Contents

Chapter 1

Introduction

1.1 Motivation

The Science and Technology Entry Program (STEP), a program under the New York State Education Department (NYSED), published a set of priorities that it wanted to see solutions proposed for when determining which institutions it would award funding to. Paraphrased, these priorities are for institutions to provide programs and services to improve:

1. The recruitment and retention of male participants;

2. The recruitment and retention of Hispanic/Latino and American Indian participants;

3. Eighth grade students' NYS Math and Science Assessment examination scores [2].

Additionally, STEP created a set of service and data requirements that funding recipients must provide. The underlying themes of the requirements are that there should be increased collaboration between the awarded institution, the local educators, students, and parents as well as providing students with assistance in gaining skills needed to succeed in learning and pursuing STEM (Science, Technology, Engineering, and Mathematics) fields [2].

Data published in numerous studies [**?**, **?**] show that American 8th grade students perform consistently below those in many other countries around the world. School Districts

in Northern New York can substantiate the STEP priorities and requirements. NYS Mathematics test data from 2010 revealed that 56% of the school districts in St. Lawrence County had below average proficiency levels for 8th grade [**?**]. At the Salmon River School District, where there is a majority American Indian demographic, 54% of students scored below acceptable proficiency levels on the 8th Mathematics assessment [4]. It is clear that STEP identified these troubling facts and is now pushing institutes of higher education to try to solve these issues.

IMPETUS (Integrated Mathematics and Physics for Entry To Undergraduate STEM), for Career Success is the result of collaboration between St.Lawrence-Lewis County BOCES, the STEM Partnership, and Clarkson University. Its primary goal is to improve and increase opportunities for underrepresented minorities and students from economically disadvantaged rural areas to realize their potential for college entry as STEM majors and for eventual career success in technically oriented professions.

Traditionally, this collaboration has been accomplished via in-person meetings between various combinations of the relevant parties. St. Lawrence County is the largest county by area in not only Northern New York, but in the whole state, yet has a population density of only $41/\text{mi.}^2$, whereas the state average is $355/\text{mi.}^2$. Combined, these circumstances do not create an environment suited towards an efficient and effective exchange of information [4].

IMPETUS has proposed the creation of a Web-based Collaborative Environment that will satisfy the STEP criteria by combining different technologies into a unified solution while increasing the convenience and accessibility of collaboration. Students who are in need of assistance, when meeting in-person is not possible, will be able to engage in web-based cooperative learning with Clarkson University students. Educators will be able to survey and receive feedback from targets regardless of their location and parents will have access to their children's data.

In 2008, the National Science Foundation (NSF) created a Cyberlearning Task Force that was charged with, amongst other objectives, determining what the key research topics were surrounding cyberlearning [1]. In a subsequent NSF publication, it was determined that one of these topics should be collecting, analyzing, and managing data about how individuals use a given cyberlearning system [3].

Due to STEP's data logging requirement and the fact that the NSF has deemed cyberlearning data collection to be an area of research interest, the IMPETUS Web-based Collaborative Environment (IWCE) will collect an extensive amount of data about not only the individuals using it, but also how they are using it.

1.2 Research Objectives

expand:

- Propose a web-based collaborative environment as a solution to overcoming the challenges in operating STEM outreach programs in geographically isolated, rural areas.
- Multiple users with varying interests – need a better understanding of how to measure functionality and user-satisfaction
- Flexible enough for subsequent administrators to make significant modifications
- Open Source for possible use elsewhere

1.3 Methodology

Creating the IMPETUS Web-based Collaborative Environment will be done following a modified waterfall model of software engineering: Requirements, Design, Implementation, and Verification; the modification being that feedback will be readily incorporated into the

phases allowing progress to go "up" the waterfall. Being that there is only one developer on the the project, it is the most basic model to follow.

As modules becomes ready for Verification, usability tests will be done to get feedback on their usefulness and design. This will require the creation of surveys and the user studies gauge how effective the interface is at allowing a user to accomplish goals.

Chapter 2

Software Overview

2.1 Goals

The primary motivations behind the creation of the Collaborative Environment are the need for increased collaboration and the need for data logging. The Collaborative Environment will be targeted at a variety of different user types (user classes), it follows that each will be permitted access to a different set of features and data. This means, that while an increase in collaboration is an overall goal for all user classes, specific collaborative goals vary based on how much access each class has:

- All users should have read access to announcements and an event calendar.
- All users should have access to an internal messaging system for one-to-one and one-to-many communication.
- All users should be able to answer surveys that are available to them.
- Students should be able to answer quizzes that are available to them.
- Students should have access to an availability schedule for teaching assistants.
- Students should have access to external educational resources.
- Teaching assistants should be able to create their own availability schedule.

- Teaching assistants should be able to create quizzes and surveys.
- Teachers should be able to create announcements and events for the calendar.

The next set of goals correspond to data logging. There is the possibility of some conceptual overlap between whether a goal is collaborative or data logging related, so the line has been drawn at whether or not the collaborative aspect is simply incidental to having to log data. If that is the case, then the goal will categorized under data logging.

- Parents should have access to their children's data.
- Students should have access to their own data.
- Teaching assistants should have access to the students that they are assisting.
- Teachers should be able to be able to view reports for the students that are available to them.
- Teachers should be able to edit data for the students that are available to them.
- Administrators should be able to create and access the entities by which data is stored.
- Administrators should be able to access user tracking data.

