<table>
<thead>
<tr>
<th><strong>Version</strong></th>
<th>1.0</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Print Date</strong></td>
<td>dd/mm/yy hh:mm</td>
</tr>
<tr>
<td><strong>Release Date</strong></td>
<td>dd/mm/yy</td>
</tr>
<tr>
<td><strong>Release State</strong></td>
<td>Initial/Core/Final</td>
</tr>
<tr>
<td><strong>Approval State</strong></td>
<td>Draft/Approved</td>
</tr>
<tr>
<td><strong>Approved by</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Prepared by</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Reviewed by</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Confidentiality Category</strong></td>
<td>Public/Confidential</td>
</tr>
</tbody>
</table>
# Document Change Control

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Authors</th>
<th>Summary of Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
List of Figures

1  Actor Diagram ................................................................. 3
2  Graphical Use Case Diagram ............................................. 4
3  Use Case Scenario Details ................................................ 5
4  Graphical Entity Relation Diagram ..................................... 7
1 Executive Summary [5 marks]

Tell the reader what is the purpose of this document. You should also identify the expected readership, version history, rationale for new version, and summary of changes in each version.

1.1 Scope

Define the product to be developed, what it does (and it does not), why the product is being developed (include a precise description of its benefits, goals and objectives)

1.2 Overview

This subsection contains an outline of the remainder of the document.

2 Introduction [10 marks]

The purpose of this section is to define the environment (hardware, software, human) in which the software will function, expected usage and an overview of the functionality of the software.

2.1 Product Perspective

Does it concern an independent product or is it part of a larger product? In the latter case, the other components should be identified, and the interfaces with those components should be described. This section also contains an identification of the hardware to be used.

2.2 Product Functions

An overview of the functions of the system to be delivered. This should be confined to an overview. A detailed discussion of the functions should be deferred to section 3 to 9 of the requirements specification.

2.3 User Characteristics

An indication of general user characteristics, in as far as these are relevant for the requirements specification. Or simply what are the characteristics of your user base. Experience, training and technical expertise of future users may influence specific requirements of the system to be developed.

2.4 Constraints

An indication of any other constraints that apply. These may concern government regulations, hardware constraints, security regulations, and so on. Again, we are concerned with the rationale at this point. A further elaboration follows in section 3 to 9 of this document.

2.5 Assumptions and dependencies

This does not concern constraints on the system to be developed, but things which may influence the requirements specification if they change. As an example, we may think of the availability of certain supporting software, such as a given operating system or a particular type of hardware. If that operating system or hardware turns out not to be available, the requirements specification will have to be adapted accordingly.

2.6 Deferred Requirements

Requirements which may be delayed until future versions of the system.
3 General User Requirements [15 marks]

This section contains a detailed description of functional requirements. It defines the system from the perspective of each type of user. Start off here with a brief overview of the section contents.

3.1 List of Functional Requirements

Requirements should be labelled and prioritised, and listed in brief, point form and suitably grouped. You should also separate the high level requirements from the low level ones.

3.2 System Architecture

This provides a high-level overview of anticipated system architecture and distribution of functions across system modules. The basic architecture is three tier. In the later sections you describe the middle tier Business Model, the underlying Database Model and the user Interface. In this section consider the different Models in the Business Model and how they interrelate in the interface and database. For instance you may highlight the reusable modules, if any.

3.3 External interface requirements

You describe in this section any external factors that will have an influence on this project. Expands on subsection 2.1.

3.3.1 User interfaces

A description of the characteristics of the user interfaces, such as screen layout, function keys, help functions. In order to support testing, verifiable requirements regarding learning time for the system functions should be included either here or in some subsection of refAttrib (Attributes).

3.3.2 Hardware Interfaces

A description of the logical characteristics of hardware interfaces, for example networking; interface protocols; or screen-oriented versus line-oriented terminal control.

3.3.3 Software Interfaces

A description of software needed, such as a certain operating system or library package. Interfaces to other application software is also discussed here.

4 Functional Requirements: Business Use-Case Model [20 marks]

In the subsections below, a description is given of how the transformation of data in to data out is achieved. The description is given for each class of functions, and sometimes for each individual function. The structure of this section may be improved by grouping the functions into categories, e.g. according to the class of users they support. To a certain extent, the contents of this section can be seen as a description of the solution to the user. This component of the requirement specification is the main starting point for the design phase.

4.1 Actor Diagrams

Introduce the diagram by explaining it in words. Insert an actor inheritance diagram here that illustrates the relationships that exist between the various actor types. This diagram is critical when abstract actor types are included in the Requirements Model. An example follows:
4.1.1 Actor Definitions

Repeat the section below for each actor.

