A Web Based School Networking System with ICEfaces MVC in the Ethiopian Context

Pineal Terefe
pinny_thf@yahoo.com

Fikreyohannes Lemma
PhD Candidate, Addis Ababa University, Ethiopia
fikreyohannes@yahoo.com

Abstract

This work came into existence to address the necessity that educational institutions found in Ethiopia have communication and resource sharing needs. Besides, it also addresses one more issue, which is the fact that the public does not have a centralized place where they can get information about the whereabouts of the educational institutions and the services that they provide. After clearly identifying and prioritizing these problems, the paper illustrates the method, architectural design techniques/technologies, and other approaches required to developing web-based software. Ultimately, it brings solutions to the aforementioned issues with the help of a software prototype. In doing so, the software's overall architectural design will be illustrated which will be incorporated by the end product, in this case the prototype. The methodology that was incorporated for the design is a combination of architecture centric and user-centered, which means that the needs of users of the system will be considered from the start of the designing process but most of the design will focus on the structural components of the system, their interaction, and significant design decisions made about the organization of the system. MVC (Mode-View-Control) architecture is chosen for the overall software design and implementation which provides modifiability, flexibility, and a clear separation of design concerns among other things. The framework used for the implementation of the prototype is ICEfaces. The detail on how this framework realizes MVC is clearly stated in this paper. Apart from the design of the software, the deployment architecture is also presented which is 3-tier architecture. Finally, using snapshots from the developed prototype, each of the system's features will be demonstrated and a conclusion will be drawn based on the results of the prototype.

Keywords: Model View Controller; School Networking System; Web Application; ICEFACES MVC Implementation; Software Architecture

1. Introduction

At this day and age, Internet is playing huge role in bringing the world closer together with adequate information. The numerous possibilities that are offered by its existence paved a way for software developers to use it and make something that helps different aspects of our life. People can communicate with each other by sending e-mails, via chat or using other ways like online discussion forums. People can also search for electronic materials, ideas and/or share electronic resources by uploading and downloading them. Information is a key to success and one way of availing it to the people is via applications that run on the Internet. The Internet by itself is nothing but a huge network and what makes it useful is the applications and software that run on it which are designed to solve problems that a particular group of people are facing. These problems can be different based on their nature, severity, applicability, and many other reasons.

This being said, educational institutions of our country starting from elementary schools to universities, all use the Internet one way or the other. Specifically speaking, they need to communicate with each other to share ideas. They need to share electronic resources like audio, video, e-books, tutorials, research papers, and the like. On the other hand, the public also needs to know about the whereabouts and other detailed information of these educational institutions. Parents, being part of the existing educational system, also need to get connected to their children's school to get frequent information about the status of their children and other essential pieces of information about the school

community. All these issues need a solution. There can be different possible ways of tackling this; for instance, by investigating for any existing work done on similar issues and customizing it to achieve all of these abilities. The other possible option is designing a new software whose goal is bringing a solution to the issues mentioned here. Both of these ways need thorough assessment of what is out there and what different methods and technologies there are to be used to attain the desired outcome. This paper summarizes the overall processes that were followed to address all of those needs that our educational institutions have regarding communication, resource sharing, providing information to the public, and creating a way for parents of students to interact with the school of their child.

2. Background

Educational institutions are one segment of the vast varieties of service providers in our country. By educational institution, it is meant to say any of the higher or lower level schools including universities, colleges, high schools, and elementary schools. These institutions share some common features/functionalities that can be generalized without affecting the way how things are actually being done at a particular institution. Obviously, each of these institutions needs to communicate with each other and/or let their members communicate with one another; they need to share electronic materials amongst each other, they need the public to know about their existence and/or their services, they need to manage student records, and they also need to stay connected with the public for mutual benefit (accepting donations or for other social reasons). Taking these common needs into account, a more generalized solution can be proposed architecturally which makes the application design more effective and operationally feasible. The major issue here is to create a structure where each of these institutions exists in a system where all the aforementioned issues are addressed.

