James Agbotta (B00310481)

Taxi Dispatch Application Proposal

Algorithms and collections Coursework 1

Contents

[1 The Problem 3](#_Toc121778445)

[1.1 A Typical Taxi Hire 3](#_Toc121778446)

[2 Purpose 4](#_Toc121778447)

[3 Specifications 4](#_Toc121778448)

[3.1 Analysis of Software Functionality 4](#_Toc121778449)

[3.2 Solution Analysis 5](#_Toc121778450)

[3.2.1 Data Structures 5](#_Toc121778451)

[3.3 UML Class Diagram of proposed solution 7](#_Toc121778452)

[4 Ethical Considerations 8](#_Toc121778453)

[5 Development 8](#_Toc121778454)

[5.1 Methodology 8](#_Toc121778455)

[5.2 Implementation 8](#_Toc121778456)

[6 Development Schedule 9](#_Toc121778457)

[7 References 10](#_Toc121778458)

Coursework 1

# The Problem

Taxi dispatchers are often overlooked in the system of private hire taxi services. They require the skill to select the right driver, in the nearest possible location for any hire that is needed urgently. This is an issue that also affects emergency personnel (police, paramedics, and fire service personnel).

## A Typical Taxi Hire

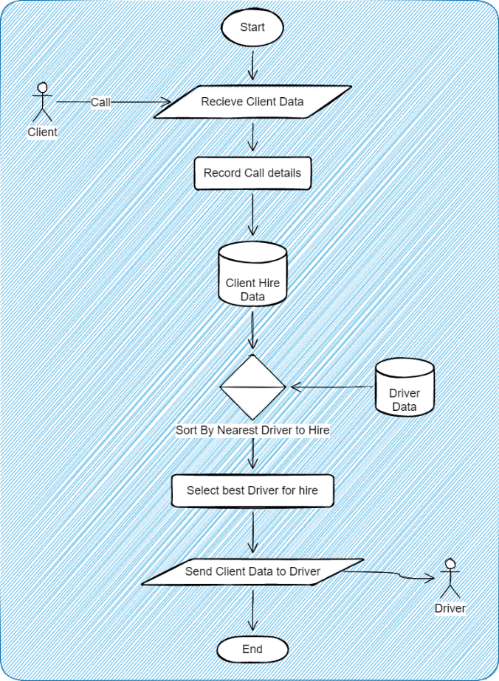
The standard taxi hire system comprises of:

Figure 1: A flow chart displaying the typical hire of a taxi from a dispatch perspective.

A typical taxi hire goes as follows:

1. A call is made to dispatch
2. Hire details are stored in the system
3. The drivers are sorted by dispatch in terms of proximity to hire
4. A driver is chosen by dispatch
5. Driver is notified of the hire

From then the driver performs the service assigned.

The issue arises when the dispatcher needs to select an available driver to carry out the fare.  
It has been noted that drivers tend to approach hires from arterials roads (smaller, less frequented, near residential areas) meaning there are not likely to be found near main roads (Nian *et al.*, 2017, pp. 685–687).   
 There may be multiple drivers in what is visually the same level of proximity to the client, but, without adequate knowledge of the road system or the driver’s directionality, the dispatcher may send clients to sub-optimal drivers, increasing wait time for more urgent fares.

Another issue, less important in the grand scheme of things, is if the driver is the right person for the fare. If a driver can be proven to give bad customer service, it is less likely for the dispatcher to be willing to give the driver the fare, even if the driver is closer to the client than one who will do a better job.

# Purpose

With drivers’ locations and travel patterns being studied to take specific routes to optimise travel and roaming having a system to prioritise drivers based on their known (or assumed location) in the assignment of said drivers with clients to reduce driver roaming time to a minimum.  
To create an application to aid in the efficient selection of drivers based on their current location, state of the driver (engaged, vacant, or out of service). This will be done by creating an aggregate score of each driver in respect to the client’s needs and the driver’s ability to provide the best possible service. This solution is primarily to offset (or remove) the need for a human dispatcher making decisions, in a time and cost saving measure.

