



Coursework Title: ASI

A Java Console-Based Railway Timetable

Module Name: Software Engineering Workshop

Module Code: 4121COMP

Level: 4

Credit Rating: 20

Weighting: 100% of module

Maximum mark: 100

Lecturer: Dr. David Lamb

Contact: If you have any issues with this coursework you may contact your lecturer.

Contact details are:

Email: D.J.Lamb@ljmu.ac.uk

Room: 7.11

Hand-out Date: w/c 18th Jan 2021 (week 17, sem 2: wk1)

Hand-in Date: 17:00 @ Friday 30th Apr 2021 (week 31, sem 2: wk15)

NOTE: Demo session is earlier: w/c 19th April 2021

Hand-in Method: Electronic submission via VLE; appropriate SCM repository; demonstration

Feedback Date: Monday 24th May 2021

Feedback Method: Summary verbal feedback at demonstration & Group feedback document, via email

Introduction

This assignment is a group task, whereby you will establish a project team of *a minimum of three and no more than five students*. Your project team must *work independently from other teams*, and following an agile, incremental Software Development Lifecycle, analyse the following specification, design and develop a solution piece of software in accordance with the practices taught within this module.

Learning Outcomes to be assessed

- 1 Understand planning for software solutions, in a larger project, as part of a software development team.
- 2 Explain software systems development through specification, design and implementation.
- 3 Analyse computer programs; determining the behaviour of the program from its source code and rewriting or adding to existing code.
- 4 Appraise the facilities offered by modern CASE tools such as integrated development environments and source code repositories

Detail of the task

Preamble: You are to analyse the following specification, document analysis, produce appropriate software designs and utilise test-driven development to implement a working solution. You are each responsible for project planning, team management, developing software and ensuring it works as specified. You must take charge of **each of these roles** through the project's duration; i.e. rotate roles so you each have planned, analysed, designed and developed – don't plan to "just" be the programmer or the project manager.

This is not an exhaustive specification: to aid analysis, **you have UP TO five opportunities to consult the client (i.e. the module leader!) regarding specification queries, or to solicit feedback on development (e.g. designs, prototypes, increments, etc) so far.** This is usually facilitated by the "client" (remotely) joining your stand-up or Scrum development meetings, or at a Sprint Review, as you deem necessary.

You should fully document these consultations and use them to shape the group's own future development. Beyond this, you can make (and must document) assumptions based on your research and what you believe feasible in the allowed time.

YOUR TEAM ARE RESPONSIBLE FOR ORGANISING, SETTING THE AGENDA FOR THESE MEETINGS, AND SENDING THE AGENDA TO THE CLIENT 24 HOURS IN ADVANCE.
ALL GROUP MEMBERS ARE EXPECTED TO PARTICIPATE.

You then have a final opportunity to present to the client (Module Leader), **where you will demonstrate your software** and the features (or parts thereof) that you've implemented. Each group member will be expected to explain and justify their contribution to the project team. Teams are advised to prepare some additional support materials (e.g. slides) for your final demonstration.

NOTE: THE MODULE LEADER WILL ASSIGN YOU A DATE AND TIME TO CONDUCT YOUR SOFTWARE DEMONSTRATION.

There are opportunities to gain extra marks in terms of:

- Using profiling tools to measure, and where feasible, optimise your software's efficiency
- Regular/appropriate use of a source code versioning (SCM/VCS) solution to manage your group's source code and maintain a revision/contribution history tagged appropriately by the contributing team member.
- Attempting and implementing intermediate and advanced functionality, as outlined below.

IMPORTANT: Your entire team takes responsibility for submitting any group components / deliverables. As such, it is strongly recommended that your team devise, agree and use a submission protocol where EVERY MEMBER checks the submitted work (on the VLE submission section) and ensures it is correct and as expected. Ensure this is not left to a single team member!

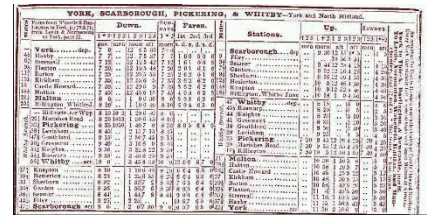
Conversely, you take responsibility for your own individual report (described later). As such, non-submission of this component will be considered a non-submission for you in this assessment.

