MASSACHUSETTS INSTITUTE OF TECHNOLOGY
ARTIFICIAL INTELLIGENCE LABORATORY

June, 1996

Prototype of a Configurable Web-based Assessment System

Miguel Hall

This publication can be retrieved by anonymous ftp to publications.ai.mit.edu.

Abstract

The MIT Educational Assessment System provides subjects and courses at MIT with the ability to perform online assessment. The system includes polices to handle harassment and electronic “flaming” while protecting privacy. Within these frameworks, individual courses and subjects can make their own policy decisions about such matters as to when assessments can occur, who can submit assessments, and how anonymous assessments are. By allowing assessment to take place continually and allowing both students and staff to participate, the system can provide a forum for the online discussion of subjects. Even in the case of scheduled assessments, the system can provide advantages over end-of-term assessment, since the scheduled assessments can occur several times during the semester, allowing subjects to identify and adjust those areas that could use improvement. Subjects can also develop customized questionnaires, perhaps in response to previous assessments, to suit their needs.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for this research was provided in part by the Advanced Research Projects Agency of the Department of Defense under contracts MDA972-95-1-0018 (North East Regional Consortium Grant) and N66001-95-C-8611 (Computer Aided Education and Training Initiative).

Prototype of a Configurable Web-based Assessment System

by
Miguel G. Hall

Submitted to the Department of Electrical Engineering and Computer Science
on May 28, 1996, in partial Fulfillment of the Requirements for the Degrees of
Bachelor of Science in Computer Science and Engineering
and Master of Engineering in Electrical Engineering and Computer Science

ABSTRACT

The MIT Educational Assessment System provides subjects and courses at MIT with the ability to perform online assessment. The system includes polices to handle harassment and electronic “flaming” while protecting privacy. Within these frameworks, individual courses and subjects can make their own policy decisions about such matters as to when assessments can occur, who can submit assessments, and how anonymous assessments are. By allowing assessment to take place continually and allowing both students and staff to participate, the system can provide a forum for the online discussion of subjects. Even in the case of scheduled assessments, the system can provide advantages over end-of-term assessment, since the scheduled assessments can occur several times during the semester, allowing subjects to identify and adjust those areas that could use improvement. Subjects can also develop customized questionnaires, perhaps in response to previous assessments, to suit their needs.

Thesis Supervisor: Carl Hewitt
Title: Associate Professor, Electrical Engineering and Computer Science

Table of Contents

List of Figures

Acknowledgments

Carl Hewitt was instrumental in getting this prototype working, providing and soliciting feedback about all aspects of the system. Anne Hunter, Mary Rowe, and Jeff Schiller were invaluable in helping formulate the policies to be used by the system. Many people, including Carl Manning, Fanya Montalvo, Marilyn Pierce, Bill Siebert, and Peter Szolovits, read initial versions of this and other documents, providing feedback and pointing out things that could be improved.

Portions of the mail sending code were adapted from the code for “Blat,” a public domain command line mailing utility for Microsoft Windows NT.

I would also like to express my appreciation to Microsoft for donating all of the software necessary to develop the MIT Educational Assessment System.

1. Introduction

The MIT Educational Assessment System is an online computer system that is intended to provide subjects at MIT with a method for enhanceable assessment. The system will enforce policies to help prevent and handle harassment and electronic “flaming” while protecting the privacy of the participants. Within this framework, individual subjects and courses will have the ability to make their own policy decisions and create their own questionnaires.

Subjects can choose to allow assessment to occur continually throughout the semester or at several specific points in the term, such as add date and drop date. Subjects can also decide whether they wish to allow staff members to participate in the assessment process. If subjects choose to allow ongoing assessment and staff members can participate, the system can serve as a forum for online discussion about the subjects.

The MIT Educational Assessment System will enable subjects to perform assessment during the term if they wish. Students may be more likely to offer constructive suggestions if they feel that they may directly benefit from them. Subjects can choose varying levels of participation, depending on their resources. While some subjects may have staff members who actively participate in the assessment process, others may simply encourage students just to submit assessments.

We designed the MIT Educational Assessment System to be configurable. Subjects or courses will be able to make individual policy decisions. They will be able to decide when assessments can occur, whether staff members can participate, and who can access the assessments. In addition they can decide how anonymous the assessments shall be and whether the assessments will be reviewed before being posted. Subjects will also be able to create customized questionnaires.

2. Background

The MIT Educational Assessment System consists of a relational database that is integrated with the World Wide Web, allowing participants to submit and review assessments through web browsers. We provide an overview of the relational database model in AppendixA. In this section, we discuss some of the issues involved in integrating the World Wide Web with a database, and we provide a brief discussion of some of the possible consequences of electronic assessment.

2.1 Interfacing to Software From The World Wide Web

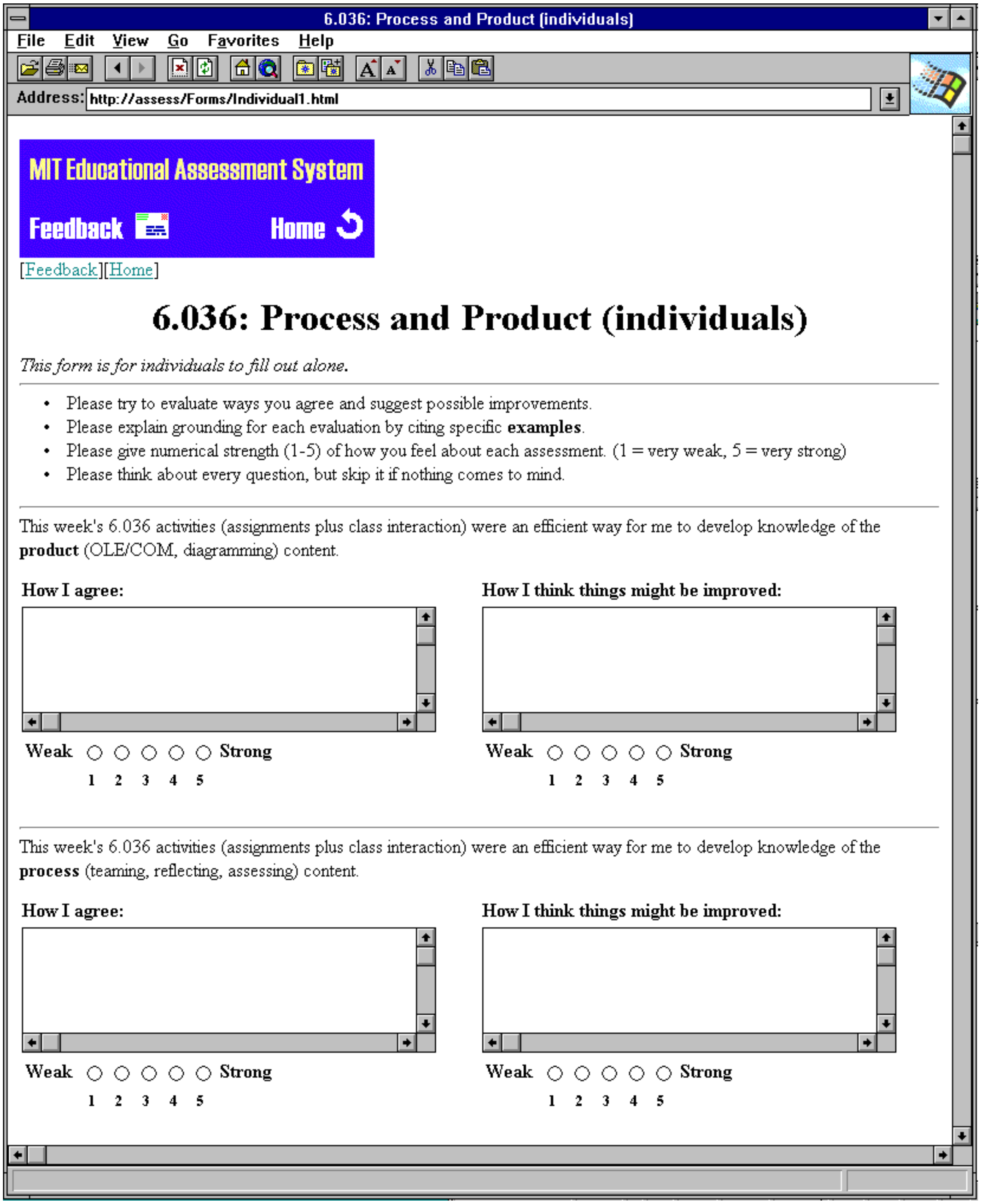
In order for the World Wide Web front-end to interact with a relational database, it must have some mechanism for communication. Fortunately, the hypertext markup language (HTML) specification defines protocols for gathering information from the user and communicating it to software.

2.1.1 HTML Forms

The HTML specification defines forms, which the web server can use to gather data from the user. All major web browsers currently in use support HTML forms, and most users are familiar with them. An HTML form is a template that allows the user to input a specified set of information and then perform an action on the data.[1] The form designer can name and place several types of input elements, including text entry fields, selection lists, radio buttons (one-of-many selections), check boxes (n-of-many selections), and buttons. The first page of a sample assessment form is shown in Figure 2.1.

Each form also contains an associated action URI (Uniform Resource Identifier), which identifies the application or script that will process the data. The web server communicates with the application using the Common Gateway Interface (CGI) which is described briefly in Appendix B.

[1] T. Berners-Lee and D. Connolly, *Hypertext Markup Language - 2.0*, RFC 1866, HTML Working Group, 1995. Page 40.


6.036: Process and Product (individuals)

File Edit View Go Favorites Help

Address: http://assess/Forms/Individual1.html

MIT Educational Assessment System

Feedback Home

[Feedback][Home]

6.036: Process and Product (individuals)

This form is for individuals to fill out alone.

- Please try to evaluate ways you agree and suggest possible improvements.
- Please explain grounding for each evaluation by citing specific **examples**.
- Please give numerical strength (1-5) of how you feel about each assessment. (1 = very weak, 5 = very strong)
- Please think about every question, but skip it if nothing comes to mind.

This week's 6.036 activities (assignments plus class interaction) were an efficient way for me to develop knowledge of the **product** (OLE/COM, diagramming) content.

How I agree:

Weak ○ ○ ○ ○ ○ Strong
1 2 3 4 5

How I think things might be improved:

Weak ○ ○ ○ ○ ○ Strong
1 2 3 4 5

This week's 6.036 activities (assignments plus class interaction) were an efficient way for me to develop knowledge of the **process** (teaming, reflecting, assessing) content.

How I agree:

Weak ○ ○ ○ ○ ○ Strong
1 2 3 4 5

How I think things might be improved:

Weak ○ ○ ○ ○ ○ Strong
1 2 3 4 5

Figure 2.1: A sample assessment form

2.2 The Effects of Electronic Assessment

The MIT Educational Assessment System will be a web-based system. Therefore, all assessments will take place electronically. People's behavior is different online than it is in either face-to-face communication or paper based assessment.[2]

In addition, the MIT Educational Assessment System will allow the possibility of anonymous assessments, and assessment will be able to occur continually throughout the semester. Both of these facts can have an impact on the behavior of the users.

2.2.1 Openness

Research into the behavior of people responding to surveys has shown that there is a marked difference in the responses to paper surveys and electronic surveys.[3] Even when both types of surveys are anonymous and self-administered, responses to the electronic surveys tend to be more honest. In particular, the responses are less biased towards socially acceptable responses. Open-ended questions also tend to elicit longer responses on the electronic surveys than on the paper ones.[4]

With the MIT Educational Assessment System, subjects will be able to allow assessments to take place anonymously. Students are much more likely to be open about their

---

[2] Lee Sproull and Sara Kiesler, *Connections: New Ways of Working in The Networked Organization*, MIT Press, 1991. Page 58.
[3] Sara Kiesler and Lee Sproull, "Response Effects in the Electronic Survey." *Public Opinion Quarterly, Vol. 50*, 1986. Page 402.

opinions in an anonymous setting. Students will be less likely to say what they think their instructors want to hear, and will be less likely to “go along with the crowd”.

