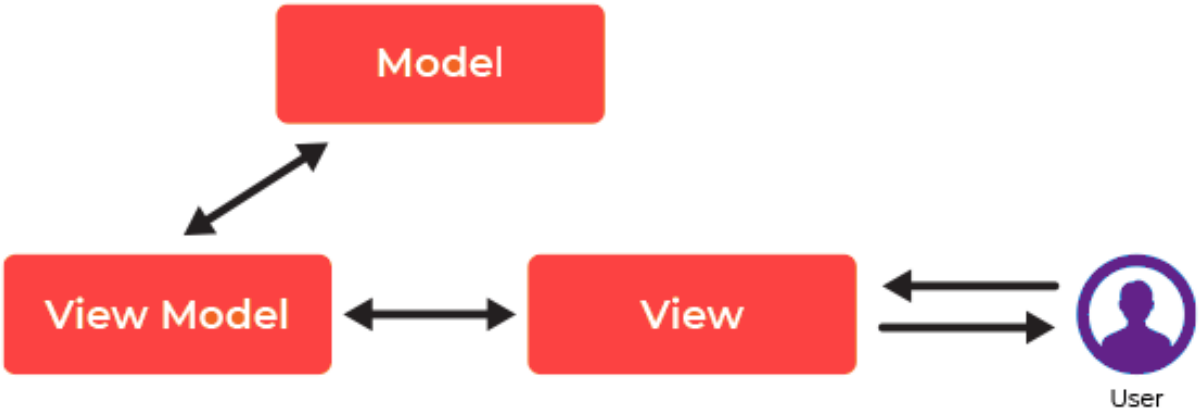


Figure 7: MVVM architecture

3.1.3 Conclusion

	Advantage	Disadvantage
MVC	• The development of the various components can be performed parallelly. • It avoids complexity by dividing an application into separate (MVC) units • It only uses a front controller pattern that processes web application requests using a single controller. • It provides a clean separation of concerns.	• Business logic is mixed with Ul • Hard to reuse and implement tests • There is a need for multiple programmers to conduct parallel programming.
MVVC	• Business logic is decoupled from Ul. • Easy to reuse components. • You can write unit test cases for both the viewmodel and Model layer without the need to reference the View.	• Maintenance of lots of codes in controller • Some people think that for simple UIs of MVVM architecture can be overkill

Table 1: Comparison between MVC and MVVM

Base on the comparison above, MVVM is more suitable for this project because it is easier to reuse components and write unit test cases.

3.2 Frontend

3.2.1 Objective-C

Objective-C currently ranks 22nd in popularity according to the TIOBE Index. While its use is declining, it remains significant in the Apple ecosystem, with around 60,000 active repositories. Introduced in the 1980s by Brad Cox and Tom Love, Objective-C is an object-oriented programming language primarily used for Apple platforms. It was licensed by NeXT Computer Inc., a company founded by Steve Jobs. Noteworthy features include classes, inheritance, dynamic binding, and a Smalltalk-like syntax for messaging objects. Despite declining usage, Objective-C remains one of the more popular programming languages in the market

Some of the great features of this language are:

- **Object-Oriented**: Supports classes, objects, and object-oriented concepts.
- **Dynamic Binding**: Resolves method calls at runtime for flexibility.
- **Message Passing**: Utilizes message passing for a natural coding style.
- **Smalltalk-style Syntax**: Simple and expressive syntax like Smalltalk for clear code.
- **Cross-Platform**: Works on macOS, iOS, and other Apple platforms, plus Windows and Linux.
- **Interoperability with C**: Integrates seamlessly with C libraries for collaboration.

3.2.2 Swift

Swift is a powerful and modern programming language developed by Apple for building applications across its ecosystem, including iOS, macOS, watchOS, and tvOS. Introduced in 2014, Swift was designed to be efficient, expressive, and easy to learn, addressing the shortcomings of its predecessor, Objective-C. With a syntax that is both concise and readable, Swift empowers developers to create robust and high-performance applications. Known for its safety features, dynamic capabilities, and versatility, Swift has quickly become the language of choice for many developers seeking to craft innovative and seamless experiences within the Apple ecosystem. In this introduction, we'll explore the key features and characteristics that make Swift a standout language in the world of software development.

3.2.3 SwiftUI

In summary, SwiftUI, developed by Apple, has become increasingly popular among iOS app developers for valid reasons. Introduced five years after Swift alongside Swift 5 and Xcode 11, it serves as a UI toolkit designed for creating software across various platforms, including iOS, macOS, watchOS, and tvOS. SwiftUI offers a declarative approach to UI design, enabling developers to describe layout and behavior using a simple and intuitive syntax. This results in more efficient and faster development of complex UIs, thanks to concise and easily readable code compared to traditional imperative approaches.