Problem Specification

Domain Background: The railway timetable has been an integral part of railway operations since the printed Bradshaw's guides from the 1830s; information useful to operators of railway services and the travelling public.

Particularly since the advent of web-based timetables, there has been much focus on presenting information in a form useful for a railway traveller; it is much more straightforward to determine the details of a desired journey.

Various key developments added useful functionality to a rail traveller; particularly so as the railway's various systems became integrated and/or interconnected, such as automated timetable filtering (i.e. you only see the bits of the timetable relevant to you), showing and filtering on station facilities – and crucially – the ability to show a live (i.e. real-time) version of the timetable, not just the (i.e. planned) “working timetable”.



Task Specification

Tasks: you are required to analyse, design and implement a Java, console-based software solution that addresses some or all of the following functional points. They are categorised according to difficulty, which, in conjunction with the marks guide later, allows you to plan an achievable project and understand the grade profile you'll aim at.

Basic:

A. Timetable Data Persistence: The timetable data (you must determine exact requirements through this specification and subject research) should be read from (a) file(s). There is no requirement that this data is editable; you can either populate a test file yourself by hand (e.g. from included example PDFs or similar alternatives), or attempt to interpret any available rail timetable data your research uncovers.

B. Matrix or Tabular View: The application must be able to display, for a given route, a timetable of services in a tabular format. The usual layout is columns show the time of day, and rows indicate stations / stops; (e.g. *timetable on Liverpool-Blackpool route from Northern PDF*)

An entire timetable will not fit on the screen, so there must be a feature to scroll forward and back through times, and up and down through stations, as required. There is no requirement to include multiple routes, but you must be able to show the variations of timetable for a given route (e.g. *Mon-Fri, Sat, Sun or similar, as appropriate*). The requirement is that this view must show stations (full names) and times (where both arrival and departure times are listed, only the departure time need be shown).

Liverpool - Wigan - Blackpool

Mondays to Saturdays

Liverpool Lime Street	d	1345	1437	1445	1537	1545	1636	1645	1709	1736	1745
Edge Hill		1349	-	1449	-	1549	-	1649	-	-	1749
Wavertree Technology Park		1351	-	1451	-	1551	-	1651	-	-	1751
Broad Green		1354	-	1454	-	1554	-	1654	-	-	1754
Roby		1357	-	1457	-	1557	-	1657	-	-	1757
Huyton		1400	1445	1500	1546	1600	1645	1700	1718	1744	1800
Prescot		1404	-	1504	-	1604	-	1704	1722	1748	1804
Eccleston Park		1406	-	1506	-	1606	-	1706	1724	1751	1806
Thatto Heath		1409	-	1509	-	1609	-	1709	1727	1753	1809
St. Helens Central	a	1412	1453	1512	1553	1612	1653	1712	1730	1757	1812
	d	1413	1453	1513	1554	1613	1653	1713	1731	1757	1813
Garswood		1419	-	1519	-	1619	-	1719	1737	1804	1819
Bryn		1422	-	1522	-	1622	-	1722	1740	1807	1822
Wigan North Western	a	1431	1506	1531	1606	1631	1706	1731	1747	1813	1831
	d	-	1506	-	1607	-	1706	-	1747	1814	-
Euxton		-	1515	-	1616	-	1715	-	1756	1823	-
Leyland		-	1520	-	1620	-	1720	-	1800	1827	-
Preston	a	-	1528	-	1628	-	1727	-	1810	1835	-
Kirkham & Wesham	a	-	1538	-	1639	-	1738	-	-	1846	-
Poulton-le-Fylde	a	-	1547	-	1647	-	1747	-	-	1854	-
Blackpool North	a	-	1553	-	1653	-	1753	-	-	1900	-

Intermediate features over the page....

