St Mary's University Twickenham London

CPS5001 – Data Structures and Algorithms – Assessment Brief

Module Code:	CPS5001
Module Title:	Data Structures and Algorithms
Module Convenor:	Prins Butt
Module Level:	5

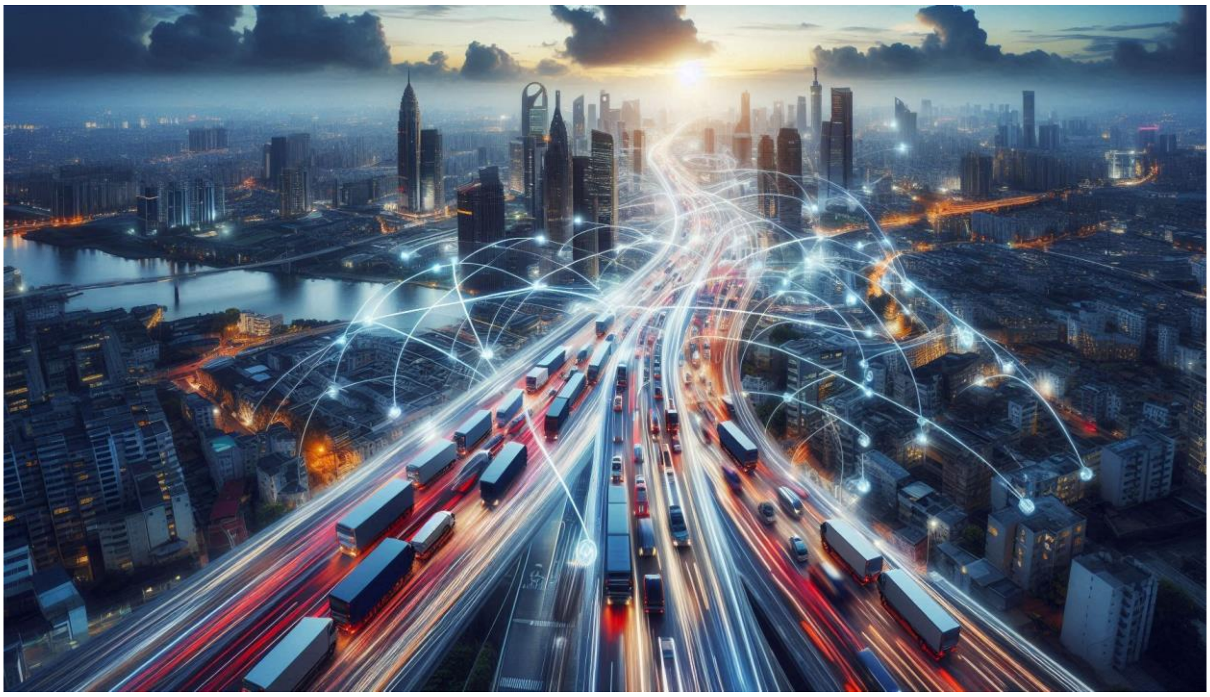
Assessment Number:	2
Assessment Title:	Intelligent Logistics and Delivery System
Assessment Weight:	60%
Assessment Individual/Group:	Individual
Assessment Type:	Software Artefact with Report
Assessment Time/Word Count Restrictions:	1 software artefact 2400 words report
Assessment Time/Word Count Limit Consequences:	It is essential that assignments keep within the time/word count limit stated above. Any work beyond the maximum time/word length permitted will be disregarded and not accounted for in the final grade.

Issue Date:	8th November 2024
Hand in Date:	6th January 2024
Planned Feedback Date:	Within 3 working weeks
Mode of Submission:	Online via Moodle
Number of copies to be submitted:	1 copy of each of the following: • a report in pdf format • a zip file containing your software artefact

Author:	Prins Butt
Internal Moderator:	Harshil Joshi
Moderation Date:	28th October 2024

Table of Contents

Introduction



You have been commissioned to design and develop an Intelligent Logistics and Delivery System (ILDS) for a logistics company. The system must optimise the movement of delivery vehicles across a city by providing efficient route planning, delivery scheduling, and congestion prediction, all while adapting dynamically to changing conditions. Your task is to determine the appropriate data structures and algorithms to achieve these objectives. You are required to implement your solution in Java and provide a detailed explanation and justification for the choices you make.