In summary, SwiftUI stands out with these key features:

- **Declarative Syntax**: Describes UI appearance easily, improving code readability.
- **Automatic Layout**: Handles UI layout automatically, reducing manual effort.
- **Dynamic UI**: Enables the creation of dynamic and interactive interfaces.

- **Cross-platform Support**: Works across all Apple devices with a single codebase.
- **Real-time Preview**: Provides live previews of UI changes during development.
- **Pre-built Components**: Offers ready-to-use UI components for quick and efficient development.

3.2.4 Conclusion

Feature	Objective C	Swift
Age	Developed in the early 1980s	Introduced in 2014
Syntax	Uses C-based syntax with Smalltalk-style messaging	Uses a modern and concise syntax
Performance	Slower than Swift due to overhead and lack of optimization	Faster than Objective-C due to optimization features
Memory Managment	Prone to memory leaks	No memory leaks due to being type-safe and memory-safe
Stability	Stable	Unstable as it is still growing

Table 2: Comparison between Objective C and Swift

Base on the comparison above, Swift is more suitable for this project because it is superior in multiple aspects when comparing to Objective C.

3.3 Backend

3.3.1 Java

3.3.2 Python

3.3.3 NodeJS

3.3.4 Conclusion

3.4 Database

3.4.1 MySQL

MySQL is a widely used database for web-based applications, offered as freeware with regular upgrades to improve capabilities and security. The freeware edition emphasizes efficiency and dependability over a comprehensive range of functionalities. Prominent characteristics encompass the ability to select storage engines, a user-friendly interface, and effective handling of extensive datasets in batches, all while maintaining modest resource use.

	Advantage	Disadvantage
,MySQL	• MySQL is accessible for free, making it cost-effective for various applications. • Multiple user interfaces can be implemented, enhancing usability. • Supports both structured data (SQL) and semi-structured data (JSON).	• The process of setting up MySQL for specific activities may necessitate a greater investment of time and effort in comparison to systems that automate certain functions, such as incremental backups. • While support is available for the free version, premium support may incur additional costs.

Table 3: Advantages and disadvantages of MySQL

3.4.2 MongoDB

MongoDB is designed for applications that handle both structured and unstructured data, and it is offered in both free and paid versions. The highly adaptable database engine establishes connections between databases and applications using MongoDB database drivers, providing a wide array of options that are compatible with multiple programming languages. Although MongoDB may encounter performance challenges when heavily utilized for relational data models, it demonstrates exceptional capabilities in managing changeable, non-relational data. Successive iterations consistently incorporate functionalities to cater to various applications, bolster operational robustness, and prioritize the protection of data.

	Advantage	Disadvantage
MongoDB	• MongoDB is fast and easy to use. • The engine supports JSON and other NoSQL document formats. • Capable of effectively storing and retrieving data of any arrangement without rigid schema prerequisites. • Allows the modification of schema without downtime, facilitating flexibility.	• Tools translating SQL to MongoDB queries add an extra step in using the engine. • The default settings may lack sufficient security measures, necessitating further tweaking to provide optimal security.

Table 4: Advantages and disadvantages of MongoDB

3.4.3 Oracle

Oracle is a prominent and long-lasting database management solution that has been in use since the late 1970s. The different versions of the software are designed to meet the specific requirements of different organizations. The current long-term release focuses on providing comprehensive support and ensuring reliability. The most recent update incorporates cutting-edge functionalities like as autonomic management, AutoML, and improved multi-model compatibility, hence strengthening its attractiveness for future use.

	Advantage	Disadvantage
Oracle	• Oracle consistently deliver cutting-edge innovations. • Oracle's tools are highly robust, offering a wide range of capabilities to meet diverse requirements. • Supports various data models, including semistructured (JSON, XML), spatial, RDF, and structured data (SQL). • Provides multiple access patterns based on the data model.	• The cost of Oracle can be prohibitive, particularly for smaller organizations. • The installation process may need a substantial amount of resources, which could result in the need for hardware upgrades.

Table 5: Advantages and disadvantages of Oracle

3.4.4 PostgreSQL

To summarize, PostgreSQL is a popular and freely available database system that is extensively utilized for web-based databases. Being one of the pioneering database management systems, it supports both organized and unstructured data and is compatible with multiple platforms, including Linux. The most recent update incorporates improvements in compression choices, backing for organized server log output in JSON format, and general benefits in performance.