2.2.2 Critical Assessments

One of the major purposes of the MIT Educational Assessment System is to allow students and staff to assess subjects during the term, allowing the subjects to improve. Since the intent is improvement, many of the submitted comments are likely to focus on the areas of a subject that are lacking in some way. The tone of the assessments will therefore be more critical than it might otherwise be.

Critical assessments serve a purpose during the term. They point out the shortcomings of a subject, and provide a forum for students and staff to discuss alternatives or enhancements. However, these critical assessments are much less appropriate for general dissemination to MIT after the semester. Subjects can choose to prepare summaries of the assessments near the end of the term. These summaries could be released to the general MIT population.

---

[4] Sara Kiesler and Lee Sproull, 1986. Page 411.

3. Design

We decided from the outset that the assessment tool would be online, since we wanted to create a system that could be easily configurable. The subjects at MIT are diverse and have diverse needs. In order to effectively meet those needs, the system would have to be customizable, allowing specific courses and subjects to adapt it to satisfy them. The system would allow subjects or courses to make their own policy decisions and create their own electronic questionnaires. Instead of trying to create a system with one set of guidelines and questionnaires that would please everyone, we decided to build a configurable framework that would allow subjects to conduct assessments in their own ways.

As an online system, the MIT Educational Assessment System would be available to all of the students and staff of MIT constantly, allowing subjects to conduct assessments without sacrificing class time. The assessments may be more comprehensive, since users can be allowed to submit assessments when they wish. In addition, Lee Sproull and Sara Kiesler have conducted some research indicating that people may be more forthcoming in online surveys than they would be either in paper surveys or in person.

3.1 The medium of choice

We initially considered basing the system on either email or the World Wide Web. Eventually we decided to create a web-based system after comparing the World Wide Web to email in the following areas.

3.1.1 Availability

All students and faculty at MIT can obtain an MIT account with email. In addition, many people can access their email accounts from home, over phone lines. People are familiar and comfortable with email.

The World Wide Web is a much more recent development than email, but it has experienced a much greater rate of progress recently, and it is beginning to become nearly as ubiquitous as email. The MIT community has access to web browsers via the MIT networks, such as Athena. A large and growing number of students connect their computers to the residential computer network, Resnet, meaning they are likely to have access to web browsers from their homes. Many students already spend a considerable amount of time “surfing” the web, either from Athena or from home.

While not yet as universal as email, the World Wide Web is becoming extremely common. We concluded that we would not be limiting the availability of the MIT Educational Assessment System by basing it on the web.

3.1.2 Usability

In an email-based assessment system, the system could mail questionnaires to participants at appropriate times. For those subjects desiring a continual assessment, the system could mail questionnaires weekly. Other subjects could choose to send out questionnaires only at specific times in the semester, such as drop date and add date. Participants could then fill out the questionnaires and send them back to the assessment system. The system would then have parse the responses to specific questions from the emailed replies. One of our concerns about an email-based system is that participants might not return the questionnaires.

Another problem with an email-based system is that users of the system would not have immediate access to the assessments. The system could either mail the submitted assessments to all of the participants automatically, or to individuals in response to requests. Even in the latter case, though, the results would not be immediately available, due to the nature of email.

For a web-based system, the paradigm would be somewhat different. Subjects could choose to allow users to submit assessments on their own schedule. More importantly, users could review the submitted assessments when they wanted to, and perhaps respond to them immediately. The system would be more immediately gratifying to the users, and therefore more likely to be used.

A web-based system would be easier to interface to a database than would an email-based one. The HTML specification allows for well-defined forms in web documents, and protocols such as the Common Gateway Interface (CGI) allow the web server to call applications from a web document. In contrast, extracting the necessary information from an email message would be more problematical, as it would be impossible to guarantee that users would not send malformed messages.

The World Wide Web also offers much more room for growth than email. For example, the system could eventually contain graphical information. New features are also continuously being added to web browsers that would allow for even greater improvement. A system based on the World Wide Web would almost certainly be more visually appealing than an email system. Visual appeal is an important aspect of any system.

We believe that more people would use a web-based system more often. In addition to being easier to use and implement, it is more appealing and provides more room for improvement. In short, the MIT Educational Assessment System would benefit from those same aspects of the web that are causing it to receive the amount of media and commercial attention that it is.

3.1.3 Authentication

As we discuss in section 3.2.2, the MIT Educational Assessment System requires the ability to perform user authentication. An email-based system would allow for a crude sort of authentication based on email usernames. Unfortunately email usernames are trivial to compromise, and many students know how to do so. In particular, users could easily impersonate others. On the other hand, forged email is not a new problem, and mechanisms exist to track down the perpetrators and deal with them appropriately. An email-based system would effectively reduce the impersonation problem to one the Institute already attempts to handle.

All web browsers offer a basic form of authentication, and some additionally offer more secure ones. While not perfect, the basic authentication is far superior to the use of email usernames.

One of the problems with the basic authentication is that it sends usernames and passwords over the network in plain text. Modern web browsers will soon contain support for the Secure Sockets Layer (SSL). When a web system uses SSL, it encrypts all data sent between the web browser and the server, including authentication information and form submission data. While configuring the MIT Educational Assessment System to use SSL would limit the availability somewhat, soon the majority of users will have access to web browsers with SSL support.

3.1.4 Conclusion

After considering the above issues, we decided to create a web-based assessment system. While access to the World Wide Web is not yet as universal as access to email, we do not feel that the MIT community will have problems obtaining access to a web-based system. In addition, a web-based solution is more robust, is more likely to be used, and has more room for future development.

The MIT Educational Assessment System consists of three major pieces. The users see the World Wide Web pages, and the assessment data is stored in the database back-end. In between the web pages and the database are some supporting applications that perform the communication. This hierarchy is shown in Figure 3.1.

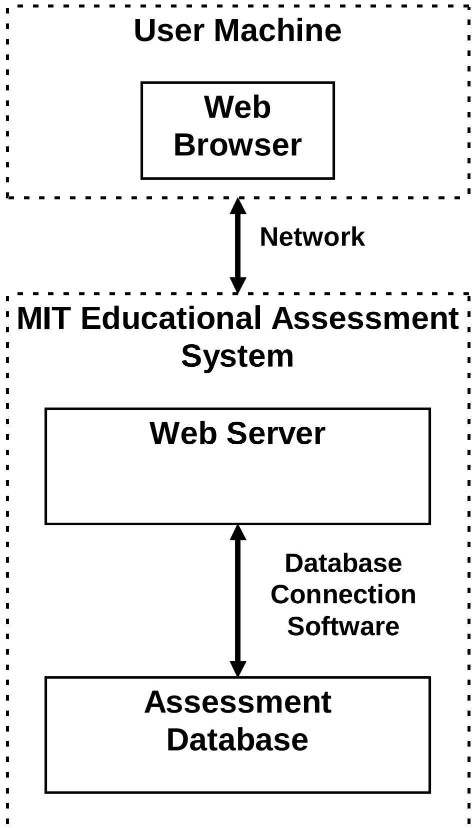


Figure 3.1: Overview of the major pieces of the assessment system

3.2 Issues of Policy

The needs of subjects at MIT vary widely. In order for an assessment system to satisfy these needs, it must be highly configurable. We designed our system to allow subjects and courses to make their own policy decisions.

3.2.1 Professionalism

The MIT Educational Assessment System must maintain a high degree of professionalism. In all cases, the system must enforce several polices.

- Prohibit harassment
- Discourage electronic "flaming"
- Protect privacy

Within the framework of these constraints, courses and their subjects will also be able to set policy.

3.2.2 Policy Choices

We have identified several policy issues that courses and subjects need to address. Our goal in designing the assessment system was to leave as many of the policy decisions as possible up to the courses and subjects.

Timing of Assessments

When should participants be able to submit assessments? Some subjects may desire to accept assessments continuously throughout the semester. One benefit of continual assessment is that participants can submit assessments whenever they have something to say. If subjects allow participants to initiate the assessments, the participants will be able to submit an assessment while issues are fresh in their minds. Also, incremental assessment may seem less overbearing than being faced with a form full of questions all at once.

Some subjects, however, may want more control over when assessments take place. These subjects could define any number of specific times for assessment forms to be available, such as drop date, add date, or the end of the semester. Since subjects could offer different assessment questionnaires at different times, they could elicit fresh insight from the participants.

Both of the above scenarios offer advantages. Most subjects will likely offer a combination of the two. Subjects could make different questionnaires available at different times throughout the semester. Subjects could develop these questionnaires in response to the previous assessments, allowing them to receive more in depth information, and the system could possibly notify participants when new questionnaires become available. In addition to these questionnaires, subjects may choose to allow participants to respond to the assessments of others, providing a means of ongoing discussion.

Staff Participation

Allowing subject staff to take part in the assessment process could create a useful discussion about the subject. If the subject staff responds to issues that the students raise, students may feel that their assessments are being taken seriously. They will likely be more inclined to submit assessments that are designed to improve the subject as opposed to simply complaining about it.

Not all assessment questionnaires may be appropriate for staff submissions. Some subjects may decide only to allow staff to respond to students' assessments, while others may have specific questionnaires for staff members.

Privacy

Courses and their subjects will be able to decide how anonymous the assessments will be. Some subjects may desire all of the student assessments to be anonymous, while others may have some anonymous questionnaires and some signed questionnaires.

Many students are apprehensive about honestly assessing a subject. Students may worry about what their instructors and fellow students will think about their assessments. By allowing anonymous assessments, subjects can alleviate these fears and elicit more honest responses.

Anonymous assessment raises issues of privacy and harassment. The MIT Educational Assessment System must have mechanisms to deal with the possibilities of harassment and electronic "flames". Therefore, even when postings are anonymous, the system will internally authenticate users. The system will require that all users obtain accounts, and will use these accounts internally to handle submissions properly. For example, if a subject allows users to vote on postings, expressing their approval or disapproval, the system must be able to determine which users have already voted, to prevent duplicate votes. However, the system will encrypt the information linking people to accounts, and make it available, in the case of offenses, only to those parties legally authorized to obtain it. The system could automatically destroy these encrypted records at the end of the semester.

Students and staff will be able to sign up through the web to use the assessment system. When they sign up, the system will prompt them for a desired username and password, as well as their current email address at MIT, as shown inFigure 3.2. When the system creates an account, it mails a confirmation to the provided MIT email address. The user must supply the confirmation to activate the account before using it the first time. By using such a confirmation process, we can reduce the possibility of impersonation during the sign-up phase.

Moderated Assessment

Some subjects may wish to have a team of moderators review assessments before posting them. By reviewing assessments prior to posting them, subjects will be able to help keep discussion focused and screen out any harassing or off-topic submissions. On the other hand, reviewing submissions requires moderators and slows down the discussion. Therefore, some subjects may not want to have the assessments be reviewed before the system posts them.

