Guide to Preparing the
SOFTWARE DESIGN SPECIFICATION DOCUMENT

© R. Buckley
CSc 190 Senior Project
Department of Computer Science - College of Engineering and Computer Science
California State University, Sacramento
Version 8.15.12

Title Page (same design/format to be used for each document)
Table of Contents (include each section and the first level subsection for each section)

1. INTRODUCTION

This is the software design specification document for the <*name of the project*> project sponsored by <*name of sponsor*>.

This project is being undertaken by the <*name of team*> development team. The team is comprised of undergraduate students majoring in Computer Science at California State University, Sacramento. The team members are enrolled in a two-semester senior project course required of all undergraduate majors. Successful delivery of the desired software product will fulfill the senior project requirement for the student team members.

PROJECT SPONSOR (if there is more than one person serving as sponsor, list each sponsor):

Name
Title
Company *or* Organization name
Contact information (phone number and Email address)

<*NAME OF TEAM*> DEVELOPMENT TEAM

List of team member names
Team contact information (phone number and Email address)

The remainder of this section is intended to introduce the reader to the document and to serve as a high level, executive-like summary to the document.

Note. Write this section last! In preparing the document, the project team may have a general understanding of the purpose and scope of the document but one that is not sufficient to inform the general reader. In most cases, it is best to leave this section to the very end, or at least reserve final review and revision until a draft of the document has been completed.

1.1. Purpose

This subsection should describe the purpose of this document. Explain why this document is being written and what the reader should expect, in general, to learn.

1.2. Scope

This subsection should describe what specifically is to be covered by the document and what is not. Also, this section should indicate that this document represents the baseline design specification for the <*named*> system. Included, should be an explanation of how subsequent changes to this document will be handled. This would be the description of the team's baseline change process, which was specified in the team's management plan.

1.3. Definitions, Acronyms and Abbreviations

This subsection serves as a glossary for the document. All technical terms used as well as all acronyms and abbreviations used are arranged in alphabetical order. The purpose of this subsection is to provide the reader with quick access to the technical references used throughout the document.

For example:

1.3.1. **Baseline**. A baseline is a work product that has been formally reviewed and accepted by the involved parties. A baseline is changed only through formal configuration management procedures.

1.3.2. **CSc**. An abbreviation used to represent computer science.

1.3.3. **SRS**. An acronym for the software requirements specification. The SRS documents the essential requirements (functions/features/uses, performance, design constraints, and attributes) of the software and its external interfaces.

1.4. References

This subsection serves the same purpose as a bibliography. Included should be a reference to this document as well as any additional documents used in the preparation of the document. If this document contains a reference to a specific bibliographic entry, the reference should be made at the location in the document where information from that specific source is used.

Samples:

Ambler, Scott W. "UML 2: Sequence Diagram Overview". Online posting.
Posted on June 13, 2003. Accessed on February 25, 2004.
http://www.agilemodeling.com/artifacts/sequenceDiagram.htm.

Bennett, Douglas. Designing Hard Software: The Essential Tasks, Manning Publishing Company, 1997.

Buschmann, Frank, et al. Pattern–Oriented Software Architecture, John Wiley and Sons, Ltd., 1996

Cusumano, Michael, et al. "Software Development Worldwide: The State of the Practice." IEEE Software 20.6 (2003): 28-34.

IEEE, Standards Collection on Software Engineering. New York: IEEE Press, 1994.

Schneider, Geri and Jason P. Winters. Applying Use Cases: A Practical Guide. Addison-Wesley, 1998.

1.5. Overview of Contents of Document

This subsection briefly describes each of the remaining sections in the document as well as the contents of each appendix. The words used should be identical to those used to introduce each section and each appendix.

2. ARCHITECTURAL DESIGN

The architectural design of the system describes what functionality is assigned to each feature, where the data will be stored, and how much communication is required through the user interface, involving both functionality and database interactions.

This section is intended to provide a road map into the software. After reading this section, the reader should have a general understanding of the system's hardware and software design architecture.