	Advantage	Disadvantage
PostgreSQL	• PostgreSQL has excellent scalability, enabling efficient management of terabytes of data. • The system is capable of handling JSON data format, which allows for versatile data administration. • Offers a range of predefined functions for diverse database operations. • Multiple interfaces are available for user interaction. • Functions as a multi-model database, supporting Spatial Data, Key-Value, Structured Data (SQL), and Semi-Structured Data (JSON, XML).	• The documentation may be incomplete or challenging to navigate • Speed may be affected during extensive bulk operations or read queries involving large datasets.

Table 6: Advantages and disadvantages of PostgreSQL

3.4.5 Conclusion

After analyzing the advantages and disadvantages mentioned, it is clear that each of the four databases has unique strengths and limitations that are suited to certain project needs. When it comes to situations where the most important factors are scalability, flexibility, and excellent performance, MongoDB is the ideal choice. MongoDB is widely recognized for its remarkable capacity to handle large amounts of data and heavy website traffic. MongoDB's document-oriented architecture is specifically

tailored for unstructured data and complex data structures. This design provides efficient access and retrieval of data, resulting in improved overall performance.

4 IMPLEMENTATION

4.1 Bus Services Integration

4.1.1 Getting a single route's information

The integrity of the data on the Buyt TPHCM website is guaranteed due to its commissioning and maintenance by the Ho Chi Minh City Department of Transport.

Several JavaScript source files power this site:

- **L.Config.js:** This is a configuration file to set up the map rendering for Buyt TPHCM. The tiles are loaded from http://map.stis.vn/bright/z/x/y.png.
- **L.RouteMap.js:** This source file is responsible for showing the stops and bus routes. We were able to get the coordinate sets that we wanted by placing breakpoints on this file.

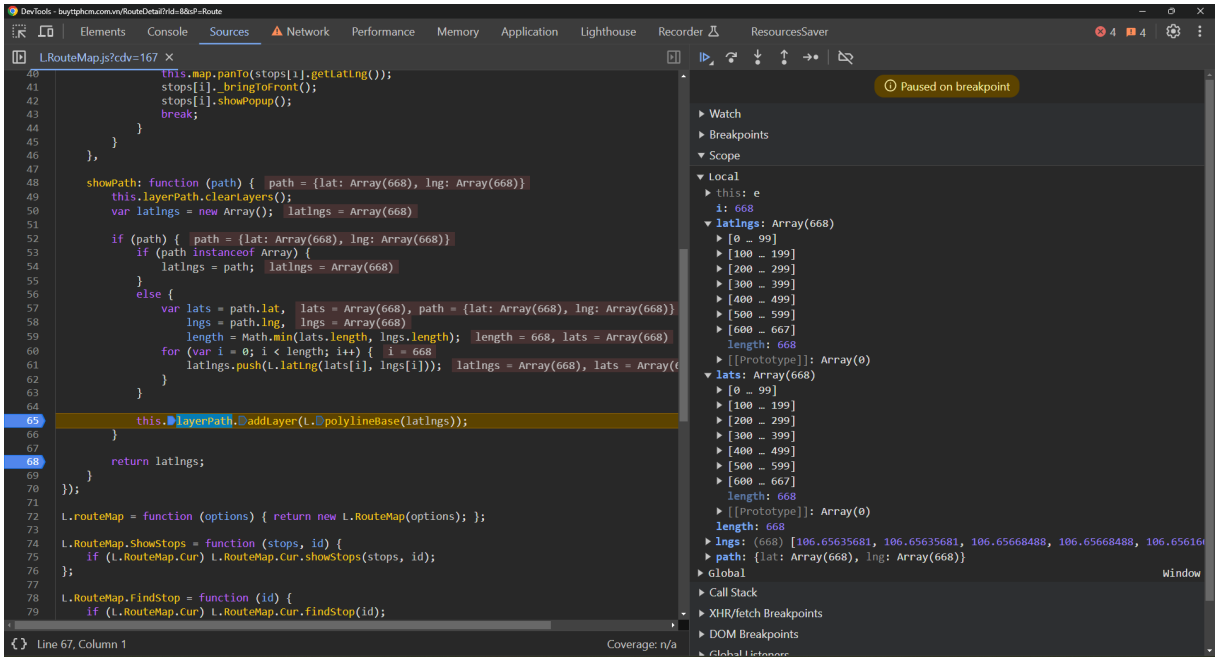


Figure 8: Capturing and observing the route data using the DevTool

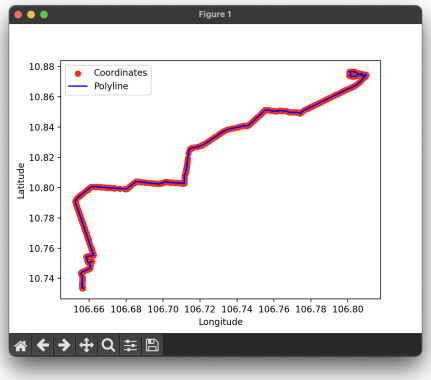
By inserting breakpoints in L.RouteMap.js, we can easily obtain the coordinates of the route, as demonstrated on the right-hand side. It is worth mentioning that while the coordinates are given in both distinct arrays and a single array with integrated data, it is more advantageous to utilize the latitudes and longitudes in separate arrays. The reason for this is that the present system implementation utilizes MongoDB, which does not allow tuples due to its implementation of BSON.