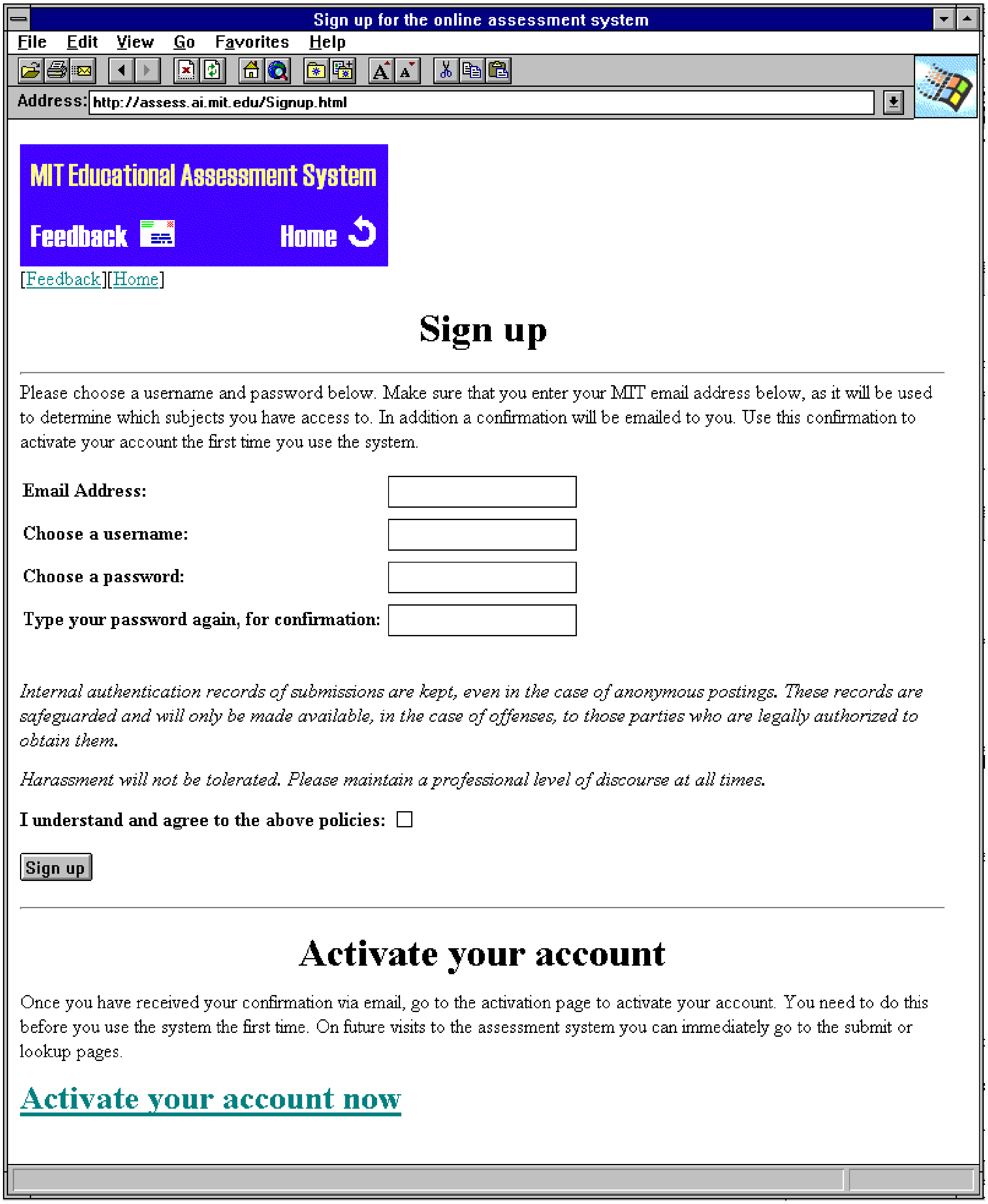


Figure 3.2: The sign-up screen

Even those subjects that do not have moderators could choose to have some mechanism to remove inappropriate postings after the fact. For example, a subject could designate a team of students or staff members that would have the power to remove postings from the system (without necessarily knowing who the authors were) and refer any harassment to the proper authorities.

Since students will be aware that they can be held accountable for their submissions, they should be less likely to abuse the system than if it was completely anonymous internally.

Access to Assessments

During the semester, many subjects will want to allow only those students and staff associated with the subject to read the assessments, since some of the students or staff might not want the remarks made by or about them to be available to everyone. However, near the end of the semester the subject assessments can be valuable to other students attempting to plan their schedules.

One possibility would be to have summaries of the assessments prepared after the semester is over. These summaries could be made available to the rest of MIT, providing students and faculty with useful information, while protecting the privacy of those involved with the subjects. An alternative would be to have an end-of-the-term assessment. Students (and possibly staff) could fill out questionnaires for the express purpose of conveying their overall impressions of the course.

Subjects with large enrollments may find assessment problematical. These subjects could choose to have a separate assessment forum for each section. Section assessment forums can more easily keep discussion focused, promote a sense of community, and discourage electronic “flaming”.

One problem with assessing the sections separately is that different sections would not be able to interact with each other, which might deprive some sections from taking part in interesting discussions. One possibility is to provide a subject-wide, moderated discussion forum. Moderators (perhaps the section instructors) could post interesting topics from the individual sections. These topics could “seed” the section discussions, and ensure that each section is exposed to the ideas brought up by other sections.

Origin of Questionnaires

We would be unable to create a single set of blank questionnaires that would satisfy the needs of all of the subjects at MIT. Therefore, the MIT Educational Assessment System will allow any subject to use customized forms. Students, subject staff, departments, schools, or the Institute might want to develop new blank questionnaires. We show two sample assessment forms in AppendixF.

4. Implementation

We have developed a working prototype of the MIT Educational Assessment System. The system will certainly undergo changes as we receive more feedback from participating subjects, and we list some possible improvements in Chapter 5. We have attempted, wherever possible, to keep our options open, so that the system can easily grow.

4.1 The World Wide Web Front-end

We decided to use the Microsoft Internet Information Server (IIS), which runs under Microsoft Windows NT Server. IIS contains the Internet Database Connector (IDC), which is built-in support for interfacing to databases. Unfortunately, however, the IDC lacked sufficient functionality for our purposes, so, as described shortly, we implemented our own connector.

Since we chose not to use the database connectivity features of IIS, our system could easily be ported to another web server if need be.

4.2 The Database Back-end

We chose to use Microsoft Access as the database for our prototype. In the future, as the system becomes more widely used, we will probably have to move to a more powerful database system, such as Microsoft SQL Server. The move from Access to another database should be fairly easy, as the connection software that we developed works for any Open Database Connectivity (ODBC) compliant database.

4.2.1 Organization of Information in The Database

We designed the database to conform to the Third Normal Form as described in Appendix A. The data for each questionnaire is contained in a separate table. For example, a questionnaire that simply allows free-style comments and responses is shown in Figure 4.1.

SubmissionNumber	ResponseTo	SubjectNumber	Username	Date	Submission
1		6.036	user1	5/3/96	This is the first comment.
2	1	6.036	user2	5/4/96	This is a response
3	2	6.036	user1	5/5/96	This is another response.

Figure 4.1: Table for a sample questionnaire

4.3 Supporting Software

Designing the web pages for the front-end and the database tables for the back-end are only a part of the problem. The supporting software that we developed is the “glue” that holds the system together. In particular, we required software to interface the front-end to the back-end and software to facilitate signing new users up to use the system.

4.3.1 Interfacing to The Database From The World Wide Web

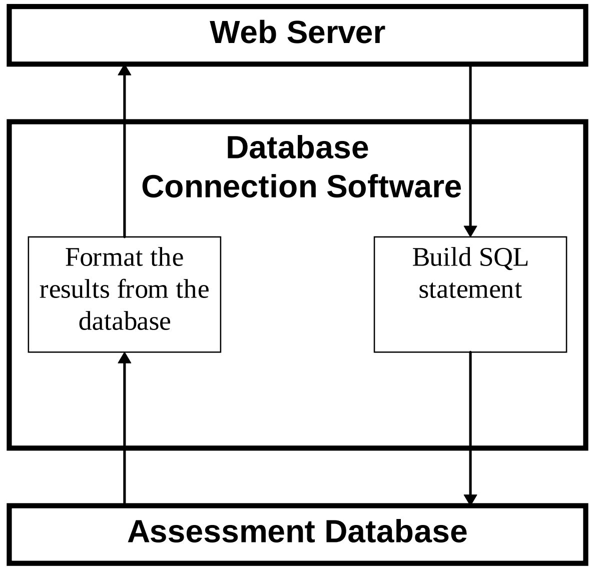


Figure 4.2: Overview of the operation of the database connection software

Although the Internet Information Server comes with the Internet Database Connector, which allows web pages to interact with Open Database Connectivity (ODBC) compliant databases, we chose not to use the built-in support. The database connector has several shortcomings that make it inadequate for our purposes, so we have developed our own connector, which provides more of the functionality we need but otherwise behaves similarly to the Internet Database Connector. The Internet Database Connector is described in Appendix C.

As Figure 4.2 shows, our database connector, in response to a request from the web server, generates and executes a Structured Query Language (SQL) statement. The results from the database are retrieved, processed according to a template, and returned to the web server as a hypertext document that can be displayed by a web browser.

When the database connection software communicates with the database, it uses the Open Database Connectivity (ODBC) protocol, which is an industry-standard protocol. Since the connector uses a standard communication protocol, we can easily move the system to a different database system, such as SQL Server, without rewriting the connection software.

The database connection software that we developed is described in more detail in Appendix D. The database connector makes use of two data files, which describe the query that the database connector is to generate and the template that it is to use when formatting the results of the query. The format of these files is described in Appendix E.

4.3.2 Signing up Users

Before they can use the MIT Educational Assessment System, users must obtain accounts on the system. We desired to make it as simple as possible for users to obtain accounts, so we developed an application, which is described in Appendix D, to allow them to sign up through the World Wide Web.

In the sign-up process, as shown in Figure 4.3, the user requests a desired username and password, while supplying an MIT email address. If the user has permission to sign up to use the system, and the desired username is not already in use, the sign-up application creates a new account and emails a confirmation to the user. Since the user must provide the confirmation in order to activate the account before using it the first time, users cannot sign up to use the system while impersonating someone else.

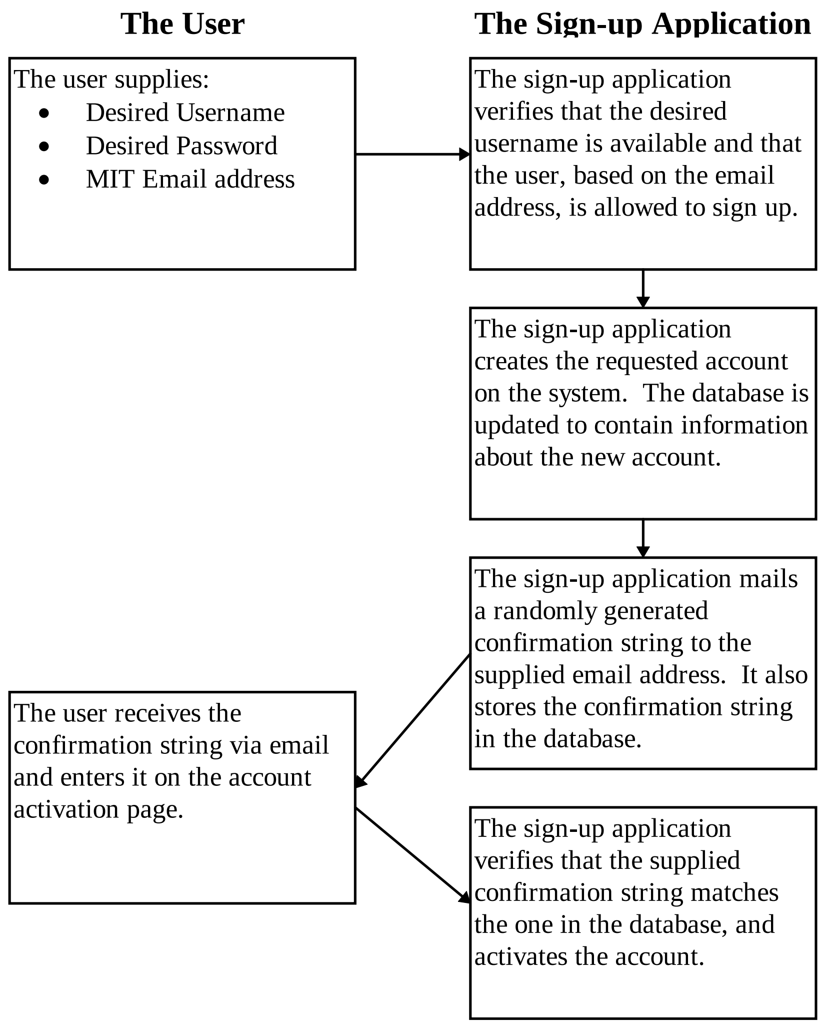


Figure 4.3: The sign-up process

4.4 Submission Example

In Figure 4.4, we show a sample form in both the blank and filled-in states. The displayed form enables users to submit free-style comments, questions, or suggestions. Other users can view the comments, sorted by titles, and respond to them.

