

## List of project titles with short explanations

### **Matching potential project supervisees with potential supervisors**

Specification, design, implementation and testing of a web-based system for matching potential project supervisees with potential supervisors

### **An assumption based truth maintenance system**

Specification, design, implementation and testing of an assumption based truth maintenance system in an object oriented language.

Language: Java or C#

### **Teaching and learning with dynamical graphical systems**

Specification, design, implementation and testing of programs to support teaching and learning with dynamical graphical systems - e.g. binary search trees expanding and contracting, 8 queens problem.

### **A comparison of the JVM and CLR**

The JVM and CLR are two alternative virtual machines on which to execute high-level programming languages. I have already supervised a very successful project on a comparison of the two, and this project will take forward the previous student's work. One possible area to focus on is the provision for exception handling. This should suit anybody who enjoyed machine code and/or compiler design.

### **Graphical animation of Z schemas**

This involves turning specifications into implementations and executing them graphically, in a manner that may be educationally helpful to students of Formal Specification. This first part is difficult in general but tractable for simple specs of the sort encountered on that unit. You will need to learn about both parsing and LaTeX.

### **Implementation of functional and logic programming languages**

How is Mercury (a modern variant of Prolog) compiled for .NET? Are there any disadvantages in targetting the CLR instead of the Warren Abstract Machine traditionally used to execute Prolog? Similar questions may be asked about F# and Haskell.NET.

### **Lazy Evaluation**

Some people identify lazy evaluation as one of the best features of functional programming languages like Haskell. But just how difficult is this to simulate in an ordinary object-oriented language like Java? Similarly, do delegates allow C# programmers the advantages of higher-order functions?

### **Programming paradigms**

How easy is it to program in one paradigm using a language designed for another? How useful are languages advertised as multi-paradigm (like Python or Pizza) compared to languages that are single paradigm? The project will involve writing the same program in different languages, selecting from the menu of available language features, and comparison using some criteria.

### **Modelling and simulation**

Modelling and simulation (e.g. of real world systems) using discrete event simulation

**Use of mobile technology for data collection in remote locations** This project involves solving the problem of government organisations in remote areas without access to the internet being able to send data to their headquarters. For example nurses in a remote clinic being able to send data on patients treated in their clinic. As this will involve significant amounts of detailed data it is not possible to send the data by simply entering the data into a mobile phone, and there would not be sufficient funds available for purchase of more advanced mobile phones.

Therefore the requirement is for mobile technology to be used to send data that has been captured in a desk top/lap top to a server in an organisation's headquarters.

### **Developing a tool for integration of software systems**

Organisations use a number of different software applications and tools, most commonly accounting packages and customer relationship management systems. Integrating these systems can be a time consuming and expensive process. In the government sector, the complexity and costs are more extensive.

The project involves developing a tool that can be used for linking any two or more systems by identifying the structure of the data in the two systems and developing a tool for transfer of the data between the two or more applications

### **Developing a tool for designing software business process requirements**

The project involves developing a tool to design and develop software. The tool would enable a user to:

1. Define the problem being solved by the software and present this as a graphical interface
2. Take the problem and solution interface and develop a business process flow, data model and database design
3. Take the business process flow and create screen mock-ups.

**Mobile Phone Application Development** - This programming project will require Android programming to allow a mobile phone to receive and process sensor data transmitted using Bluetooth.

### **Data Structures Teaching Tools**

Many applets on the web exist for illustrating data structures concepts, (e.g. AVL trees, B-trees, hash tables). However, many are more concerned with functionality than usability, and are thus not as usable for teaching as they could be. With a teaching tool it's important that any tool is very usable, with good instructional design to support students' learning objectives. There many possibilities for projects to develop good tools for learning about data structures.

### **Educational software for mathematics**

The advent of computers in the classroom allows many mathematical concepts to be explored and illustrated in different ways. Software can be developed for such topics as fractions, angles, vectors, geometric transformations, or other topics from the curriculum.

### **Individualising datasets for statistics assignments**

Producing sets of data for assignments on statistics modules can be a tedious task, even more so is marking such work. This project would involve writing software to produce and mark such datasets, one for each student. This will involve programming, implementing some standard statistical algorithms, and could involve using a variety of other computing techniques, such as human-computer interface design, interfacing with other statistical packages like SPSS or Excel, email APIs.

### **Developments in case-based reasoning**

A case base of solution outcomes to complex, multi-parameter problems can be used as the basis of a computerised system for improving the solutions process. A similarity function can be used to identify the correlation between the case base parameters and the new situation parameters.

This project would select an area of expertise in which to develop a case base and reasoning mechanism for an expert system.

Required: an interest and expertise in AI and knowledge-based systems.

### **Improvements to web / database interfacing**

Interfacing a web site to a server-based database system is a challenging task. A computerised system is envisaged that interactively assists the web designer during the process of setting up the database interfacing commands and protocols.

This complex project would require considerable internet, database and server knowledge to establish a mechanism for user guidance and set-up procedures.

### **Analysis of missing data**

Data is at the core of many computerised analysis systems but often some of the data is corrupted, noisy or missing. The analysis of the data and the missing values is an interesting area as there are many methods for estimating the missing data that can be usefully compared. This project would require a little knowledge of statistics to understand the estimation procedures which would lead to the analysis of missing data using existing and devised programs.