In the above list, there are two terms which require definition: *data* and *entities*. *Data* is defined as all personal information, survey results, and quiz results for the user it is in reference to, whereas *entities* are the fundamental objects that provide the basis for all relationships in the user data.

2.2 Features

High level overview of each. Details of each will follow in §3.2.

expand:

- Academic year switching
- Quiz creator and result viewer
- Survey creator and result viewer
- Messaging system
- Analytics
- District editor
- User editor

2.3 User Classes

In the Collaborative Environment, there exists a hierarchy of roles where each user will be a member of a particular level. These roles naturally correspond to the type of user that they describe and are listed in order of least access to most (also note that any level's access to features is a proper superset of its preceding level's features). Further, there is a distinction between non-privileged and privileged classes: users of a non-privileged class have access to zero or one user's data (typically their own or their child's) whereas users of a privileged class have access to zero or more users data (typically all of their students).

Non-privileged

Anonymous

An anonymous user is any user that has not presented any authentication credentials. They have read access to announcements, calendars, and schedules.

Parent

Has access only to their student's educational data, quiz results, and attendance. They are able to participate in surveys and send messages to any user.

Student

May participate in quizzes and view their own various attempts at quizzes.

Privileged

Teaching Assistant

For any district that they are enrolled as an assistant for, they may view any Student's data. They may also create surveys and quizzes.

Teacher

May read and write student data for only those that belong to the same district as themselves. This data includes survey results, quiz results, and detailed student educational information. A Teacher may also enroll a new Teaching Assistant into their district and create new Students and Parents.

Administrator

Responsible for creating the fundamental Entities which all other users interact with and have access to all user data and reports. They may create new academic years and districts, as well as the specific student activities, courses, and exams that may be tracked. They may also create a user of any role and assign Teachers to their respective districts.

Chapter 3

Software Design and Implementation

3.1 Background

This section will provide details of the existing software and services that are leveraged to create the Collaborative Environment. For each, we provide the reasoning for its use as well as a technical description.

3.1.1 Symfony

The Collaborative Environment is built using Symfony Standard Edition, an open-source, object-oriented PHP framework designed around the Model-View-Controller (MVC) software architecture. Using Symfony to create the Collaborative Environment encourages the use of design patterns that are well understood and allows for more of the development time to focus on application features rather than reimplementing standard components of web applications.

The MVC architecture separates an application's business logic, that is, all of the algorithms which process the exchange of data between an interface and a database, from its user interfaces. The *model* consists of object representations of application data (Entities) and a persistence layer, which will store and retrieve entities via a databases management system. The *view* will render a model object into a user interface, such as a web page. The

controller is what mediates transactions between the user and the application. The typical control flow of a basic MVC application is:

1. Client makes a request
2. Controller receives the request and transforms it into a manipulation of the Model
3. The updated Model is saved to the database by the persistence layer from the Controller
4. A View is generated by the Controller which makes the changes to the Model visible
5. The Controller responds to the Client's request with the generated View.

Symfony Standard Edition comes with many software bundles which extend the core Symfony framework to provide an enterprise-grade application skeleton. Three of the key bundles included are the Security, Doctrine, and Twig bundles, which provide user-rights management, an object-relational mapper, and a template engine respectively.

Security

Security in Symfony is handled by a two-step process: authentication and authorization. The authentication step identifies a user and is typically accomplished by having a user first visit the website, at which point they are authenticated as an anonymous user, and then having them provide a set of credentials to authenticate them as a specific user. Authorization occurs when a user attempts to access a resource of an application. Symfony allows for a set of user roles to be defined, of which each user has a set of, as well as an access control list, to determine exactly what resources any given user has access to.

Controllers are aware of what the current user's authentication is and, as such, can be configured to only allow processing to occur if and only if the current user has a given role. While an Entity will generically describe all instances of a piece of data in an application, if the currently authenticated user should only have access to an exact instance of an entity, an entry can be added to an access control list for that particular user-entity instance pair.

These features provide the Collaborative Environment with a powerful and robust mechanism that is essential to securing the Student data being tracked.

Doctrine

In a dynamic web application, Entities are constantly being created, read, update, and deleted. To keep track of these changes, it is natural to consider using a database management system (DBMS) such as MySQL or PostgreSQL. In an object-oriented program, these Entities often contain non-scalar data (e.g. lists, arrays) that must be persisted by a DBMS, which can be problematic because a DBMS can often only store scalar data (e.g. strings, integers). Using an object-relational mapper (ORM), such as Doctrine, solves this problem by managing the transformation to and from scalars/non-scalars when persisting an object.

As an example of how an ORM operates, let's consider an object-oriented system where a Student is enrolled in a set of Courses. Naturally, we would have two Entity objects, Student and Course. The fields of the Student entity could be an ID, a name, and a list of Courses that they're enrolled in (consider this to be a many-to-many relationship). The fields of the Course Entity could be a name and number. When a new Student has been created and they are enrolled in, perhaps, two different Courses and the ORM is told to persist this data, it will store the scalar data in the DBMS as its native types (names map to varchar, IDs and numbers map to integer), but for the Student's list of enrolled courses, since we established that there is a many-to-many relationship, the ORM will manage a Student ID-Course number tuple (Student 1 is enrolled in Course 1, Student 1 is enrolled in Course 2). Of course, when this data is requested from the DBMS, the ORM will convert the data back into Entity objects.