When it is desirable to work with a single array of coordinates, the stored data can be retrieved and combined together. Below is an example where we zip the arrays and plot them using Matplotlib (please note that since there are over 600 elements in each array, 668 to be exact, a figurative number was put in the code excerpt)

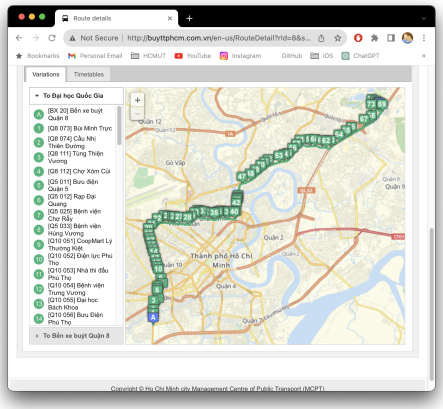
```
lat = [10.001, 10.002] # the latitudes
lng = [106.001, 106.002] # the longitudes
coordinates = [(lat[i], lng[i]) for i in range(min(len(lat), len(lng)))]
```

```
    # Extract latitudes and longitudes from the coordinates
    latitudes, longitudes = zip(*coordinates)

    # Create a scatter plot to mark the coordinates
    plt.scatter(longitudes, latitudes, color='red', marker='o', label='
    Coordinates')

    # Create a line plot to draw the polyline
    plt.plot(longitudes, latitudes, color='blue', label='Polyline')

    # Set labels for the x and y axes
    plt.xlabel('Longitude')
    plt.ylabel('Latitude')

    # Add a legend
    plt.legend()

    # Display the plot
    plt.show()
```

The result is as follow:



(a) The Matplotlib representation of the bus route 8.



(b) The actual route 8 from Buyt TPHCM.

Similarly, information about the bus stops can be obtained by placing breakpoints in L.RouteMap.js. Each route comes with a list of stops and their corresponding data. This will be useful in designing our data schema later on when we integrate the gathered information into our database.