2.1. Hardware architecture

Provide a *diagram* and *description* of the hardware configuration on which the software will operate. The representation should explain the nature of the client / server hardware tiers. The following subsections describe some of the possible configurations that might apply.

For example:

Two tier architecture. This architecture describes a distributed design where multiple client machines make requests of a single server machine.

Three tier architecture. The client machine hosts the user interface application while the central data base host is the server. The third tier is the local application server. In this case, the client machines are connected via a local area network to a local application server, which in turn communicates with the central database server.

N-tier architecture. This architecture represents the norm in most organizations, representing the linkage of local area networks, wide area networks, the Internet and the World Wide Web. The distinction between client and server depends on the type of transaction processing being run.

A simple figure (e.g. showing client machines and servers) should be included which represents what you are describing in this section.

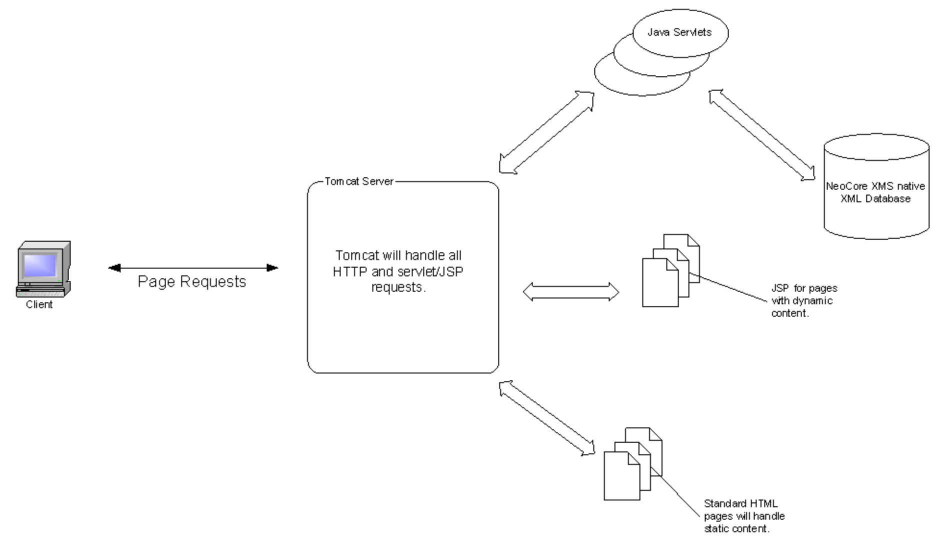
2.2. Software Design Architecture

This subsection should describe the following, as appropriate:

- The scope of the software
- The concepts used to design the software
- The *business* procedures contained in the software (these procedures contain the project specific processing logic that has been designed to deliver the software's functionality to the user)
- The interactions between each of the software components

Also, provide a *diagram* and *description* of the deployment of the software across the multi-tiered hardware architecture described above. The software is grouped into at least three layers, each of which is described in the following subsections.

For example, for each feature:


Java Servlets
Tomcat Server
Tomcat will handle all HTTP and servlet/JSP requests.
NeoCore XMS native XML Database
Client
Page Requests
JSP for pages with dynamic content.
Standard HTML pages will handle static content.

2.2.1 Presentation Layer. This layer resides at the "edge" of the software system. Its job is to capture external event requests and to perform some degree of editing of incoming data. It also is charged with presenting the event responses to the outside world (e.g. the screens and reports). This layer is usually allocated to the client machine or, in the case of a Web based application, to the Web server and is represented by the system's Use Cases.

Note. The software functionality associated with each feature will be provided to the system users through the interface. A general description of each feature is provided in this section.

2.2.2. "Business" Logic Layer. This layer contains the coded functionality, which executes and enforces the business rules. This software layer may be deployed on the client machines, on the server, or any machine on the network.

Note. The general description of the software functionality as it is distributed to each of the system's features is included in this subsection.