Requirements

You are required to design and implement a system in Java that addresses the challenges of city logistics. The solution must include a range of features designed to ensure efficiency, scalability, and real-time adaptability. The assessment is divided into several components, each with specific requirements.

1) City Logistics Network Representation

The first component requires you to represent the city's logistics network as a graph. The nodes in the graph will represent delivery hubs and customer locations, while the edges will represent roads, weighted by factors such as distance, congestion levels, and travel time. Your system must support the dynamic addition and removal of nodes and edges, as well as updates to edge attributes. You must carefully select an appropriate data structure for the graph, considering factors such as efficiency and scalability, and provide a detailed justification for your choice.

2) Route Optimisation

The second component involves implementing a route optimisation feature to determine the best delivery routes. The routes should account for factors such as the shortest distance and minimal delivery time, even under changing traffic conditions. You are required to select and implement one or more pathfinding algorithms. In addition, you must extend the algorithm(s) to handle constraints such as vehicle capacity and delivery deadlines. Your report must include an analysis of the performance of your chosen algorithms, highlighting their strengths, limitations, and suitability for this scenario.

3) Delivery Scheduling and Vehicle Management

The third component focuses on delivery scheduling and vehicle management. Your system should assign deliveries to vehicles based on location and capacity, prioritise time-sensitive deliveries, and ensure that vehicles are not overloaded. You must use appropriate data structures to manage scheduling. It is essential to provide a detailed justification for your choice of data structures, explaining how they contribute to efficient scheduling and vehicle management.

4) Congestion Prediction and Adaptation

The fourth component requires the implementation of a congestion prediction feature. This feature should use historical data or simulated traffic conditions to predict potential bottlenecks and adapt delivery routes accordingly. You are encouraged to use graph traversal algorithms or other predictive techniques, to detect and respond to congestion. Your system must include functionality to suggest alternative routes and should be tested with realistic traffic scenarios.

5) Design Documentation and Analysis

Finally, you must submit a report that includes a detailed explanation of your chosen data structures and algorithms. The document should also include a complexity analysis (using Big-O notation) for all major operations, as well as diagrams illustrating your system architecture and workflows. Additionally, you should reflect on potential ethical and practical considerations, such as the fair allocation of resources, scalability, and real-world constraints.

Environment and Tools

You are required to use the following tools for this assessment:

- **IntelliJ IDEA** (or an equivalent Java IDE): as your integrated development environment for Java.
- **Java SE 21+**: as the programming language and standard library.
- **Draw.io** (or an equivalent tool): to create diagrams for your design and documentation.
- **Microsoft Word** (or an equivalent tool): to author your report and export it as a PDF.
- **Adobe Acrobat Reader** (or an equivalent tool): to view your final report saved as a PDF file.
- **Git Tools and GitHub**: for version control, ensuring you track and evidence the development of your project.

Additionally, the following Java libraries and packages may be imported and utilised:

- Any standard Java data structures and utilities provided by the Java SE library.
- Appropriate Java libraries for the visualising your system

No additional external libraries or frameworks should be used without the written permission of the module convenor. This restriction ensures that the focus remains on your understanding and application of core Java concepts, data structures, and algorithms.

Submission

The assessment must be completed individually. You must not share, in part or whole, your assessment with another party other than the module convenor and for the purpose of submission to the university. You must ensure that the University's academic misconduct guidelines are followed in their entirety.

You should use the assessment submission link on the module's Moodle page to submit the following files:

- A **PDF** file for your report. This should not be included in the zip file but instead submitted as a separate file. Failure to do so may result in zero being awarded.
- A **Zip** file of your software artefact. This should contain your software solution and any relevant files to open and execute your solution.

You should ensure that you make a timely submission by the deadline stated at the start of this assessment brief.