When a user fills out the form and clicks the "Submit Comment" button, the following things happen:

- The database connection software retrieves the form input data from the web server. In this case, the data consists of the comment, its title, and the subject number. In addition, the web server sends the connection software the user's username.
- The database connection software ascertains that the user has permission to submit assessments for the chosen subject.
- The database connection software builds a database query from the data. When the query is executed, it causes the database to be updated. Figure 4.5 show a sample entry in the database. It shows the data that the query places in the database, and the data types that the database (in this case Access) expects each field to be. A portion of the database schema is shown in Figure 4.6, showing the relationships between the table containing the submitted comments, a table with general information about a subject, and a table allowing users to vote on comments.

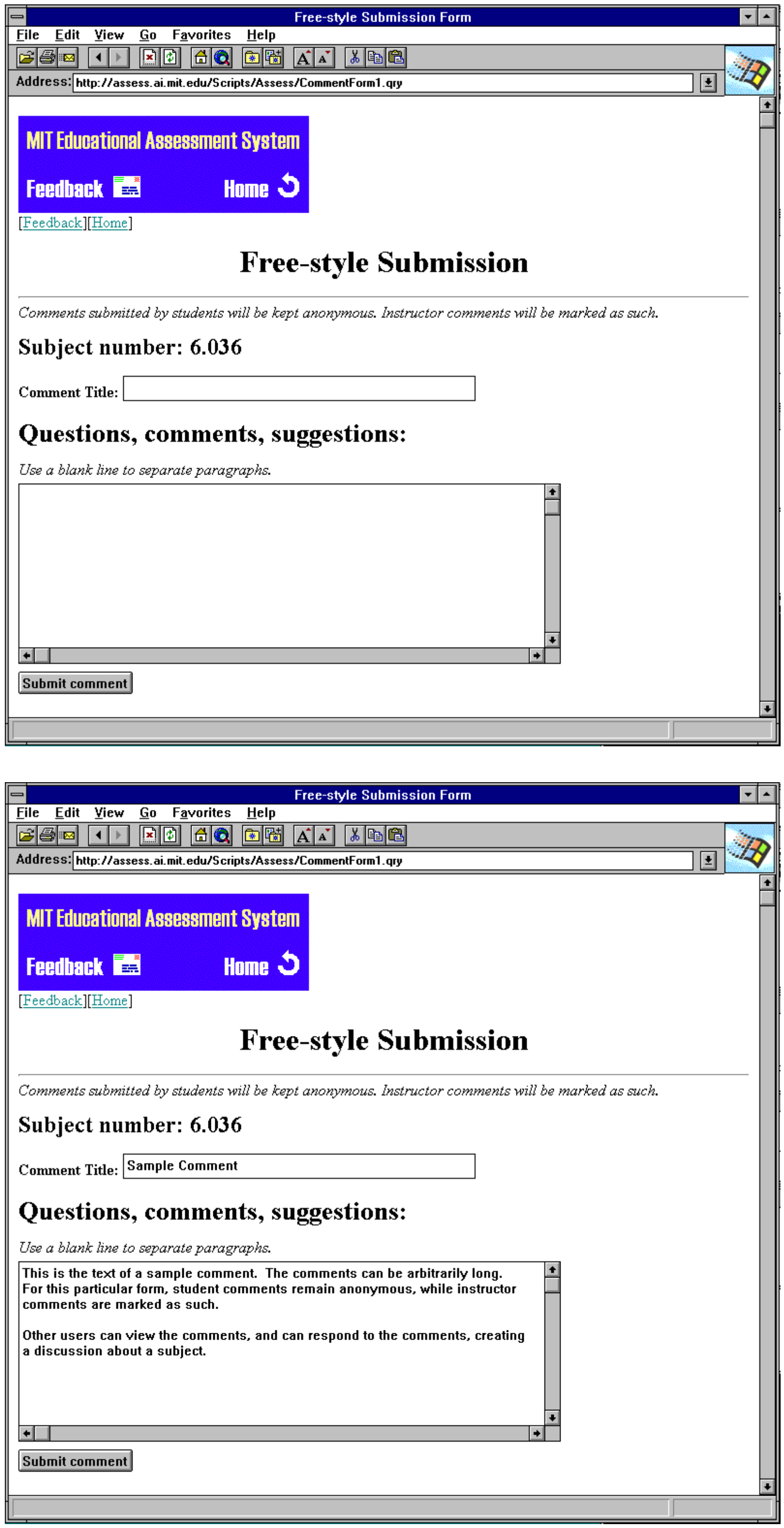


Figure 4.4: Sample comment form being filled out

Field Name	Access Data Type	Data
SubmissionNumber	AutoNumber	23
ResponseTo	Number	
Title	Text	Sample Comment
SubjectNumber	Text	6.036
Username	Text	user1
Date	Date/Time	5/15/96 16:23
Submission	Memo	This is the text of a sample comment. The comments can be arbitrarily long. For this particular form, student comments remain anonymous, while instructor comments are marked as such. Other users can view the comments, and can respond to the comments, creating a discussion about a subject.

Figure 4.5: Sample entry in the database

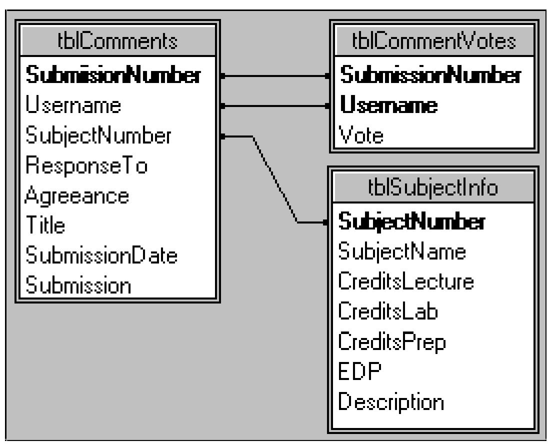


Figure 4.6: The database schema for the comment form.

5. Future Improvements

While functional, the current implementation of the MIT Educational Assessment System is not complete. The system is sure to undergo changes as more subjects participate and provide feedback.

Essential Improvements

- A server machine must be acquired.
- The system must be made secure

Optional Improvements

- The creation of new questionnaires could be simplified.
- The assessment system could be made more interactive.

5.1 Making The System Secure

In its current implementation, the MIT Educational Assessment System requires that users obtain accounts on the system before they can use it. The system authenticates all submissions internally, even if the postings are anonymous. Before the system can be used by MIT subjects in general, it must have a higher level of security. In particular, the information linking accounts to people must be encrypted, available only to those who are legally authorized to obtain it.

Also, we will use the Secure Sockets Layer (SSL), even though it will reduce the compatibility of the system with older web browsers. Fortunately both Netscape and Microsoft browsers now support SSL. SLL provides data encryption for the communication between the web server and the web browsers, increasing the security of the MIT Educational Assessment System considerably.

5.2 Simplifying The Creation of New Questionnaires

One possible method of simplifying the creation of new blank questionnaires would be to create a simplified questionnaire development front-end, executed from the web, which would allow authorized users to create new questionnaires through the web.

Such an application could simplify the development of basic questionnaires. Even if the application did not allow the same amount of flexibility as using an Access front-end, it would still be useful to most people. Many questionnaires consist of a sequence of questions, the responses to which are either text or multiple choice.

The simplified front-end could allow the user to specify how many questions the questionnaire would have, and the types of the questions (e.g., text, multiple choice, etc.). For each question, the user could provide a title and description. Even such a simple front-end would satisfy the needs of many users.

Once the user has designed the questionnaire, the front-end could create a new table in the database, and create the necessary files on the web server, which would include the data files used by the database connection software.

The simplified questionnaire front-end would provide questionnaires that are sufficient for some users. Others may wish to use them as starting points, and further customize them using more sophisticated front-ends.

5.3 Enhanced Interactivity

Netscape Navigator, the most widely used web browser, contains support for the Java programming language. Soon, Microsoft's Internet Explorer will support Java, also. By using some Java applets, we could provide a more appealing, elegant interface. We would also be able to achieve a much higher degree of interactivity.

The Internet Explorer will also soon support ActiveX, which is a set of tools designed by Microsoft that will allow interactive content. ActiveX will also eventually be supported by other web browsers.

These technologies could allow for fancier, more interesting assessments. Not only might that entice students to submit assessments with greater frequency, but they might enable new types of questionnaires.

Appendix A. The Relational Database Model

E.F. Codd developed the relational database model in the early 1970's. In this model, a database consists of a set of tables. The rows and columns of each table are unordered, and the rows are distinct. Each table contains a primary key, which is column or set of columns containing unique values that distinguish between the rows.

The tables can be manipulated by nonprocedural operations, which themselves return tables. Common operators such as INSERT, UPDATE, and SELECT are part of a data manipulation language known as Structured Query Language, or SQL.

The relational database model allows a great deal of flexibility. However, the same versatility that provides such great functionality also allows several types of internal inconsistencies. Further constraints can be imposed upon a relational database so that it achieves a more consistent structure, eliminating the possibility of certain types of inconsistencies. Normalization is the process of modifying the design of a database to satisfy these constraints. There are several levels of normal forms, each building upon the previous ones to impose stricter constraints. Codd originally defined three normal forms[5], and several others have since been defined to eliminate even more possible inconsistencies.

---

[5] E.F. Codd, "Further Normalization of the Data Base Relational Model." In *Data Base Systems*, Courant Computer Science Symposia Series, Vol. 6, Prentice-Hall, 1972.

1. First Normal Form

A table is in the First Normal Form (1NF) if all of the column values are atomic[6]. For example, Figure 6.1 contains a table that is not in 1NF, since the *Items* column does not contain atomic values. The table in Figure 6.2, on the other hand, satisfies 1NF, since each value is atomic. In addition to adding the *Quantity* column to create atomic values, we added the *Item Number* column. The primary key for the table in Figure 6.2 consists of the *Order Number* and *Item Number* columns, since the *Order Number* column by itself would not be unique.

Order Number	Items
1	2 hammers
2	1 hammer, 2 screwdrivers
3	1 saw

Figure 6.1: A table violating First Normal Form

Order Number	Item Number	Item	Quantity
1	1	hammer	2
2	1	hammer	1
2	2	screwdriver	2
3	1	saw	1

Figure 6.2: A table in First Normal Form

2. Second Normal Form

A table satisfies the Second Normal Form (2NF) if it is in 1NF and every non-key column is fully dependent on the entire primary key[7]. Thus, if we were to add a *Payment Method* column to the table in Figure 6.2, as shown in Figure 6.3, the table would violate 2NF, since the payment method would not depend on the *Item Number* column, which is part of the primary key. A table that satisfies 2NF contains information about a single entity, which is fully described by the primary key.