# Specifications

## Analysis of Software Functionality

A picture containing arrow

Description automatically generated

Figure 2: A UML Use Case Diagram showing the system of ordering a taxi, in which a dispatch personnel is involved.

## Solution Analysis

Graphical user interface, text, application, email

Description automatically generated

Figure 3: A MoSCoW Analysis done in Notion

In the hierarchy of needs to produce this application, a means to contain hire, driver, and street data, as well as manipulate the data (create, read, update delete, search and sort).  
From there, creating a system that allows the dispatch admin access to manipulate the data, and a means to present the data in a user friendly manner is welcome, but of a lower priority.

### Data Structures

As mentioned in the MoSCow analysis, data structures to contain hire and driver data are must. To This end, all hires will be stored in a node-based queue, while drivers will be stored in a linked list.  
The client hires will be listed on First Come First Serve, with each new hire having to wait for the hires previously accepted to be assigned drivers. In the event scheduled hires are added to the system, hire data will be stored as an encrypted file on local storage to be retrieved on load up.

Drivers are stored on local storage as a JSON file, which will be loaded as a graph (based on driver location).   
After sorting for most applicable driver to a specific here, the resultant stack generated will be queried to ensure that driver is available, if not, the driver will be popped from the stack and the next driver is queried.

The map system will be a graph of streets as nodes and junctions/intersections as edges. Each street node will hold the following information:

* Street Length: This will be used as the weighting system for searching for the quickest route from driver to hire. The length is calculated from intersection to intersection. In the event Open Street Map is implemented, the number of nodes between intersections will be used as a secondary weight.
* Hires: A list of hire IDs on that street. This is on the contingency that multiple people need vehicles on the street for any reason.
* Drivers: A list of drivers currently halted on that street. This is on the contingency that multiple vehicles are waiting the street (E.g. they are at a depot or taxi rank).

## UML Class Diagram of proposed solution

Below with the UML Class diagram of the prototype system.

Things to note: this a skeleton of classes to implement MVC into the system to ensure well-formed software.

Graphical user interface

Description automatically generated

Figure 4: Overview of a Class Diagram for the dispatch application.

