WebApps Group Project

Second Year Group Project
Department of Computing
Imperial College

Summary

Your task is to implement a multi-user web program or app that maintains a consistent internal state via use of a database. Examples of suitable programs include online-games, collaborative publishing tools and any multi-user app. You are encouraged to think outside of the box and try to come up with something original.

Requirements

You are free to design your program in any way you wish, however it must fulfil *all* of the following requirements:

- Your program must be multi-user and should allow for rich user interactions (e.g. messaging, scoreboards, collaboration).
- Your program must run on at least one of the Internet browsers installed on the lab machines (e.g. Firefox, Internet Explorer, Chromium, Konqueror). Alternatively, if you are designing an app, it should run on one of the devices available on loan from the lab. Ideally your app should be accessible on at least one of the browsers as well.
- Your program must use a database to maintain some internal state, such as user specific data.
- Any additional plugins required for your program must be freely available and users should be clearly told how to install them.
- You must not break the college or the department's rules for the use of computers (these can be found on the college website).
- You must not break any copyright laws by using third party code or content without consent.

If this is your first time building a web program or mobile app you may wish to treat the following suggestions as additional requirements:

- It is suggested that you use either the department's main Apache 2 server 'www.doc.ic.ac.uk', a local Tomcat 6 server tomcat6, or the Microsoft ASP/IIS server 'www-asp.doc.ic.ac.uk'. You are free to use your own servers, or external servers, but CSG will not provide support for these servers.
- It is suggested that you use the postgresql database provided to you by CSG.
- It is suggested that your database code work on at least one of postgresql or Microsoft SQL Server database.
  Note: while the Microsoft database and web server work together, you can expect problems trying to use them with non-Microsoft software.

An Example Game

To give you some ideas, we provide an example setting and some basic rules for a web based trading game. This should be treated as a suggestion only, you are free to design your program in any way you wish. Your program does not need to be a game. Be creative!

Setting: It is the dawn of a new age of space travel. Scientists at Imperial College London have developed a revolutionary jump drive that allows a ship to travel light-years in mere days. Mankind has spread out into the stars and colonised a number of nearby star-systems. Following in their wake, a wave of enterprising merchants have taken to the stars hoping to make their fortune trading with the new colonies.

Map: The game is played among the 10 star-systems nearest to Earth, each a number of light-years away. Each star-system has a type: Agricultural, Mineral, Industrial or Spice. Sol (our solar-system) always starts the game as an Industrial system. In total there will be 3 Agricultural systems, 3 Mineral systems, 4 Industrial systems (if you include Sol) and 1 Spice system. The type of each star-system determines the resources it wants and the resources it produces. Star-system types are discussed further under 'Trading'.

Turns: Each player plays the game in turns, but there is no synchronisation of turns between players. On a turn a player can either make a transaction (with a star-system or player), move one of their spaceships to another star-system, or resign from the game. Each turn a player consumes 1 unit of food (unless they resign from the game). Note that if a player had insufficient resources for an action, they will be unable to perform that action.

Spaceships: The only way to move resources between star-systems is to use a spaceship. A spaceship is equipped with jump engines that let it move rapidly through space. Making a jump between two star-systems takes only one turn, but consumes 1 unit of fuel per light-year covered. Spaceships have enough cargo space to carry 50 units of resources (food, ore, tools and spice). They also have fuel tanks that can hold a maximum of 50 units of fuel.

Player Setup: Each player starts in the Sol system with their spaceship, 100 Imperial Credits (100IC), 20 Food, 10 Tools and 25 Fuel.

Communication: Any player can send a message to one or more players. These messages are stored, and it is up to each player to check their own messages. Messages can be used to chat, make offers and negotiate trade prices. Sending messages does not count as a turn.

Trading: Players can trade with the star-system they are visiting, or with other players also in that star-system. In order to trade, both players involved have to be playing. Trading between players can take any form, players may even gift resources, credits and fuel to one another if they wish. Trading between players and star-systems is always performed at a fixed rate as described in the table in Figure 1.

Victory Conditions: A Player loses the game if they run out of food or fuel and are unable to acquire more in their current star-system. The winner is the last player left in the game, or the richest player still in the game when an agreed end point has been reached. This can be after each player has had a set number of turns, after some time-limit is reached, or by agreement of all players in the game.

The Database

As mentioned in the requirements, you will need to track the state of your game in a database. You will have been allocated a database in your project area that you should modify to include all of the tables needed to support your game. To access the database you will need to use the password allocated by CSG for the group (details of this will be in a group setup e-mail sent to you be CSG).

An example database structure is provided in the file `trader.sql` which you can download from Piazza and is also available at `http://www.doc.ic.ac.uk/~mjw03/webapps/trader.sql`. This file contains definitions of the tables to create in the database and the commands to load data into these tables. You may find it helpful to read and edit this file to suit your own purposes. To load the table definitions and initial data specified by the `.sql` file start `psql` specifying your database name, and from the `psql` prompt type `\i trader.sql`.