Assessment Criteria

Your assessment will be graded according to the following criteria:

Grading criteria	Functionality (50%)	Documentation (30%)	Professional Practice (20%)
Mark band			
80-100 Pass (1st)	Fully functional system addressing all requirements, including advanced features like dynamic congestion prediction and multi-agent collaboration. Demonstrates excellent use of efficient algorithms and advanced data structures.	Comprehensive report with detailed justifications, including complexity analysis, system diagrams, and evaluations of design choices. Clear discussion of ethical considerations and innovative features.	Extensive version control showing industry-standard practices such as feature branching, pull requests, and commit tagging. Evidence of professional project management and adherence to best practices in software development.
70-79 Pass (1st)	Fully functional system addressing all requirements. Includes advanced algorithms like A* or Dijkstra's for optimisation and features such as route suggestion. Strong use of object-oriented principles and scalable design.	Detailed report with complexity analysis, supported by clear diagrams and well-explained design choices. Includes a discussion of professional and ethical considerations.	Strong version control evidence, including feature branches and detailed commit messages. Demonstrates effective task management and adherence to professional development standards.
60-69 Pass (2.1)	Mostly functional system addressing core requirements such as route optimisation and delivery scheduling. Demonstrates appropriate use of data structures like graphs, queues, or heaps. Strong application of object-oriented programming principles.	Detailed documentation explaining the implementation with justifications for data structure and algorithm choices. Includes diagrams and some complexity analysis.	Good version control practices, with clear commit messages and use of branches. Evidence of iterative development and problem-solving through version history.
50-59 Pass (2.2)	Functional system meeting core requirements like basic network representation and package delivery. Some attempt at route optimisation or scheduling. Demonstrates basic object-oriented programming principles.	Reasonable documentation explaining implementation choices with limited analysis of complexity or performance. Some supporting diagrams or outputs included.	Basic version control evidence, showing development progression with meaningful commit messages. Limited use of advanced practices like branching or pull requests.
40-49 Pass (3rd) (*Threshold*)	Partially functional system with basic features such as package delivery and graph representation. Lacks optimisation or advanced functionality. Demonstrates minimal application of object-oriented programming.	Basic documentation summarising implementation with limited justifications and no complexity analysis. Diagrams, if included, lack detail or relevance.	Minimal version control evidence, such as commit history showing basic development stages. Does not demonstrate consistent or professional use of version control practices.
30-39 Marginal Fail	Incomplete or partially functional system with significant gaps in meeting core requirements. Demonstrates limited	Inadequate documentation with vague explanations and no meaningful analysis or justification. Lacks diagrams or evidence of critical thinking.	Poor or no version control evidence, with commits showing inconsistent or poorly organised development.

	understanding of data structures and algorithms.		
0 – 29 Fail	Little to no functionality demonstrated. Incorrect or inappropriate use of data structures and algorithms.	Missing or wholly inadequate documentation, with no justifications, diagrams, or evidence of analysis.	No evidence of version control or professional practices in development.

Additional Guidance

Basic Solution

The solution should simulate a basic logistics network consisting of delivery hubs, customer locations, and roads. Delivery vehicles must collect packages and transport them to specified hubs, with the network represented using an appropriate data structure such as a graph. The solution must be implemented using object-oriented programming principles, demonstrating basic functionality.

The project should include evidence of version control, such as commit logs, to demonstrate the stages of software development.

The report should provide a summary of the implementation, including justifications for the chosen data structures and algorithms. The report should include basic diagrams illustrating the system design and operation. Evidence of version control, such as screenshots of commit history, must be included.

Comprehensive Solution

The logistics network simulation should include all features required for a pass grade, with an additional attempt to incorporate route optimisation or delivery scheduling. The solution should demonstrate the application of object-oriented programming principles with evidence of modularity and encapsulation.

Version control should demonstrate best practices, such as the use of feature branches and meaningful commit messages, to track the development process effectively.

The report should provide a detailed explanation of the implementation, including rationale for the design choices and supporting evidence such as diagrams and examples of code output. Comprehensive version control evidence should be included, demonstrating an understanding of effective version control practices.

Strong Comprehensive Solution

The logistics system must attempt all core requirements of the assessment, including network representation, route optimisation, and delivery scheduling. It should showcase strong object-oriented programming principles, with an emphasis on reusable and scalable code design. Efficiency improvements should be evident, with complexity analysis for major operations included in the documentation.

Version control should demonstrate advanced practices, including the use of feature branches, tagged releases, and detailed commit messages, highlighting progress and iterations.

The report must include a thorough explanation of the implementation, with extensive justifications for the selection of data structures and algorithms. Evidence of effective version control and ownership of the artefact should be detailed, including examples of branch merging, resolved conflicts, and iterative development.

Advanced Solution

The logistics system must comprehensively address all specified requirements, including dynamic congestion prediction and alternative route suggestions. The implementation should demonstrate advanced object-oriented design, employing appropriate patterns such as MVC or Singleton where appropriate.