Twig

When a Controller has to return a new response, it is common for the View to be defined as a template that will be rendered though an engine, such as Twig, rather than explicitly crafted in the desired output format. For example, this could be accomplished by designing a Twig template that will output specific Model Entity variables and have it be processed into HTML by the Twig engine instead of just writing a mixture of HTML and PHP directly.

There are many benefits to using a template engine, such as:

Template inheritance Defining a parent or a layout template that can be inherited from children allowing for a consistent theme across an application.

Syntax Provides numerous shortcuts for applying common patterns to variables such as escaping output and modifying control flow.

Speed After compiling a template, the result will be optimized and low-overhead PHP when compared to freehand coding.

3.1.2 Khan Academy

The Khan Academy is a nonprofit organization that aims to better global education by providing free, high-quality resources to anyone anywhere. [**?**] On their website, they provide in excess of 2700 videos that teach varying topics in STEM fields. The organization has been provided with the financial support and public accolades of large entities such as The Bill & Melinda Gates Foundation and Google [**?**] indicating that they produce noteworthy content. The videos that Khan Academy hosts are all created by Salman Khan who holds an MBA from Harvard and three science degrees from MIT. Each is produced in a fairly consistent manner using a software whiteboard, a screen capture program, and a microphone while typically lasting about 10 minutes.

As stated in §2.1 and §2.2, students using the Collaborative Environment should have access to supplemental learning resources and will have the ability to take quizzes given by their teachers and teaching assistants. The Khan Academy videos are ideal for this use as they are concise and short enough to be associated with individual quizzes that are created in the Collaborative Environment. If a student, while taking a quiz, feels that they are in need of some assistance, a link to the an appropriate Khan Academy video will be readily available to them.

3.2 Features

We now describe the main features of the Collaborative Environment as previously outlined in §2.2. Each subsection henceforth provides the concept, design, and implementation of an individual feature. They have been arranged in such a way that each successive feature will build on the previously introduced ones.

3.2.1 Academic Year

In a traditional document-based teaching environment, student data is stored in numerous grade books or spreadsheets which are held by many different people in many different places. Each teacher, for instance, would have their own set of student records and a district will have a permanent record for each student containing information such as their standardized test scores. If an administrator managed to collect all of this data into a single place, however frustrating that may be, the next thing they would have to do to it, before any type of analysis, would be to group it all by what year the record corresponds to.

As the Collaborative Environment will be collecting data year after year, an Academic Year entity is required. Teachers, Teaching Assistants, and Students all have data associated with them that can change on an annual basis. Students will be taking different classes, have

taken different standardized tests, and participated in different activities as well as be in a different grade altogether or have graduated. Each year can also bring with it a different round of quizzes and surveys and even new Teachers or Teaching Assistants.

Design

The Year entity is really very simple, just a primary key identifier, *id*, and the year itself as an integer, *year*. Explicitly storing the years that the Collaborative Environment will be tracking in a table allows consistency throughout the entire application and customization by an administrator. An alternative implicit design would have been to keep a *date* column in each entity that was going to have time-sensitive content added to it and the server would be responsible for getting the current year from the operating system, but problems arise if the system's time and date are not correct.

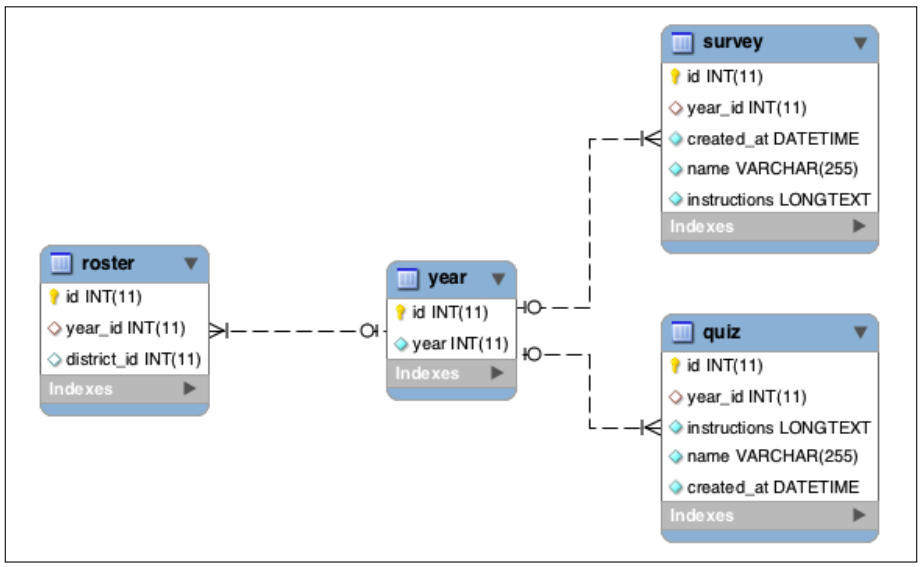

Figure 3.1: Year Enhanced Entity-Relationship (EER) Diagram

We see in Figure 3.1, that the year entity appropriately maintains a one-to-many relationship with each of the tables that reference it. In an EER diagram such as this, each box represents a table in a database and each row in a box corresponds to a column in that

table. A connecting line indicates that a relationships exists between its connected tables. If we take the relationship between the year and survey tables as an example, the line, which is drawn using Crow's Foot notation, indicates that a survey can be given during one year and that a year can have many surveys given during it. This diagram additionally indicates that many quizzes (see §3.2.4) and many rosters (see §3.2.3) can also only be associated with only one year.

