BEAR Advisor

Brown Electronic Access to the Registrar Advisor

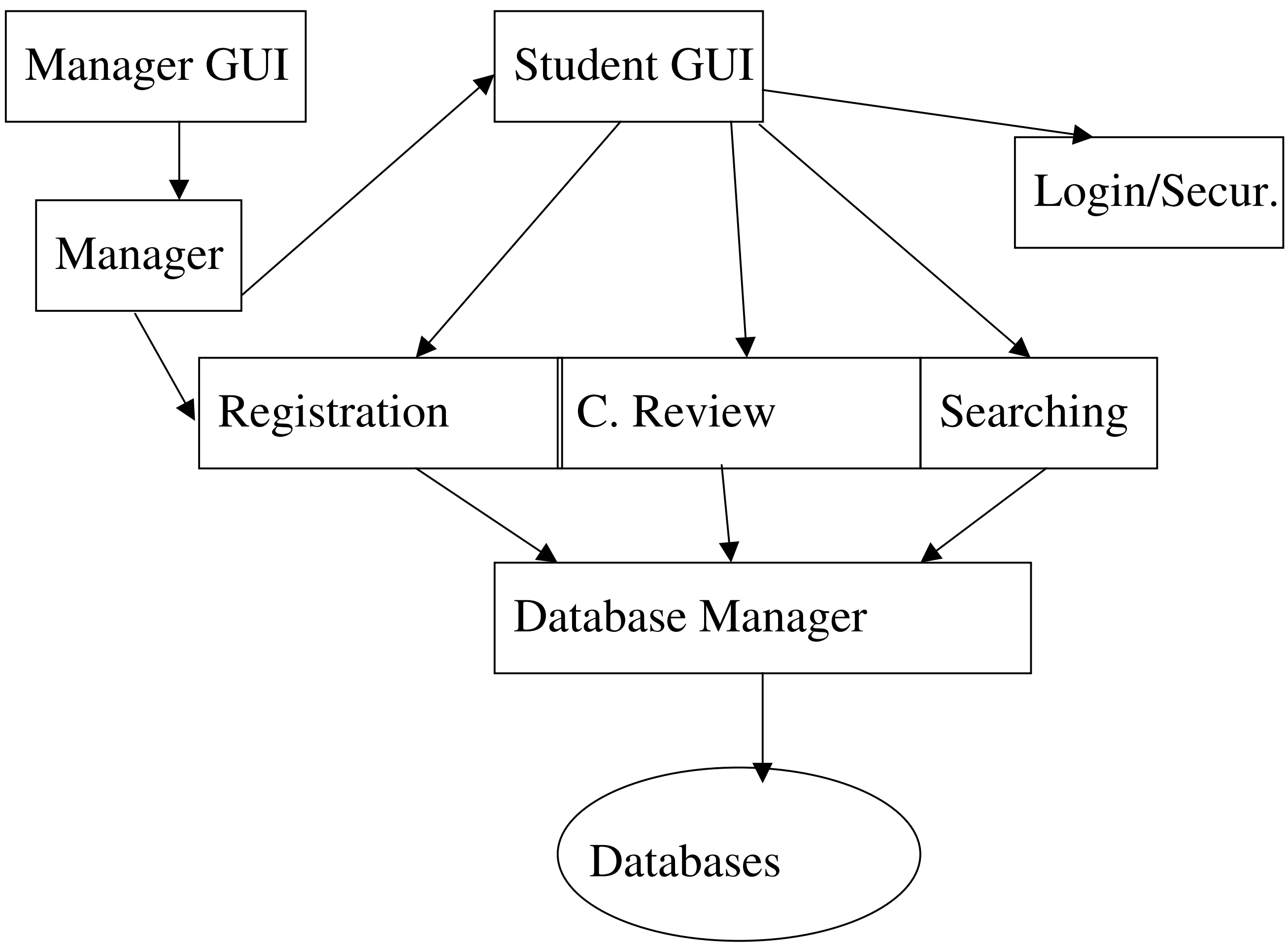
Top-Level Design

February 25, 2000
Author: Melissa Cheng

1.0 Document Overview

This document contains top-level design details for BEAR Advisor, Brown External Access to the Registrar Advisor via the World Wide Web. The details contain views of various top-level components of design, descriptions of these components, possible external dependencies, a task breakdown, a group organization, and a planned implementation schedule.

2.0 Components Overview



2.1 GUI components

The **Student GUI** component handles input from the student user and output to the student user. The Student GUI generates HTML files of consistent “look & feel” and “intuitive” navigation based on information from Login/Security or Logic components. The Student GUI must also handle student user input by passing it to either Login/Security or one of the Logic components.

The **Manager GUI** component handles input from the manager user and output to the manager user. The Manager GUI generates HTML files of consistent “look & feel” and “intuitive” navigation based on information from Login/Security or Manager components. The Manager GUI must also handle manager user input by passing it to either Login/Security or Manager components.

2.2 Login/Security component

The **Login/Security** component checks the access privileges of the student or manager user based upon username and password. The Login/Security component encrypts the username and password information sent over the network from the Student GUI and Manager GUI components, subsequently checks for Kerebos authentication, and returns a message of success or failure to the Student GUI and Manager GUI components.

2.3 Logic components

The **Logic** section is made up of several components: Registration, Course Review, (Smart) Searching. Each component gets information from the Data Manager component to process requests from the Student GUI component and returns appropriate information to the Student GUI component.

