Final SRS/SDS/Project Plan

Team Members:

Jason Prideaux

Helen Shen

Willy Suhali

Christian Tan

TABLE OF CONTENTS

FIGURES

TABLES

Software Requirement Specification

Purpose Of The Document

This document is an official description of the requirements of the book-exchange website, *Bookster*. The system will be built by the guidelines of this document.

Problem Statement

The University of Oregon (UO) Bookstore charges a high price on used books, but it pays little on buyback. In order to buy cheap books or sell books at a reasonable price, students can only post messages either on the Oregon Daily Emerald or on the bulletin board in the buildings on campus, which is not very efficient. Currently, the University campus supports a large network of computers that can be used by all students to communicate and share information. All students have free access to this computer network, which is spread all over the campus area.

Proposed Solution

Bookster is a website that provides UO students an environment to post a list of books to sell at a price which they deem reasonable. Besides listing the books for sale, *Bookster* users can also search for books they wish to purchase. *Bookster* also provides an email communication forum for buyers to notify the sellers of their interest to purchase a book.

Bookster will be implemented as a web-based application. This is to provide a wider target market of UO students, as opposed to limiting to UO students who knows how to install and run a Java application. However, since a fully functional web-based application will be large and complex, the *Bookster* system development is divided to three phases: proof of concept, client-server application, and web-based application. Given the time frame for this project, the proposal involves the completion of phase 1 only.

Phase 1: Proof of concept

A system which has the same functionalities as the proposed website to be built. However, this system is installed and operated in the local host. This phase is designed to demonstrate the intended system and how it will perform. The purpose of this phase is to prove the concept that *Bookster* is able to perform the desired functions in the local host.

Figure 1. Phase 1: Proof of Concept.

Local database. Limited to users knowledgeable in Java.

This *Bookster* system in this phase will be a desktop application and is a proof of concept. The database is interactive only to the local machine, since it is not connected to a server. The system in this phase is a Java application, so the usage of *Bookster* in this phase is limited to users who know how to install and run a Java application.

Phase 2: Client-server application

A working *Bookster* system will be built. This client-server application will use the User Interface (UI) of Phase 1 (the client) to communicate with the servlets (server) to be built in Phase 2. Users can use the system to buy and sell the books after installing the *Bookster* software.

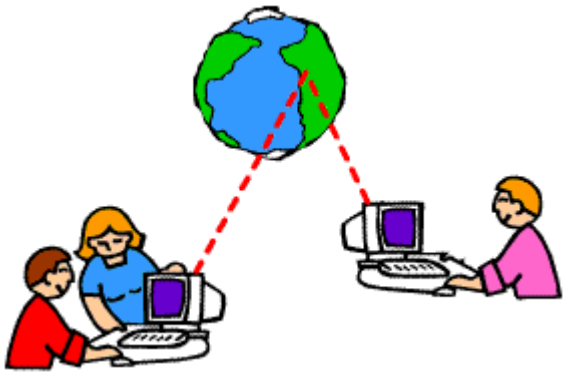

Figure 2. Phase 2: Client-server application.

Working Interactive database. Limited to users knowledgeable in Java.

The *Bookster* system in this phase will be a desktop application, however, it has a working database residing on a server. Since this phase implements a shared database on a server, the database is interactive to any machines running the Bookster UI of this phase. Like in Phase 1, the system in this phase is a Java application, so the usage of *Bookster* in this phase is limited to users who know how to install and run a Java application. Users are also required to have an Internet connection since it is connecting with the database on the server.

Phase 3: Web-based application

The *Bookster* system will be transported into a web-based application. Users are then able to use the system via web browsers instead of installing the *Bookster* software.

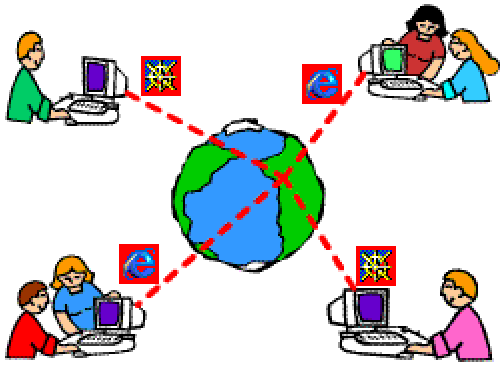

Figure 3. Phase 3: Web-based application.

Working Interactive database. Users now expanded to include anyone who knows how to use a web browser.

The *Bookster* system in this final phase will be a complete web-based application. The database will reside on a server as implemented in Phase 2. Since Phase 3 is a web-based application, users are not required to install any *Bookster* application. Instead, they can use their favorite Internet browsers, such as Netscape's Communicator or Microsoft's Internet Explorer, to use *Bookster*. Since this final phase of *Bookster* do not require users to install any Java application, the target audience is greatly increased to anyone who has an Internet connection and an Internet browser.