### **Data mining applications in engineering**

Data mining is a powerful tool for extracting information or 'knowledge' about a subject when data sets are available. This project would require a student to have a knowledge of and possible access to engineering data that can be analysed. Methodologies and algorithms can be explored to derive patterns and rules from the data.

### **Public key encryption scheme**

Cryptography: Implement a public key encryption scheme such as RSA or El-Gamal)

### **Investigate the Data Encryption Standard**

Investigate the Data Encryption Standard (DES), perhaps by implementing it (standard code is available) and looking at how it can be broken.

### **An authentication system for secure communication**

Cryptography: Implement an authentication system for secure communication across a network

### **Syntax Checker**

Theory of Computation: Develop a simple syntax checker for a subset of a familiar language, such as Z, B, HTML or Pascal

### **Graphic Editor for Z**

Formal Specification: Create a graphic editor for Z that automatically generates a skeleton specification from a given state schema

### **Computer tools for Management Science**

A tool for solving simple optimisation problems in management science, such as linear programming programs, or critical path analysis.

### **Practical use of nested datatypes and/or efficient folds**

Functional programming: Investigate the practical use of either nested datatypes or so-called efficient folds.

### **Simple games implemented in functional programming**

Functional programming: Implement some simple games

### **Manipulating graphics in Haskell**

Functional programming: Implement a program for manipulating graphics images in Haskell to create PowerPoint-like animation effects

### **Algorithm collections on a particular datatype**

Functional programming: Implement a collection of algorithms (eg sorting) on a particular datatype (eg trees or lists) using higher order functions to illustrate essential similarities.

### **Online tutorial system**

#### **Games to Encourage School Children**

An educational game to encourage schoolchildren to be tempted by the prospect of a computing or maths degree. For example, see the games at [www.musicoftheprimes.com](http://www.musicoftheprimes.com))

#### **Re-usable Machine Learning Components**

Build and test a range of drop in components for Machine Learning (e.g. ID3, Neural Nets, etc.)

Programming language(s): Delphi or Java

Skills Required: Good programming skills

Skills to be learnt: Component production, ML

#### **Re-usable Knowledge Acquisition Components**

Build and test a range of drop in components for Knowledge Acquisition (e.g. Sorting, Rep Grids, etc.)

Programming language(s) : Delphi or Java

Skills Required: Good programming skills

Skills to be learnt: Component production, KA

### **Re-usable Data Visualisation Components**

Build and test a range of drop in components for Data Visualisation (e.g. Bar charts, picto-representation.)

Programming language(s): Delphi or Java

Skills Required: Good programming skills

Skills to be learnt: Component production, Data Visualisation

### **Parallelising k-Nearest Neighbour**

k-NN is a basic ML algorithm. The aim of this project would be to build a parallel version of k-NN using sockets or the PVM or MPI libraries and to investigate dimensionality reduction via sub-space mapping.

Programming language(s): C or Java

Skills Required: Good programming skills, basic networking

Skills to be learnt: Basic ML, MPI or PVM

### **Parallelising Artificial Neural Networks**

ANNs can take a considerable time to train. The aim of this project would be to investigate parallelising strategies for ANNs using sockets, or the PVM or MPI libraries

Programming language(s): C or Java

Skills Required: Good programming skills, basic networking

Skills to be learnt: Basic ANNs, MPI or PVM.

### **Intelligent Firewalling**

Firewalls today are often basic packet filters with little or no intelligence. This project would involve looking at network traffic and dynamically deciding what packets to filter. It would involve the areas of both networks and AI.

Programming language(s): Perl or C

Skills Required: Good programming skills, a knowledge of network protocols

Skills to be learnt: Firewalling, packet analysis, a basic AI technique.

### **Enlarging Tux's dominion**

Take a device that there currently isn't a Linux driver for, or a feature that you would like to see in the Linux kernel and implement it.

Programming language(s): C or C++

Skills Required: Good programming skills, understaing of OS operation.

Skills to be learnt: driver or kernel hacking.

### **Delphi interface to libpcap and/or libnet**

Libpcap and libnet well known and used libraries - libnet creates packets, libpcap captures packets. Unfortunately these libraries are written in C and not easily useable in other languages. This project would involve creating a set of Delphi or Kylix modules that would allow a programmer to easily use these libraries and also some sample applications to demonstrate their use.

Programming language(s): Delphi or Kylix

Skills Required: Good programming skills

Skills to be learnt: Component production, network protocols

### **Cyberkaleidoscope**

This software could display patterns with symmetries other than those found in a conventional kaleidoscope, and could be used as an educational tool for 2-D symmetry.

**Software to manage coach trips**

This software would need to check each child had a responsible adult, issue acknowledgements to parents, etc.

**Argument Recorder**

System to record argumentation that occurs during Open Source development. E.g. you can add propositions arguments for against or relative to those propositions, comments, rebuttal, support of those arguments etc.

**Advice recorder**

Software to record advice given to students in office hour sessions, maybe web-enabled to allow students to access their own notes. Among other requirements would be ability to type up the notes during the consultation so that the student can see what you are typing for them but not the advice you have given to other students. Could also be done as an add-in for Outlook.

**Enforcer**

A program that allows you to specify dependencies between packages and classes in java and checks that your actual code respects those dependencies.