The user's choice of which year they want to have as their current context for data viewing and manipulation will have to persist between each page of the Collaborative Environment. Having to constantly select what year to interact . Therefore, once a user initially visits the web site, a PHP session variable will be established that tracks the year they select which will then be used by subsequent SQL queries to retrieve the requested data.

Implementation

As established in the previous section, the selected year must be persisted from page-to-page as well as be accessible in such a way that it will not require a user to constantly have to interact with a selector.

This was implemented by having a year selector list appear in the upper-right corner of each page in the Collaborate Environment, as shown in Figure 3.2. The list is populated by making an asynchronous HTTP GET request to the URL `/year` which will both provide all available academic years that have been configured and indicate which is the current year as selected by the user.

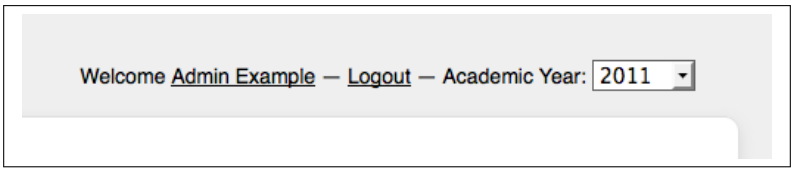

Figure 3.2: Year Selector Highlight

Upon choosing a year from the selection, if it differs from the currently selected year, then an HTTP POST request will be sent to `/year/change/selected-year`. This will cause the year PHP session variable to be updated to *selected-year*. If the server responds with a success message, then the current web page will be refreshed and any changes to the data that are dependent on the year will be reflected.

3.2.2 User Management

A User entity is perhaps the most fundamental object in the Collaborative Environment. Users of each class, as defined in §2.3, with the exception of anonymous, will be required to have an account with this system, and that account is what the User entity represents. The set of users minus anonymous will be referred to as "registered users".

This entity facilitates the fulfillment of the Collaborative Environment's goal of logging user data by allowing the storage of a student's personal data such as gender, ethnicity, if they've graduated, and what college they are attending, in addition to student educational data such as standardized test scores, courses enrolled in, and after-school activities.

The collaborative functionality of the system also requires a User entity. Quiz attempts, survey submissions, and messaging need a way of differentiating one user from another.

Design

The Collaborative Environment internally refers to the set of user classes as "roles". Roles are easily stored in the database by using a role table such that its *name* is stored in a column. The User entity, as modeled in Figure 3.3 stores two types of data: required information that applies to all registered users and personal/educational data that applies strictly to students. Required information consists of a user's *username*, *password*, *salt*, *firstName*, and *lastName*.

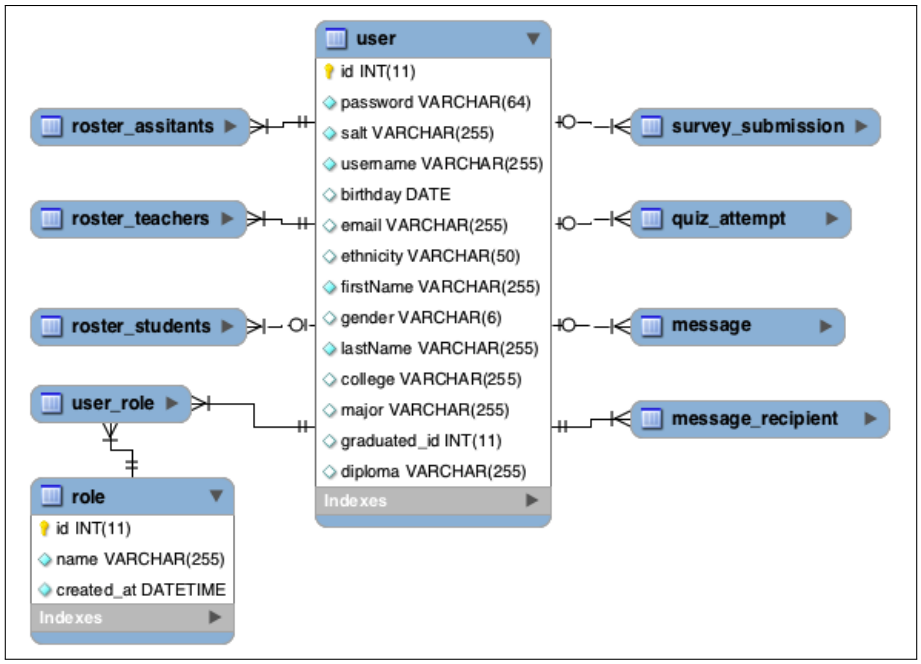

Figure 3.3: User and Role EER Diagram

Each of these attributes has an obvious use except, perhaps, for *salt* which is a precautionary security measure. All user passwords are hashed using using the SHA-256 cryptographic hash function. This means that if two users had the same password, then they would obviously result in identical hashes. This fact can be used exploited by people that have large masses of precomputed hashes. A salt, which is a randomly generated string, is append to the plaintext of a password before it is hashed. Now, even if the user's password is something common, what is hashed is, most likely, something unique.