User Characteristics

Target users are all of the UO students who have a valid UO email account. Since this system is to be built specifically for UO students, a valid email account is required for verification. The users must also be able to send and receive emails since communications between buyers and sellers will be done via emails. Users also need to have the basic knowledge of using the Internet

and web browsers (Microsoft's Internet Explorer or Netscape Communicator) in order to use the *Bookster* system. Furthermore, in the proof of concept and the client-server application phases, users are required to know how to compile and run java programs.

User Scenarios

Selling A Book Through Bookster

Sam is graduating and had just finished his last class, PSY 203. He does not need his PSY 203 textbook anymore. He remembers buying the book used at $70 at the bookstore and was outraged that the Bookstore only charges $20.25 for buyback. He remembers buying his CIS 422 textbook from the online book market system, *Bookster* last term, so he decided to sell his PSY 203 textbook there.

Sam surfs himself onto the *Bookster* website, entered his username and password, and logs himself in. He navigates to the page where he can list his book for sale. In a form provided, Sam enters the information of his PSY 203 book, and the price he would like to sell it at ($45!). He then logs out and prepares for his graduation party.

Two days later, Sam receives an email from *Bookster*, notifying him that a *Bookster* user by the username of Janney has expressed an interest in the book he is selling. He sends an email to Janney and schedules a time and place for the transaction. The successful transaction will make Sam $24.75 richer.

Buying A Book Through Bookster

Jane is taking PSY 203 next term. She goes to the bookstore and is shocked to find out that they are charging $70 for a used PSY 203 textbook. She remembers her friends raving about this web site for UO students called *Bookster* where they can buy and sell course books. She decides to give it a try.

At the *Bookster* web site, Jane creates a new account for herself, and used Janney as her *Bookster* username. After verifying with her valid UO email address, Jane logs in to *Bookster*. She navigates to the page where she can search for books that are listed for sale. In the form provided, Jane enters "Drew Westen" as the author of the PSY 203 textbook, and selects $50 as the highest price she would want to search for. At the request, *Bookster* returns her with a list of sellers. Jane notices from the list that the *Bookster* user, IamSam, is selling that textbook at the lowest price of $45. Jane selects the user IamSam and requests *Bookster* to inform IamSam of her interest in the textbook.

The next day, Jane receives an email from IamSam, requesting for a time and place to meet for the transaction. Jane is excited, as a successful transaction will save her $25!

Functional And Non-Functional Requirements

System non-functional requirements

Databases

The system needs databases to store user information and book information.

SMTP service

The system provides Simple Mail Transfer Protocol (SMTP) service to send notification emails to sellers, as well as for sending a temporary password to new users.

Security

The system will provide security features, such as Secure Socket Layer (SSL) and encryption of the password in the user database. This requirement is not applicable in the proof of concept phase.

User non-functional requirements

Search history

A search history list is stored so that a user is able to easily search for books that he or she has previously searched for, without having the user type all the book information again. The search history list will store up to ten books.

User-friendly interface

A user-friendly interface is provided for general users. The window layout and the pull-down menu are easy to use and familiar to computer users. Popup windows with alert messages will users of pertinent information.

User and system functional requirements

New user registration

Users who are using the system for the first time are guided through the registration process. The system will store the user information to the database and send a temporary password via email to the user's UO email account. In addition, the system will prevent duplicate registrations.

Login

A registered user needs to login whenever he or she wishes to use the system. The system will check the database to verify the validity of the user.

Password retrieval

A registered user is able to retrieve his or her password if he or she happens to forget the password. The system will send out an email containing the password to the user's registered email address.

Sell books

A registered user is able to post information of the books he or she wishes to sell. The system will update the database by adding one entry of the book-for-sale information in the book database. The system will also prevent the user from adding duplicate books to the database.

Edit books

A registered user is able to edit the book information he or she has put up for sale, such as selling price and the condition of the book. The system will replace the previous book information with the updated one in the book database.

Delete books

A registered user can delete the book information he or she had listed for sale. The system will remove that book information from the database.

Search books

A registered user can search the database for books that he or she wishes to buy. The system will perform a search based on the search criteria and display a list of matches found. Further, the system will update the search history list with the new search criteria.

Email sellers

A registered user is able to send out emails to inform the seller that he or she would like to buy the book. The system provides a SMTP service that sends out emails that will contain the buyers email address. This is to facilitate the seller with a means of communication with the buyer.

Edit personal profile

A registered user can change his or her password and email address. The system will update the user information in the database.

Other Requirements:

Bookster has been divided into three phases as described in the Proposed Solution section on page 1. Phase 1 and Phase 2 uses Java for *Bookster's* UI. Because the UI of Bookster is created in Java, the application is hence platform independent. However, since it is a Java application, it requires the users to have Java version 1.3 to be already installed on their computers before they can use *Bookster*. Since Phase 3 is a web-based application, the users are required to have an Internet connection and a web-browser in order to use *Bookster*.