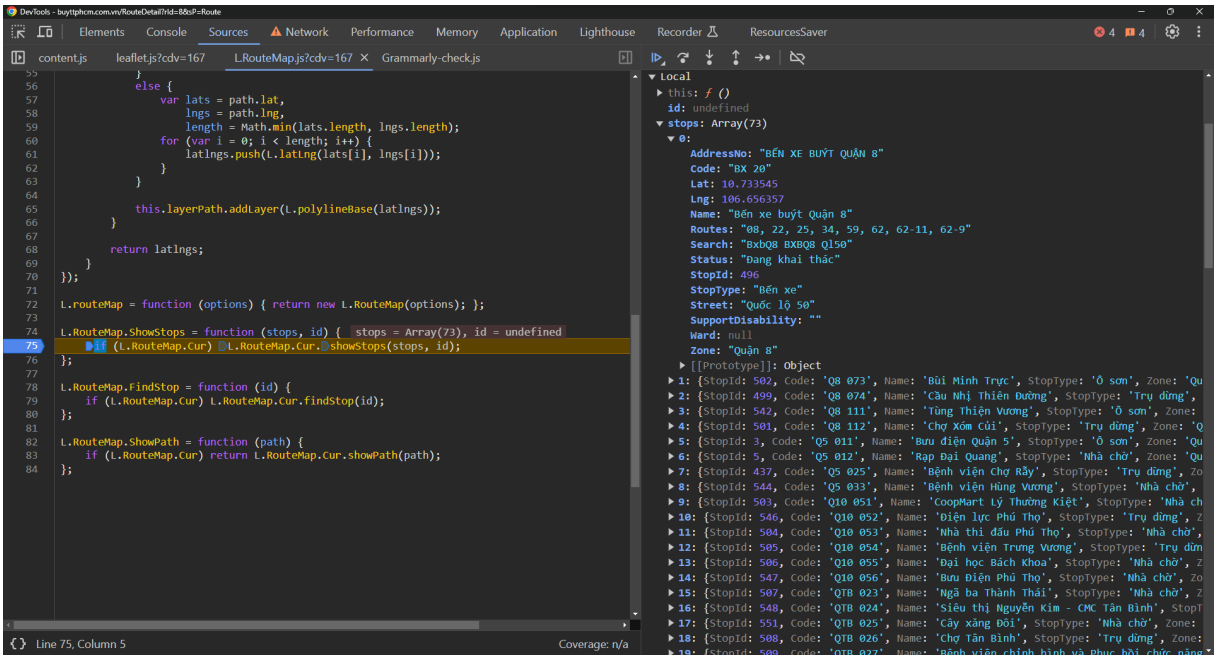


Figure 10: Capturing and observing the bus stops data using the DevTool

4.1.2 Automation of the mining process

Since there are about 130 routes currently displayed on the website, it is impossible to place breakpoints and collect data by hand. This necessitates the use of automatic scripts to gather information.

After careful examination, we found several places where we could mine the data. This introduces us to the routevar.js file, which contains the three functions: loadVarsByRou(rouId), loadStopsByVar(prtId, rouId, varId), and loadPathByVar(prtId, rouId, varId)

4.1.3 Data storage of bus services

4.2 Bus API Development and Server Deployment

We have previously discussed the introduction of the API additions in the UTraffic system, and we have taken the steps to deploy them on the live server of UTraffic. The endpoints are now live and working as intended.

4.2.1 The Bus API

The following are the details of the new API endpoints. An example URL and the corresponding response are also provided for each endpoint.

- **Get nearby bus stops**.
  - URL: https://api.bktraffic.com/api/bus/nearby-stops.
  - Query parameters: `lat`, `lng`, `radius`.
  - Example request URL: https://api.bktraffic.com/api/bus/nearby-stops?lat=10.7721&lng=106.6579&radius=500

- Example response:


```
{
    "code": 200,
    "message": "success",
    "data": {
        "type": "FeatureCollection",
        "features": [
            {
                "type": "Feature",
                "geometry": {
                    "type": "Point",
                    "coordinates": [
                        106.657698,
                        10.772603
                    ]
                },
                "properties": {
                    "name": "  i    h c  B ch Khoa",
                    "type": "Nh   c h "
                }
            },
            {
                "type": "Feature",
                "geometry": {
                    "type": "Point",
                    "coordinates": [
                        106.657829,
                        10.771331
                    ]
                },
                "properties": {
                    "name": "  i    H c  B ch Khoa(
c ng  t r  c )",
                    "type": " T r   d ng "
                }
            },
            // ...
        ]
    }
}

```


- **Get nearby disability friendly bus stops**.
  - URL: https://api.bktraffic.com/api/bus/disability-friendly-stops.
  - Query parameters: `lat`, `lng`.
  - Example request URL: http://api.bktraffic.com/api/bus/disability-friendly-stops?lat=10.77037&lng=106.69868
  - Example response:


```
{
    "code": 200,
    "message": "success",
```

```
                "data": [
                    {
                        "_id": 1344,
                        "address_no": "65G (75)",
                        "code": "Q1 119",
                        "lat": 10.766785,
                        "lng": 106.696011,
                        "name": " C u   ng  L nh",
                        "routes": "139, 140, 31, 46, 72",
                        "status": " ang  khai  th  c",
                        "stop_type": "Nh   c h  ",
                        "street": " N g u y n  Th  i  H  c ",
                        "support_disability": true,
                        "ward": " P h   ng  C u   ng  L nh",
                        "zone": " Q u n  1",
                        "location": {
                            "type": "Point",
                            "coordinates": [
                                106.696011,
                                10.766785
                            ]
                        }
                    },
                    {
                        "_id": 7276,
                        "address_no": "277-279-275Y",
                        "code": "Q1 190",
                        "lat": 10.76767,
                        "lng": 106.690941,
                        "name": "T  n  T h t  T  ng",
                        "routes": "03, 04, 102, 109, 140, 18, 19,
    20, 28, 34, 36, 39, 46, 52, 65, 69, 72, 75, 88, 93, D4",
                        "status": " ang  khai  th  c",
                        "stop_type": " T r   d ng ",
                        "street": " P h m  N g  L  o",
                        "support_disability": true,
                        "ward": " P h   ng  P h m  N g  L  o",
                        "zone": " Q u n  1",
                        "location": {
                            "type": "Point",
                            "coordinates": [
                                106.690941,
                                10.76767
                            ]
                        }
                    },
                    // ...
                ]
            }

```


- **Get the details of a bus stop**.
  - URL: https://api.bktraffic.com/api/bus/stop-details.