<table>
<thead>
<tr>
<th>Documentation</th>
<th>Brief description of the actor and its role in the system. This description should be no more than a small paragraph and should give the reader an understanding of the role of the actor in the organisation. Mandatory.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aliases</td>
<td>Any other names by which this actor may be known. A simple list is sufficient. Say ‘None’ if there are none.</td>
</tr>
<tr>
<td>Inherits</td>
<td>The ancestors for the actor. Some actors may be specialised types of other actors. A simple list of the names of the ancestors will suffice. Say ‘None’ if there are none.</td>
</tr>
<tr>
<td>Actor Type</td>
<td>Active/Passive and Person/External System. Mandatory.</td>
</tr>
</tbody>
</table>

4.2 Business Use Cases

4.2.1 Use-Case Listing

Cover every use case that this application has been scoped to handle. If it is deemed inappropriate to handle a use case or scenario then highlight the fact and state which external application should. Include the top-level use case diagram here to provide some overall context. Note: Exclude user-interface related scenarios here, defer user-interface to section 6.

<table>
<thead>
<tr>
<th>Use case Id</th>
<th>Use Case Name</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>xxxxxxxxxxx</td>
<td>xxxxxxxxxxxxx</td>
<td>xxxxxxxxxxxxxxx</td>
</tr>
</tbody>
</table>

4.2.2 Graphical Use-Case Diagram

This section presents all the business use-cases of the subject area in a graphical package. Provide the top level view of all your Use Cases for the Business Model here.
4.2.3 Use Case Scenarios

Insert each business scenario definition for your use cases here. **These may be lower level than the top level Use Cases described above.** If you have too many, insert a sample Use Case scenario here and place the rest in the Appendix. Give us a variety of Use Cases. For instance if each top level Use Case from section 4.2.2 breaks down into a View, Edit, Delete and Search use case at the next level down, provide an example "View" Use Case for one module, "Search" for another module, and so on.

**Documentation** Brief description of the business scenario and its context. This description should be no more than one paragraph and should give the reader an understanding of the role of the scenario but not the detail. Mandatory.

**Actors** List the actor names that perform the scenario. Ensure they appear in an actor definition template in section 4.1. Mandatory.

The following sections can be generated by Visual Paradigm

**Super Use Case** Name any Use Case of which this is a subset

**Preconditions** Specify any conditions that must be met before this scenario can be performed. These should be set out as a numbered list. The list of preconditions should be stated in a declarative fashion (eg. ‘the alarm must be on’). This will help put the business scenario in context and initiate test cases. If there are none then say ‘None’.

**Use-Case Text** Narrative of the scenario. The scenario narrative should state the uses and why it is needed for each step of the scenario. Each step should be numbered (eg. 1, 2, 3). When referencing other scenarios, the key word uses, and the scenario name following it, should be italicised. Begin the step with the uses and then explain why it is needed. For example.

1. Use Maintain Customer to update the postal address
2. Calculate commission fee (see note 4).
Details of business rules should not be included in the text. Instead refer to a note for details.

Any error messages should also be presented as part of the scenario. Any alternatives should be referred to as alternative n (where n is the number of the alternative as specified in the alternative section) and italicised in the text.

The text should not generally extend for more than one to two pages.

**Alternative Courses** Narrative of alternative scenario. This section should start with the prerequisite conditions necessary for the alternative scenario to occur as the first flow of events [either from actor or system]. Each alternative scenario should be numbered sequentially (eg. 1, 2, 3) and it should be ensured that the alternative is used in the scenario narrative above. Alternative courses have a similar layout and structure as scenario text.

**Extends** Include the comma delimited names of any other business scenarios that this scenario extends. If there are none then say ‘None’.

In additional to Visual Paradigm attributes, we want you to include under the printed Use Case Description, the following:

**User Interfaces** Indicate which section 6 user-interface ids used in accomplishing this scenario. If there are none then say ‘None’.

**Constraints** Describe any constraints or requirements that will not be met or are out of scope. Such explicit statement avoid confusion and helps clarify any limitations implicit in the business scenario. These should be numbered sequentially (eg. 1, 2, 3). If there are none then say ‘None’.

**Questions** State any questions that need to addressed before the business scenario can be considered complete. Delete the question once it has been answered within another part of the business scenario. The business scenario cannot become final if there are questions still in this section. Include the name of the person who is charged with resolving the question. If there are none then say ‘None’.

-----

<table>
<thead>
<tr>
<th>Description</th>
<th>Login</th>
</tr>
</thead>
<tbody>
<tr>
<td>Super Use Case</td>
<td>Sub case of this</td>
</tr>
<tr>
<td>Author</td>
<td>Cat Kutay</td>
</tr>
<tr>
<td>Date</td>
<td>23/03/2006 1:57 PM</td>
</tr>
<tr>
<td>Brief Description</td>
<td>Has to be done to start this</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Post-conditions</th>
<th>Actor Input</th>
<th>System Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Student enters user name and password</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>System compares with database and accepts user</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>System compares with database and accepts user</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Exception 1</th>
<th>Actor Input</th>
<th>System Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>System compares with database and warns user</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>User retries. Login</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 3:** Use Case Scenario Details
5 Functional Requirements: Entity-Relationship Model [10 marks]

Entity-relationship modelling is regarded as a mainstream approach for conceptual data modelling. One of the primary factors for its popularity is the belief that entities and relationships are natural modelling concepts in the real world.