This being said, currently there are a number of technologies and design patterns that software developers can incorporate in their software designs. Specially, for web based software, there are new and advanced web technologies that are emerging these days. On the other hand, architectural patterns are the other issues that a software developer should identify and integrate into the designing process. Choosing the right architectural pattern demands knowing the nature of the software to be developed and picking the right pattern among many possible architectural solutions. This can be done by stating the pros and cons of each possible architectural style and choosing the one that best fits the design considering the architecturally significant requirements set for the project and the design methodology chosen for the overall project. Just like the designing process, the deployment is also a crucial aspect of software development. It involves identifying the software and hardware components and how they are going to be mapped to different physical processes.

Among the many development technologies, ICEfaces is the one that was selected to be incorporated for this work. ICEfaces is a JSF extension that adds rich UI and AJAX features to JSF applications. It also supports AJAX which eliminates the full-page-refresh problem that most of the existing frameworks have. AJAX plus rich UI are often referred to as *Web 2.0* or *RIA* [1]. Thus, ICEfaces is a framework for creating Web 2.0 applications (or RIAs). The other thing is facelets support. Facelets is a view technology for JSF that replaces the standard view handler which is JSP. It has amazing benefits like: Fast Templating/Decorators for Components and Pages, their APIs aren't dependent on a Web Container; they provide the ability to specify UIComponent trees in separate files (UICompositions), XML configuration files aren't necessary, and so much more. ICEfaces supports MVC architectural pattern. MVC is a design pattern which provides flexibility and improves the performance of the application by separating the application into three layers: Model, View, and Controller.

The design methodology used in this work is mainly architecture centered design method (ACDM). As it is explained in [4], ACDM provides techniques and a structure for designing the architecture and then using the architectural design to guide the programmatic aspects of a project.

3. The Proposed Solution

The output of the research is a web based school networking software. The ultimate goal of the development is to bring a solution to all of the issues that are stated in the previous section which are issues of communication and resource sharing between different types of educational institutions and involving the public and parents of students in the communication and resource sharing aspects as well. The different types of users that the software encompasses are: School Administrators (Representatives), Teachers, Students, Parents, and Public users (non-registered users). Except the public users all the rest will be registered into the system and they will have their own accounts to use the system.

3.1 Architecturally Significant Requirements

1. The system should be usable and should address usability attributes like ease of use, learnability, and consistency.
2. The system shall respond to search actions in not more than 5-7 seconds depending on the specificity of the search keys provided by the user.
3. The architecture of the system must support modifiability.
4. The system must incorporate role based security.
5. The system should use cryptography to secure user account.

3.2 Architectural View

The business process view is depicted by UML use-case diagram to show major business processes of the system as shown in Figure 1.