The database server is regularly backed up, but if you want to save the database before making a major change you can use the `pg_dump` command to create an `.sql` file that can be used to restore the database later. The command,

```
pg_dump --host=db --inserts *ourdatabsee* --file saved.sql
```

Agricultural:	1 tool	→	5 food
	10IC	→	1 food
	1 tool	→	10IC
	1 spice	→	50IC
Mineral:	1 food + 1 tool	→	5 ore
	20IC	→	1 ore
	1 tool	→	10IC
	1 food	→	10IC
	1 spice	→	50IC
Industrial:	1 food + 1 ore	→	5 tools
	1 food + 1 ore	→	20 fuel
	50IC	→	1 tool
	5IC	→	1 fuel
	1 food	→	10IC
	1 ore	→	10IC
	1 spice	→	50IC*
Spice:	1 food + 1 ore + 1 tool + 1 fuel	→	1 spice
	1 food	→	10IC
	1 ore	→	10IC
	1 tool	→	10IC
	1 fuel	→	10IC
	1 spice	→	50IC

*spice traded with Sol is worth an extra 100IC (for a total of 150IC).

Figure 1: Trading Rates with Star-Systems

writes the commands and data that will recreate '`*ourdatabase*`' to the file '`saved.sql`'.

The SQL standard does not include any commands to change the definition of an existing table in a database. Postgresql has an extra command `ALTER TABLE` that allows you to add or remove a column in a table or to change names and types in a table. More information about postgresql can be found in the postgresql documentation at `http://www.postgresql.org/docs`.

An Example App

To give you some more ideas, we provide an example app description and some basic user interactions for an exam invigilator communication system. As before, this example should be treated as a suggestion only. You are encouraged to come up with your own ideas.

Purpose: During the main summer exam season, due to extra time candidates and the popularity of certain courses, it is common for candidates of an exam to be split over multiple rooms. It is important that the invigilators in these rooms be able to communicate with one another in order to synchronise start/finish times, announce clarifications/corrections and enable examiner's to answers queries from candidates. This can be particularly troublesome in rooms with just a single invigilator. Mobile phones can sometimes be useful, but not everyone has their phone on them, not everyone knows one-another's numbers and people's phones will necessarily be on silent and so messages can easily be missed. We suggest an app that allows for a phone or tablet in each room to be connected to a central communication server that relays messages and status information silently between the rooms.

Requirements: Here is a partial list of requirements that the exams communication system would need to fulfil:

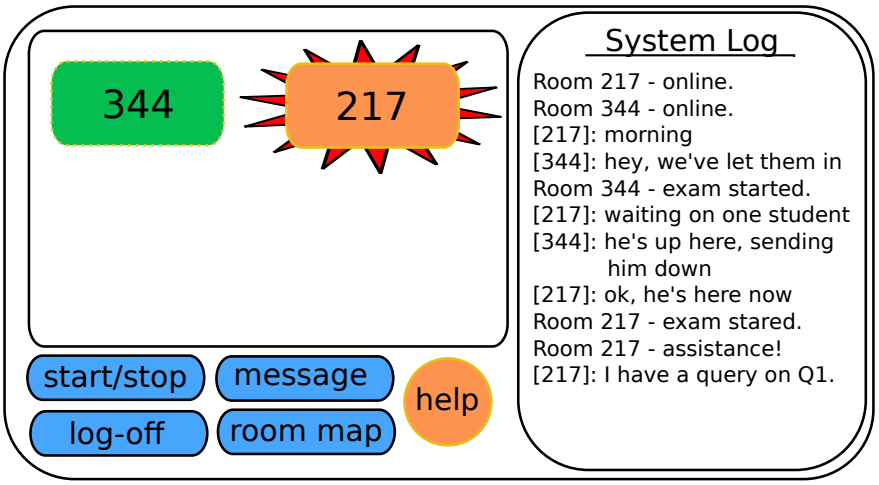
- The system must operate silently. Candidates must not be disturbed during their exams.
- The system should make continuous alerts so that messages are not 'missed'.
- Messages need to be receipt confirmed so that the sender knows that their message has been received.

- Each room should have a status (e.g. stopped,under-way,assistance) displayed at all times during the exam. These status's need to update in real-time across the system.
- The system should display a log of status changes and messages.
- The system needs to be able to handle that examiners may be moving between rooms at any time during the exam.

User Interactions: Here is a partial list of user interactions expected to be supported by the system:

- Examiners/invigilators should be able to see at a glance which room(s) currently need assistance.
- An invigilator needs to be able to notify the system when the exam in their room starts/stops. The room's status should be appropriately updated.
- An invigilator needs to be able to ask for an examiner to come to their room to answer a question. Note that there may be multiple examiners for a single exam, and that multiple exams may be running in a single room.
- An invigilator needs to be able to ask for an invigilator to assist them in their room (e.g. a candidate may need to be escorted to the bathroom).
- An examiner/invigilator needs to be able to acknowledge that they have received a request to come to a particular room. Additionally this receipt should be broadcast so that requests are not over-serviced.
- An examiner needs to be able to ask for an announcement to be read out in all rooms that contain a certain exam. Each room should be required to acknowledge when the announcement has been made.