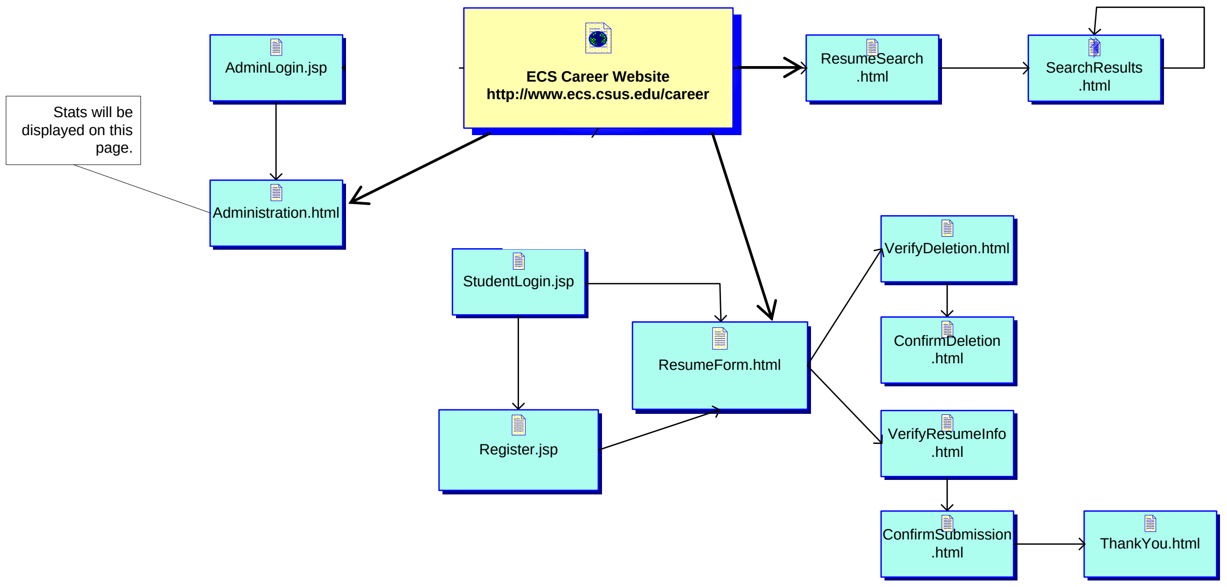
2.2.3. Data Management Layer. This layer provides access to the stored data. It manages concurrent requests to read and write to the database. In the case where data is distributed throughout the system, this layer would handle the synchronization of these distributed data elements. In almost all cases, this software layer is defined by the data base management system used.

3. INTERFACE DESIGN

This section specifies the look, feel and behavior of that portion of the system that is visible to the user. If the project team prepared a design prototype, this section would provide the design specifications for that prototype.

