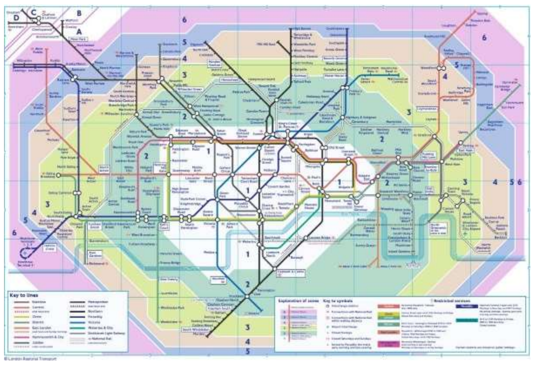
Southampton Solent University
Coursework Assessment Brief

Assessment Details

Module Title:	Object Oriented Design and Development
Module Code:	COM528
Module Leader:	Dr Craig Gallen
Level:	5
Assessment Title:	Practical Assignment – Underground Train Ticket System
Assessment Number:	AE2
Assessment Type:	Software Development
Restrictions on Time/Word Count:	N/A
Consequence of not meeting time/word count limit:	There is no penalty for submitting below the word/count limit, but students should be aware that there is a risk they may not maximise their potential mark. Assignments should be presented appropriately in line with the restrictions stated above; if an assignment exceeds the time/word count this will be taken in account in the marks given using the assessment criteria shown.*
Individual/Group:	Individual
Assessment Weighting:	60%
Issue Date:	Week Commencing 5th October 2020
Hand In Date:	COM528 6th January 2021, 4:00 PM (16.00 local time)
Planned Feedback Date:	Within 4 weeks of submission
Mode of Submission:	on-line
Number of copies to be submitted:	1
Anonymous Marking	This assessment: is exempt from anonymous marking.

Assessment Task

Case scenario (London Underground Ticket System)



The London underground is a rapid transport system serving Greater London.

Different charges apply depending upon the time of day (peak and off peak), on the day of the week (e.g. weekends and bank holidays are off peak) and the number of zones travelled. Tickets are issued which allow travel within1,2,3,4,5 or 6 zones. The number of zones travelled depends on the start and end destination.

All charging rates are subject to review and the ticket machines must regularly download their ‘rate schedule’ from a central server. Each ticket machine is identified by an ID number and will download its rate schedule based upon its ID.

Customers (i.e. passengers) must purchase a ticket for the journey before they depart. They can only use credit cards with ‘touch and go’ for payment (i.e. no pin required). Credit cards transactions are validated against a bank service and if approved, a ticket for the duration and destination should be issued.

Customers can use their ticket to enter and exit the system using ticket gates.

You are required to create a ticketing system consisting of a Controller, Ticket Machines and Gates which allow:

- Station managers to use the controller to add new ticket machines and gates to the system at given stations.
- Station managers to use the controller to create time schedules which set the charging scheme for ticket machines in the system. Ideally, they would use a GUI to modify these schedules.
- Users to select their origin and destination stations and calculate the charge based upon the day of the week, time of day and number of zones travelled. The machine should issue a ticket with a validity time, allowed zones and price and a validation code which checks the ticket is valid.
- Users to validate their ticket at gates on gated entry and exit using the validation code (to simulate the ticket reader). If the validation code, time and zone do not match the ticket should be rejected.

Consider the following features in the development of the system:

(a) Central Controller Service

- Use a java ReST technology (e.g. JAX-RS/Jersey) to implement a simple central schedule server. Ticket machines should request the schedule for their location based upon their ID.
- As a minimum, the manager should be able to provide an xml file with the stations, zones and schedule for the system. However for more credit you may implement a web UI to modify the central schedules.

(b) Ticket machine

- Use a Java Web Technology (e.g.JSP/Tomcat) to implement the ticket machine.
- The ticket should on a regular cycle use a ReST service to request an updated charging schedule. This information should be persisted locally.
- The ticket machine should show the destination stations and allow the user to select a destination.
- The ticket machine should show the total price for the journey selected and allow the user to present a credit card. (You may enter the number in your demo. There is no need to simulate the bank transaction.)
- All transactions and parking tickets issued should be logged in a separate transaction log
- Each ticket should have a unique id, date of issue number of zones and duration and should be displayed for the user.

(c) Gate Service

- The gate service simulates a gate card reader. Use a single JSP to allow entry of a ticket validation code and details. If ticket is valid, in the valid time period and for the correct zone, the gate should open.

Design Process Requirements

(a) Use Cases and Test plan

You should document use cases for each actor in your system. These use cases should also be reflected in any integration and unit tests you create for your design. You should create a simple test plan (which can be manually performed) which illustrates the correct functioning of your use case. You will be asked to perform a selection of these tests as part of your presentation of your system.

(b) Model

You should construct a model for your system documenting the core data types and interfaces you will need to implement for each component. Ideally you should use this model to generate the key interfaces and data elements but even if this is not possible, your programmatic elements should correspond closely to the model you have created.

(c) Data Flow diagrams
You should draw data flow diagrams to document how data is passed between the entities in your system including external entities such as the bank.

(d) Implementation

- You may use the skeleton multi module maven project introduced in class to give your project structure.
- You should provide unit tests for each layer/module in your design
- You should use a logging framework to help with debugging
- You should document all your classes with Javadoc

(e) Packaging and Handing in

- Submit your code and your report in a zip file on SOL BEFORE the submission date
- All of your submitted code must be maintained in a project on Github. You MUST TAG your code using Git on Github as a submitted release and you must submit the tagged version on SOL.
- You should apply an appropriate open source licence and copyright to all of your code
- You should include instructions for a user to build and run and use your submission.
- You should include documentation describing the key features of your solution

(f) Report
Your analysis, design, testing and code artefacts must be accompanied by a report. This should include discussion of the following (half a page to a page for each - I am not looking for reams and reams of text!):

- Decisions you made when drawing up your domain model and use-case texts
- Any decisions made when drawing up your diagrams
- Detail on places where your code did not match your design, and why.
- Rationale for your test strategy and test plan
- Critical evaluation of your code and/or design.
-

(g) Demonstration

- A demonstration of your software will follow submission on an agreed date with the tutor
- Please note the demo represents 20% of the overall grade for this assessment and you will only be graded based on tutor’s understanding of your software and report if absent for a demo.