- Query parameters: `id`.
- Example request URL: https://api.bktraffic.com/api/bus/stop-details?id=2931.
- Example response:


```
            {
                "code": 200,
                "message": "success",
                "data": {
                    "_id": 2931,
                    "address_no": "432",
                    "code": "QTP 117",
                    "lat": 10.794048,
                    "lng": 106.628799,
                    "name": "T  n  S  n  Nh  ",
                    "routes": "41",
                    "status": " ang  khai  th  c",
                    "stop_type": " T r    d ng ",
                    "street": "T  n  S  n  Nh  ",
                    "support_disability": false,
                    "ward": " P h   ng  T  n  S  n  Nh  ",
                    "zone": " Q u n  T  n  Ph  ",
                    "location": {
                        "type": "Point",
                        "coordinates": [
                            106.628799,
                            10.794048
                        ]
                    }
                }
            }

```


- **Get all bus routes**.
  - URL: https://api.bktraffic.com/api/bus/routes.
  - Query paramaters: none.
  - Example request URL: https://api.bktraffic.com/api/bus/routes.
  - Example response:


```
            {
                "code": 200,
                "message": "success",
                "data": [
                    {
                        "_id": "01",
                        "route_name": " B n  Th nh -  B n  xe bu t
    C h   L n "
                    },
                    {
                        "_id": "03",
                        "route_name": " B n  Th nh -  T h nh  Xu n"
                    },
                    {
```

```
                    "_id": "04",
                    "route_name": " B n  Th nh -  C ng  H a -
    B n  xe An S  ng"
                },
                // ...
        }

```

- **Get the details of a bus route**.
  - URL: https://api.bktraffic.com/api/bus/route-details.
  - Query parameters: `id`.
  - Example request URL: http://api.bktraffic.com/api/bus/route-info?id=08
  - Example response:

```
        {
            "code": 200,
            "message": "success",
            "data": {
                "_id": "08",
                "route_name": " B n  xe bu t  Q u n  8 -  i
    h c  Q u c  gia",
                "agencies": " H p  t c  x  v n  t i  xe bu t
     Q u y t  T h ng ,  T : (028)38.642.712",
                "route_type": " P h  th ng - C  t r  gi ",
                "distance": "32.70 km",
                "vehicle_type": "68 seats",
                "operation_time": "Operation time: 04:40 -
    20:30",
                "fare_price": "- V  l  t  t r  gi : 7000
    VN \n     - V  l  t  t r  gi  HSSV: 3000 VN \n     -
    V  t p : 157500 VN ",
                "total_trips": "348 Trip/day",
                "trip_time": "80 - 90 minutes",
                "trip_spacing": "3 - 10 minutes"
            }
        }

```

- **Get the bus stops along a bus route's path**.
  - URL: https://api.bktraffic.com/api/bus/route-stops.
  - Query parameters: `id`, `leg`.
  - Example request URL: https://api.bktraffic.com/api/bus/route-stops?id=08&leg=forward.
  - Example response:

```
        {
            "code": 200,
            "message": "success",
            "data": {
                "type": "FeatureCollection",
                "features": [
```

```
                    {
                        "type": "Feature",
                        "geometry": {
                            "type": "Point",
                            "coordinates": [
                                106.656357,
                                10.733545
                            ]
                        },
                        "properties": {
                            "name": " B n  xe bu t  Q u n  8",
                            "stopType": " B n  xe"
                        }
                    },
                    {
                        "type": "Feature",
                        "geometry": {
                            "type": "Point",
                            "coordinates": [
                                106.656346,
                                10.73646
                            ]
                        },
                        "properties": {
                            "name": "B  i  Minh  T r c ",
                            "stopType": "   s  n"
                        }
                    },
                    // ...
                ]
            }
        }
```


- **Get the path of a bus route**.
  - URL: https://api.bktraffic.com/api/bus/route-path.
  - Query paramaters: `id`, `leg`.
  - Example request URL: https://api.bktraffic.com/api/bus/route-path?id=08&leg=forward.
  - Example response:


```
        {
            "code": 200,
            "message": "success",
            "data": {
                "type": "Feature",
                "geometry": {
                    "type": "LineString",
                    "coordinates": [
                        [
                            106.65635681,
                            10.7335453
                        ],
```

```
                            [
                                106.65635681,
                                10.7335453
                            ],
                            [
                                106.65668488,
                                10.73355579
                            ],
                            // ...
                        ]
                    }
                }
            }
```

4.2.2 Deployment of the Bus API

The deployment process was made possible first by accessing the server via the SSH protocol. The server is running Ubuntu 20.04 LTS with the NodeJS version 12.22.12 installed. It was crucial to first upgrade this NodeJS version to a more recent one due to this version being out of the active support. We have upgraded the NodeJS version to version 16.16.0 (LTS) instead of the newer version 18 or above, because the server hardware is also not very powerful, and the newer versions of NodeJS are not compatible. We also noticed that there was no Node Version Manager (NVM) installed on the server. This is a tool that allows us to install and manage multiple versions of NodeJS on the same machine, therefore, we installed the new node version with NVM.