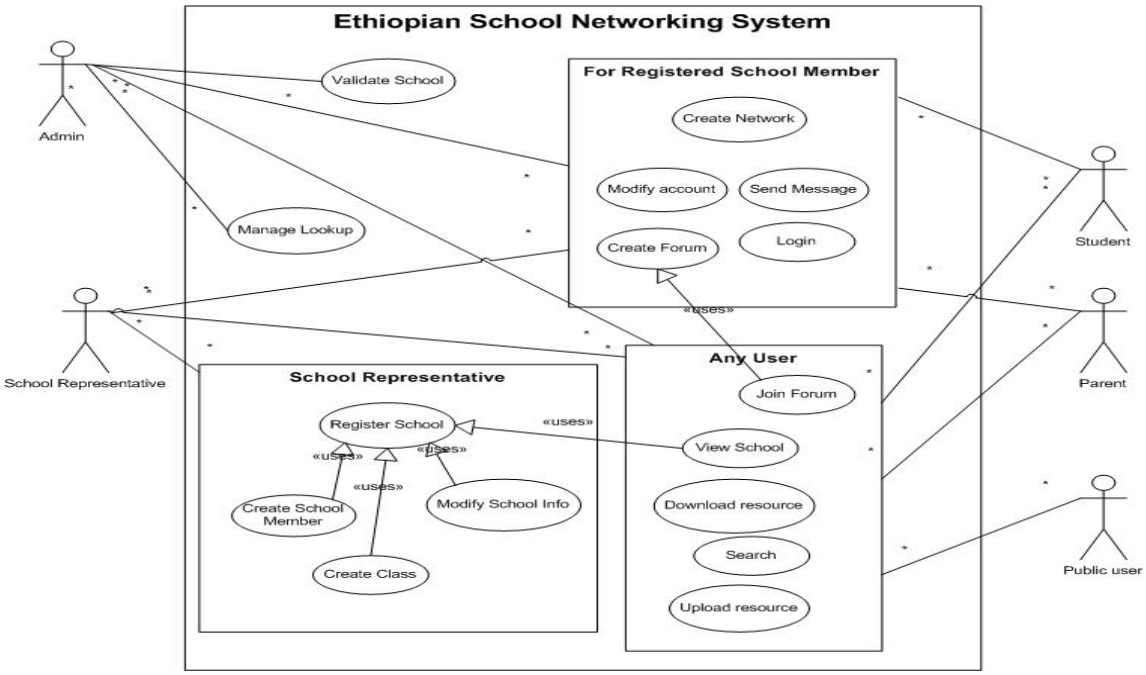

Figure 1: System use-case model

3.3 Deployment

Three-tier-architecture is chosen for the deployment. This architecture has three distinct tiers: presentation, application, and data tiers which represent the client side browsers, applications servers with Apache Tomcat (glassfish) servers installed on it, and a database server with MySQL 5.0 installed on it respectively as shown in Figure 2.

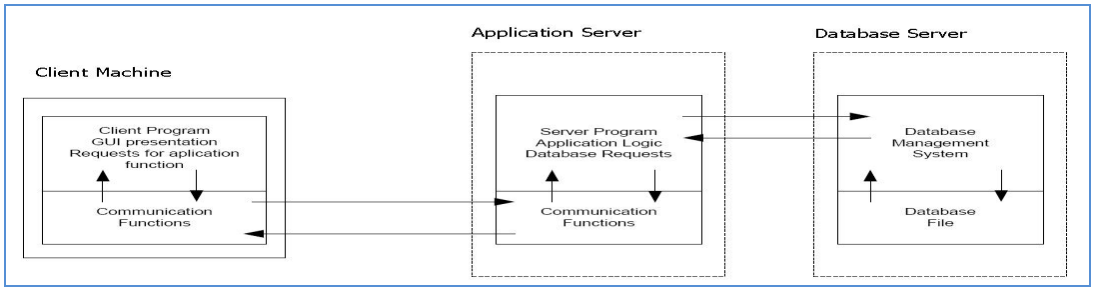

Figure 2: Three tier architecture

3.4 Chosen Architecture

The architecture chosen for the design of the software is MVC (Model View Controller). This pattern proposes a clear separation of concerns by separating the application into three which are model, view, and controller. The model layer of MVC is responsible for managing data. JSF implements the model layer with its managed Bean facility. The second layer of JSF's MVC framework is the View Layer which defines the user visible aspects (User Interface) of an application. JSF MVC architecture uses EL (Expression Language) to bind the view to the model. The third and the last layer of JSF's MVC architecture is the controller. It handles user interactions and navigation between views. JSF implements the controller layer with navigation-handler which by default supplies action plus outcome style navigation rules.

4. Prototype

The prototype is developed using ICEfaces MVC framework. The view handler used is facelets which adds advanced features to JSP which is the default view handler of ICEfaces. Templating feature of facelets makes the design modifiable, readable, and makes the process of software design easy and manageable. To give a highlight on how the software operates, the administrators/representatives of a particular educational institution will submit their personal information and the information of the school that s/he represents. This request arrives at the web-admin and s/he will examine the validity of the submitted information and will either accept or deny it. If the information is accepted, then the corresponding school representative's account will be activated which will allow him/her to login to the system and do all the tasks the system allows them to do. These tasks include registering class (which identifies a batch of students uniquely (e.g., Veterinary_1st_year), students, teachers, and parents. Once each of these different user types are registered and have their account, they will be able to login and access the system and perform tasks that are visible to them based on their role. The features of the software are defined in the previous section; hereafter we will briefly present how each feature is implemented by the prototype.