Intermediate:

C. Origin/Destination filtering: The tabular view should have an additional feature such that it is possible to filter the timetable such that it shows only rows for a specified origin and destination station. This must filter out services that do not call at BOTH origin and destination stations (e.g. expresses).

D. Interchangeable station codes: All input facilities function with either 3-letter “CRS” station codes or the full station name (e.g. *LIV* = *Liverpool Lime St*) – i.e. typing *LIV* should indicate to the system that *Liverpool Lime St* is the required station. The mappings *must not* be hardcoded; they must be drawn from a station codes document at start-up. It is available for download from here: <https://bit.ly/4121Stations>

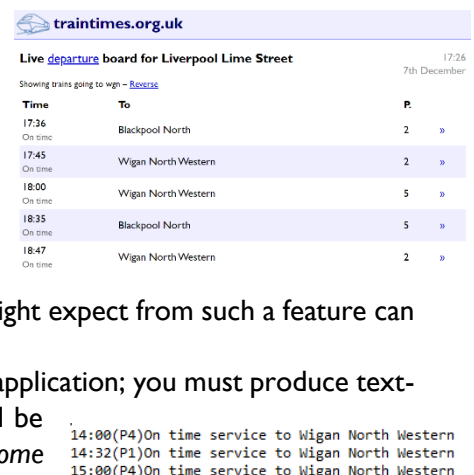
E. Journey Duration calculation: Given the known start and end times, when a filtered timetable is shown, the software must calculate and display the journey time, in hours and minutes. For an unfiltered timetable, the total (end-to-end) journey time should be displayed.

F. Station facility display & filtering: A feature to filter stations in tabular view based on whether it has the following facilities: car parking, bicycle racks/storage, and step-free access/disability assistance. Additionally, when in unfiltered mode, the view should indicate whether a station has these facilities; utilising a key such as STATION NAME [P, B, D] would indicate the station has all 3 facilities; STATION NAME[B, D] would indicate it has bicycle racks and step-free access, but does not have car parking.

Advanced:

G. Real-Time LDBs (Live Departure Boards): The software should be able to present a Live Departure Board for the selected departure station, such that it shows only those services calling at the desired destination station. This should be implemented in a feature similar to (or an extension of) that provided for the Intermediate feature, *C. Origin / Destination filtering*. The results should show the high-level details (i.e. *scheduled vs. expected departure time, status (on-time, delayed, cancelled) and platform, if available*), of any matching service. This will require you to connect to a service that provides this information. Examples of the data output you might expect from such a feature can be seen at: <https://bit.ly/trainldb> & <https://bit.ly/NREldb>

Obviously, you are constrained somewhat by the limitations of a console application; you must produce text-only output – console output such as the screenshot to the right would be acceptable. Please see the guidance for this and the following feature for some guidance on how this data should be acquired.



Time	To	P.
17:36 On time	Blackpool North	2
17:45 On time	Wigan North Western	2
18:00 On time	Wigan North Western	5
18:35 On time	Blackpool North	5
18:47 On time	Wigan North Western	2

14:00(P4)On time service to Wigan North Western
14:32(P1)On time service to Wigan North Western
15:00(P4)On time service to Wigan North Western

H. Enhanced LDB: The software should allow the user to select any service shown in the LDB feature above, and show more details for this service. This should include, but is not limited to:

- Calling points (and calling times) for each service
- Train length (where available) / other service-specific data
- Delay or cancellation reason (where applicable)

I. Your chosen feature: Once you have attempted ALL OTHER FEATURES, you may add one more feature of your choosing. This is an opportunity for your team to demonstrate creativity and show off their skills. You *must* discuss and refine this feature at a client meeting, before attempting it. You should consider features relating to obtaining, processing and/or caching live data, location-sensitive reporting, alerting, and/or presentation of results.

Guidance for Live Departure Boards and related advanced features

Various web providers have an intuitive web-facing service providing much real-time information, but place strict restrictions on how you may use the data. In the *traintimes* and *nationalrail* examples, both prohibit “reuse”, so while it would be technically viable to *web scrape* these sites; **it would be a violation of the service’s terms of usage**. Before using any online service that purports to provide you with relevant information, you should fully investigate and *beware of violating any terms of service*. This is particularly important during testing when your software may inadvertently send requests more frequently than you’d intended!