- The **Registration** component handles online registration procedures (add/drop/change grade option) which change the student user’s current class status.
- The **Course Review** component checks the validity of incoming requests to add a course review (i.e. makes sure that the student user has previously taken the course).
- The **(Smart) Searching** component queries the Data Manager component for database information on courses, professors, and course reviews. If Smart Searching is implemented, this component will need to search while taking into account student user information (e.g. past courses taken, concentration, and date of graduation).

2.4 Data Manager component

The **Data Manager** component handles requests from Logic components to read from and write to the various databases. Requests from the Registration component will attempt to read from the Course Database, the student user’s past class status, and the student user’s current class status, or write to the student user’s current class status.

Requests from the Course Review component will only attempt to write to the Course Review Database. Requests from the Searching component will only attempt to read from the Course Database and the Professor Public Information Database (e.g. biographical information). If the Smart Searching is implemented, the component will additionally attempt to read from the student user's current class status, the student user's past class status, and the student user's academic preferences files, or write to the student user's academic preferences files.

2.5 Manager component

The **Manager** component handles requests from the Manager GUI component to change deadline constraints used in the Registration and Smart Searching components, and make announcements to all student users. Additional functionality to allow the Manager to alter the Student GUI component and rules used in the Registration component may be more difficult to implement.

3.0 External Dependencies

Testing is extremely important, especially because maintainability is so important to the target managers. Effective testing will largely depend, however, on being able to find dummy data that will simulate "real" data. Also, it will be somewhat difficult to communicate with the Registrar about atypical cases (they know of many).

4.0 Task Breakdown and Group Organization

The **Leader** is the final word on all inter-component decisions. The purpose of this is not to stifle democratic group decision making but instead to make sure the group effectively makes decisions rather than spends too much time in the decision making process. Being sensitive to all group members' needs, strengths, and weaknesses, the leader is able to keep the group on schedule, provide motivation, and keep up group morale. The leader is also responsible for providing resources (e.g. him/herself, someone else, online documents, etc.) and handling/directing group members' questions. In addition, the leader is responsible for administrative needs and recruiting user testers.
Prospective Leader (1 person needed):
Joe Fuqua, interested in administrative/team-building/non-coding

The **Architects** are responsible for the design and implementation of whole components. Architects need to be in close communication with the leader, especially discussing progress or asking for help with problems. Each architect owns one or two components and has progress checks with the leader and works with his/her implementers to code and document the components. Although others may contribute ideas to component design, the architect makes the final component design decisions.

Prospective Architects (3-4 needed):

Lisa Cozzens, interested in general programming;
Melissa Cheng, interested in UI and general programming;
Neelu Bedi, interested in networking and general programming, has strong ownership emotions because her proposal was similar;
Emily Leventhal, interested in UI;
David Yun, interested in UI and databases

The **Implementers** are assigned to work with architects on specific components. Each implementer closely communicates with his/her architect to contribute ideas to and generate feedback for component design; however, the architect makes the final decisions. Each implementer is usually assigned (by the architect) one or more portions of the component to code and document.

Prospective Implementers (3-4 needed):

Lisa Cozzens, interested in general programming;
Melissa Cheng, interested in UI and general programming;
Neelu Bedi, interested in networking and general programming, has strong ownership emotions because her proposal was similar;
Emily Leventhal, interested in UI;
Brett Heath-Wlaz, interested in coding low-level classes and data structures;
David Yun, interested in UI and databases

The **Tester** writes component and system test scripts (similar to cs16 grading scripts) that attempt to break the components with special cases to test functionality, speed, and security. The tester is also responsible for analyzing results of component, system, and user testing, and to publicly report bugs and other feedback. Although this responsibilty may be passed to the leader, the tester may need to talk to the Brown University Office of the Registrar about getting dummy data sets.

Prospective Testers (1-2 needed):

Rikard Grafstroem, interested in user's perspective, not in too much coding
Brett Heath-Wlaz, interested in coding low-level classes and data structures;

The **Editor** incrementally (alongside the tester) checks and updates developer documentation written within code as well as writes component READMEs, system READMEs, and user help file documents. The editor may work with the tester and user testers to improve the "readability" of documentation.

Prospective Editor (1 needed):

Rikard Grafstroem, interested in user's perspective, not in too much coding

General guidelines*:* All group members need to ask questions, make suggestions, and be in communication with the leader.

* NOTE: I do not mean to assign people to roles to only to say which roles I believe they might be interested in based on preferences listed in their CS190 student profiles.

5.0 Planned Implementation Schedule

Below is a tentative implementation schedule for the BEAR Advisor project. As reflected in the schedule, I feel that prototyping is an extremely useful and effective tool early regarding UIs.

f 3/3	Initial group meetings, discuss overall design, finalize group organization.
m 3/6	Finalize role assignments.
f 3/10	Finalize overall project design
w 3/15	Finish first draft of interface, GUI prototypes.
f 3/17	Discuss coding conventions
f 3/24	Finalize interface. Finalize “look & feel” of GUI.
m 4/3	In smaller groups (architects with implementers, etc.) discuss design
w 4/5	Finalize detailed designs
f 4/7	Start coding components! User testing on GUI prototypes
m 4/17	Initial integration, component testing
f 4/21	Whole group meeting to check on progress, people’s happiness, etc.
f 4/28	Functionality Freeze, Full Integration
w 5/3	Finish system testing
f 5/12	Finalize documentation

6.0 Assumptions

One assumption was made that the manager user will be able to directly update the databases through already existing database management programs such as FileMakerPro and Access. That functionality is not included in the Manager GUI and Manager components.