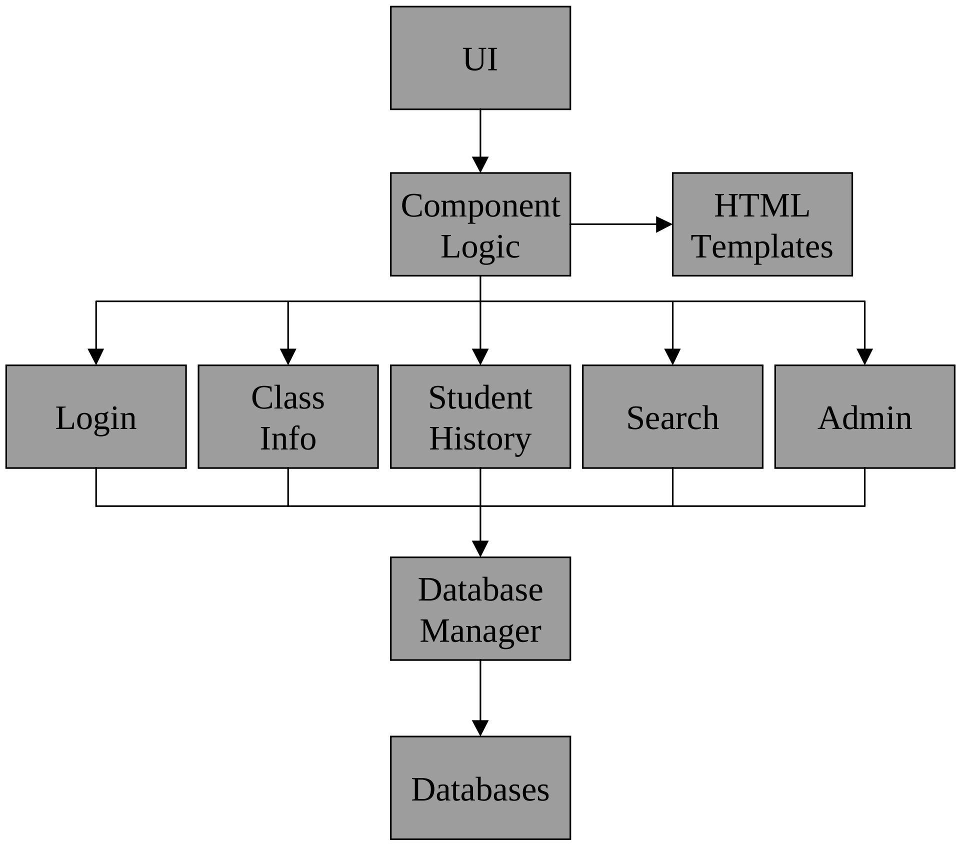
BEAR
Brown **E**lectronic **A**ccess to the **R**egistrar

Top-Level Design
Version 1
February 27, 2000
Author: Brian Chuck

1.0 Document Overview

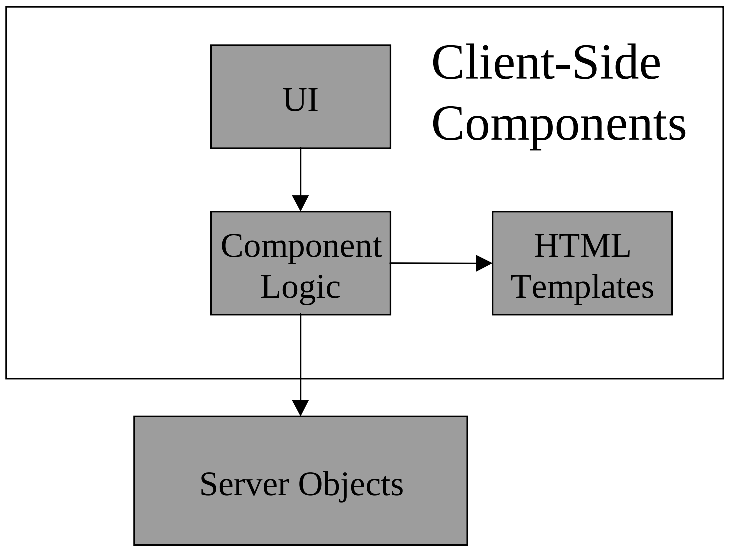
This document contains a top-level design for the BEAR Advisor, the Brown Electronic Access to the Registrar Advisor. It is derived from the specifications documents written by Lisa Cozzens and Melissa Cheng. The design contains the various components of the project, component descriptions, external dependencies, task breakdown, group organization, a preliminary project schedule, and assumptions that went into the design.