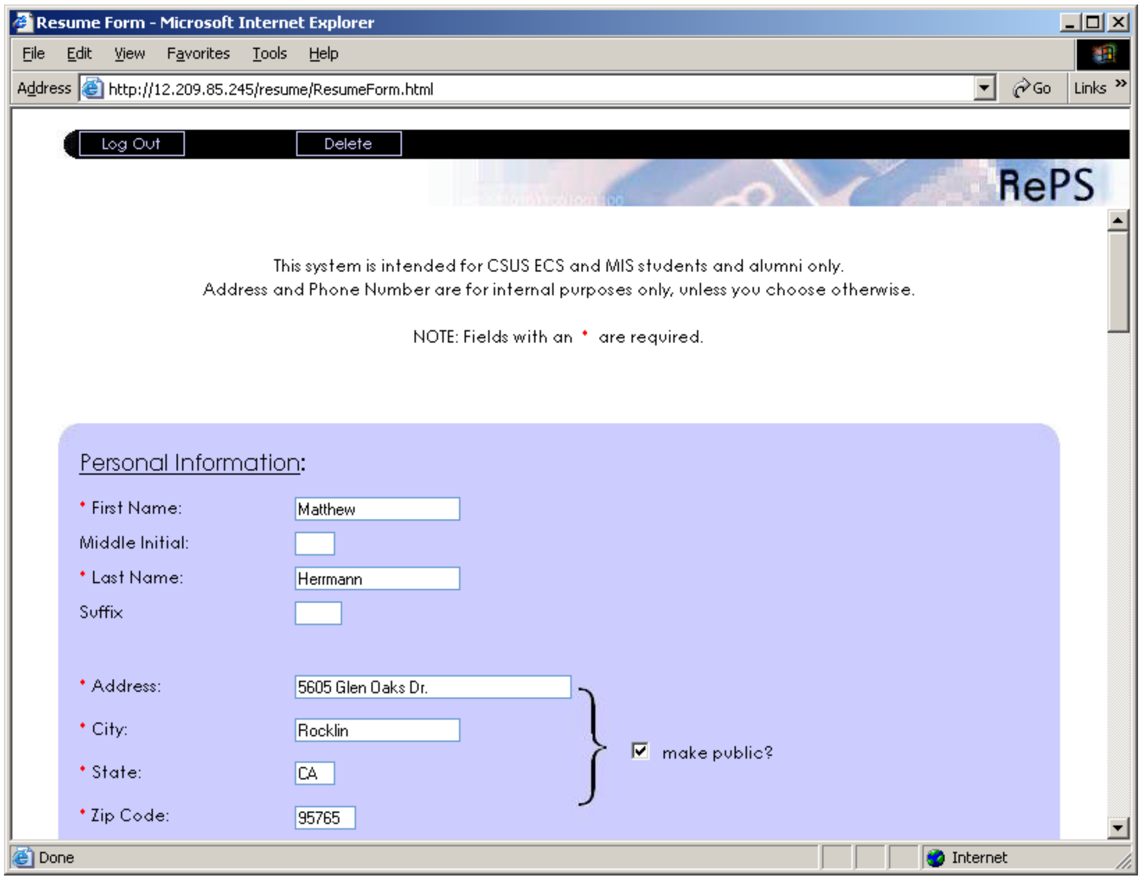
To introduce this section, provide a table that lists each Feature with a description of capabilities provided to user (e.g. this could be the same description that is contained in the SRS). In addition, list each webpage associated with each Use Case. Also, include the system's Webpage Navigation Diagram showing each feature and the linking relationships.

The following is a sample diagram from a previous senior project design document (features: Administration, Resume, and Resume Search).



Display the screen print for each Feature and briefly describe its purpose and content beginning with subsection 3.1. The following is an example of a subsection containing the screen print for the “Resume” feature that appears in the sample navigation diagram above.

3.1. Resume Form. Provides the capability for the student to enter specific resume data.


Resume Form - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://12.209.85.245/resume/ResumeForm.html
Go Links
Log Out
Delete
RePS
Done
Internet

This system is intended for CSUS ECS and MIS students and alumni only.
Address and Phone Number are for internal purposes only, unless you choose otherwise.

NOTE: Fields with an * are required.

Personal Information:

* First Name:	Matthew
Middle Initial:	
* Last Name:	Herrmann
Suffix	
* Address:	5605 Glen Oaks Dr.
* City:	Rocklin
* State:	CA
* Zip Code:	95765

[✓] make public?