Note: where it is clear that you are violating the terms of usage for a service (e.g. scraping the above sites or exceeding your usage cap), your team will not be eligible for marks for the affected features.

National Rail provide a free-to-use (*registration required*) SOAP Web Service that provides this information; its terms of service permit 5000 requests / hour without charge. More details are available at: <https://bit.ly/NRE-OpenLDB-Docs> - registration is available at: <https://bit.ly/NRE-OpenLDB-Reg>. Successful registration will provide you with an access token that you must supply with every query.

As this is an advanced feature, your team will need to research how to connect to a SOAP Web Service, and how to use several Java tools to generate some relevant code for you. Help is available from module staff, but your team need to take a lead. To get you started, you are advised to look at Java’s command-line *wsimport* tool. While this is feasible in any recent version of Java with additional libraries, it is easiest with JDK8, as this includes the relevant libraries. *wsimport* is a tool that can generate client code from the web service’s WSDL file.

Relevant helpful links are given in *Useful Links* towards the end of the specification.

Marking Summary (see *additional grading / attainment rubric document*)

While a per-section breakdown is provided over the page, please ensure you consult, study and understand the provided **attainment rubric** for a detailed breakdown of how marks are allocated and the grading expectations – **PLEASE ASK IF YOU DO NOT UNDERSTAND**.

Group Mark Assignment, Individual Reports and Weightings

As one of this assessment’s learning outcomes measures your ability to work in a software development team, to allow you each to experience the various specialisms required to complete this coursework; gaining and sharing skills with other team members. As such, while it is understood that teams will want to capitalise on their strengths; you are each **required** to take a lead and complete at least one series of design, implementation and evaluation tasks for a given Feature.

You must each, individually, also complete a brief report outlining the aspects of the project that you personally contributed to, and in what capacity you did so.

You will also be expected to comment on aspects where working as a team helped the project, and where working as a team caused difficulties, along with how they were resolved. It is recommended you keep a personal work diary to assist you in doing so.

In this individual report, you will also assess your peers against a number of team-working criteria (as they will you!). You’ll be provided with a template to structure this report and the peer review. This peer review, along with evidence gathered by the module leader, will be used to weight the final group mark to give each member an individual grade – this is notable where a member has underperformed and others have had to significantly increase their contribution. In the rare event there is significant dispute between peer reviews, the module leader may arrange a short *viva voce* examination to assess an individual’s contribution and share of knowledge.

What you must hand in

You should prepare the following assessed components:-

- **GROUP:** Project documentation, including the requested project/team, analysis, design and testing deliverables below. You must use tasks in the table below as section headings within your document. Further details on each deliverable type will be / is provided in the referenced delivery sessions below.
 - **This must be provided as a single, professionally structured technical report in PDF or Word DOC/DOCX format; other documents – separate diagram files, a ZIP-file, etc, will not be marked**
- **GROUP:** Source code via a private source code repository (e.g. GitHub, GitLab) setup and shared with module leader (**guest access is not sufficient**).
 - Your software (both versions) should provide a repeating menu that offers access to all features you have implemented. Features **must** be identified using the Feature letters from the spec (i.e. A, B, C, etc)
 - Your Repository and (Eclipse) project **must** be titled as follows: **2021-4121COMP-yourgroupname** (where *yourgroupname* is replaced by your group's name)
 - The application entry point (i.e. *where is your application's main method*) must be clearly identified in a README.TXT/MD or similar file in your project's root folder
- **INDIVIDUALLY:** each team member, an **individual** groupwork and peer reflection report as outlined above: (using provided template; see separate document)
- **GROUP:** A group project and software demonstration; format details of which will follow in Session 9

Failure to submit your individual report will count as a non-attempt in this assessment.

Please ensure that you are each familiar with how you are expected to submit each of these deliverables in ample time for the deadline. If you have any doubts or questions, please ask.