Software Design Specifications

Design: High-level View

The design of the *Bookster* system is highly dependent on the requirements of the system. The system requirements imply a design that consists of several databases and a controller. The controller handles requests from a single user interface or from an outside source if this system is running on a remote server. Thus, in order to fulfill the requirements mentioned in the SRS, *Bookster* contains the following components:

- **User Interface** – An interface that takes in all commands from the user, and displays the desired information that the user requests.
- **Data Controller** – Controls the requests from the User Interface and returns the information pertaining to the requests back to the User Interface. It has direct and exclusive control over the databases and email components.
- **Database** - **User Database** contains all the current users of the *Bookster* system, and their information, such as email, password, and preferences.
  - **Book Database** is comprised of all the books currently for sale.
  - **Data** are the objects that are stored in the databases, including the **User** and **Book** objects that represent users and books respectively.
- **Email** – This component facilitates the need to send out messages to users, notifying them when someone wants to buy their book and to verify email addresses of first-time users.

Design Phases

The abovementioned components are needed for the design of this system during all three phases; however, there are a few slight differences. Phase 1 uses the high-level design previously described, and is covered in more detail in the rest of the SDS.

In phase 2, the Data Controller will be divided into two control modules. One control module will be on the user's system and the other will reside on a remote server where the databases are located. The two controllers will allow for unified communication between the application and server databases. Thus, the user terminals can easily send requests to the databases located on the server.

In phase 3, there will only be one Data Controller. However, in this phase, the UI will be implemented in HTML and will be accessed from any web browser. Requests will then be sent from the user's browser to the servlets that contains the Data Controller and Databases.

System Architecture

Together the four main modules and their sub-components make up the system architecture as shown in Figure 4. The User Interface is one main module that displays all necessary information to the user and receives all necessary information from the user. The User Interface only communicates with the Data Controller, which controls all information going to the User Interface and coming from the User Interface. The Data Controller has direct access to the Database, which is actually more than one database — one for users and one for books. For example, when the Data Controller receives a login request from the User Interface, the Data Controller queries the User Database for the user's account information. Likewise, when the Data Controller receives a request for some sort of action, the Data Controller can access the book database to retrieve or modify any data it needs to. The Data Controller also controls the Email module to send out emails to users for password confirmation and to notify users of an interested buyer. Additionally, there are small components such as User and Book that are data encapsulation objects. These objects represent users and books in the system.

