Master's Project Proposal

Master's project website:
A comparison of server-side technologies

Prabhakaran Nagarajan

February 2004

1. Summary

Any Graduate student in the Computer Science department at RIT has to complete either a Thesis or Project as part of their degree requirements. One of the requirements for a project is for it to have suggestions for enhancements so that future students can use the project as a baseline and improve on it. A common repository that allows students to browse submitted project reports and proposals would therefore be desirable. The most common form for dissemination of information these days is through the World Wide Web. However, the technology used to store this information and display it to the user is diverse. Each technology has its own specializations, has its own pros and cons.

There currently exists a **Masters project website** designed by Dr. Schreiner. While Dr. Schreiner's MS website is one possible implementation using servlets, some competing and some complementing technologies exist that allow for equally elegant solutions. This project aims to select two technologies that are either extensively used, and/or are defining technologies of the future, with the aim of comparing the technologies used to implement them. The implementations will be compared against Dr. Schreiner's solution as well. The comparison will focus on the ease of learning/using the new technology, the flexibility of the API, as well as the perceived performance.

One of the most commonly used server-side scripting technologies today is **PHP.** PHP, short for PHP: Hypertext Preprocessor, is an open source initiative headed by the Apache Software Foundation, and is one of the more widely adopted scripting languages. PHP by itself provides for dynamic page creation, and is available on multiple platforms. **MySQL** is an open source relational database widely used in conjunction with PHP. Considering that the CS department at RIT is in the process of migrating the department website to PHP and MySQL, choosing PHP as one of the implementations serves the dual purpose of comparing a popular scripting technology, as well as integrating it with the rest of the CS website. The first experimental server will employ PHP and MySQL.

XML Web services are fundamental building blocks in the move towards distributed computing on the Internet. The primary advantage of the web service architecture is that it allows programs written in different languages and on different platforms to communicate with each other in a standards-based way. Web Services are significantly less complex than earlier approaches that made the same promises, and as an additional advantage, work with standard protocols – XML, HTTP and TCP/IP. **ASP.NET** makes building XML Web Services very easy. The infrastructure provided by ASP.NET allows for building web services conforming to industry standards such as SOAP, XML and WSDL, while leveraging ASP.NET's performance, state management and authentication functions. This project will use XML Web Services and ASP.NET as the second server side technology. The ASP.NET solution can only be hosted on a Windows platform, and for this project, we will use the ASP.NET development environment's (Visual Studio) built-in web server. Emphasis will be on the ease with which web services can be created, and the complexity involved in a client-side invocation.

The web browser - based user interface will be as similar as possible for both parts of the project.

2. Overview

2.1. General Overview

The Internet has been the most popular ways for sharing information in recent years. As the number of publicly available sites increase, the users demand for more dynamic content has also increased. This has lead to a proliferation of technologies available to serve up dynamic content. Though the choice of technology depends on the target environment, a wide range of implementations can be had on any given platform, including various cgi implementations, server side JavaScript (livewire), ColdFusion, PHP, Active Server Pages, .Net, Java Server Pages, servlets and web services. The server side technologies facilitate creation of dynamic web pages, but often do not operate on their own. Databases often play an integral part in storing the information, and the scripting languages query the database to create the dynamic web page. While Object databases (which allow for object persistence) can be used to store data, they are not mature enough to be used in production quality systems. The more robust and well-understood relational databases are normally used for the persistence layer. A variety of vendors cater to this, and the popular ones are Oracle, DB2, Microsoft SQL server, Sybase, MySQL, Postgre, and MS Access.

2.2. Existing Solution

The current Masters project website was designed and implemented by Dr. Schreiner using servlet technology. The implementation consists of two sites, a static site[2] and an admin site. The static site contains generated html – pages containing project details, as well as pages that group the projects by faculty, year or student name. The admin site is servlet based, with functionality to create project and manage files. There is no persistent data store; the information is stored as XML, validated against the ms.dtd document type definition. The HTML for the static site is generated by from XML by periodic cron jobs. Security is based on the user account of the logged in faculty.

The same functionality provided by Dr. Schreiner's implementation will be offered by this project, using two different technologies, and the pros and cons of both implementations will be compared against Dr. Schreiner's solution. This project was conceived from Dr. Schreiner's presentation[1] of the implementation at a Project Seminar.

2.3. The PHP / MySQL Platform

The PHP-MySQL combination is a cross-platform, open-source implementation used to create dynamic web pages. PHP is a server-side scripting language, consisting of PHP code enclosed in PHP tags. The web server processes the PHP code. MySQL is a Relational Database Management System, compliant with the American National Standards Institute (ANSI) SQL, and has multithreading abilities. This combination is one of the most popular platforms for implementing websites. A detailed overview of both is outlined below.