---

[6] Paul Litwin and Ken Getz, *Microsoft Access 95 Developer's Guide, Second Edition*, Sybex, 1996. Page 54.
[7] C.J. Date, *An Introduction To Database Systems*, Addison-Wesley Publishing Company, 1975. Page 99.

Order Number	Item Number	Item	Quantity	Payment Method
1	1	hammer	2	cash
2	1	hammer	1	visa
2	2	screwdriver	2	visa
3	1	saw	1	check

Figure 6.3: A table violating Second Normal Form

3. Third Normal Form

A table is in the Third Normal Form (3NF) if it is in 2NF and all non-key columns are mutually independent.[8] For example, if we augment the table in Figure 6.2 with an *Item ID* column as shown in Figure 6.4, the resultant table would violate 3NF. The *Item* column and the *Item ID* column are not mutually independent. To normalize the table, we would create a second table, consisting of a mapping from *Item ID* to *Item*, as shown in Figure 6.5.

Order Number	Item Number	Item ID	Item	Quantity
1	1	47	hammer	2
2	1	47	hammer	1
2	2	52	screwdriver	2
3	1	33	saw	1

Figure 6.4: A table violating Third Normal Form

Order Number	Item Number	Item ID	Quantity
1	1	47	2
2	1	47	1
2	2	52	2
3	1	33	1

Item ID	Item
47	hammer
33	saw
52	screwdriver

Figure 6.5: Two tables satisfying Third Normal Form

[8] C.J. Date, Page 98.

4. Higher Normal Forms

There are normal forms beyond 3NF (including the Boyce-Codd, Fourth, and Fifth Normal Forms), but they add little value to 3NF for our purposes. The higher normal forms are all supersets of 3NF, and most tables that satisfy 3NF also satisfy the Boyce-Codd Normal Form and perhaps even Fourth or Fifth Normal Forms.[9]

5. Costs And Benefits of Normal Forms

Databases that are in one of the normal forms have added restrictions on them, above and beyond the stipulations made by the relational database model. The benefit of adhering to these stricter guidelines is that the database is easier to manipulate. Relational databases can have several types of inconsistencies. An *insert anomaly* exists if the act of inserting data causes an inconsistency.[10] For example, if a table contains a calculated column and we insert data into another column, the calculated column may be in an inconsistent state.

A *delete anomaly* exists if the deletion of one row deletes more information than was intended.[11] For example, if we were to delete the last row from the table in Figure 6.4, we would not only delete the third order, as intended, but also the relationship between *Item ID* 33 and “saw”.

---

[9] Jeffrey D. Ullman, *Principles of Database Systems, Second Edition*, Computer Science Press, 1982. Page 234.
[10] C.J. Date. Page 106.
[11] C.J. Date. Page 106.

An *update anomaly* exists if a change in one value in the real world causes many changes in the database.[12] For example, in the table in Figure 6.4, if we decide to change the *Item ID* of a hammer from 47 to 48, we would have to change two occurrences. If the tables were larger, there might be many more occurrences, and if we failed to update all of them the database would be in an inconsistent state.

The normal forms discussed above greatly reduce the number of anomalies present in the database. Higher normal forms eradicate even more anomalies, but for our purposes the Third Normal Form is sufficient.

Sometimes, however, the normal forms place undue restrictions on us. For example, some databases can achieve large performance improvements by storing a calculated index in a table.[13] If the index requires large amounts of calculation, it might be unreasonable to compute during each query. Thus, although a calculated column violates 3NF, the database designer might want to stray slightly from strict adherence to the normal form.

---

[12] C.J. Date. Page 106.
[13] Paul Litwin and Ken Getz. Page 72.

Appendix B. The Common Gateway Interface

When the user completes a form and submits it, the HTTP server communicates with the application or script specified in the action URI. A standard communications protocol is the Common Gateway Interface (CGI). When using CGI, the server passes information about the request through environment variables.[14] These variables can contain information such as the remote host name and the username of the user, if authenticated.

The web server may pass the form data through environment variables or through the standard input file descriptor, depending on the form's request method. In either case, the application can return a document via standard output.

[14] David Robinson, *The WWW Common Gateway Interface Version 1.1*, 1995

Appendix C. The Internet Database Connector

The Internet Database Connector uses two types of files for describing interactions with databases. An IDC file contains the information necessary to perform the query, such as the name of the data source and the SQL query statement. An HTML extension (HTX) file contains a template for the returned data. The IDC file permits the use of CGI variables in the SQL query string, allowing the query to be dependent on data submitted from an HTML form.

An HTX file is an HTML file with a few control codes to allow the system to properly format the data returned from the query. It allows for a single merged data section, which the database connector processes once for each row of data. The database connector also allows special tags that reference the CGI variables, some environment variables, two internal variables, and any data returned from the query. The database connector supports simple conditional statements, which allows the behavior to be dependent on the variables.

1. Shortcomings of The Internet Database Connector

The SQL query statement in an IDC file can reference CGI form variables, but not CGI environment variables, although they are available in the HTX file. Since user authentication makes use of a CGI environment variable, this shortcoming poses serious problems for us. We require the authentication information in order to be able to return data that is dependent on the username.

We can work around the lack of access to environment variables in IDC files by using them in an HTX file and placing them as a hidden element on a form. Thus, when the user submits this new form, the web server passes the authentication information to the IDC file as a CGI variable. This solution, however, poses two major problems. First, it requires that the user submit an extra form, making the assessment process less intuitive and elegant. Second, and more significant, this change makes forgery trivial. Users could easily write a web page, hosted on any server, that would submit a form identical to the designed one, except using a username other than their own. They could potentially impersonate other users. We would greatly prefer to retrieve the authentication information from the environment variables automatically.

A second shortcoming of the built-in Internet Database Connector is that it does not allow us any means of pre-processing the data that it retrieves from the database before it processes the HTX file. As a consequence, we do not have the ability to convert special HTML control characters that users may include in submitted assessments. For example, if a user submitted a response to a questionnaire that contained malformed HTML tags, the response could conceivably cause problems when the database connector attempts to process the HTX file.

An IDC file allows for the description of a single SQL statement. This statement can contain CGI form variables and can reference tables and queries stored in the database, allowing for arbitrarily complex queries. However, even the most complex SQL query has severe limitations, due to the nature of SQL. For example, when updating the database, we occasionally desire to perform several updates to different tables, some perhaps conditional upon the successful completion of others. The Internet Database Connector allows only one SQL statement per file, and provides no method of automatically chaining the files without some user intervention.

A possible solution is to use several IDC files, and require the user to interactively perform several steps. However, we cannot guarantee that the user will perform all of the steps. It may be, therefore, impossible to maintain the integrity of the database. There is also no method, from an HTX file, of determining whether an action query is successful, rendering some conditional updates problematic.

A final shortcoming of the Internet Database Connector is not a design flaw, but rather a bug. The HTX files can contain `if` statements, allowing conditional behavior. However, the `if` statements cannot be safely nested, limiting their functionality.

2. Differences Between ODBCLink and The IDC

ODBCLink, the database connection software that we developed, uses a QRY file instead of an IDC file. The format of the QRY files is described in Appendix E. One significant difference between QRY files and IDC files is that QRY files allow CGI environment variables to be referenced in the SQL statement. As we discussed earlier, the lack of such functionality in IDC files makes authentication problematic.

Both IDC files and QRY files have a template field that specifies the HTX file for the connector to use when processing the data returned from the query. As an alternative, QRY files allow the specification of two QRY files. The connector executes one of the two QRY files based on the result of the SQL query in the original QRY file. This chaining of QRY files allows arbitrarily complex conditional behavior, without any intervention by the user. The connector can execute multiple update queries in sequence. The final QRY file in any chain can provide an HTX file as a template for the final results.

The standard HTX file format used by the Internet Database Connector defines two internal variables,`CurrentRecord` and `MaxRecords`, which can be used in conditional statements. The `CurrentRecord` variable allows behavior that is conditional on which row of data is being processed. In particular, it allows the HTX file to recognize when the SQL query did not return any data. Update queries, however, return no data regardless of their success. Consequently, HTX files have no means of altering the behavior based on the success of an action query. The HTX file format that ODBCLink uses adds a third internal variable,`RowCount`, which, in the case of action queries, contains the number of rows affected by the query.

As described earlier, in section1, the user might submit responses that contain malformed HTML tags. The standard HTX file cannot pre-process the data returned from the query to eliminate the tags. The new HTX file format used by ODBCLink can convert any of the returned SQL variables, translating HTML control characters to harmless sequences which display the characters.

Finally, ODBCLink processes HTX files in a slightly more robust manner than the Internet Database Connector. Conditional statements may be arbitrarily nested, allowing for more complex behavior. Also, the environment, CGI, SQL, and built-in variables may be used throughout the HTX file, whereas the Internet Database Connector restricts some of them to conditional statements.

Appendix D. Software Details

The two major pieces of software that we required for this system were a database connector and a sign-up application. The database connector enables the web server to communicate with a back-end database, and the sign-up application allows users to obtain accounts on the system.

1. The Database Connection Software

We developed a Common Gateway Interface (CGI) application that allows the web server to communicate with the database. The application, ODBCLink, enables the web server to pass Structured Query Language (SQL) queries to an Open Database Connectivity (ODBC) compliant database by means of two files, QRY and HTX. A QRY file contains the information necessary to generate a SQL query and connect to the database. An HTX file is an HTML extension file that describes a template for displaying the information retrieved by the query. These files are described in more detail in Appendix E.

Whenever the URL passed to the web server references a QRY file, the web server executes ODBCLink, passing any appropriate CGI variable and environment data. Figure 6.6 shows a brief overview of the operation of ODBCLink.

ODBCLink first attempts to open and parse the QRY file. If the file does not exist or is malformed, ODBCLink outputs an error message. If the current user does not have permission to read the QRY file, ODBCLink informs the server, which may give the web browser an opportunity to authenticate itself.

When it parses the QRY file, ODBCLink constructs the SQL statement, filling in any necessary variables. If the QRY file specifies required parameters that are not present, ODBCLink relays an appropriate message.

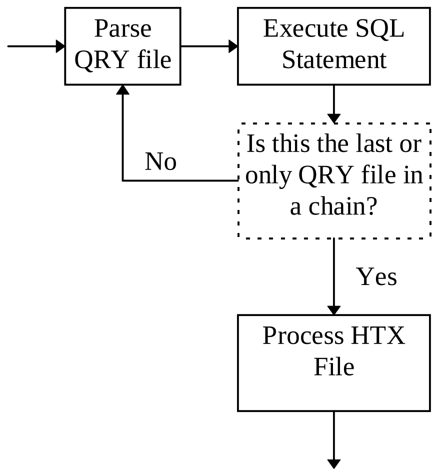


Figure 6.6: Overview of the operation of ODBCLink

If ODBCLink encounters no errors while processing the QRY file, it allocates the necessary handles for an ODBC connection to the data source specified by the QRY file. It connects to the data source using the username and password that were specified in the QRY file, and it attempts to execute the constructed SQL statement. In the event of an error, ODBCLink returns an error message.

After successful execution of the SQL statement, ODBCLink performs one of two actions. If the QRY specified additional QRY files using the `IfQryThen` and `IfQryElse` fields, ODBCLink opens the proper QRY file, based on the data returned by the SQL statement, and processes it. If the original QRY file specified an HTML extension file in the `Template` field, ODBCLink opens and processes the HTX file.

ODBCLink opens the HTX file and scans for the special delimiters "<%" and "%>". It outputs, to the web server, all data in the HTX file that is not within delimiters. For each tag within the delimiters, ODBCLink performs the appropriate action. For variables, whether environment, CGI, internal, or SQL, ODBCLink outputs the value of the variable in place of the tag.