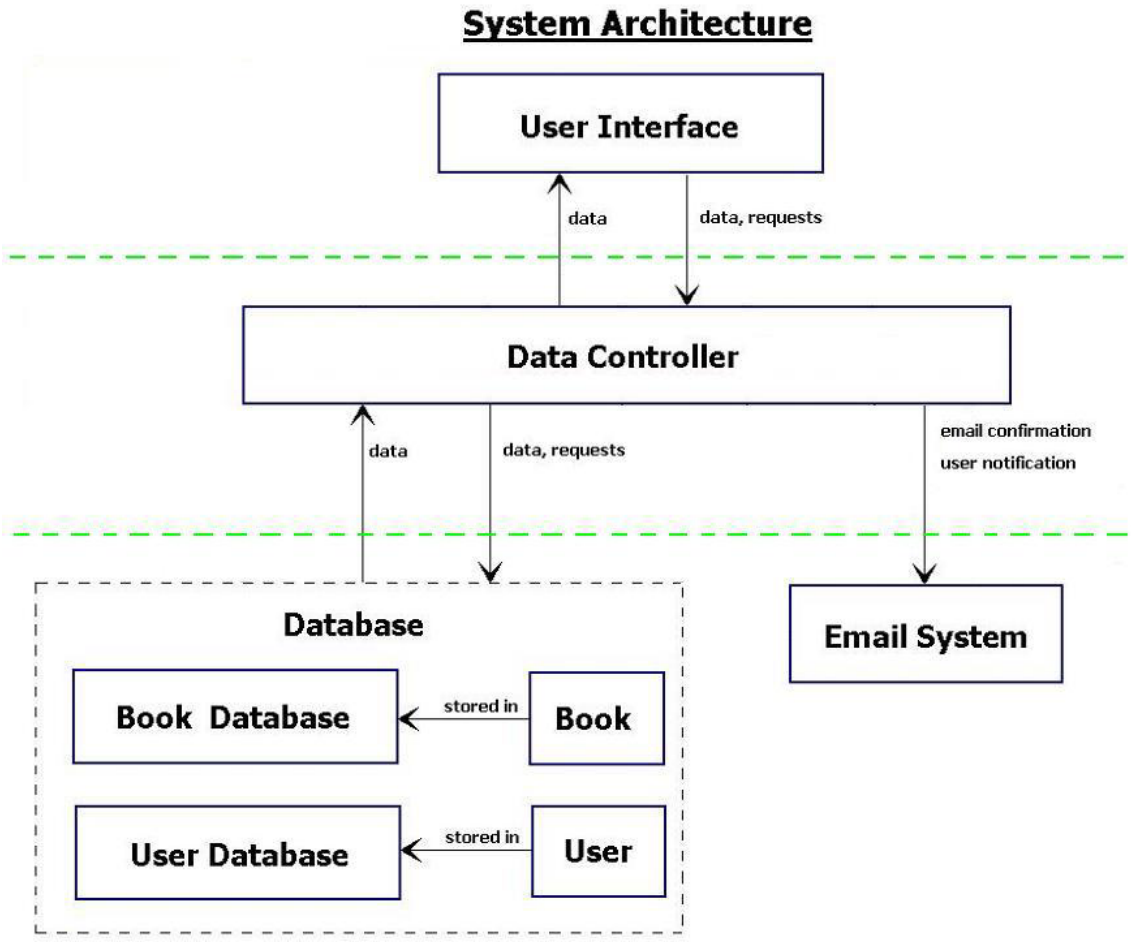

Figure 4. System architecture: high-level view.

Structure: Low-level View Of Individual Components

The structure of the *Bookster* software system in Phase 1 consists of the four main components, with the Database component comprising of several sub-components. A visual of all the system objects is shown in Figure 5. The following is a description of the main system components, some of which contain multiple classes in order to divide the code into smaller, more manageable pieces.

User Interface (UI):

The UI is the only component that the user can interact with. It contains an instance of the Data Controller module inside it, and all requests that UI makes will be sent to the Data Controller. The UI consists of several different JPanels, JDialogs, and JFrames. The UI features are described in detail in the *Bookster Tutorial*.

Data Controller:

The Data Controller module in this system is only accessed/called upon by the UI. However, the Data Controller has access to all other components, and has sole control over them, including the Email and Database components.

Email:

The Email component is a simple object instantiated by the Data Controller. The Email component is called upon by the Data Controller to send email verifications containing a password to new users. This is designed to ensure that users have a valid *uoregon.edu* email address. Note: Although this component is not a necessity in phase 1, it does give a sense of how the system will work when it is based on a server.

Databases:

All databases will be similar in design and functionally; the main differences will be the information stored in each database and how searches are preformed.

User Database:

The User database contains information pertaining to each user of the *Bookster* system. All users are stored in the User Database as User data objects described later. The database implements functions to allow the Data Controller to get user data, add user data, remove user data, and update user data.

Book Database:

The Book Database contains information pertaining to every book being sold on the *Bookster* system. All books currently listed for sale on *Bookster* are stored in the database as Book data objects described later. The Book Database implements functions to allow the Data Controller to get book data, remove book data, and add book data.

Data:

All information stored in the databases is either in the form of a User object or Book object. These objects encapsulate all information about a User or Book.

User:

The User data object contains all information for each user including user name, preferred email address, password, and a history of all wanted books. The User object also contains functions for getting information and updating information inside the object.

Book:

The Book data object contains all information regarding a particular book. This includes title, author, ISBN, price, and the condition of the book. The Book object also contains functions for getting information and updating information inside the object.