For the sake of simplicity, we decided that the personal/education attributes *birthday*, *email*, *ethnicity*, *gender*, *college*, *major*, *diploma*, and *graduated* as year-independent data. This means that if and when these attributes have a value, that it will not vary based on what academic year the client has selected to be the current year; their values will persist through all years. On the other hand, there is strictly educational data that we do treat as year-dependent: activities, courses, exams, and grade level.

Figure 3.4, outlines the general model to achieve a year-dependent relationship between

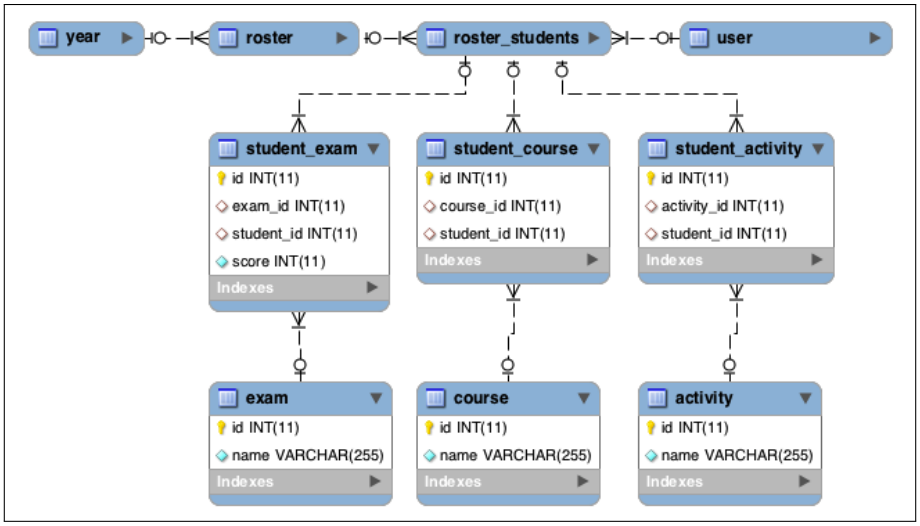

Figure 3.4: Student user EER Diagram

a student and their educational data and will be explained in depth in §3.2.3. Any given year is associated with many rosters, each of which contain a set of students in varying grade levels. Each student in a roster can then be associated with many exams, courses, and activities.

Implementation

User management is broken up into three processes: creation, updating, and reading. As an interface to each of these processes, privileged users have access to a user management interface seen in Figure 3.5. This interface will only display the users that the currently authenticated user has access to. For administrators, this will be all users, whereas a teacher or a teaching assistant will only see the students that belong to the district that themselves belong to. For example, if a teacher is listed as being a teacher in the district "Public School #1" for the year 2011, then the user list will only show students that also enrolled at the district during that year.

This idea of teachers and teaching assistants only having access to the students that are on the same roster as them is persistent throughout the application. The details regarding

districts, rosters, and permissions will be explained in §3.2.3.

With this data now being tracked by the Collaborative Environment, trends and reports will be able to be generated. Tracking student participation in activities, courses, and exams is now possible for the set of all students or even subsets such as minorities, males, or females.

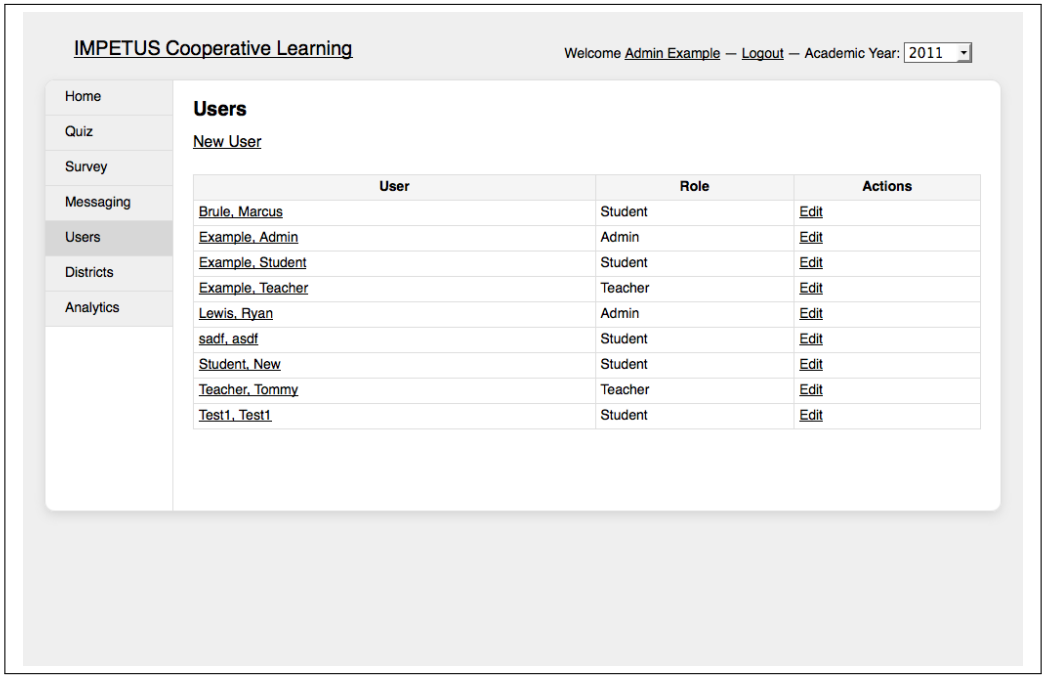

Figure 3.5: A privileged user's view of all the system's users.