2.3.1. Overview of PHP

PHP is an HTML-embedded scripting language. Started as a quick Perl hack, it has evolved into a language that allows web developers to write dynamically generated pages

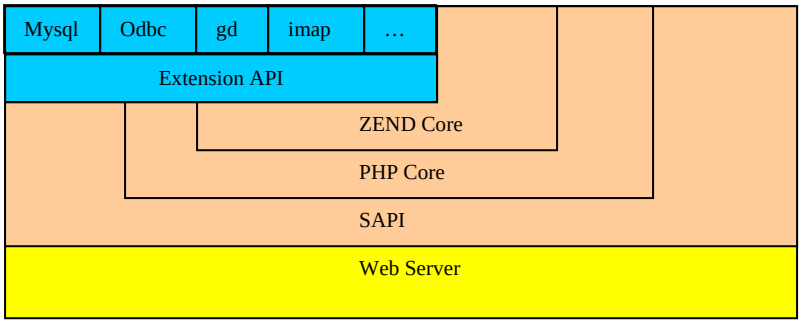
quickly. When compared to other scripting languages like ASP or ColdFusion, PHP is open source and cross platform, supported by a large group of open-source developers. PHP offers excellent connectivity to most common databases like Oracle, Sybase, MySQL and many others through ODBC. According to some surveys, PHP is today's fastest growing technology for dynamic web pages. PHP 4.x, the latest stable version of PHP, is significantly different from the previous releases in the following aspects:

- The scripts are "compiled first and executed later".
- Web server independence – the web server interaction has been abstracted to its own layer.
- Is thread-safe.
- Built-in session management capabilities.
- Has a high performance internal function API and independent output layer.

When free CGI solutions exist, why PHP? For starters, CGI scripts are not scalable. Each new request for a CGI script requires a new process in the kernel, requiring memory and CPU time. Besides being free, PHP is faster to code and execute. It is available under a BSD style license, and runs on *nix, Windows, and Mac OS X, and is designed to integrate with Apache, the most popular web server on the Internet. PHP is modifiable; the source code is readily available (even if not modifying the source, it helps if you can inspect it) and is designed for expansion. Unlike C or Java, PHP was specifically written for web development. Support for PHP is free and readily available.

PHP gets installed on top of the web server software. It works with versions of Apache, Microsoft IIS, Netscape Enterprise Server and other server software packages. A PHP page consists of HTML code containing embedded PHP code. When the browser makes a request to a PHP page, the web server defers processing of the page to the PHP server. The PHP server interprets and processes the PHP code. The resulting HTML is sent to the browser and displayed to the user.

PHP 4 architecture:



The language parser itself is a self-contained component called the ZEND core. PHP function modules or extensions are also self-contained. The Web server abstraction layer, SAPI simplifies the task of adding native support for new web servers. The SAPI (Server API) module currently has implementations for Apache, Roxen, Java (servlet), ISAPI (Microsoft IIS and Zeus) and CGI. All of PHP's functions are part of one of the layers with a side facing up in the architecture figure. Most functions such as the MySQL support are provided by an extension (MySQL and ODBC support is built in to

PHP from version 4.3). Most extensions are optional. They can be linked into PHP at compile time or built as dynamically loadable extensions that can be loaded on demand.

The PHP programming language includes, as might be expected, variables of various types, arrays, functions and control structures and the same set of operators as C. PHP is an object oriented programming language with most of the standard mechanisms but without any standard classes.