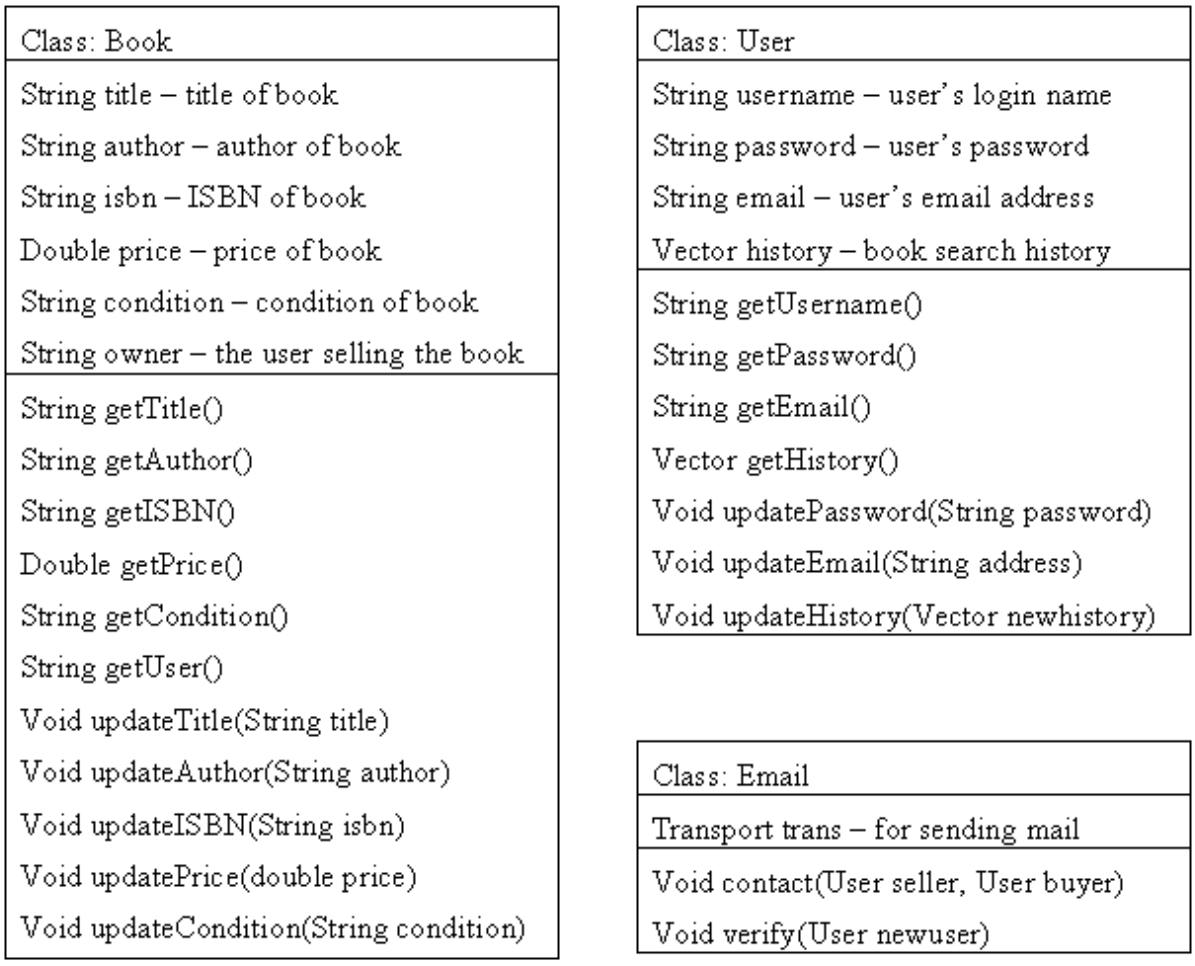

Figure 5. Object Model of the Bookster System

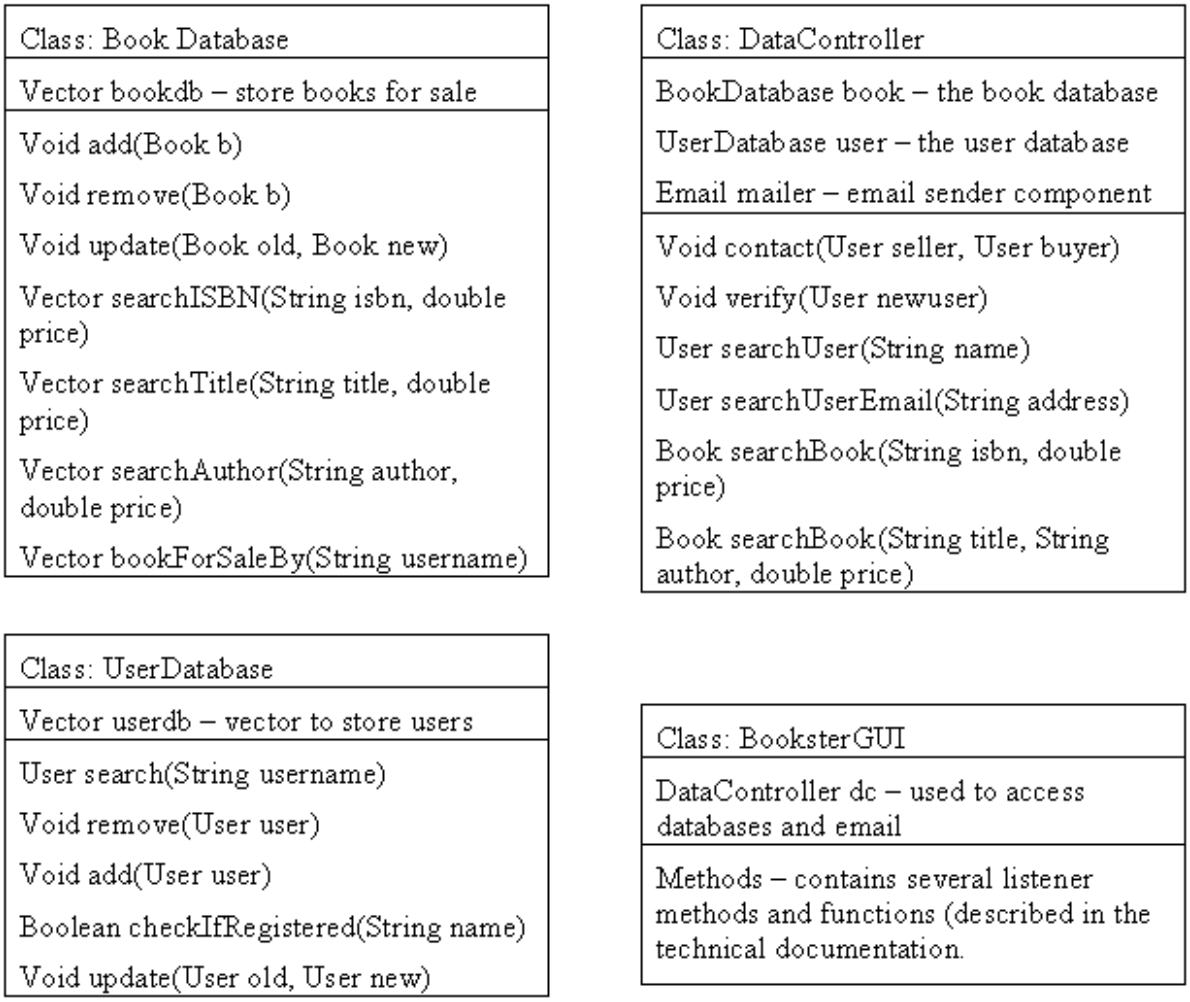

Figure 5. (Continued) Object Model of the Bookster System

Design requirements

Since *Bookster* must run on the user's system in phase 1, the system requires additional Java APIs not included in the regular Java SDK or JRE. While these APIs could have been avoided, they will provide a helpful framework for the expansion of the email system in phases 2 and 3. The extra API components are packaged with *Bookster*. However, if developers need documentation on the additional APIs, they must download and install the complete Java APIs on their system. The APIs used are:

- Java Mail 1.2
- Java Beans Activation Framework 1.0.1

Organization Principles

The overall organization of the *Bookster* system modules is based on hierarchical and centralized control design patterns. The system is hierarchical in that the User Interface is at the top, the Data Controller is underneath the User Interface, and everything else is below the Data Controller. The system also uses the centralized control pattern in that the Data Controller controls several components. The Data Controller is the central controller for the database and email components.

As for the organization of the building process, we used the Spiral Process model as shown in the Project Plan (page 15). While time was a major constraint, it was a necessity to have a process, which allowed for revisiting the software requirements and design. Some parts of the design needed modification during the implementation phase, so this process was the most useful in adjusting the schedule for such changes. Also, since this is only phase 1 of the *Bookster* software system, a process is needed that will make it easy to go back and make revisions for future phases of development.

Rationale

The design of this system has a two-fold purpose. First, the User Interface should not need to know how the databases are set, how many there are, or how to send out emails to users. Thus, having a centralized control structure around the Data Controller becomes a necessity. The User Interface needs only to worry about one component, the Data Controller.

This design also serves another purpose. While thinking ahead, we realized that the system must be designed so that the code can be easily modified when building the phase 2 or phase 3 *Bookster* system. In phase 2, where *Bookster* is a client-server application, the only modifications needed are to the Data Controller. The Data Controller will have to be modified to take in requests from several users at a time. In phase 3, where the system is to be web-based, the only changes necessary is to make the UI web-based as well. Thus, the design of this system is simple and efficient, and makes future modifications relatively easily.

Project Plan

Introduction