When a privileged user must add or edit a user, the interface appears as it does in Figure 3.6, except all of the fields are simply blank in the case of adding a new user. The interface has has been split horizontally into two sections, the top is for personal/educational data whereas the bottom is for strictly educational data. In the top half, input fields has been grouped into sets of required and supplemental (optional). This allows the user to readily identify which fields they must have a value for and not have to wait for the program to return an error.

The supplemental fields for *ethnicity*, *diploma*, and *major*, provide the user with a limited selection of options rather than a freely fillable text input because this will allow for consistent data analysis. If users were able to enter whatever they wanted for these, it would become

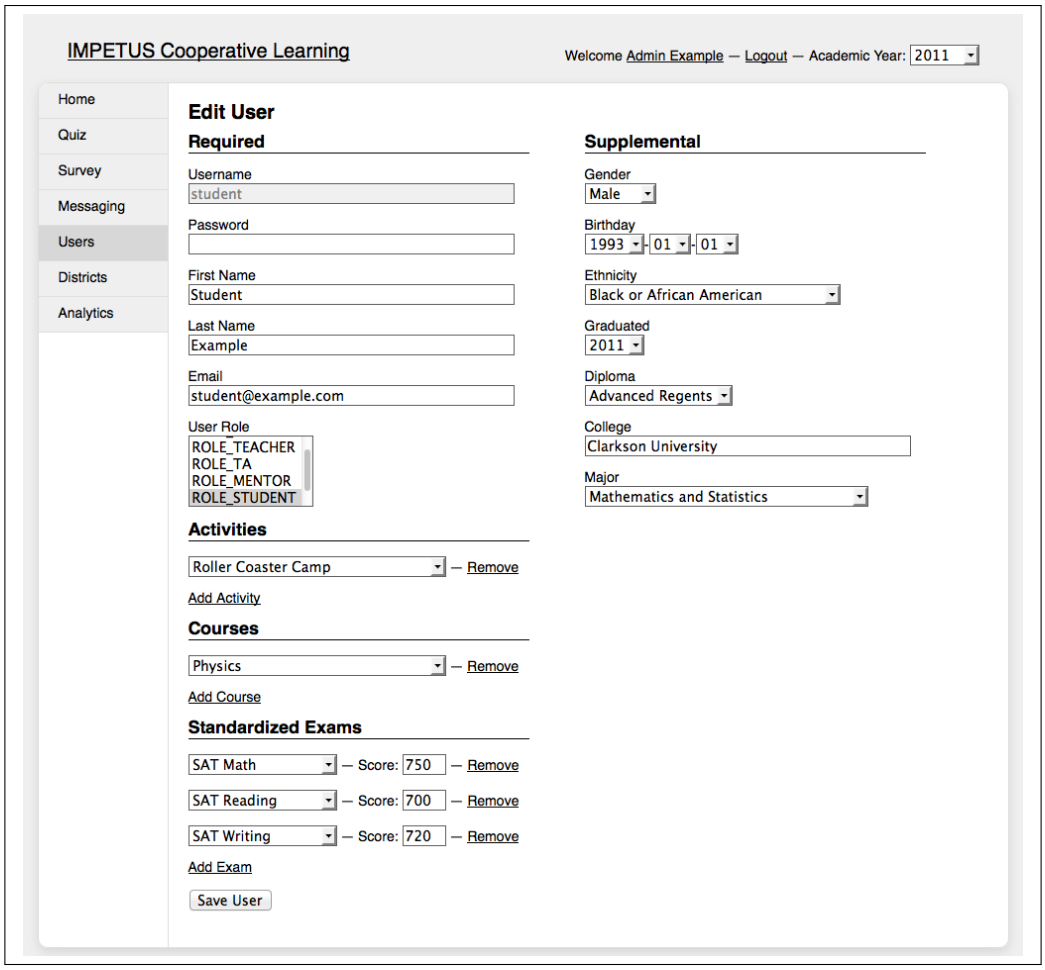

Figure 3.6: A privileged user editing a student's account.

difficult to use that data to generate accurate reports. For example, if one person were to enter that a student was going to major in "Mathematics" and another were to enter "Math".

Since the bottom half of the screen only applies to students, where strictly educational data is entered, it will only appear if the user is assigned the Student role. As stated in the Implementation section, *activities*, *courses*, and *exams* are year-dependent. This means that if a user were to choose different academic years while editing a student's data, they would see these fields change appropriately. A student can be associated with as many year-dependent fields as necessary by clicking the Add button for the appropriate section.

Lastly, viewing user data in a more organized fashion is shown in Figure 3.7. This simply provides an view into the data stored for a user. Just as the add/edit screen would update its data when the current academic year is changed, so will the view screen.