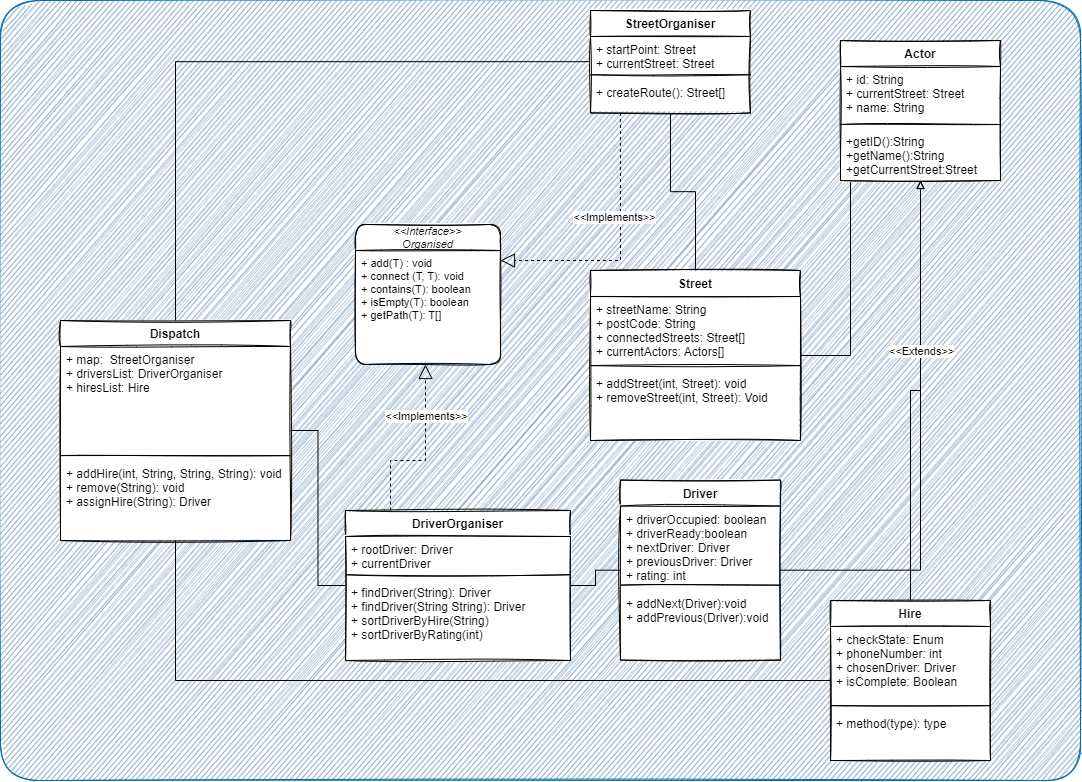


Figure 5: UML Class Diagram showing the model Package

# Ethical Considerations

The following should be considered:

* The absolute minimum data should be used, regarding potential clients. This is to build trust, reduce the risk of misuse, theft, and the sale of data to third parties without consent.
* The data of Driver and hire should be encrypted and secured in accordance with GDPR 2016, Data Protection Act 2018 and only kept for as long as is necessary for taxi hire, no more. Specifically:
  + The users of the software must opt in to have their data used,
  + Have the right to have their data completely forgotten,
  + Have their data correct and it's use fully disclosed to the users,
  + And the right to have their data protected.
* The starting locations of Taxis may be near their home addresses. These locations may need to be obscured to protect their privacy.
* All open source or licensed software must be reported or listed as per their respective licenses. Any materials used that require paid licences should be duly paid for and licences listed.
* The applications production, maintenance and use must comply with BCS Code of Conduct for members. Including but not limited to:
  + **Public Interest:** In protecting health privacy, and wellbeing of others, as well as protection for legitimate rights of others (a reiteration of the above considerations)
  + **Professional competence and integrity:** in complying with legislature and carrying our personal responsibilities, and only taking work within one’s professional competence.

# Development

## Methodology

The proposed new system will be implemented using a Dynamic Systems Development Method. DSDM entails making snapshots of the points made in MoSCoW Analysis and create achievable milestones from those points. Each milestone is then implemented and tested iteratively to ensure progress towards a viable product is made.

## Implementation

The main dispatch system will be implemented in Java. In the event map-based integration is made, OpenStreetMap’s API will be included in the production to facilitate map integration.

To test, and or implement, a driver response system, React with JSX will be used to create a client server system with Java for both the dispatch server and the driver clients to communicate with each other.

# Development Schedule

The software will be worked on in segments based on the Order of Priority found in the MoSCoW Analysis. Each milestone will be recorded as in, and any delays will cascade the lower priority tasks in the system. The initial deadline is February 3rd with a two-week buffer for issues.

The progress of software development will be noted in Notion, a software whose features include note taking, task management, timeline creation, and progress logging.

Text

Description automatically generated with medium confidence

Figure 6: Table of Tasks created in Notion; A project management and note taking application.

# References

*About | OpenStreetMap Blog* (no date). Available at: https://blog.osmfoundation.org/about/ (Accessed: 8 October 2022).

*BCS Code of Conduct for members - Ethics for IT professionals | BCS* (no date). Available at: https://www.bcs.org/membership-and-registrations/become-a-member/bcs-code-of-conduct/ (Accessed: 11 December 2022).

*Chapter 2: Choosing DSDM* (no date). Available at: https://www.agilebusiness.org/dsdm-project-framework/choosing-dsdm.html (Accessed: 18 October 2022).

*Data Protection Act* (2018). Available at: https://www.legislation.gov.uk/ukpga/2018/12/contents/enacted (Accessed: 12 November 2022).

Nian, G. *et al.* (2017) ‘Analyzing behavior differences of occupied and non-occupied taxi drivers using floating car data’, *Journal of Shanghai Jiaotong University (Science)*, 22(6), pp. 682–687. Available at: https://doi.org/10.1007/s12204-017-1890-9.

‘REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL’ (2016) *Official Journal of the European Union* [Preprint]. Available at: https://eur-lex.europa.eu/legal-content/EN/TXT/PDF/?uri=CELEX:32016R0679 (Accessed: 12 November 2022).