Marking Scheme/Assessment Criteria

Task	Assessment Criteria	% weighting
0	Project and Team Management (see Sessions 1, 2, 10)	20
0.1	Team effectiveness (evidenced via e.g. group agreements, meeting logs)	5
0.2	Project management / SDLC effectiveness (evidenced via, e.g. Sprint Planning and Review documentation, copy Backlogs)	5
	Individual group reflection	10
1	Analysis (see Sessions 2, 3)	20
1.1	Functional and non-functional requirements / Brief User Stories	5
1.2	Use Cases and Descriptions	10
1.3	Client Consultations / response to feedback	5
2	Design (see Sessions 3, 8)	20
2.1	Data Design (Class Diagram)	10
2.2	Basic UI Designs / Storyboarding	5
2.3	Behaviour Design (Sequence or Activity Diagrams)	5
3	Implementation (see Sessions 4, 5, 7, 8)	20
	Source Code – functional implementation	15
	Appropriate usage of a source control repository	5
4	Evaluation, Testing and Profiling (see Sessions 6, 7, 9)	20
4.1	Evidence of TDD, unit testing, and/or UAT	10
4.2	Evidence of profiling	5
	Client Demonstration	5

You are also issued a detailed assessment rubric / grading grid alongside this specification, which forms part of the assessment. Your team's Product Owner is **strongly recommended** to use this to regularly self-assess your team's progress, share this with your team, and help to recommend what tasks are to be prioritised.

Glossary

- API – Application Programming Interface
- CSV – Comma-Separated Value (file format that separates fields with a comma)
- JDK – Java Development Kit
- LDB – Live Departure Board
- NRE – National Rail Enquiries
- OpenLDBWS – The National Rail Open Live Departure Board Web Service
- SOAP – Simple Object Access Protocol (used in some web services)
- WSDL – Web Service Description Language (a machine-readable format that specifies the services that a web service provides and how to access them. Tools such as *wsimport* can decode a WSDL description and generate Java code that will connect to and access a service)

Useful Links

Rail timetable information

Northern Timetables: <https://bit.ly/4121NorthernTTs>

Stations CSV Page: <https://bit.ly/4121Stations>

Dynamic (i.e. real-time) rail-transport data sources:

Traintimes LDB <https://bit.ly/trainldb>

NRE LDB <https://bit.ly/NREldb>

Machine-readable dynamic rail live data:

NRE OpenLDBWS API documentation <https://bit.ly/NRE-OpenLDB-Docs>

NRE OpenLDBWS API registration <https://bit.ly/NRE-OpenLDB-Reg>

NRE WSDL page (descriptor) <https://bit.ly/NREWSDL>

Java and SOAP WSDL:

Baeldung tutorial: <https://bit.ly/4121WSDL>

mkyong tutorial: <https://bit.ly/mkyongwsimport>

JDK 8 Download: <http://bit.ly/JDK8Download>

Personal/Extenuating Circumstances: If something serious happens that means that you will not be able to complete this assignment, you need to contact the module leader as soon as possible. There are a number of things that can be done to help, but we can only arrange this if you tell us.

To ensure that the system is not abused, you will need to provide some evidence of the problem.

More guidance, including on the Personal Circumstances process, is available in the appropriate Academic Framework document, available here:

<https://www.ljmu.ac.uk/about-us/public-information/student-regulations/guidance-policy-and-process>

Any coursework submitted late without the prior agreement of the module leader will receive 0 marks.

Academic Misconduct: The University defines Academic Misconduct as ‘any case of deliberate, premeditated cheating, collusion, plagiarism or falsification of information, in an attempt to deceive and gain an unfair advantage in assessment’. *This includes attempting to gain marks as part of a team without making a contribution.*

The Faculty takes Academic Misconduct very seriously and any suspected cases will be investigated through the University’s standard policy. If you are found guilty, you may be expelled from the University with no award.

<https://www.ljmu.ac.uk/about-us/public-information/student-regulations/academic-misconduct>

It is your responsibility to ensure that you understand what constitutes Academic Misconduct and to ensure that you do not break the rules. If you are unclear about what is required, please ask.