2.0 Components Overview



There are 10 components that make up the Top-Level Design. They are: the User Interface, Component Logic, HTML Templates, Login, Class Info, Student History, Search, Admin, Database Manager, and Database. These 10 components can be grouped into two sub-groupings: Client and Server. The Client components are the User Interface, Component Logic, and HTML Templates. The other seven components make up the Server side components.

3.0 Client-Side Components



3.1 User Interface

As this application is web-based, the user will interact with the application through a web browser. Said web browser is responsible for displaying the web pages provided it by the Component Logic. Any input from the user is entered into an HTML form. Once the user decides to submit this form, the web browser gives the form to the Component Logic to handle.

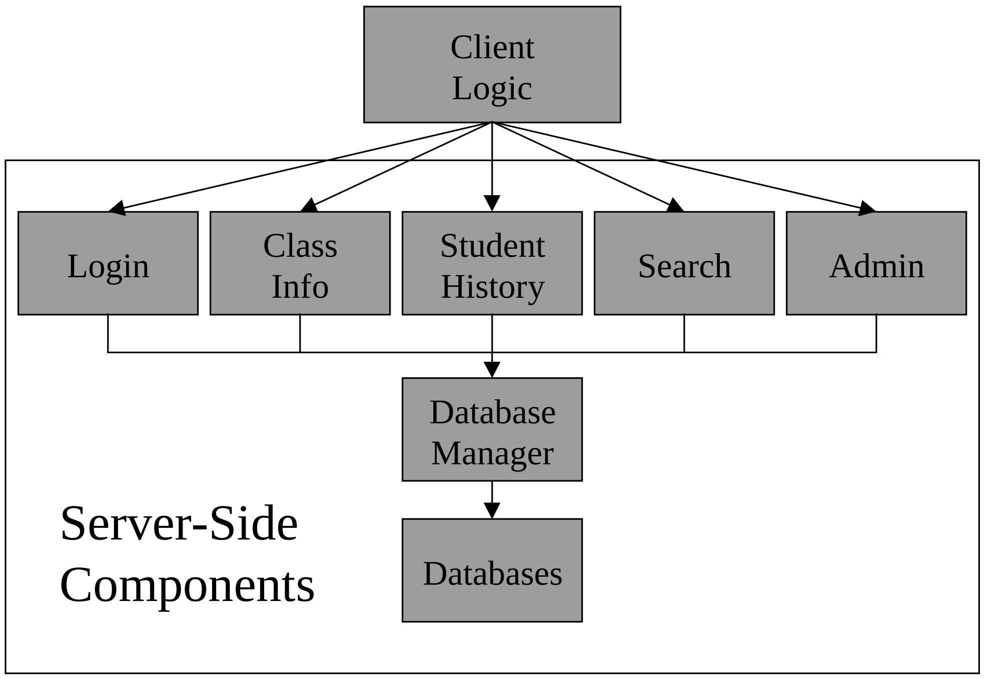
3.2 Component Logic

This component is responsible for taking the user input and communicating with the proper Server objects to either store or retrieve the proper information. Once said information is stored or retrieved, the Component Logic retrieves the proper HTML template, fills it in with the proper information, and gives it to the User Interface to display. The Component Logic always knows what is the proper page to display next. For instance, if the user requests to view his/her transcript, the Component Logic will receive from the User Interface the user's request to view a transcript online. The Component Logic will interact with the proper Server object to get the user's transcript and then the Component Logic will get the transcript HTML template and fill it in with the user's grades and courses taken. The Component Logic then gives this HTML page to the User Interface for it to display.

3.3 HTML Templates

This component will contain templates for all the possible HTML pages displayable. There will also be some pages for which a template is not necessary (e.g.: the login page). The HTML Templates component will keep these pages as well. The responsibility of this component is to return the proper HTML page or template to the Component Logic.

4.0 Server-Side Components



The above components are all Server-side components. Thus, their main functions are dealing with the user's information (i.e.: course registration info, grades, etc.). The top layer of components all use the Database Manager to make database queries and changes to find/enter the proper information for the user.

4.1 Login Component

The Login Component is responsible for allowing a student to log in and out of his/her account. Each time a user accesses the BEAR website, he/she is required to present the system with a login name and a password. The Login Component must verify that the (username, password) pair are legitimate. Once verified, the Login Component lets the Component Logic know that the user is okay to enter the site. The Login Component must also allow for users to change passwords; it must also allow a user to learn what his/her password is, in the event that a password is forgotten. This can be achieved either by sending the user an email with his/her password or by giving the user a hint. It also handles user permissions (i.e.: is the user a student or a professor?).