This project plan was used to assist the team for staying on the right track during the process of developing the system. The project manager and his backup were responsible in using the project plan to enforce milestones to the team. This project plan also acts as information for the client on the estimated completion date of the product. Any changes in the plan and schedule are reflected in the progress review sheet (Table 4. Project Tasks) at the end of this section.

Project Organization

Organizational Structure

The team is coordinated and directed by the project manager, Christian Tan, who is also responsible for administrating the project schedule. The team is made up of 4 people: Jason Prideaux, Helen Shen, Willy Suhali, and Christian Tan. All suggestions the team members make are considered thoroughly and clearly, and agreed on by every team member before a decision is made.

Team Responsibilities

The team's responsibilities are divided into seven team roles: project manager, quality control, system architect, technical documentation, user documentation, user interface, and configuration control. Each role is assigned in favor of the members' skills and experiences. There are at least 2 members assigned to each role. This will ensure that when the primary member faces any difficulties in the role, the secondary member will be able to help solve the problem. Table 1 presents the members' roles and responsibilities for each task.

In addition, for the roles of programming, the system is divided into four components, where each member is responsible for a specific component. (GUI, database, email, data controller).

Roles	**Primary**	**Secondary**
Project Manager	Christian	Jason
Quality Control	Willy	Helen
System Architect	Jason	Christian
Technical Documentation	Willy	Helen
User Documentation	Helen	Christian
User Interface	Christian	Willy
Configuration control	Jason	Helen

Table 1. Team Role Assignments

Team Communication

The team will have meetings at least 3-5 times a week, and additional meetings will be made whenever necessary. The meetings were held at the Deschutes computer lab and at the peer advising room. The progress report was checked against the project plan and milestones, and was updated accordingly during the meetings or via email by the primary and secondary Project Managers. Occasional online meetings via MSN Messenger were used daily to communicate between members. The team also used Yahoo Groups to share files and update their work, located at *http://groups.yahoo.com/group/422project/*

Yahoo Login name: ProfessorHornof
Password: hornof
Yahoo Group: 422project

Risk Analysis and Risk Reduction Strategies

Risks are prevalent in software development and may impede the team from delivering the software on time. To solve this matter, the team used risk analysis and risk reduction strategies. Table 2 presents some possible risks occurred and could have occurred during the development, and the corresponding plans to handle those risks.