Screen Shot Mock-up:



As with the previous example, the system information will need to be stored in a database.

Project Management

A portion of the assessment will focus on how your group is organised. Moreover, a well organised group will be able to do more and produce a better final product. You should divide the work amongst your group members and have a clear group structure, in particular there should be a group leader. Be sure to keep a log of what each group member is doing as you will find this useful for your final report.

Version Control

It is important that you backup and manage your group's code using a concurrent version control system, such as `git` or `svn`. This will allow you to more efficiently work together as a team and also to maintain a working version of your project at all times. Note that as part of your 'Milestone Report' you will be assessed on your use of a version control system. You can find CSG's version control guide at `http://www.doc.ic.ac.uk/csg/guides/version-control`.

Testing

It is important to test your work regularly. Make sure that your program produces the expected output, but also make sure that it can handle unexpected inputs gracefully. It might be a good idea to get your friends to try out your program, or to try out each others' programs. You should find and use a unit testing library for the language you choose to work with.

Working Away from the Department

If you choose to work remotely on your project be sure to keep in mind the following:

- Don't spend too much time trying to install and debug software on your machine that is already available on the lab machines.
- Keep in regular contact with your group so that you remain on task and avoid duplicating work.
- Make sure you back up your work regularly. You should also use a version control system as mentioned above.
- Make sure that any software you use is available in the department (for the demonstration) *before* you start using it.
- If you choose to use another database (such as MySQL) avoid using non-standard SQL so that your program can be converted to using Postgresql.
- Test your work on the department's server and allow time to find and fix bugs *before* your demonstration.

Timetable and Assessment

The timetable for the WebApps project, and the weighting of marks, are given below:

Date	**Task**	**Mark %**
28th March	Organise your Group	–
28th May	Milestone Report	10%
11th June	Project Report	30%
16th - 20th June	Demonstration and Presentation	60%

Organise your Group

Your first task is to set up a group of *three or four* people for the project and choose one person to be the group leader. Note that you do not need to make a submission on CATe, just make sure you set up your group in the normal way.

The groups created on CATe will be used to assign shared group directories for the WebApps project. By 4th April you will receive an e-mail from CSG telling you the location of your group project directory and database, along with how to access them.

Some introductory notes on web based programming, taken from Rob Chatley's Software Engineering course, can be found on Piazza as well as at the following urls:

- `http://www.doc.ic.ac.uk/~mjw03/webapps/webappsIntro.pdf`

- http://www.doc.ic.ac.uk/~mjw03/webapps/distributionIntro.pdf

CSG also provide a guide on web hosting http://www.doc.ic.ac.uk/csg/services/web and a guide on database usage http://www.doc.ic.ac.uk/csg/services/databases which you may find helpful during the project.

Milestone Report

The milestone report exists to make sure that you have made a start in your project. You will need to write a short document, explaining your implementation strategy, a description of your program/app and the expected user interations, and submit this to CATe. You will also be asked to demonstrate to a lab helper that you are using a version control system and have constructed a prototype database that can store user specific data and update a web-page.

More details of what is required for this report can be found in the 'Milestone Report' specification available from Piazza and also at http://www.doc.ic.ac.uk/~mjw03/webapps/milestone.pdf.

Project Report

Your group must submit to CATe a report of between 5 and 10 pages on your project. The report can be produced using any software you wish, but must be machine readable (scanned handwritten reports will *not* be accepted). Documents generated using LaTeX are the preferred format (you can find a guide for getting started with LaTeX at http://www.maths.tcd.ie/~dwilkins/LaTeXPrimer/).

Your report should:

- explain your choice of implementation languages;
- provide an overview of the implementation (describe the structure of your program and its user interface);
- acknowledge any third party resources that were used in developing the project;
- and include a conclusion describing what you have learned and what you would have done different or would change if you had more time.

More details of what is required for the final report can be found in the 'Project Report' specification available from Piazza and also at http://www.doc.ic.ac.uk/~mjw03/webapps/report.pdf.

Demonstration and Presentation

Your last task is to give a combined demonstration and presentation of your program and how it was designed/implemented. The presentations will take place in week 8 of term on 16th - 20th June. Each presentation should be no more than 25 minutes long and *all* members of your group should be present and should talk at some point in the presentation. The presentation should cover the same material as your project report, but you should also demonstrate your program/app being used by two or more users.

More details of what is required for the presentation can be found in the 'Demonstration and Presentation' specification available from Piazza and also at http://www.doc.ic.ac.uk/~mjw03/webapps/presentation.pdf.

Details of presentation times and rooms will follow closer to the time.