The conceptual model is a representation of the structure and constraints of a database that is independent of the software and its associated data model that will be used to implement the database. An E-R model is often used as a tool of communication between Database designers and End Users.

5.1 Graphical E-R Model

Include your E-R diagram/s here. The three basic constructs used are:

1. entities in the business environment
2. relationships (or associations) among those entities
3. attributes (or properties) of both the entities and their relationships.

Your ERD may be done in Visual Paradigm, as in Figure 5.1, or you may wish to use another tool. However make sure your attributes are general and do not make the diagram too cluttered and hard to read. The relationships in the Visual Paradigm tool are labelled by right hand mouse click – > "OpenSpecification" and enter a "Phrase" to describe the link.

5.2 E-R Traceability

Just as all use cases must be traceable back to the original requirements, so too must your E-R diagrams. In this section, make explicit the connections between what you’ve put in your E-R diagrams with its corresponding requirement(s).

6 Functional Requirements: User Interface [10 marks]

These are the user interface requirements. It details what interactions should occur rather than how those interactions should be accomplished. These requirements will directly influence how users will interact with the system.
6.1 Interface Listing

Cover all user-interface interactions with the system. For example:

<table>
<thead>
<tr>
<th>Id</th>
<th>Description</th>
<th>Type</th>
<th>Data Source and Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>UIF001</td>
<td>Stock delivery input</td>
<td>User input screen</td>
<td>Data is sourced from the Stock Database. Additional comments, must also relate this back to the Use-case Ids in section 4.</td>
</tr>
<tr>
<td>UIF002</td>
<td>Stock dispatch check</td>
<td>Barcode scan</td>
<td>Stock Database. Add more comments here.</td>
</tr>
</tbody>
</table>

6.2 Graphical Use-Case Diagrams

Where appropriate, use case diagrams may be used to specify the interactions defined in the table listed in the previous section. You may use any drawing tool or software you like to produce these diagrams.

6.3 Reports

List the reports that will be produced giving them ids, names and a description. Include screen shots - these may be part of an appendix.

<table>
<thead>
<tr>
<th>Report Id</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UREP001</td>
<td>Stock Demand Summary</td>
<td>Summarises and ranks popular products</td>
</tr>
</tbody>
</table>

7 Reliability and Performance Requirements [15 marks]

These encompass the non-functional requirements, including both static and dynamic requirements. Static requirements concern, amongst other things, the number of terminals to be connected and the number of users that can be handled concurrently. Dynamic requirements concern the operational performance of the system: how frequently will certain functions be called for and how fast should the system’s reaction be. It is important that these requirements be stated in measurable terms.
7.1 Design constraints.

Design constraints may result from such things as the prescribed use of certain standards or hardware.

7.1.1 Standards compliance

Which existing standards or regulations must be followed, and what requirements result from these. For example, certain report formats or audit procedures may be prescribed.

7.1.2 Hardware limitations

A description of the characteristics of the hardware environment, in as far as they lead to software requirements. An example of this might be the amount of memory available.

7.2 Static requirements

Static requirements concern, amongst other things, the number of terminals to be connected and the number of users that can be handled concurrently. This section should deal with your requirements for your prototype. If you require networking think how you will emulate this in the lab. You may with to use more than one computer on the cse network.

7.3 Dynamic requirements

Dynamic requirements concern the operational performance of the system: how frequently will certain functions be called for and how fast should the system’s reaction be. It is important that these requirements be stated in measurable terms.

8 Quality Attributes [10 marks]

In this section, particular attention is paid to quality aspects. These requirements must be measurable and verifiable. They must be stated in objective terms. The subsections below by no means comprise a complete list of such attributes.

8.1 Availability

Factors that guarantee a certain level of availability, such as restart procedures. In this subsection we may also enlist requirements regarding fault tolerance (with respect to both hardware failures and software failures).

8.2 Security

Requirements regarding unauthorised access and other forms of misuse. Certain cryptographic techniques may be prescribed, and we may put constraints on the communication between different parts of the system.

8.3 Maintainability

Requirements to guarantee a certain level of maintainability of the system. How dependent is one system (or components of the system) upon another (coupling).
9 Other Requirements [5 marks]

A description of requirements that are specific to certain software, and which have not been discussed yet.

WARNING: There is potential 20 mark deduction for failure to comply with typesetting rules; producing a document that has poor grammar and spelling, and is of low overall quality.
A  Glossary

This subsection contains definitions of all the technical terms, acronyms and abbreviations used in the document. Special attention should be paid to the clarification of terms and concepts from the domain of application.

B  References

References to all documents that are referred to in the remainder of the requirements specification.

C  Other Supporting Documents

C.1  Task Schedule

Attach a task schedule as of the Deliverable 2 deadline.