Risk	Probability	Effects	Strategy
Falling behind the schedule	High (happened)	Serious	Delivering regularly and as soon as possible; milestone are strictly enforced. Fell slightly behind schedule, however, everyone pickup up the pace, and did his or her part.
Loss of a team member.	Low	Serious	Assigning two members for each task, the secondary member will still be working on the task if the primary person got sick or leaves the team.
Diverging away from the requirements	High	Serious	Try to come out with prototype early, use the spiral model software process. The design of the system underwent many changes. Working overtime was necessary to overcome this hurdle.
Lack of communication	Moderate	Tolerable	Using the chat program MSN Messenger to communicate between members. This provides real-time communication between members while at home.
Loss of important materials such as programming code and documentation.	Moderate (happened)	Serious	Use web-based file storage such as Yahoo groups. This will allow members to upload and download their work to and from the Internet. Yahoo went down on 3/4/02, however, all members kept backups on the CS servers.
Conflict between team members	Low	Serious	Manage conflicts constructively; try to make a unanimous decision or consult a higher authority, such as the Professor.

Table 2. Risk Analysis and Risk Reduction

Work Breakdown and Project Schedule

Milestones

The project will have fifteen main milestones as described in

Table 3 below, some of which contain multiple tasks to be completed. The system was completed on March 6, 2002. The milestones fell behind schedule as of February 25. The falling behind in the schedule is a result of the coding to be longer than anticipated. However, the system was still completed on time. With the aid of the Progress Review sheet, the schedule was updated as necessary, and the deadline of March 6 was met.

Milestones	**Dates**
Start the project plan	Feb 8, 2002
Finalize SRS/SDS/Project Plan	Feb 18, 2002
Preparing presentation material	Feb 20, 2002
Begin construction on GUI module (Christian, Willy) Begin construction on Data Controller module (Jason) Begin construction on Book/User Database module (Helen) Begin construction on Email module (Jason) Begin construction on Data module (Jason, Helen)	Feb 21, 2002
Complete GUI module (Christian, Willy) Complete Data Controller module (Jason) Complete Book/User Database module (Helen) Complete Email module (Jason) Complete Data module (Jason, Helen)	Feb 25, 2002
Testing of all 4 modules separately (All)	Feb 25, 2002
Fixing bugs and errors if found	Feb 26, 2002
Integrating the 4 modules into one system (Jason, Christian)	Feb 27, 2002
Testing of the entire system (All)	Feb 28, 2002
Fixing bugs and errors if found	March 1, 2002
Finalizing and getting a deliverable system	March 2, 2002
Finishing user documentation and technical documentation	March 3, 2002
Finalizing the SDS/SRS/Project plan	March 4, 2002
Product delivered	March 6, 2002

Table 3. Project Milestones

Project Process Breakdown

The software process for this project follows a Spiral Process (see Figure 6). There are six main phases in the development of the software, however, given the subtle changes made mostly to the UI, some of these phases were revisited. Risks were analyzed, evaluated, and solved at the beginning of each process iteration.

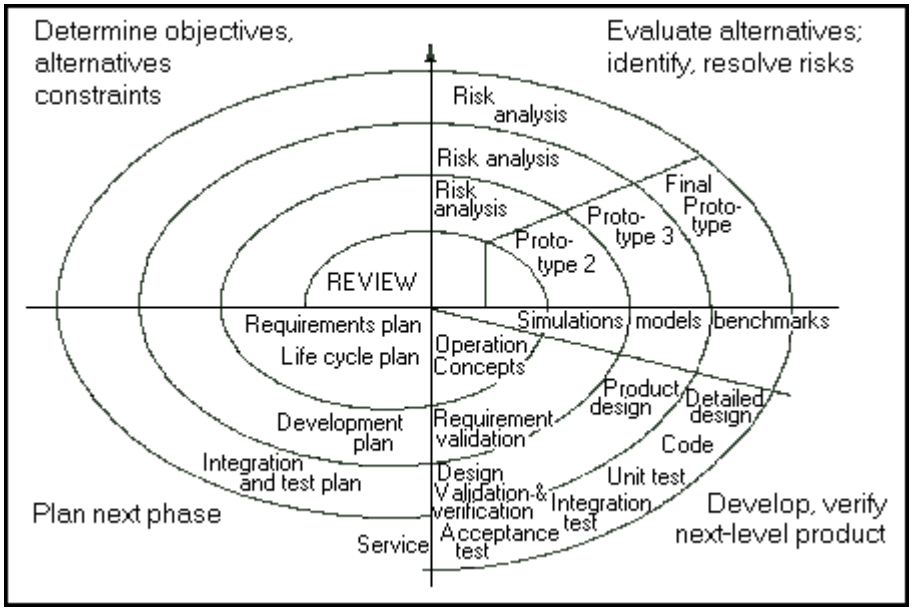

Figure 6. The Spiral Process Model

(Boehm's spiral model: ©1988 IEEE)

Software Process Model

Planning

The planning phase includes setting up the project schedule, delegating responsibilities/roles, and defining the problem statement for the Bookster project.