4.2 Class Info

This component is responsible for handling all of the user’s course issues. The duties include allowing a student to pre-register for a course, add or drop courses, and change grade options for a course. It also contains the functionality for a student getting permission to register for a course. For instance, if a student wishes to pre-register for CS190, he/she will fill out the proper HTML form and submit it. The Component Logic will then tell the Class Info component to pre-register the student in CS190. The Class Info component will check to make sure that the student is not already pre-registered for a class that meets at the same time as CS190. This pre-registering is a database query, so the Class Info component will tell the Database Manager to make the necessary database change. The Class Info component will then notify the Component Logic of the success or failure of the pre-registration. This component also allows the user to declare a concentration, as well as to change concentrations.

4.3 Student History

The Student History component is responsible for all dealings with a user’s history. It is through this component that a user will be able to access past grades and view transcripts online. The user has the options to view his/her grades within his/her concentration, or view all of his/her grades. It should be possible to later on add functionality for the user to order hard copies of his/her transcript.

4.4 Search

This component contains all the functionality of searching for courses for the user to take. The search engine is a “smart” search engine that will allow the user to search for possible courses to take based on a variety of criteria. The user can weight each search type. This component will interact with the Database Manager to find the courses that fit the search criteria; it will then return the courses from the search to the Component Logic.

4.5 Admin

This component allows the system administrator to perform changes to the system’s databases. This includes such actions as backing up the databases and adding on new databases.

4.6 Database Manager

This component is the interface to the Database. The previous five components all require interactions with the database. Rather than directly interfacing, those components all interface with the Database Manager. The Database Manager has functions that allow the other components to make database queries/changes. In the case of a query, the Database Manager queries the database and takes the returns from the database and puts them in a neat format, such as a list, and returns said list to the querying component. This component allows the other components to make database queries using a high-level query language, rather than having to use a low-level DB language. They merely ask the Database Manager for some information and let the Database Manager handle the actual retrieval.

4.7 Database

This is where all the information will be stored on disk. This project will require having a variety of databases. These databases will interact with the Database Manager. They will store all the necessary information, such as each student's course registration, grades, course permission, etc.

5.0 External Dependencies

The major impediment for this project is the cooperation of the Registrar, or rather, their lack of cooperation. This project depends almost entirely on the databases that contain all the necessary information about the students, their courses, and their grades, as well as the databases containing all the course information. As the Registrar thus far has refused to allow us access to their databases, an extensive amount of "dummy" data must be created in order to simulate the above mentioned databases.

6.0 Task Breakdown and Group Organization

6.1 Project Leader

The Project Leader is responsible for the overall progress of the project. His/her duties include monitoring group members' progress and setting/adjusting deadlines accordingly, keeping track of the overall progress of the group, facilitating communication between the different group members, and making sure everyone in the group knows what is going on. Though he/she is responsible for setting all the group deadlines, the entire group must discuss these dates jointly; the Project Leader is the person who takes in the input of all the group members and makes the final decision on the deadlines. He/she is also responsible for keeping group morale up and making sure that everyone in the group is aware of his/her duties.
# of people: 1
suggested person(s) to fill the role: Joe Fuqua

6.2 Architect

The Architect is responsible for the design of the project. Said person will know the project design intimately, and he/she will know all the details and about the design for each piece of the project. He/she is responsible for making sure the project retains its "conceptual integrity." Each group member will have a say in the design of the project, but the Architect will have the final say. Any design changes that may arise during the project implementation must be approved by the Architect.
# of people: 1
suggested person(s) to fill the role: Lisa Cozzens, Rikard Grafstroem, Emily Leventhal

6.3 User Interface Architect

A significant portion of this project relies on the user-system interface. Listed as a high priority is the ease-of-use of the application. This places an emphasis on the User Interface. Therefore, there should be a User Interface Architect who is responsible for designing all of the user interfaces. These include all of the web pages seen by the users, which include students, professors, as well as administrators. The duties of the User Interface Architect will be to consult the members of the group on their opinions as well as getting the input of several students and faculty to get a feeling of how the user interface should appear. The interaction with possible users is critical, since the users should be able to interface with the application as easily as possible. Thus, the User Interface Architect should consult as broad a section of the student body as possible when designing the user interface. The rest of the group members can provide input as well, but the User Interface Architect has the final word.
# of people: 1
suggested person(s) to fill the role: Brian Chuck, Rikard Grafstroem, Melissa Cheng