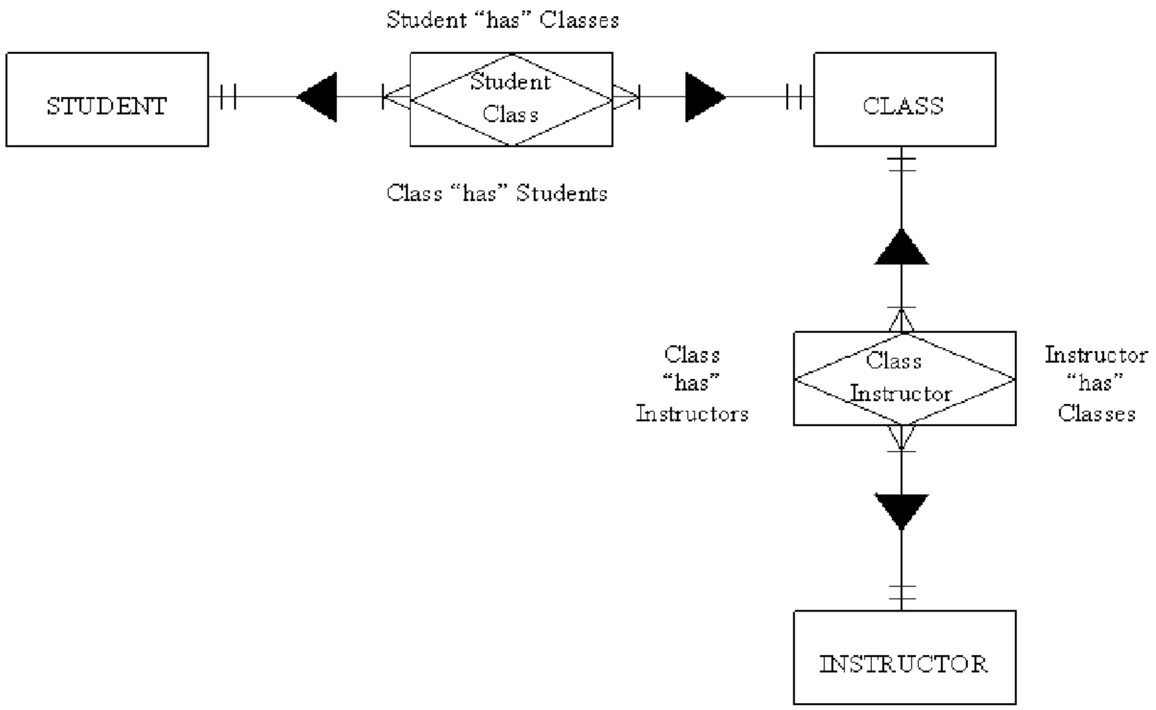
3.2 THROUGH 3.n.

If the system has *n* Webpages, each would have its own subsection (beginning with 3.1 (above), 3.2, … through to 3.*n*). Section 5.2 will contain detailed specifications in subsections for each of these interface components. Each subsection should have a title that matches the names as they appear in the navigation diagram and the screen prints.

4. DATABASE SCHEMA

This section provides the translation of the informational model contained in the Software Requirements Specification (SRS) into a relational database.

4.1. ERD diagram. This diagram depicts the data in terms of the entities and relationships described by the data. The figure below is a segment of the ERD for a hypothetical student registration system. The entities depicted are students, instructors and classes.



The relationships are represented by the Student-Class and the Class-Instructor tables. The Student-Class relationship associates 1) students with the classes they are taking, and 2) classes with their enrolled students. The Class-Instructor relationship associates 1) classes with the instructor or instructors teaching the class, and 2) the instructors with the classes they are teaching.

4.2 Creating the database. This section contains the CREATE statements for the DBMS. The following statements would implement the relationships depicted in figure above.

4.3 Triggers and/or stored procedures. This section would contain the specifications for 1) procedures embedded in a table that are automatically "triggered" by updates to another table, and 2) procedures stored in a table that can be invoked from the application (via the SQL statements).

Examples:

Using the student registration system example, if a student withdraws from the university and is deleted from the STUDENT table, a procedure would be triggered to delete all corresponding records in the STUDENT-CLASS table.

A stored procedure might be invoked to verify whether a student's major attribute would allow him or her to take the class he or she is registering for.

It is important to note that these procedures will be executed on the data base server. If your design does not make use of DBMS trigger and stored procedure capability, this section would be omitted.

Note. The relational database will be comprised of tables, each table being a series of columns, which represent individual data elements. The data records in the table form the rows. Each table has a primary key. Tables are related to each other by 1) embedding the primary key from one table into another as a foreign key, or 2) joining primary keys from different tables into a separate table. Foreign keys enable the relational database management system to enforce referential integrity. Referential integrity insures that no row in a" parent" table can be deleted if it is still referenced in a row of a "child" table.