The keywords `begindetail` and `enddetail` mark the beginning and end of the merged data section. ODBCLink retrieves the first row of data returned from the SQL statement before processing the HTX file. It retrieves an additional row, and increments CurrentRecord, after each iteration of the merged data section. If the SQL statement did not return any rows of data, ODBCLink does not process the merged data section at all.

Any `if` statements are evaluated. ODBCLink only processes the proper branch of the conditional statement. Nested `if` statements are handled appropriately. ODBCLink silently ignores any unrecognized tags found between the delimiters.

2. The Sign-up Application

We wanted users to be able to sign up to use the MIT Educational Assessment System via the web. We developed a Common Gateway Interface (CGI) application that enables people to obtain accounts on the system. However, when the web server executes CGI applications, it executes them in the context of the web user, which does not have the authority to create a new user. Therefore, we split the sign-up application into two pieces. The first piece is a standard CGI application that is executed by the web server. The second piece is a Windows NT service that runs continuously in the system context, waiting for a request from the CGI portion of the application to create a new account.

The service portion of the application opens a named pipe and listens for requests. When it receives a username and password as a request, it attempts to create a new user with the privileges necessary to access the database.

When a user signs up, he or she specifies the following:

- An email address (in the .MIT.EDU domain)
- A desired username for the MIT Educational Assessment System
- A desired password for the MIT Educational Assessment System

The CGI portion of the sign-up application, when executed, contacts the database and determines whether the user (as based on the email address) is allowed to sign up for the MIT Educational Assessment System. If the user has already signed up, he or she is not allowed to sign up a second time. In addition, if the requested MIT Educational Assessment System username is in use, the user is prompted for a new one.

If the user has permission to sign up and the username is available, the CGI application requests that the user account be created by sending a request to the service portion of the sign up application. If the account creation is successful, the application updates the database and mails a confirmation to the user.

The confirmation that the application sends to the user is a random string of letters and numbers. The user must enter the confirmation string the first time that he or she signs on to the system. By emailing a confirmation to the user, the system provides a safeguard against impersonation. One user could not sign up while pretending to be someone else, since the confirmation would be sent to the wrong email address.

Appendix E. ODBCLink File Formats

1.   Query (QRY) File Format

We designed the QRY file format to be as compatible as possible with Microsoft's IDC file format, which the Internet Database Connector uses. A QRY file has the following fields:

- **Datasource**

  The Datasource field contains the name of the ODBC data source. The data source should be a system data source, as opposed to a user data source, since it must be visible to the web server.

- **Username**

  The Username field specifies the username to use when connecting to the data source.

- **Password**

  The Password field specifies the password to use when connecting to the data source.

- **Template**

  The Template field contains the filename of the HTX file that ODBCLink will use as a template for the results. This field is required unless the IfQryThen and IfQryElse fields are present.

- **RequiredParameters**

  If present, this field contains a comma delimited list of parameters that must be defined. If any are not defined, then ODBCLink returns an error to the browser.

- **DefaultParameters**

  If present, the DefaultParameters field contains a comma delimited list of pairs of parameters and values. Each pair contains a parameter and a default value, separated by an equal sign. The default value will be used if the parameter is not supplied.

- **IfQryThen**

  This field is used in conjunction with IfQryElse as an alternative to the Template field. If the SQL statement returns a non-empty result set, or modifies a non-zero number of rows, ODBCLink will execute the QRY file specified in this field.

- **IfQryElse**

  This field is used in conjunction with IfQryThen. If the SQL statement returns an empty result set, or does not update any rows, ODBCLink will execute the QRY file specified in this field.

  The IfQryThen and IfQryElse fields do not exist in the Internet Information Server, and no way exists to achieve the same functionality. Future versions of IIS may have an equivalent mechanism.

- **SQLStatement**

  This field contains the SQL statement to execute. The SQL statement may span more than one row. On the second, and subsequent, rows, the first character must be a plus sign. Parameters (CGI variables) may be referenced by placing their names between percentage signs. For example, "%param1%" references the CGI variable named "param1". CGI environment variables can be referenced by prefacing the variable name with "http." and treating it like a CGI variable. For example, "%http.REMOTE_USER%" references the CGI environment variable named "REMOTE_USER".

Figure 6.7 shows a sample QRY file. The QRY file describes a query that is used when a user wishes to submit a comment on the free-style comment form. If the user is responding to a comment, the query retrieves the comment so that it can display it for the user. If the user is submitting a new comment, the query makes sure that the user is allowed to do so. The QRY file specifies “CommentForm.htx” as the template file, which is shown in Figure 6.9.

```
Datasource: Assessment
Username: admin
Password:
Template: CommentForm.htx
RequiredParameters: ResponseTo, SubjectNumber
SQLStatement:
+SELECT *, IIF(%ResponseTo%=NULL, 0, CommentNumber) As IDCResponseTo
+FROM qryGetCommentsAndVotes
+WHERE (SubjectNumber=’%SubjectNumber%’ OR %ResponseTo% = NULL) AND
+(CommentNumber = %ResponseTo% OR %ResponseTo% = NULL) AND EXISTS
+(SELECT * FROM qryAccessComments WHERE
+SubjectNumber = ’%SubjectNumber%’
+AND Username = ’%http.REMOTE_USER%’);
```

Figure 6.7: Sample QRY file

2. HTML Extension (HTX) File Format

The HTX file is referenced in a QRY file. It is an HTML extension file, with placeholders for the data returned from a query. The format and behavior of our HTML extension files are almost identical to those of the Internet Information Server. The keywords for the HTML extension file are delimited with “<%” and “%>”. ODBCLink will process any tag between the delimiters before sending it to the server. The following are the keywords:

- **`<%begindetail%>`**

  The `begindetail` keyword marks the beginning of the merged data section of an HTX file. ODBCLink processes the merged data section once for each row in the query result set.

- **`<%enddetail%>`**

  The `enddetail` keyword, used in conjunction with `begindetail`, marks the end of the merged data section of an HTX file.

- **`<%if`** ***arg1 op arg2*** **`%>. . .[<%else%>]. . .<%endif%>`**

  The `if` statement allows for conditional behavior in an HTML extension file. The `else` tag is optional. The values in the `if` statement, *arg1* and *arg2*, can be constants or any of the available variables detailed below. Variables referenced within an `if` statement are not enclosed in delimiters. The possible operators are shown in Figure 6.8.

`Eq`	True if *arg1* is equal to *arg2*
`lt`	True if *arg1* is less than *arg2*.
`Gt`	True if *arg1* is greater than *arg2*.
`Contains`	True if *arg1* contains *arg2*. This operator is only applicable for string arguments.
`In`	True if *arg1* is contained in *arg2*. This is only applicable for string arguments.

Figure 6.8: The operators available for if statements `if` HTX files

The Internet Database Connector does not support the `in` operator. In the case of string arguments, all of the listed operators are case-insensitive unless the operator is specified entirely in uppercase. For example, `EQ` tests for case-sensitive equality. This behavior is different from that of the IDC, in which all operators are case-insensitive.

- **<%*SQL-variable*%>**

  When ODBCLink finds a SQL variable name (a column name from the result set) between the delimiters, it outputs the value of the specified column for the current row.

- **<%html.*SQL-variable*%>**

  If the name of the SQL variable is prefaced with "html." ODBCLink converts the data before passing it to the server. Any characters that are special in HTML, such as "<" and ">" are converted to sequences that cause the respective characters to be displayed. Also, blank lines are transformed to the "<p>" sequence to create a paragraph break in HTML.

- **<%idc.*Query-variable*%>**

  Any of the variables that were passed to the QRY file by the server can be referenced by prefacing them with "idc." and enclosing them within the delimiters. The prefix "idc." is used for compatibility with the Internet Database Connector.

- **<%http.*Environment-variable*%>**

  Any environment variable can be referenced by prefacing it with "http." and enclosing it within the delimiters. This is incompatible with the behavior of the Internet Database Connector. The IDC does allow access to some of the environment variables, but it does not preface them with "http." We chose to preface them in order to be consistent with the behavior of the QRY file.

- <%**CurrentRecord**%>

  The internal variable CurrentRecord contains the number of the record currently being processed. It is zero initially, and is incremented once each time the merged data section of the HTML extension file is parsed. Thus, after the merged section, `CurrentRecord` contains the total number of rows returned by the query.

- **<%MaxRecords%>**

  The internal variable `MaxRecords` is reserved for future use. The Internet Database Connector uses it to indicate the value of the `MaxRecords` field of the IDC file. We do not currently use this field, but, for compatibility, we reserve the `MaxRecords` keyword.

- **<%RowCount%>**

  The internal variable `RowCount` contains the number of rows affected by an update query. If the query was not an update query, `RowCount` is equal to -1. The Internet Database Connector does not have `RowCount` or an equivalent.

In the Internet Database Connector, the internal variables may only be present in `if` statements. Our HTML extension files do not have the same restriction. The internal variables can be used anywhere the other variables could be used.

The internal variables take precedence over the SQL variables. Thus, if a SQL result column is named either `CurrentRecord`, `MaxRecords`, or `RowCount`, it will be inaccessible in the HTML extension file (unless it was prefaced with "`html.`", in which case it would be converted.).

Figure 6.9 shows a sample HTX file, which could accompany the QRY file that was shown in Figure 6.7. ODBCLink will process the merged data section, as denoted by the `begindetail` and `enddetail` tags, once for each matching row.