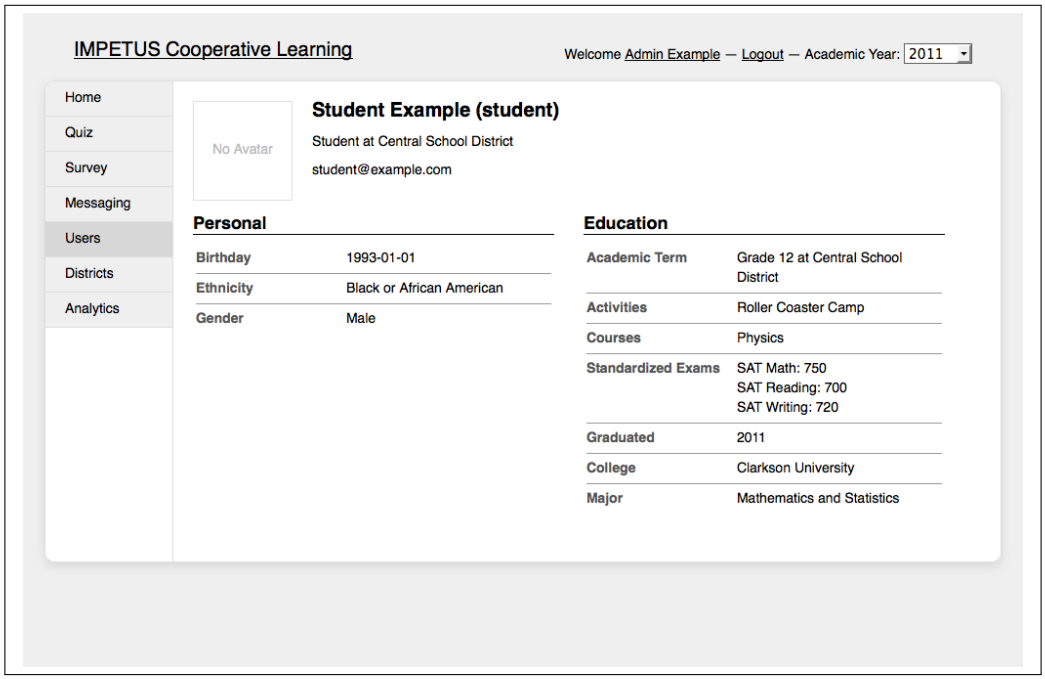

Figure 3.7: A privileged user's view of a student's profile.