5. COMPONENT DESIGN SPECIFICATIONS

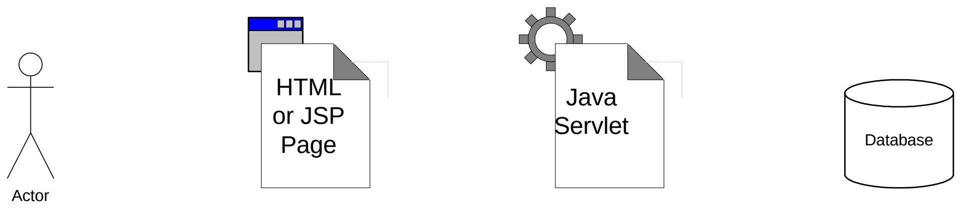
This section provides the detailed description for the design of the software. The section begins with a traceability matrix that map use cases to the web page and the components needed to serve the web pages. The following table is an example where the supporting components of java servlets.

	Use Case COMPONENTS		**Database**
Feature	**Webpage**	**Use Case (function)**	**Table/Relations**
Create New Resume	StudentLogin.jsp, Register.jsp, ResumeForm.html, VerifyResumeInfo.html, ConfirmSubmission.html, ThankYou.html	CreateUser.java, SubmitResume.java, FinalizeResume.java, Logout.java	
Delete Resume	StudentLogin.jsp, ResumeForm.html, VerifyDeletion.html, ConfirmDeletion.html	UserLogin.java, DeleteResume.java	
Search Resumes	ResumeSearch.html, SearchResults.html	SearchResume.java, RetrieveResume.java	
Update Resume Information	StudentLogin.jsp, ResumeForm.html, VerifyResumeInfo.html, ConfirmSubmission.html, ThankYou.html	UserLogin.java, SubmitResume.java, FinalizeResume.java, Logout.java	

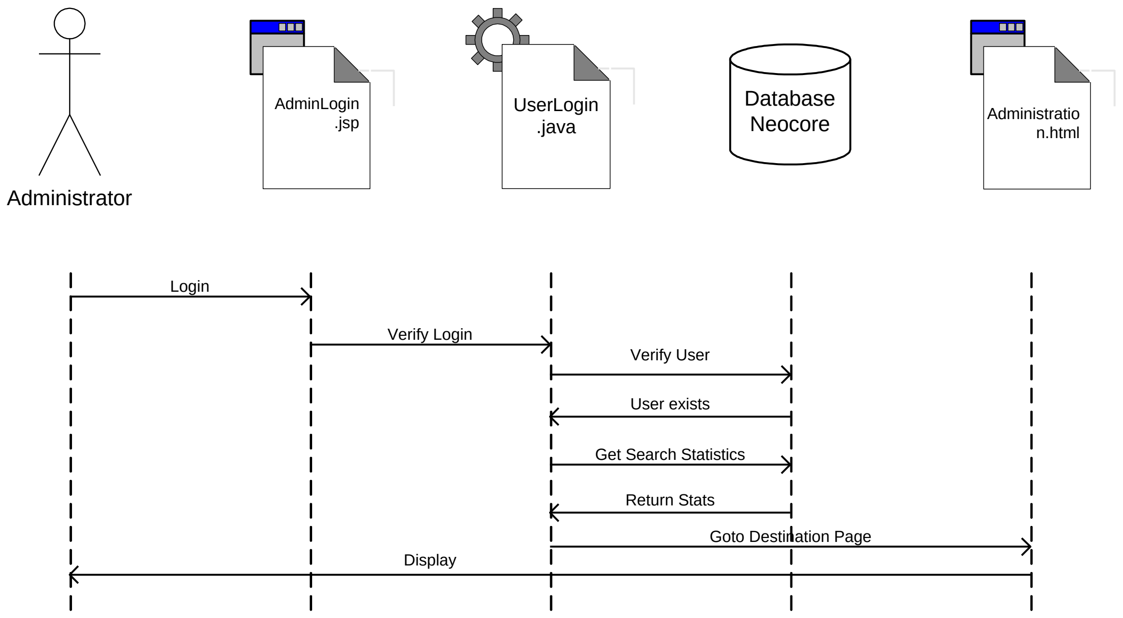
5.1. Sequence Diagrams. For each feature, include the sequence diagrams that illustrate the message interaction between the components that are required to deliver each specific use for that feature. The six features represented in the table above would be described in the following subsections:

5.1.1. Administrator Login Sequence Diagram
5.1.2. Create New Resume Sequence Diagram
5.1.3. Delete Resume Sequence Diagram
5.1.4. Search Resumes Sequence Diagram
5.1.5. Student Login Sequence Diagram
5.1.6. Update Resume Information Sequence Diagram

Note. The introductory section, 5.1, should provide instructions on the notation and how to read sequence diagrams. For example, sequence diagrams are read from left to right and top to bottom. The diagrams in this section use various icons. Pictured below are the icons used in the sequence diagrams.



The following is the example of a sequence diagram that models the design of the administrator login Use Case.



5.2. This subsection introduces the design specifications for each webpage and use case (function) associated with each feature. A table listing each of functional components, the feature or features that it serves, and the database tables/relations required must be included. Using the example above, subsections would be defined for each of the following twenty functional components:

5.2.1. Administration.html
5.2.2. AdminLogin.jsp
5.2.3. ConfirmDeletion.html
5.2.4. ConfirmSubmission.html
5.2.5. CreateUser.java
5.2.6. DeleteResume.java
5.2.7. FinalizeResume.java

5.2.8. Logout.java
5.2.9 Register.jsp
5.2.10. ResumeForm.html
5.2.11. ResumeSearch.html,
5.2.12.RetrieveResume.java
5.2.13. SearchResults.html
5.2.14. SearchResume.java
5.2.15. StudentLogin.jsp
5.2.16. SubmitResume.java
5.2.17. ThankYou.html
5.2.18. UserLogin.java
5.2.19. VerifyDeletion.html,
5.2.20. VerifyResumeInfo.html

Each component should have a unique identifier. Naming rules should be established and where appropriate a hierarchical naming scheme should be used to identify the parent of the component. The identifier should reflect the purpose and function of the component.

As indicated above, subsections are required to complete the design specification necessary for each functional component. For example, the following specifications would be included for each of the twenty components listed in the example above:

Preconditions. A description of the conditions that must be met before this module can be executed.
Interface specifications include the appropriate "signatures" for invoking the module and parameters and their data types.
Processing specifications for the module would be described using pseudocode, activity diagramming, or some suitable diagramming method for specifying the processing requirements. If the database is to be accessed, provide the appropriate SQL statements.
Screen print for the web page components only.
Database requirements. This would list table(s) and include a description of the table relationships and how database access is controlled, if appropriate.
Postconditions. A description of the conditions that must be met before this module can be completed. Typically, this refers to the state of the system that must exist for module execution to be completed.

Note. Each of the twenty subsections would provide the design specifications for the twenty subsections included in Section 3.0 of the SRS.

6. PERFORMANCE ANALYSIS

This section provides details on any performance issues or constraints that may have arisen during the design and implementation phase. If the SRS contained nonfunctional specifications relating to the performance of the delivered software, this section would also explain how these requirements were met.

7. RESOURCE ESTIMATES

This section should contain a summary of the computer resources required to build, operate and maintain the software.

8. SOFTWARE REQUIREMENTS TRACEABILITY MATRIX

The matrix relates the design components to their associated requirements by listing the subsection numbers in this document and the associated subsection numbers in the Software Requirements Specification Document.

9. APPROVALS

This section contains the sign-off sheet that is used to indicate approval of *and* agreement to the design specifications contained in this document. The signatories should include, at a minimum, each member of the project team and the project's faculty adviser. The sign-off sheet should include the name, full title, and full name of the parent organization for each of these signatories.

APPENDICES

APPENDIX A.	Database tables and attributes.
APPENDIX B.	Alphabetic listing of each attribute and its characteristics.

Note. References to both Appendix A and B must be included in the appropriate section or subsection in this document.