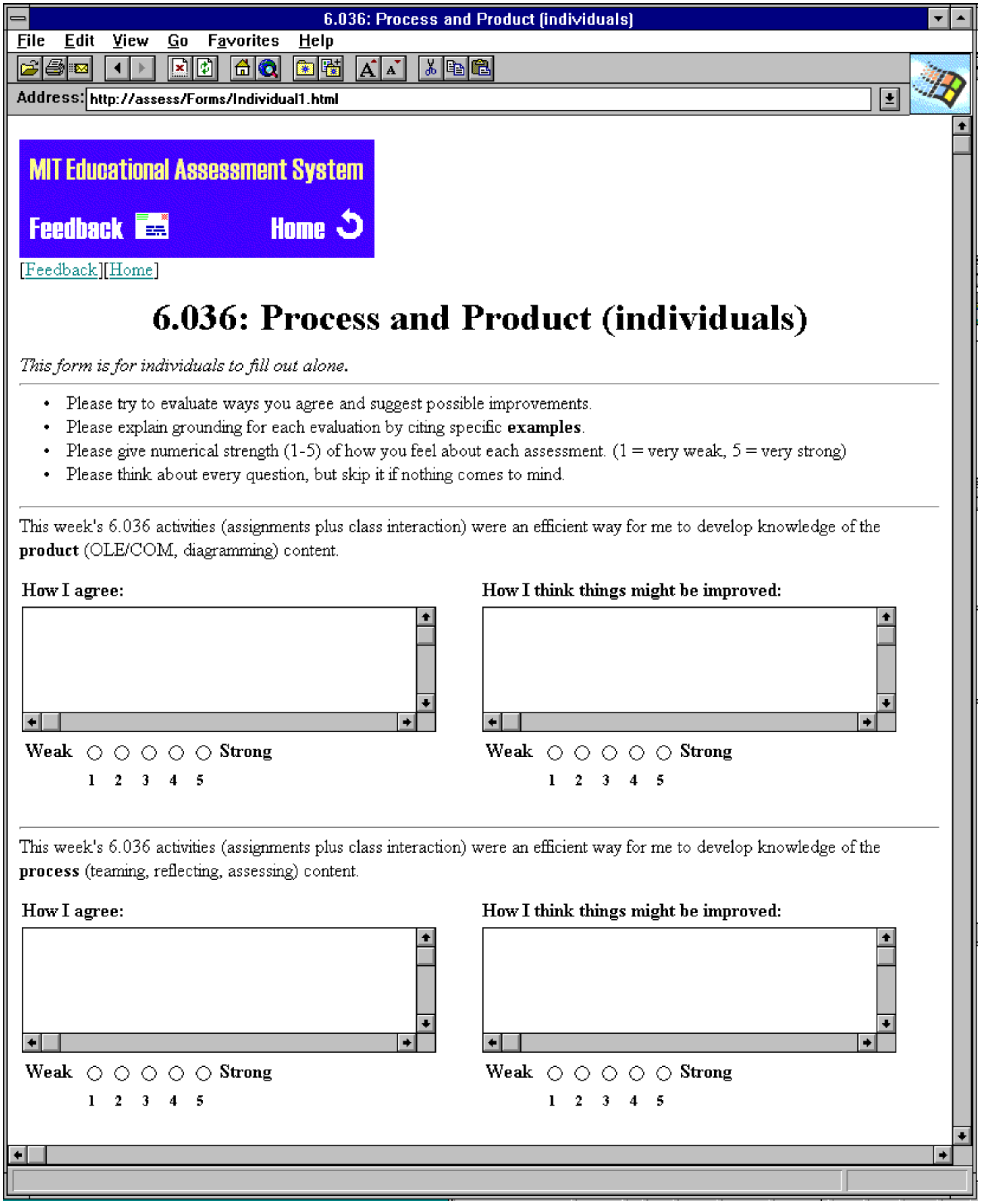
The HTX file shown in Figure 6.9 displays an HTML form for the user to fill in. If the user is responding to a comment, the HTML page will display the old comment. Otherwise it will simply display a blank form. Figure 6.10 show what the displayed form might look like when the user is responding to a comment.

```
<html>
<body bgcolor=ffffff>
<title>Free-style Submission Form</title>

<!--#INCLUDE file="/header.inc"-->

<center><h1>Free-style Submission</h1></center>
<hr>

<%begindetail%>
<%if CurrentRecord eq 0%>
<h3>Subject number: <%idc.SubjectNumber%></h3><p>
<%if IDCResponseTo eq 0 %>
<%else%>
<dl>
<dt><b>Comment #<%IDCResponseTo%><%if Poster eq NULL%><%else%>
(<%Poster%>) <%endif%>, <%CommentDate%></b>
<dd>
<%html.Comment%>
</dl>
<%endif%>
<hr>

<form METHOD="POST" action="/Scripts/Assess/SubmitComment.qry">
<%if IDCResponseTo eq 0 %>
<input type="hidden" name="ResponseTo" value="NULL">
<input type="hidden" name="Agreeance" value="NULL">
<h2>Questions, comments, suggestions:</h2><p>
<%else%>
<input type="hidden" name="ResponseTo" value="<%CommentNumber%>">
<b>Level of agreeance to comment #<%CommentNumber%></b>
<select name="Agreeance" size=1><option value="NULL" selected><option
value="0">Disagree Strongly<option value="1">Disagree<option
value="2">Indifferent<option value="3">Agree<option value="4">Agree
Strongly</select>
<p>
<h2>Reply:</h2><p>
<%endif%>
<i>Use a blank line to separate paragraphs.</i>
<input type="hidden" name="SubjectNumber" value="<%idc.SubjectNumber%>">
<textarea name="Comment" rows=10 cols=80></textarea><br>
<input type="submit" value="Submit comment">
<input type="reset"  value="Clear all fields">
</form>
<hr>
<%endif%>
<%enddetail%>

<%if CurrentRecord eq 0%>
There was an error.  Either you do not have permission to submit
comments for the class you
specified (<%idc.SubjectNumber%>) or the system was unable to locate the
comment you were trying
to respond to.
<hr>
<%endif%>
</body>
</html>
```

Figure 6.9: Sample HTX file

Figure 6.10: Sample free-style comment submission form.

Appendix F. Sample Assessment Forms



6.036: Process and Product (individuals)

File Edit View Go Favorites Help

Address: http://assess/Forms/Individual1.html

MIT Educational Assessment System

Feedback Home

[Feedback][Home]

6.036: Process and Product (individuals)

This form is for individuals to fill out alone.

- Please try to evaluate ways you agree and suggest possible improvements.
- Please explain grounding for each evaluation by citing specific **examples**.
- Please give numerical strength (1-5) of how you feel about each assessment. (1 = very weak, 5 = very strong)
- Please think about every question, but skip it if nothing comes to mind.

This week's 6.036 activities (assignments plus class interaction) were an efficient way for me to develop knowledge of the **product** (OLE/COM, diagramming) content.

How I agree:

Weak ○ ○ ○ ○ ○ Strong
1 2 3 4 5

How I think things might be improved:

Weak ○ ○ ○ ○ ○ Strong
1 2 3 4 5

This week's 6.036 activities (assignments plus class interaction) were an efficient way for me to develop knowledge of the **process** (teaming, reflecting, assessing) content.

How I agree:

Weak ○ ○ ○ ○ ○ Strong
1 2 3 4 5

How I think things might be improved:

Weak ○ ○ ○ ○ ○ Strong
1 2 3 4 5

Figure 6.11: Process and Product assessment form (page 1)

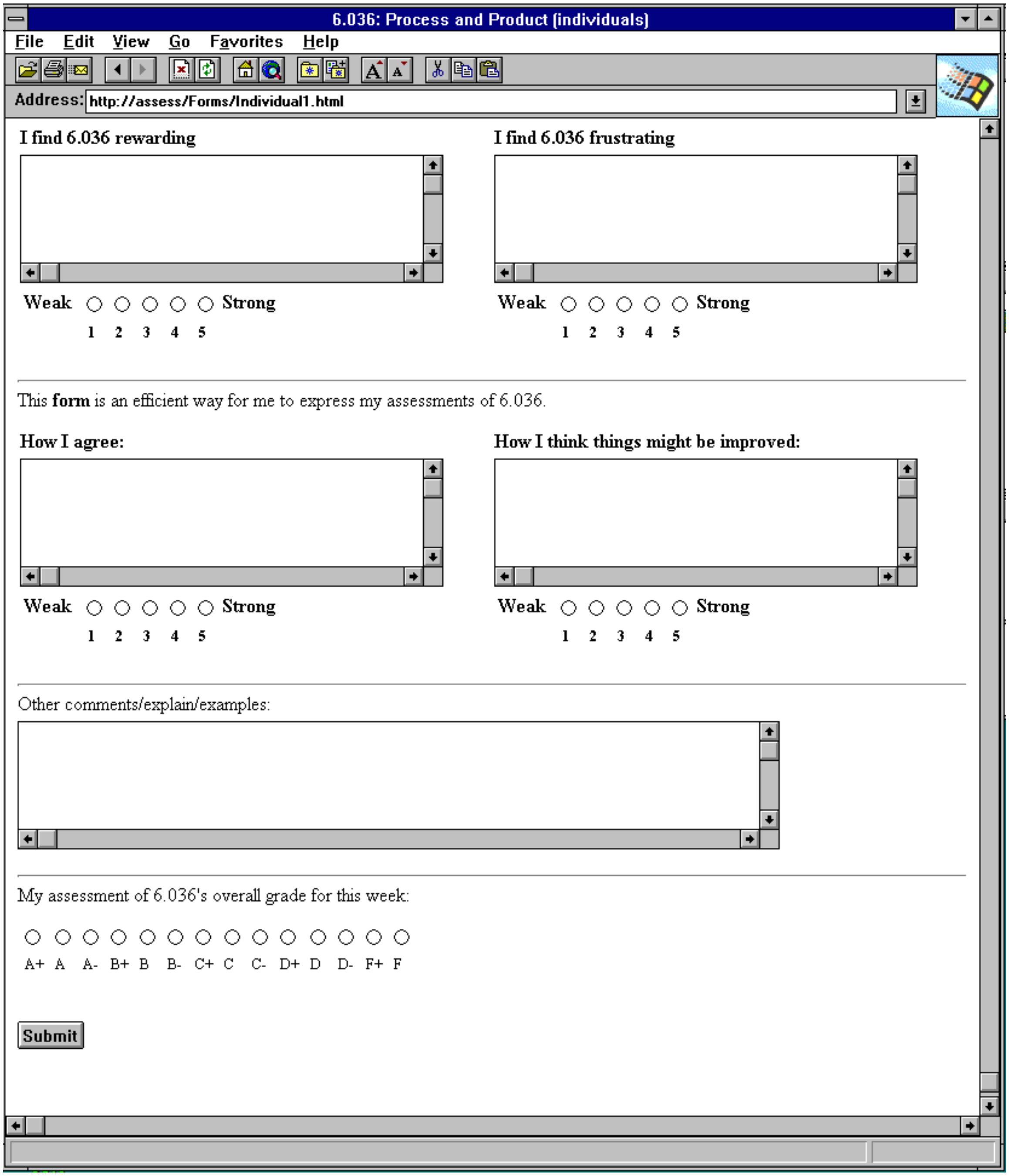
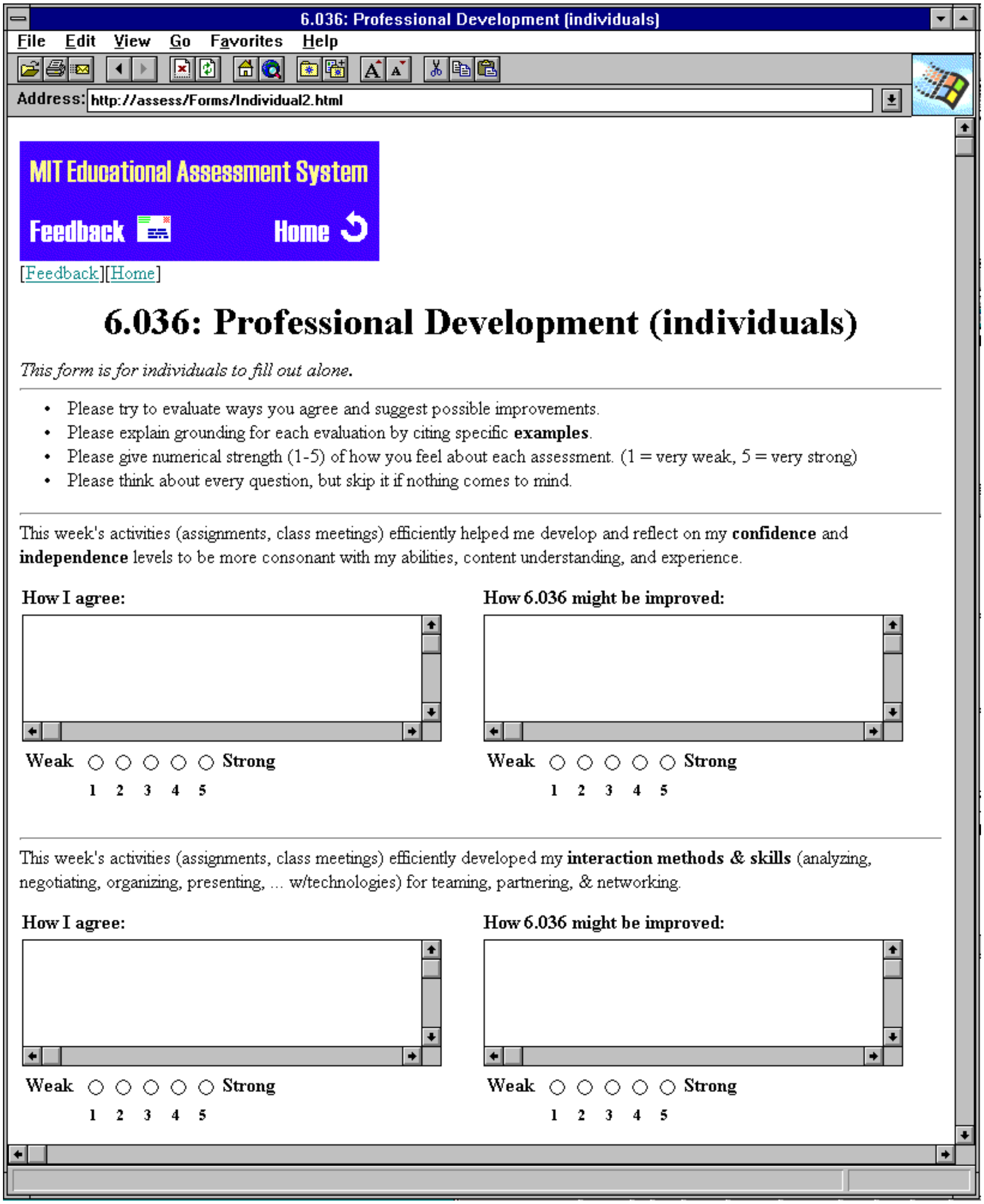


Figure 6.12: Process and Product assessment form (page 2)


6.036: Professional Development (individuals)

File Edit View Go Favorites Help

Address: http://assess/Forms/Individual2.html

MIT Educational Assessment System

Feedback Home

[Feedback][Home]

6.036: Professional Development (individuals)

This form is for individuals to fill out alone.

