# Overview

You will write a client-server software system that demonstrates how a database system normally operates in a network environment. Generally, a database system runs as a service that responds to requests from an arbitrary list of sources. It is up to the database system to make sure data is secure and integrity is maintained. **For example, only those sources that are permitted should access or update the data; and only one source can update data at a time – even though several sources may be requesting to update at the same time.**

## Primary Goals

* Practice and enhance your file I/O basics, data validation and error handling
* Simulate a database service, and discover key problem areas that dedicated DBMS’s address

## Secondary Goals

* Document a software assignment
* Demonstrate the operation of a software project that you developed to a stakeholder

# Mandatory Functional Requirements

1. The database file consists of records with the following fields:
   1. MemberID (as an integer)
   2. FirstName (as a string)
   3. LastName (as a string)
   4. DateOfBirth (as a Date format supported by your OS)
2. Write a server program that will run continuously, listening for requests to write to (or read from) the database. There are only 3 requests that the server can respond to:
   1. INSERT - allows the insertion of new data. The data should be provided only as the FirstName, LastName and DateOfBirth. The MemberID is automatically generated and written to the file with the rest of the data. The automatically generated MemberID must be sequential. The server should only handle up to 40,000 records. This means that IDs of 1-400,000 would be valid. If the INSERT command is not successful, an error should be returned to the client.
   2. UPDATE – allows the modification of existing records. The client must provide a valid MemberID, FirstName, LastName and DateOfBirth. Even though the data may not change, all four values should be in the parameter list. If the UPDATE command is not successful, an error should be returned to the client.
   3. FIND – allows a client program to get the information of a specific MemberID. The server should return all four data fields. If the FIND command is not successful, an error should be returned to the client.
3. Bulk Data Client - Write a client program that can be run multiple times concurrently from one or more computers. The purpose of this program is to randomly create the data that will be written to the database and to demonstrate that you can handle multiple requests simultaneously.
   1. This client must contain the logic for generating ‘random’ sample data. In the interests of time, there are no hard and fast constraints on how you generate 400,000 sample records… you could procedurally generate all fields or use one of many sample files on the Internet (please cite sources in documentation/code comments!).
   2. For our purposes, you should focus on running the client app from just one machine. If you venture down the path of socket-based communications over the network, it could boost your discretionary functionality mark for this assignment, but you’ll not be penalized if you don’t include TCP based communications.
4. Interactive Client - Write a client program that can query the database (use the FIND command) and will allow you to update a specific record.

# Discretionary Functional Requirements (Select 2 of these below)

1. Doxygen/Wiki style Documentation – Provide a comprehensive, electronic documentation resource for your project. Use of Doxygen (or similar) is HIGHLY encouraged.
2. LAST MINUTE REQUEST! Although scope creep is a terrible thing it happens. In our scenario, a Product Manager in our company promised our client that this file-based data management system would have ‘dashboard’ functionality that would show an administrator several important statistics about this application. Sadly, this is all we got from that Product Manager before he flew off to Tokyo for a conference.
3. Use a new language! Develop all the same, required functionality, just do it in a language/platform we haven’t covered yet in SET (E.g. C#/.Net, Java, Python, etc.)
4. Advanced Mandatory Functional Requirements – Is there some way that you think you can you can demo a significantly better deliverable than those listed in the previous section? If so, talk to me to get an OK/approval for your idea, and run with it!
   1. This would include client/server functionality over TCP/IP demonstrated on multiple PC’s during demo day.
5. An idea so crazy, it just might work! – I am happy to listen to any other great idea you have to improve or add onto this project. Please avoid disappointment on demo day, by discussing the idea with me and getting approval first.

# Demonstration Day

As you progress through SET, you will find that we rely less on multiple choice and short answer exams, and more on demonstrations/investigations of your functioning software product. After all, what better way to determine how you are progressing as a software developer than to assess your functioning product?

We will assign one of our class periods (refer to schedule/Instructional Plan) for the demo day. It is VITAL that you are present for your demo on this day. Absences on this day will only be dealt with through official college policies for missed assignments (e.g. documentation from a physician, etc.).

The demo day may be stressful, and it may also be a lot of fun. Ultimately it should be VERY informative in terms of the concentrated feedback you are going to get on your application. This is extremely valuable, so do your best to be ready to go on demo day.

## What gets marked on Demo Day?

We will be marking most of your assignment 1 on this day! While watching you and/or your partner progress through these steps, I will be able to provide feedback and a grade on MOST of the assignment’s components. After demo day, I will finalize marks by reviewing documentation, code comments, and the like, so do not neglect those other, non-demo’ed components.

## How will a demo be conducted?

1. Server and client programs all launch/run from compiled releases. Do not run either of these from your IDE/Visual Studio or penalties may be applied to the grading.
2. Demonstrate that your DB file is ‘clean’ as we start (e.g. prove you have nothing in your DB as we start the demo.
3. Run your Bulk Data Client to populate the DB with sample data.
   1. Optional – You can do your concurrency test here (see below) proving that your solution easily handles multiple clients’ requesting to bulk insert data. For this purpose, it is fine to go beyond 400,000 records.
4. Demonstrate your Interactive Client
5. Concurrency and Error Test – Demo your solution’s ability to manage two clients simultaneously sending requests to the server. Pick one:
   1. Clients hit the same record for an update – How do you protect the data from problematic conditions and changes (e.g. Race conditions, deletion anomalies, etc)
   2. Two or more Bulk Data Clients quickly, and easily get their data loaded to the DB (see above).
6. Demo any other requirements (e.g. Discretionary items).

# What to submit to the Assignment dropbox

**NOTE** – The due date for this assignment is set to just before our demo period will start. All items requested below must be uploaded prior to this target time, otherwise late penalties will begin to accumulate.

Please follow SET Submission standards to clean/prepare your source code and/or project. If you did your project in Visual Studio, for example, the Submission Standards review how to clean your project up before zipping. This allows me to easily conduct a review of your code to finalize marks for this assignment.

* All source code/project files
* Documentation (PDF, Word, or HTML files/folder if you did something like Doxygen)
* Any supplemental ‘README’ style notes that will help me to assess your project. This might include notes about where you sourced your random names list or even ‘Installation and Operation instructions’ that would be important to me to run your project on my own to assess some functionality that was missed in demo.

**Please review the additional information in eConestoga concerning this assignment’s marking scheme.**