Requirement analysis

The requirement analysis phase includes analyzing the problem statement to create our list of requirements as described in the SRS. The list of requirements will then be used to describe the design of the system. This phase will be visited more than once to make sure that the process does not diverge away from the requirement.

Design

The design phase includes creating a system architecture, designing the UML diagram to assist the team in developing the system, identifying the 4 main modules and sub-components which are data controller, database, email, and GUI. Specifying when the modules should be built and by whom, is also included in this phase.

Development

At this phase, every group member will have a very specific programming task, and through regular meetings, the progress of each group member will be monitored. Since the system is divided into four components, each member was responsible for specific components. At a scheduled date, Feb 28, all code was integrated to create the system.

Testing

Individual group members did testing during the completion of his or her module. After the integration of the system, two group members that are assigned as quality control managers will perform the alpha-test of the system. A review of the system was done involving all group members, and adjustments were made as reflected in the *Bookster* Testing and Bug sheets.

Delivery

The system was completed on Mar 6, and delivered to the client in person on Mar 7.

Monitor and Report Progress

The team has kept a record of each task assigned, to whom the task was assigned, when it was assigned, the scheduled due date of the task, and when the task was completed. The team members signed off on the progress report when each task is completed and verified by the other members. The following table is the list of tasks that have been completed.

Task	Assigned to and completed by	Date assigned	Scheduled completion	Actual Completion
Meeting: Discuss project proposal	ALL	Feb 7	Feb 7	Feb 7
Meeting: Confirm projects proposal	ALL	Feb 8	Feb 8	Feb 8
Meeting: Analyzing requirements	ALL	Feb 12	Feb 12	Feb 12
Meeting: Finalizing requirements	ALL	Feb 14	Feb 14	Feb 14
Meeting: Start SRS/SDS/Project plan	ALL	Feb 12	Feb 12	Feb 12
Finalize SRS	Helen	Feb 14	Feb 17	Feb 17
Finalize SDS	Jason	Feb 14	Feb 17	Feb 17
Finalize Project Plan	Willy	Feb 14	Feb 17	Feb 17
Create GUI mock-ups	Christian	Feb 15	Feb 16	Feb 16
Combines SRS/SDS/PP together	Christian	Feb 18	Feb 18	Feb 18
Meeting: go over final SRS/SDS/PP	ALL	Feb 18	Feb 18	Feb 18
Setup architecture and interfaces	Jason	Feb 18	Feb 19	Feb 19
Meeting: assign programming roles	ALL	Feb 19	Feb 19	Feb 19
GUI	Christian, Willy	Feb 19	Feb 25	Feb 25
Book/User Databases	Helen	Feb 19	Feb 25	Feb 25
Controller, Email, Data	Jason	Feb 19	Feb 25	Feb 25
Meeting: progress review	ALL	Feb 25	Feb 25	Feb 25
Meeting: progress review w/ professor	ALL	Feb 26	Feb 26	Feb 26
Finish GUI	Christian, Willy	Feb 25	Feb 28	Mar 2

Finish Book/User Databases	Helen	Feb 25	Feb 28	Feb 28
Finish Controller, Email, Data	Jason	Feb 25	Feb 28	Feb 28
Meeting: progress review	ALL	Feb 28	Feb 28	Feb 28
Initial integration	Jason, Willy	Feb 28	Mar 1	Mar 2
Meeting: integration progress review	ALL	Mar 2	Mar 2	Mar 2
Revise/Finish GUI	Christian, Willy	Mar 2	Mar 4	Mar 5
Debugging	Jason, Christian	Mar 2	Mar 4	Mar 6
Revisions	Jason, Christian	Mar 2	Mar 4	Mar 6
User Doc/Tutorial	Helen	Mar 2	Mar 6	Mar 6
Readme file	Willy	Mar 2	Mar 5	Mar 5
Final SRS/PP	Jason	Mar 2	Mar 6	Mar 6
Final SDS	Christian	Mar 2	Mar 6	Mar 6
Tech doc	Jason, Christian	Mar 2	Mar 5	Mar 5
Integrate SRS and SDS/PP	Christian	Mar 2	Mar 6	Mar 6
Meeting: progress review / testing	All	Mar 4	Mar 4	Mar 4
Meeting: progress review / testing	All	Mar 5	Mar 5	Mar 5
Meeting: progress review	All	Mar 6	Mar 6	Mar 6

Table 4. Project Tasks