6.4 Coder

The Coder will do the actual coding and implementation of the application. Each Coder will be given a specific component or set of components to implement. He/she will have the responsibility of making sure his/her component(s) get implemented. Once said component(s) has been implemented, he/she will have the responsibility of fixing any bugs found in his/her code. The Coder can get his/her portion implemented by any means that he/she deems reasonable. If he/she wishes to have other members of the group help him/her, it is allowable, however not at the expense of another person's duties.
# of people: 4
suggested person(s) to fill the role: David Yun, Emily Leventhal, Lisa Cozzens, Brett Heath-Wlaz, Melissa Cheng

6.5 Documenter

The Documenter is responsible for creating the documentation for the project. This includes a User Manual, design documentation, and documentation pertaining to each component and its implementation. The design documentation and implementation documentation should be mostly created by the Architect and Coders; thus the Documenter's duties in these areas will consist mainly of gathering the already written documents, proofreading and editing them, and organizing them in a presentable fashion. Therefore, the main duty of the Documenter is to write the User Manual.
# of people: 1
suggested person(s) to fill the role: Neelu Bedi, Rikard Grafstroem

6.6 Tester

The Tester is responsible for testing not only the entire application, but also for testing each component's functionality. At every deadline and integration step, the Tester must test the application and record any bugs he/she finds. He/she is responsible for maintaining a list or database of all the bugs found, as well as those that get fixed. There are many phases to testing, though, so the Tester will not be responsible for all of them. In the instance of the project implementation, each Coder is responsible for testing out his/her portion prior to any integration step. This requires the Coder to write some test code to test out the functionality of his/her component(s). In addition to the Tester's testing, he/she should arrange for prospective user's (i.e.: various, random students) to test out the application, starting from early implementation, to see if they can find bugs.
# of people: 1
suggested person(s) to fill the role: Neelu Bedi, Rikard Grafstroem

6.7 Miscellaneous

Though each person has an assigned role for the project, that does not mean that each person is confined to his/her role only. For instance, the Project Leader, while keeping track of the overall progress of the project and maintaining group morale, he/she can also participate in other functions. For instance, his/her services are available to help any member of the group out. In addition, all members of the group have a say in any decision, to the extent that each person can give his/her input. However, the final say in each decision falls on the person to whom that responsibility is given. For instance, the Architect has the final say on all design issues.
# of people: 9
suggested person(s) to fill the role: Joe Fuqua, Brian Chuck, Rikard Grafstroem, David Yun, Emily Leventhal, Lisa Cozzens, Melissa Cheng, Brett Heath-Wlaz, Neelu Bedi

7.0 Assumptions

As it was not clear in the specifications, it is assumed that professors as well as students are potential users of the system. However, it is assumed that professors can only check online to grant permission to a course. As a later addition to the project, functionality for professors to record grades and write student performance reviews can be added with relative ease. It was also assumed that the Admin tools would be online as well.

8.0 Schedule

3/3 – Top-level design selection

- Project responsibilities assigned

3/10 – Final top-level design

- All revisions of top-level design collected and made
- Top-level design frozen
- Project assignments frozen

3/15 – Interface proposals

- Each component's Coder produces a proposal for their external interfaces

3/20 – Interface comments

- Each member should go over all interface proposals, including ones that do and don't affect their components and provide feedback, comments and suggestions

3/24 – Final interface definition

- Each component owner finalizes his/her interface by taking into account all comments made
- Approval for each interface must be granted by both the Project Leader and the Architect
- Interfaces are frozen and can only be changed through a thorough review process

4/5 – Detailed designs

- Each Coder should have a finished design approved by the Architect and project TA
- Once approved, design should be recorded
- Designs are frozen

4/17 – Initial systems integration

- Initial viewing of the project
- Most of the UI should be here, with the core implementation ready

4/21 – Initial system tested, debugged

- By this date, Tester as well as users should have tested the initial system and recorded any bugs found
- Coders should have fixed most of, if not all, bugs found

4/28 – Full system implementation

- System should be fully implemented and functional
- In-class demo should be possible
- Functionality frozen

5/1 – Debugging, Testing

- Extensive testing should be done by Tester, as well as each member of the group, and as many users as possible
- Any bugs found should be duly recorded and fixed

5/4 – Public Demo

- Project demo in Lubrano

5/4 – 5/12 – More debugging, testing

- Testing should be going on to find all bugs in the system
- Fix any bugs found
- Documentation should have been created ongoing as the project develops; any documentation still required should be produced

5/12 – Final system submission

- Hand in of system
- All coding, testing, and debugging must be finished at this point
- All documentation must be finished and handed in