- Please try to evaluate ways you agree and suggest possible improvements.
- Please explain grounding for each evaluation by citing specific **examples**.
- Please give numerical strength (1-5) of how you feel about each assessment. (1 = very weak, 5 = very strong)
- Please think about every question, but skip it if nothing comes to mind.

This week's activities (assignments, class meetings) efficiently helped me develop and reflect on my **confidence** and **independence** levels to be more consonant with my abilities, content understanding, and experience.

How I agree:

Weak ○ ○ ○ ○ ○ Strong
1 2 3 4 5

How 6.036 might be improved:

Weak ○ ○ ○ ○ ○ Strong
1 2 3 4 5

This week's activities (assignments, class meetings) efficiently developed my **interaction methods & skills** (analyzing, negotiating, organizing, presenting, ... w/technologies) for teaming, partnering, & networking.

How I agree:

Weak ○ ○ ○ ○ ○ Strong
1 2 3 4 5

How 6.036 might be improved:

Weak ○ ○ ○ ○ ○ Strong
1 2 3 4 5

Figure 6.13: Professional Development assessment form (page 1)

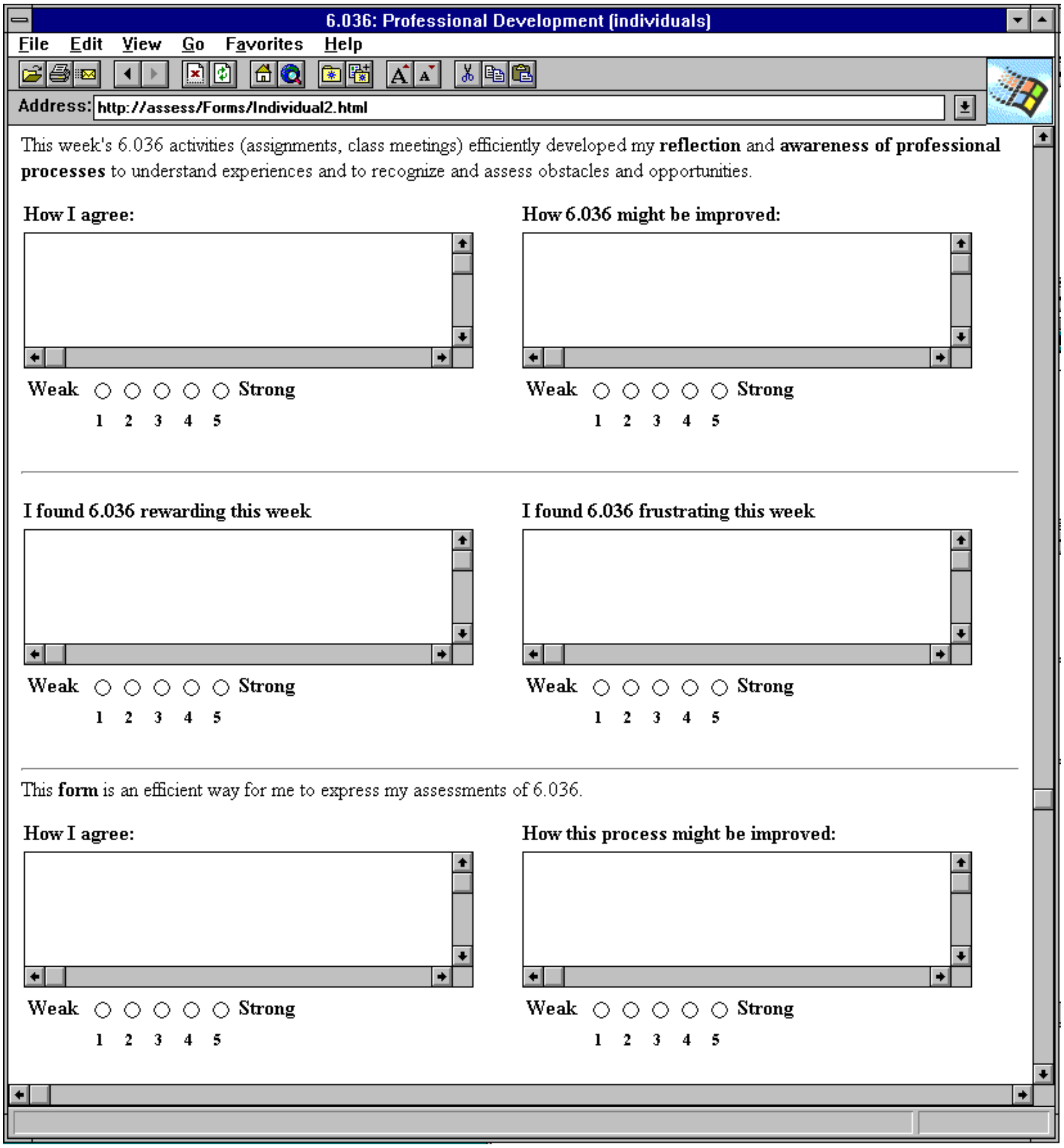
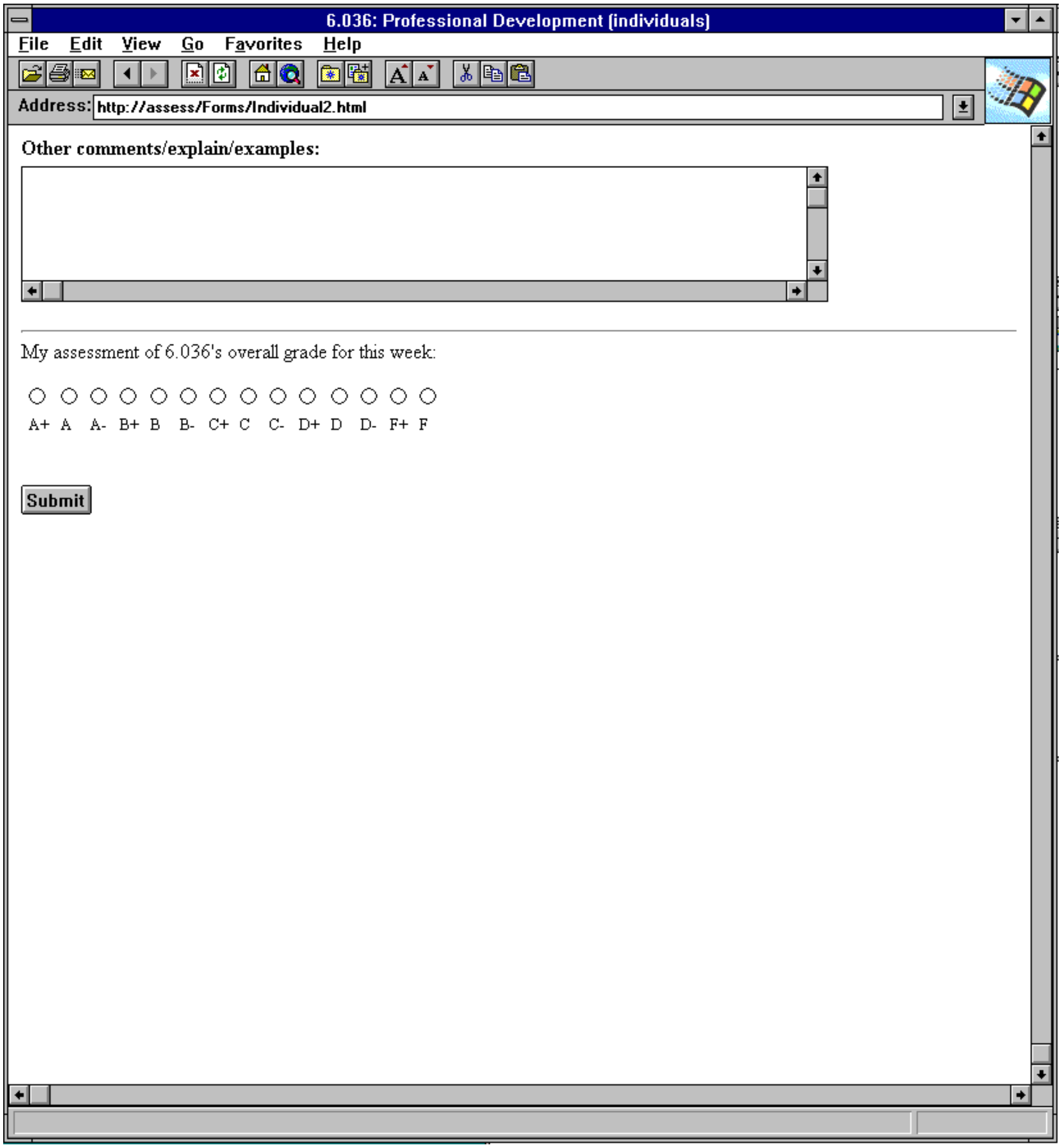


Figure 6.14: Professional Development assessment form (page 2)



6.036: Professional Development (individuals)

File Edit View Go Favorites Help

Address: http://assess/Forms/Individual2.html

Other comments/explain/examples:

My assessment of 6.036's overall grade for this week:

A+ A A- B+ B B- C+ C C- D+ D D- F+ F

Submit

Figure 6.15: Professional Development assessment form (page 3)

Bibliography

Berners-Lee, T. and Connolly, D., *Hypertext Markup Language - 2.0*, RFC 1866, HTML Working Group, 1995.

Codd, E.F., "Further Normalization of the Data Base Relational Model.", *Data Base Systems*, Courant Computer Science Symposia Series, Vol. 6, Prentice-Hall, 1972.

Date, C.J., *An Introduction To Database Systems*, Addison-Wesley Publishing Company, 1975.

Kiesler, Sara and Sproull, Lee, "Response Effects in the Electronic Survey." *Public Opinion Quarterly, volume 50*, 1986.

Litwin, Paul and Ken Getz, *Microsoft Access 95 Developer's Guide, Second Edition*, Sybex, 1996.

Postel, Jonathan, "Simple Mail Transfer Protocol", RFC 821, USC/Information Sciences Institute, 1982.

Robinson, David, *The WWW Common Gateway Interface Version 1.1*, 1995.

Sproull, Lee and Kiesler, Sara, *Connections: New Ways of Working in The Networked Organization*, MIT Press, 1991.

Ullman, Jeffrey D., *Principles of Database Systems, Second Edition*, Computer Science Press, 1982.

Undergraduate Association Course Evaluation Guide, Spring 1995, Massachusetts Institute of Technology, 1994.