The next step was to upgrade PM2. PM2 is a process manager for NodeJS applications. It allows us to keep the application running in the background, restart the application automatically when it crashes, and monitor the application's resource usage. We have upgraded PM2 to accompany the new NodeJS version successfully.

We use GitLab to host our source code. After the code was ready on the GitLab repository, we issued a `git pull` on the server's terminal to pull the latest code to the correct directory. We then installed the dependencies using `npm install` and restarted the process with `pm2 restart BKTrafficApi`. Figure **??** shows the running server process.

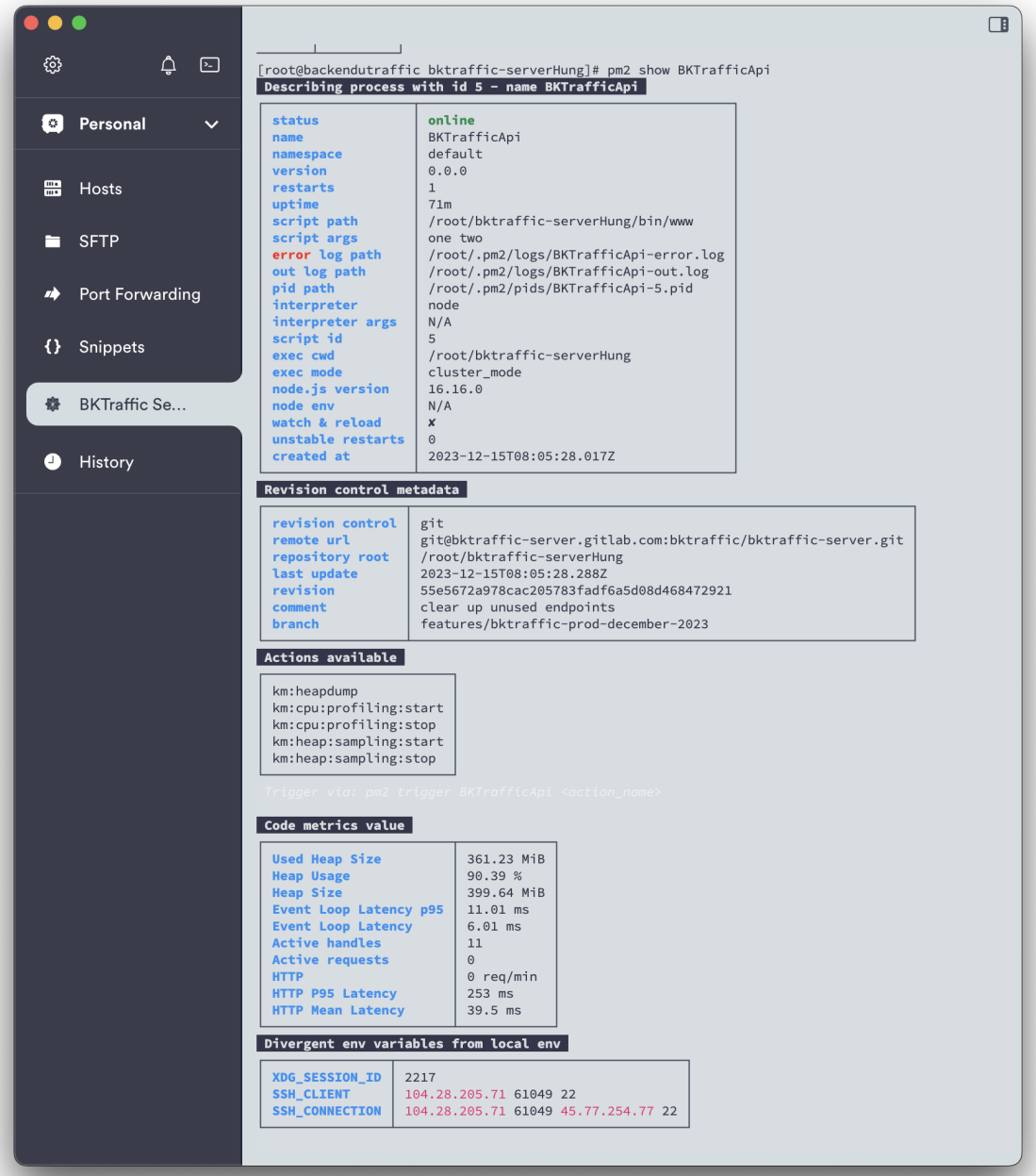


Figure 11: The server information and the running NodeJS process.

