Project Report

Bus Timetable and Ticketing

System Busy-Bees

Programmers

M00774667 - Alam Rincon - **Team Leader**

M00951627 - Artem Halis - **Secretary**

M00843428 - Charlie Dovey - **Developer**

M00916537- Petar Atanasov - **Developer**

M00829986 - Teon Morgan - **Tester**

Introduction:

Transportation is the key to any community, keeping people connected and on the move.

That's why we're excited to introduce our group project: the Bus Timetable and Ticketing System. Developed by a team group of five: team leader, secretary, 2x developers, and tester. Each with their own designated tasks, this system aims to simplify and enhance the bus travel experience.

At its core, our system aims to simplify and enhance the bus travel experience by providing a seamless platform for route searching, ticket purchasing, timetable viewing, and ticket tracking. Moreover, each user is equipped with a personalized account, offering a convenient hub for conducting all operations mentioned above.

Our project is mainly user-oriented, hence all operations are defined by users.
Initially, users can sign up or log in to their account to reach specific operations. They also have a predefined balance that they can use to purchase tickets. Here is the breakdown of menu functionalities that users have: find a route, show obtained tickets, show timetable at bus stop, account settings and logout.

All data for buses, stops and routes are loaded to the vectors from reading .csv files.

In this report, we'll take you behind the scenes of our project, explaining how we've designed the system, tested its functionality, and what we've learned along the way. We'll wrap up with some thoughts on where we go from here, and how our system can continue to evolve.

Report layout contains:

- Introduction:
  - brief description of the project
- Design:
  - justification of selected data structures and algorithms
  - analysis of the algorithms which provide the key functionality
- Testing:
  - testing used approach
  - table of test cases
- Conclusion:
  - summary of work done
  - limitations and critical reflection
  - how we would change our approach to similar tasks in future
- References:
  - Used methodologies for project development

Design:

Data Structures

In our project, we used various data structures, such as vectors(one dimensional, two dimensional) which were efficient for their random access to elements, constant-time access to the back element and insertion, and removal of elements in the end.

In our project, the main beneficial usage of vectors is keeping an eye on all items in a large dataset and being able to fit, retrieve, and remove selected items.

Another DS we used is a hash table for their efficient insertion, deletion and lookup operations. Moreover, in our project, we needed to organise and group key-value pairs of data. Therefore the hash table was strongly efficient for our project.

Usage:

One-dimensional vectors:

- Users list
- Stops list
- Tickets purchased list
- Buses list
- Route stops order
- Routes list

Two-dimensional vectors:

- Load data from the provided file path into a two-dimensional vector

Hash tables:

- Store assigned buses in a hash table
- Store required files, like buses.csv, stops.csv, and routes.csv in a hash table

Algorithms

Quick Sort	**Binary Search**
The quick sort algorithm was chosen for its efficiency and perfect usage with large datasets. It does not require extra memory and is one of the fastest algorithms available. **Usage:** Once data is loaded into the system initially from .csv files	Binary search was used for the sake of its simplicity and efficiency. Fast searching of sorted data, minimal space complexity and ease of implementation were the beneficial keys to why we chose it. **Usage:** Once we need to search for specific tickets, assigned routes for busses, stops that are assigned to routes etc.

Testing:

Approach Type: Unit Testing

For testing the functionality and ensuring the robustness of the project, **Catch2** unit tests were implemented. Unit testing is a software testing approach where individual components or units of a software application are tested independently to ensure they function correctly. Catch2, a modern C++ testing framework, was utilised to write comprehensive unit tests that validate the correctness of individual components within the project.

Test Case Description	Expected Outcome
Construction of all 5 main classes(User, Bus, Route, Ticket, Stop)	All classes instantiated successfully with correct attributes initialized.
Loading all data from a provided file	Data from the file is loaded into the system without errors.
Registering a new user	A new user account was created successfully with valid input.
Logging in an existing user	Existing users authenticated successfully with valid credentials.
Purchasing a ticket	The ticket was purchased successfully with valid user input and sufficient balance.

These test cases cover various aspects of the system, ensuring that each component functions as intended and meets the specified requirements.

Quick Sort Time Complexity Analysis Binary

Search Time Complexity Analysis

Conclusion:

Summary

Our team, comprising five members, completed the project, implementing all planned features and conducting thorough testing. Throughout the project duration, we maintained effective communication channels, including daily stand-ups, group calls with screen-sharing capabilities and a dedicated group chat for ongoing discussions and idea sharing. These practices facilitated collaboration, enabling us to progress steadily towards our goals.

In addition to our development efforts, we conducted comprehensive research to understand existing systems and best practices. This included studying websites such as TFL (Transport for London) and other similar platforms that provide APIs, allowing us to gain insights into their functionalities and approaches. Furthermore, at the project's outset, we utilized

resources such as the GeeksforGeeks website, which featured a Bus Reservation System implemented in C++. This served as a valuable reference point, inspiring ideas and providing guidance on potential features and functionalities to incorporate into our system.

Limitations and Critical Reflection

While we achieved our objectives, one notable limitation was related to time management. Balancing schedules to find suitable meeting times occasionally posed challenges, impacting our efficiency. Reflecting on this, we recognize the need for improved time management strategies in future projects. Additionally, clearer prioritization of tasks could help mitigate similar issues.

How Would Change Approach on Similar Task in Future

In a future of similar scope, we would change our approach by placing greater emphasis on structured meetings and enhanced preparation. This includes establishing clearer agendas for discussions, ensuring all team members come prepared, and maximizing the productivity of each session. By implementing these improvements, we aim to optimize our workflow and streamline project execution.

References:

Banerjee, P. (2022). *Quick Sort Explained | C++ STL*. [online] ALgo 101. Available at: https://medium.com/algo-101/quick-sort-explained-c-stl-c8d105cbaade [Accessed 19 Apr. 2024].

Halli, O. (2024). *Binary Search in C++ (code with explanation)*. [online] Medium. Available at: https://medium.com/@omkareshwarhalli/binary-search-in-c-codewithexplanation-5d a4bbf62eee [Accessed 19 Apr. 2024].

TFL, GOV.UK. (2024). *APIs: List*. [online] Transport for London - API. Available at: https://api-portal.tfl.gov.uk/apis [Accessed 19 Apr. 2024].

developer.transportapi.com (2024). *Examples | TransportAPI*. [online] developer.transportapi.com. Available at:

https://developer.transportapi.com/examples#tapi-bus-information [Accessed 19 Apr. 2024].

www.geeksforgeeks.org (2023). *Bus Reservation System in C++*. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/bus-reservation- system-in-cpp/ [Accessed 19 Apr. 2024].

Banerjee (2022) explained the Quick Sort algorithm, highlighting its efficiency and suitability for large datasets. This resource informed our decision to utilize Quick Sort in our project.

Halli (2024) offered insights into the implementation of the Binary Search algorithm in C++, emphasizing its simplicity and efficiency. We referenced this resource when selecting an appropriate search algorithm for our system.

TFL, GOV.UK (2024) and developer.transportapi.com (2024) provided valuable information on APIs related to transportation, such as those offered by Transport for London. Our research into these APIs helped us understand industry standards and best practices.

The example provided by geeksforgeeks.org (2023) of a Bus Reservation System in C++ served as an initial reference point for our project, inspiring ideas and guiding our early development efforts.