Assessment criteria

	A1-A4	B1-B3	C1-C3	D1-D3	F1-F3
Analysis and design (20%)	Work fully complete; additional considerations beyond the basics have been made in your design. Analysis and design artefacts all consistent with each other. At least over 10 use-cases presented	Work complete, analysis and design correct and artefacts all consistent with each other (a small number of inaccuracies or inconsistencies are permissible for a lower B). At least, up to 10 uses cases are presented	Work complete, diagrams predominantly correct and consistent with each other, but with a number of inaccuracies. Over 5 use-cases are presented	Work mostly complete; significant inaccuracies and/or inconsistencies in your analysis and design. Up to 5 use-cases presented	(F1) Some parts of the analysis and design completed, but others incomplete. (Lower F) minimal effort.
Implementation (20%)	An implementation of all specified use cases which makes use of the more advanced implementation technologies covered in the unit. Robust error handling and a user-friendly interface. UI for entering schedule also attempted Implementation matches design, or if not, the reasons for this are explained clearly in the write-up.	An implementation of all the specified use cases. There may be room for improvement in your error handling. Some evidence of use of the more advanced implementation technologies covered in the unit. Implementation matches design, or if not, the reasons for this are explained clearly in the write-up.	At least three out of five use cases implemented. Little error handling. Little evidence of use of the more advanced implementation technologies covered in the unit. Implementation matches design, or if not, the reasons for this are explained clearly in the write-up.	At least 5 use cases implemented. Implementation matches design, or if not, the reasons for this are explained clearly in the write-up.	A minimal effort; up to one use case successfully implemented.
Testing **(20%)**	Comprehensive range of JUnit tests	A wide range of JUnit tests undertaken. There may be a small number of omissions	Significant number of JUnit tests undertaken but with a number of omissions	A small number of JUnit tests undertaken with significant omissions	Little or no testing undertaken
Report (20%)	Clear justifications of decisions made when drawing up your analysis and design artefacts including insightful comments. Considerations beyond the basics are made. Clear rationale for	Clear justifications of decisions made when drawing up your analysis and design artefacts. Clear rationale for tests.	Largely clear justifications of decisions made when drawing up your analysis and design artefacts but unclear at times. Rationale for tests mostly clear.	Write-up clear and accurate in some places but unclear and/or inaccurate in others. A significant number of omissions.	Predominantly unclear and/or inaccurate write-up. Little understanding demonstrated.

	tests.				
Test plan and Demo 20%	Good test plan and demo of all features	Good test plan with a demo of most features	Good test plan with demo of some features	Demo of some parts of the solution. May be stubbed features	No working demo

Learning Outcomes

This assessment will enable students to demonstrate in full or in part the learning outcomes identified in the unit descriptors.

Late Submissions

Students are reminded that:

i. If this assessment is submitted late i.e. within 5 working days of the submission deadline, the mark will be capped at 40% if a pass mark is achieved;
ii. If this assessment is submitted later than 5 working days after the submission deadline, the work will be regarded as a non-submission and will be awarded a zero;
iii. If this assessment is being submitted as a referred piece of work (second or third attempt) then it must be submitted by the deadline date; any Refer assessment submitted late will be regarded as a non-submission and will be awarded a zero.

http://portal.solent.ac.uk/documents/academic-services/academic-handbook/section-2/2o-assessment-policy-annex-1-assessment-regulations.pdf?t=1411116004479

Extenuating Circumstances

The University's Extenuating Circumstances procedure is in place if there are genuine circumstances that may prevent a student submitting an assessment. If students are not 'fit to study', they can either request an extension to the submission deadline of 5 working days or they can request to submit the assessment at the next opportunity (Defer). In both instances students must submit an EC application with relevant evidence. If accepted by the EC Panel there will be no academic penalty for late submission or non-submission dependent on what is requested. Students are reminded that EC covers only short term issues (20 working days) and that if they experience longer term matters that impact on learning then they must contact a Student Achievement Officer for advice.

A summary of guidance notes for students is given below:

http://portal.solent.ac.uk/documents/academic-services/academic-handbook/section-4/4p-extenuating-circumstances-procedures-for-students.pdf?t=1472716668952

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. The University's Academic Handbook includes the definitions of all practices that will be deemed to constitute academic misconduct. Students should check this link before submitting their work.

Procedures relating to student academic misconduct are given below:

http://portal.solent.ac.uk/support/official-documents/information-for-students/complaints-conduct/student-academic-misconduct.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 an ethics release or an ethical approval prior to the start of the project.

The Ethics Policy is contained within Section 2S of the Academic Handbook:
http://portal.solent.ac.uk/documents/academic-services/academic-handbook/section-2/2s-university-ethics-policy.pdf

Grade marking
The University uses a letter grade scale for the marking of assessments. Unless students have been specifically informed otherwise their marked assignment will be awarded a letter grade. More detailed information on grade marking and the grade scale can be found on the portal and in the Student Handbook.

Policy: http://portal.solent.ac.uk/documents/academic-services/academic-handbook/section-2/2o-assessment-policy.pdf

Guidance for online submission through Solent Online Learning (SOL)

http://learn.solent.ac.uk/onlinesubmission