The solution should also showcase advanced algorithms and may include innovative features. The artefact should reflect professional development standards, with extensive evidence of version control and the application of industry best practices such as code reviews or continuous integration.

The report must provide a detailed evaluation of the solution, including complexity analysis for all major components, a discussion of design patterns used, and critical reflection on the performance and scalability of the system. Strong consideration of professional practice and ethics should be demonstrated, such as discussing the implications of using selected algorithms or resource allocation fairness.

Comprehensive version control evidence should be provided, illustrating professional development techniques such as detailed commit histories, feature branching, and effective use of pull requests or merges.

Learning Outcomes

This assessment will enable students to demonstrate the following learning outcomes as stated in the module outline:

[Module Learning Outcome 2]

- Identify and analyse different algorithmic problems and select and implement appropriate solutions using a variety of data structures and algorithms.

How is this learning outcome addressed?

The assessment requires students to design a logistics and delivery system, a complex, real-world problem. To achieve this, students must identify and analyse algorithmic challenges, such as route optimisation, dynamic scheduling, and resource allocation. They are tasked with selecting suitable data structures (e.g., graphs, priority queues) and algorithms (e.g., Dijkstra's, A*). By implementing these solutions, students demonstrate their ability to match algorithmic techniques to specific problem requirements.

[Module Learning Outcome 5]

- Design and implement algorithms from scratch, using a variety of data structures and search algorithms, and evaluate their performance using appropriate metrics.

How is this learning outcome addressed?

Students are required to implement algorithms such as graph traversal, shortest path finding, or task scheduling from scratch, applying data structures like adjacency lists, heaps, or hash tables. They must evaluate algorithm performance using metrics such as time and space complexity. The report provides a platform to critically discuss their design choices, supported by complexity analysis and rationale for algorithm selection.

[Module Learning Outcome 6]

- Work independently on complex algorithmic problems and communicate solutions effectively to both technical and non-technical audiences, adhering to professional communication standards.

How is this learning outcome addressed?

The assessment emphasises independent work, where students are responsible for designing, implementing, and testing their system. The documentation criteria assess their ability to explain and justify their solutions clearly, using diagrams, analysis, and technical explanations for a specialist audience. Additionally, ethical considerations and real-world applications are discussed in the report, showcasing their ability to communicate to a broader, non-technical audience effectively.

[Module Learning Outcome 7]

- Be prepared for further study and research in computer science and related fields by having a solid foundation in advanced data structures and algorithms with the ability to apply knowledge ethically to real-world problems.

How is this learning outcome addressed?

This assessment challenges students with a real-world scenario requiring the application of advanced data structures (e.g., graphs, heaps) and algorithms (e.g., Dijkstra’s, A*, dynamic programming). The inclusion of ethical considerations ensures students recognise the broader implications of their designs. By addressing practical challenges and adhering to professional practices such as version control, the assessment prepares students for further study or research in computer science.

Regulations, Policies, and Guidelines

Guidance for online submissions
https://www.stmarys.ac.uk/policies/online-submissions.aspx

Academic Misconduct
Any submission must be students' own work and, where facts or ideas have been used from other sources, these sources must be appropriately referenced. Please find a link to the academic misconduct policy below:
https://www.stmarys.ac.uk/policies/academic-regulations.aspx

Ethics Policy
The work being carried out by students must be in compliance with the Ethics Policy. Where there is an ethical issue, as specified within the Ethics Policy, then students will need ethical approval prior to the start of the project. Please find a link to the ethics policy below:
https://www.stmarys.ac.uk/research/students/ethical-review-process.aspx

Extenuating Circumstances
The University's Extenuating Circumstances procedure helps students facing challenges in assessment submission. To request an extension or deferment, submit an EC application with evidence. Approved cases will not incur academic penalties. For longer-term issues, contact Student Services. Please find a link to the EC policy below:
https://www.stmarys.ac.uk/policies/extenuating-circumstances.aspx

Reassessment
If a student fails to meet the assessment criteria, they may be eligible for a reassessment. The reassessment will follow the same guidelines as the original assessment, requiring the student to reattempt the tasks outlined. Students should review any feedback provided and make improvements to meet the expected learning outcomes. In case of reassessment, it is important to consult with the module convenor for clarity on the areas of improvement and ensure compliance with submission deadlines and regulations.