3.2.3 District Management

test

3.2.4 Quiz System

test

3.2.5 Survey System

test

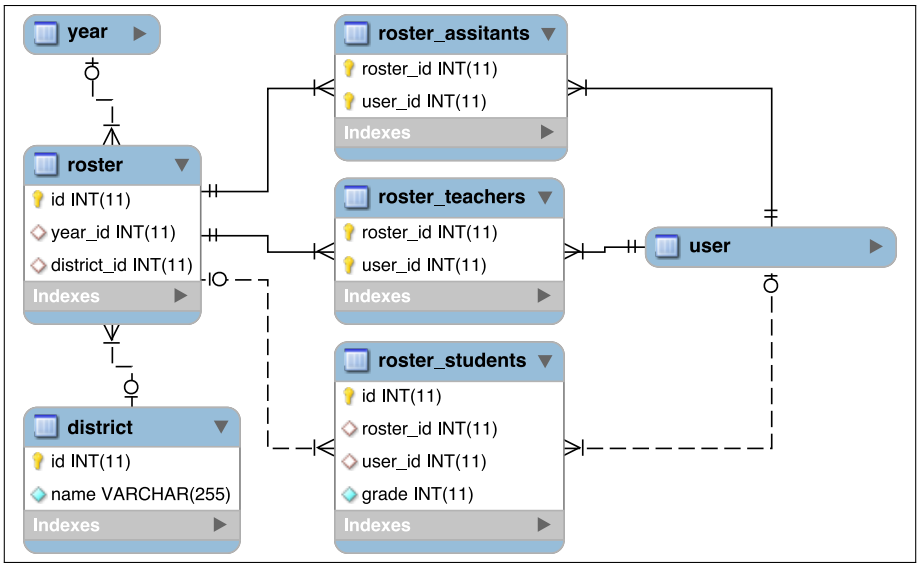

Figure 3.8: District and Roster EER

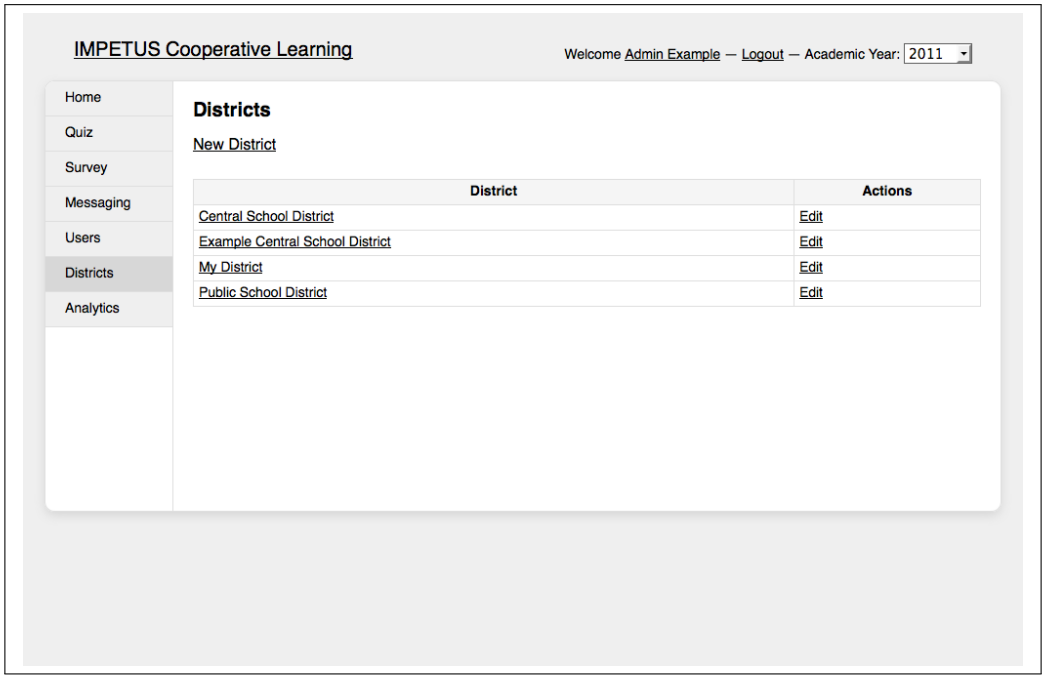

Figure 3.9: A privileged user's view of all the system's users.

3.2.6 Messaging

test

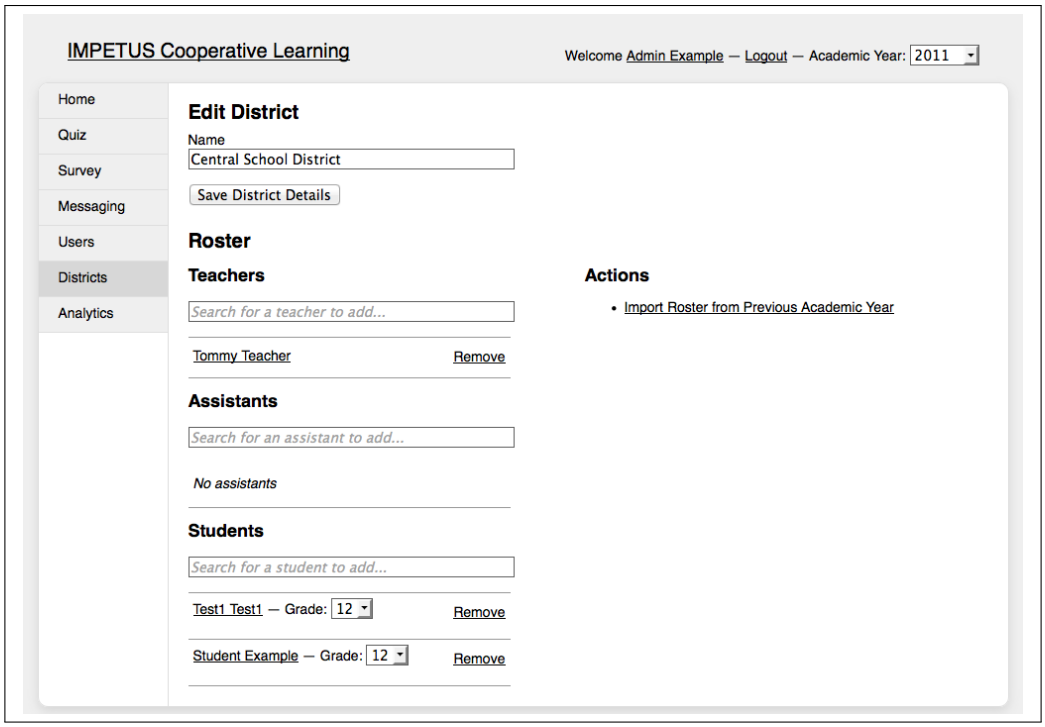

Figure 3.10: A privileged user editing a district.

3.2.7 Analytics

test

3.3 Known Issues

Chapter 4

Usability

4.1 Test Topics

4.2 Surveys

4.3 Reports

4.4 Significance

Chapter 5

Summary and Conclusions

5.1 Conclusions

5.2 Future Work

Bibliography

[1] Christine L. Borgman, Hal Abelson, Lee Dirks, Roberta Johnson, Kenneth R. Koedinger, Marcia C. Linn, Clifford A. Lynch, Diana G. Oblinger, Roy D. Pea, Katie Salen, Marshall S. Smith, Alex Szalay, and NSF Task Force on Cyberlearning. Fostering Learning in the Networked World: The Cyberlearning Opportunity and Challenge. Report, National Science Foundation, 08 2008.

[2] New York State Education Department. STEP Operations Manual. Website, 2011. `http://www.highered.nysed.gov/kiap/step/`.

[3] National Science Foundation. Cyberlearning: Transforming Education Cyberlearning, 09 2011. Solicitation 11-587.

[4] Clarkson University. Northern New York Partnership for Rural Integration of Science and Mathematics, 2010.

Appendix A

User Manual

Appendix B

Administrator Manual