2.3.2. Overview of MySQL

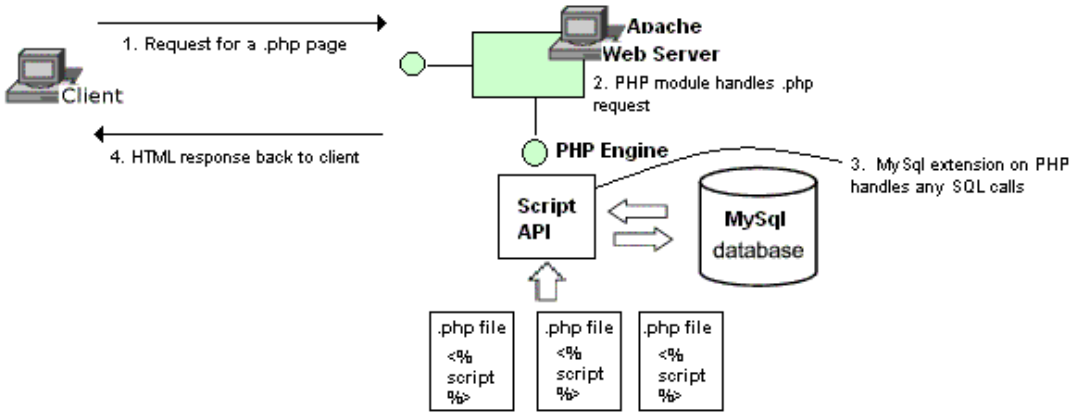
MySQL, the most popular Open Source database, is a very fast, multi-threaded, multi-user and robust relational database server with a SQL front end. Initially designed to work with medium sized databases (10-100 million rows, or around 100mb size) on small computers, work is ongoing to support terabyte-sized databases, and to add extensions to SQL and support non-SQL features. The latest stable version of MySQL is 4.1, which is currently SQL-92 and ODBC 3.51 compliant, is being upgraded to be SQL-99 compliant. Some of the features (or lack thereof) are explained below:

- MySQL supports both transactional (InnoDB and BDB transactional engines) and non-transactional (MyISAM) table types. Therefore, a choice on what paradigm to use can be done on a per-table basis. For ISAM type tables, the operations are atomic, and table-level locks are also provided.
- Supports Foreign Keys for data integrity
- No support for stored procedures and triggers in 4.1. This feature will be available only from version 5.0 onwards.
- No support for views.

The MySQL team has been mainly concentrating on speed and reliability. The feature set currently supported by MySQL is sufficient for heavy web/logging usage and mission-critical 24x7 usage.

For the purposes of this project, MySQL is completely adequate as a database server. It integrates very well with PHP and will be used as the database server for both implementations.

Life Cycle of a Web Request in Apache/PHP/MySQL platform:



1. User clicks on a link to a .php page on the browser.
2. The web server receives the request. The "LoadModule", "AddModule" and "AddType" directives instruct Apache to delegate the request to the PHP module.
3. The PHP engine receives the request, interprets the php code enclosed within the tags.
    a. Support for ODBC and MySQL is inbuilt in the PHP engine, and any MySQL calls are sent to the MySQL database engine through the MySQL driver.
    b. The MySQL engine executes the SQL and returns the results. The MySQL driver does appropriate data type conversions.
4. The PHP engine completes the translation of PHP code to HTML and sends the result back to the client browser.

2.4. The .NET / Web Services Platform

The Microsoft .NET Framework is a multi-language platform for building, deploying, and running Web Services and applications. ASP.NET is a compiled, . NET-based environment; you can author applications in any .NET compatible language, including Visual Basic .NET, C#, and JScript .NET. . In the .NET vision, an application is constructed using multiple Web services that work together to provide data and services for the application. This project will use ASP.NET web services that will expose methods to manage the project database, which will be accessed by ASP.NET Web Forms. A detailed overview of the various technologies involved in this solution is outlined below:

2.4.1. Overview of Web Services

A simplified definition of web services is: programmable application logic accessible using standard Internet protocols. Unlike previous component technologies, web services are not object-model specific protocols such as the Distributed Component Object Model, Remote Method Invocation or Internet Inter-Orb Protocol. Web Services typically use HTTP as their transport (while other protocols can be used) and XML as their data format (other mime-friendly formats are possible).

- XML Web services expose useful functionality through a standard web protocol, most likely SOAP.

- XML Web services provide a way to describe their interfaces in enough detail to allow a client application to talk to them. This description is usually provided in a XML document called a Web Services Description Language (WSDL) document
- XML Web services can be registered so that potential users can find them easily. This is done with the Universal Discovery Description and Integration (UDDI).

Unlike regular web pages, where the client is a person, the target for a web service is typically another software program. Furthermore, since a web service is strictly defined in terms of the messages it accepts and generates, consumers of the service can be programmed in any language, as long as they can create and consume messages defined in the web service interface.

For this project, we will use create Web services using the .Net platform, and the consumer for these web services will be an ASP.NET web application. Since this web service is for internal use only, there will be no requirement to create a discovery document and use UDDI to publish it.

2.4.2. WSDL Basics

The web service architecture makes it possible for any consumer with XML support to integrate with Web Service applications. However, in order to accomplish this, consumers must know precisely what the interface is prior to making the call. XML Schema partially fill this need, as it allows defining the structure of xml messages, but cannot communicate the additional details involved in the communication. For example, a schema can give the structure for and "Add" element, and another called "AddResponse", but cannot convey the relation that the "AddResponse" is a message returned on an "Add" message call.