4.3 iOS App Development

The iOS app is developed using Swift 5.9 and Xcode 15. The app is compatible with iOS 16.0 and above.

First and foremost, this app must at least offer the basic functionalities like that of the Android counterpart. This includes authentication, viewing the traffic status report, and the ability to communicate with the server. Other requirements are that the app must also be lightweight, performant, and

consistent with the ongoing design of the Android app to an extent. During the development of this prototype, we have also taken into consideration the possibility of future expansion, such as adding more features and improving the user experience, as well as following Apple's best practices when it comes to iOS app development.

4.3.1 The landing page and permission requests

Because the UTraffic app must know the user's real-time location in order to gather GPS data and also display nearby traffic status, it is crucial that we ask the users for location permissions upon the first app launch. This is done by presenting a dialog box that asks for the user's permission to access their location. The choice is saved into `UserDefaults`, so that this process will not be repeated on subsequent app launches.

4.3.2 Account registration and login

We adapt most of the user interface from the Android app to the iOS app in order to provide a consistent experience of the app across platforms.

4.3.3 Viewing the traffic status report

4.3.4 Integration with other groups' research projects

5 WORK EVALUATION AND FUTURE PLANS

5.1 Work Evaluation

No.	Content	Duration
1	**Study existing systems**	**1 week**
1.1	Study UTraffic's services	0.5 week
1.2	Study other existing services	0.5 week
2	**Prepare necessary knowledge**	**4 weeks**
2.1	Prepare necessary Front-end knowledge	1 week
2.2	Prepare necessary Back-end knowledge	1 week
2.3	Prepare necessary Database knowledge	1 week
2.4	Prepare necessary Model knowledge	1 week
3	**Implementation**	**6 weeks**
3.1	Gather bus-related data	2 weeks
3.2	Bus services APIs	2 weeks
3.3	Implement application's iOS Front-end	2 weeks

5.2 Future Plans

No.	Content	Duration
1	**Implement bus services**	4 weeks
1.1	Add appropriate APIs to the system	2 weeks
1.2	Implement features for searching and filtering bus stops and routes	1 week
1.3	Implement features for suggesting bus routes between locations	1 week
2	**Implement parking services**	4 weeks
2.1	Add appropriate APIs to the system	2 weeks
2.2	Implement features for registering, managing, and searching for parking lots	2 weeks
3	**Implement iOS application**	4 weeks
3.1	Import existing features from Android	2 weeks
3.2	Implement bus services	2 weeks

REFERENCES

[1] T. H.-T. D. /, *Traffic congestion costs ho chi minh city $6 billion each year*, vi, https://tuoitrenews.vn/news/society/20221001/traffic-congestion-costs-ho-chi-minh-city-6-billion-each-year/69343.html, Accessed: 2023-12-13, Oct. 2022.

[2] *Utraffic mobile*, vi, https://bktraffic.com/home/mobile-app, Accessed: 2023-12-13.

[3] *UTraffic - homepage*, vi, https://bktraffic.com/home/, Accessed: 2023-12-13.

[4] *Mkgeojsonobject*, Apple Developer Documentation. [Online]. Available: https://developer.apple.com/documentation/mapkit/mkgeojsonobject (visited on 12/14/2023).

[5] Apple Inc, *Updating apps that use web views - latest news - apple developer*, en, https://developer.apple.com/news/?id=12232019b, Accessed: 2023-12-13.

[6] *Trang chủ*, vi, http://buyttphcm.com.vn/, Accessed: 2023-12-14.

[7] *MongoDB manual*, vi, https://www.mongodb.com/docs/v3.0/reference/bson-types/, Accessed: 2023-12-12.

[8] M. Martin, *MVC vs MVVM – difference between them*, en, https://www.guru99.com/mvc-vs-mvvm.html, Accessed: 2023-12-13, Nov. 2023.

[9] Nimble App Genie and N. Sharma, *Objective c vs swift : Which one is the best?*, en, https://www.nimbleappgenie.com/blogs/swift-vs-objective-c/, Accessed: 2023-12-13, May 2023.

[10] *Educative answers - trusted answers to developer questions*, en, https://www.educative.io/answers/swift-vs-objective-c, Accessed: 2023-12-13.

[11] *The pros and cons of 8 popular databases*, en, https://www.keycdn.com/blog/popular-databases, Accessed: 2023-12-14.