The software will have the following distinct features:

1. *Educational institution management:* This feature handles the following tasks:
   - Requesting school account by submitting school information
   - Approving submitted school
   - Updating school information
   - Blocking/Denying school
   - Viewing school information
2. *User management:* tasks included in this feature are:
   - Registering teacher, student, parent
   - Communicating parents of a student with:
     - School representative
     - Teachers
     - Other parents
3. *Class management:* the tasks of this system feature are:
   - Registering new class
   - Modifying existing class
   - Assigning teacher(s)/instructors to a class
   - Assigning/removing or changing students to/from a class
4. *Upload download management:* tasks handled here are:
   - Uploading resources (e-book, audio file, video)
   - Downloading resources

5. *Message management:* tasks handled here are:
   - Sending message to friends
   - Checking for messages (Inbox)
6. *Profile management:* the tasks included in this feature are
   - Updating personal information
   - Changing profile picture
   - Changing login password
7. *Network management:* this feature includes the following major tasks:
   - Sending friendship request to other users (school representative, teacher, student, parent) separately.
   - Approving or denying sent friendship requests.
   - Viewing friends to either send message or disconnect friendship.
8. *Forum management:* this feature includes the following:
   - Creating new forum
   - Posting to a forum
9. *Search management:* tasks included in this feature are:
   - Searching for a resource, person, forum, school.
10. *Locale management:* this feature is responsible for:
   - Handling locale of the system by changing language (font)

5. Related Work

Based on the problems that were identified to bring solutions to and the time allocated for the design and implementation of the entire work, a few related works were examined.

The first work is software called Moodle. Based on the definition collected from [2], Moodle is a web-based system that helps educators to create interactive online courses without having to know the ins and outs of web design and development. According to [2], Moodle is a versatile tool that allows students and teachers to work collaboratively online and it covers issues like: upload video, audio and links relevant to a lesson, engage students in a discussion forum, guide a real-time discussion or "chat", create, conduct and grade quizzes, prompt students with survey questions, post classroom blogs, create and maintain a classroom glossary related to a lesson, assign, collect, review and grade assignments, work collaboratively on a classroom Wiki or group-edited document.

Most of the things that Moodle does relate to the problems that this work tries to address, but again, Moodle does not provide information to the public about any of the educational institutions registered in it (regarding location, departments/faculties, courses, teachers and the like) and/or parents do not have a role to play in either knowing about the status of their children or by communicating with the rest of the school. These two features are proposed to be parts of this work.

The second related work that was reviewed is web-based software called TemariNet. Based on [3], temarinet.com is an educational social Learning Website created to promote a Healthy and Motivated Responsible Student Community. It mainly focuses on providing a medium where teachers and students communicate (colleges and universities) and learn from one another by communicating via forums and blogs. Based on the review, like Moodle, temarinet.com only focuses on the students and teachers of a particular college or university but not the public or parents of the students.

Temarinet.com somehow covers the two major problems that this work tries to bring solutions to which are communicating members of educational institutions and allowing resource sharing. But, it gives services to colleges and universities only. Plus, it does not let the public get information about the colleges or universities or the services that are using it. Parents are not included here as well; just like the previous one. On the other hand, temarinet.com lets users from anywhere to sign up and the administrator has to approve them individually so that they can get access to the system which we believe is too much burden on the administrator(s). This task could have been handled by the school itself using the representative of the school. But again, since temarinet.com does not register school information, the one way to handle sign ups is probably by the administrators.

6. Conclusion

Using the implemented prototype, the public can get information about registered educational institutions, students/teachers, the rest of the school members can interact with members of other schools to communicate and share resources, and parents can interact with the school of their children. On the other hand, this work contributes to the science of software engineering by showing how to design software using MVC architectural style. It also illustrated the overall software designing process following the appropriate steps and procedures. In addition to these, it incorporates ICEfaces technology for the implementation of the prototype which has advanced features as mentioned in the previous sections. For these reasons, this work contributes to the society and the science of software engineering.

References

[1] http://www.smart-soft.com/ice/gen/aa_Day1/bb_WhatIsIcefaces/index.html, last accessed on March 12, 2012.

[2] http://orvsd.org/tutorial/what-moodle: last accessed on January 20, 2012.

[3] http://en.wikipedia.org/wiki/Service-oriented_architecture: last accessed on March 23, 2012.

[4] Anthony J. Lathanze, "Architecting Software Intensive Systems", CRC Press, London, 2009.