WSDL provides a way to group messages into operations and operations into interfaces. It also provides a way to define bindings for each interface and protocol combination along with the endpoint address for each one. A complete WSDL document contains all the necessary information to invoke a web service. Similar to the IDL file for COM and CORBA, a WSDL file can be considered as a contract between client and server.

2.4.3. SOAP Basics

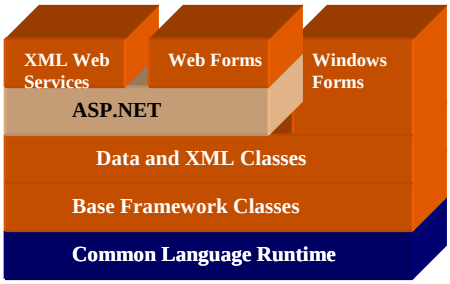
The Simple Object Access Protocol (SOAP) defines a lightweight communications protocol for information exchange. Part of the SOAP specification defines a set of rules for how to use XML to represent data. Other optional parts of the SOAP specification define an extensible message format, conventions for representing remote procedure calls (RPCs) using the SOAP message format, and bindings to the HTTP protocol. (SOAP messages can be exchanged over other protocols, but in most cases HTTP is the only protocol supported. Some implementations support MSMQ and SMTP).

Different vendors provide SOAP implementations built to SOAP specifications, called SOAP toolkits. Microsoft SOAP Toolkit 2.0 translates COM function calls to SOAP and the Apache toolkit facilitates translation of Java function calls to SOAP. The WebMethods framework revolves around mapping SOAP messages to methods on a .NET class.

2.4.4. .NET basics

The .NET framework is a multi-language environment for building, deploying and running XML Web services and applications. It consists of three main parts:

- Common Language Runtime (CLR): The CLR is responsible for managing memory allocation, starting up and stopping threads and processes, enforcing security policy, as well as satisfying component dependencies.
- Unified Programming Classes: The framework provides developers with a unified, object-oriented, hierarchical and extensible set of class libraries (API). All .NET languages, including VB, C#, J# and JScript have similar access to the framework, allowing developers to choose the language they want to use.
- ASP.NET: builds on the programming classes of the framework, extending it to a web application model with a set of controls and required infrastructure. Using XML web services features, ASP.NET developers can write their business logic and use the ASP.NET infrastructure to deliver services via SOAP.



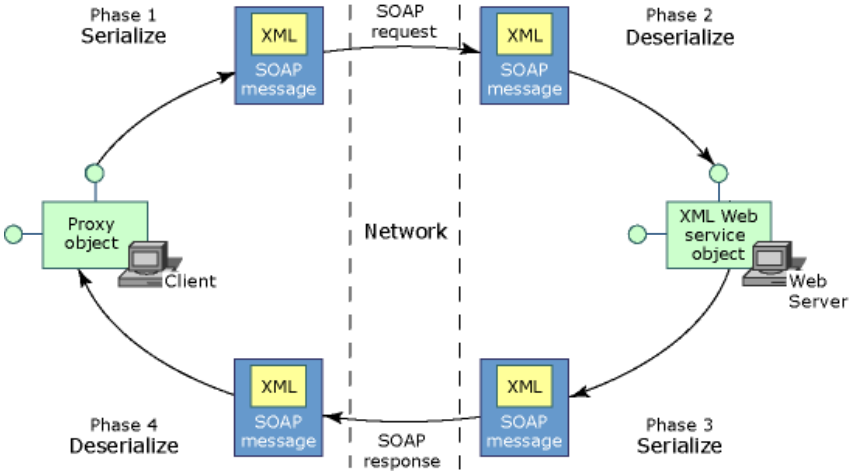
ASP.NET framework simplifies creation of web services. When you build an XML web service using ASP.NET, it automatically supports client communications through the HTTP-GET, HTTP-POST and SOAP protocols. Since HTTP-GET and HTTP-POST pass parameters through url-encoded name/value pairs, the data type support is not as rich as would be through SOAP. Normally, the complex data types are defined using XML Schema Definition (XSD) schema, however, developers using ASP.NET for creating web services do not have to explicitly define XSD schemas. Rather, they can build a managed class. ASP.NET handles mapping class definitions to a XSD schema and mapping object instances to XML data in order to pass it back and forth across the network.

XML Web services consist of two parts: the XML Web service entry point and the code that implements the XML Web service functionality. In ASP.NET, the .asmx file is a text file that serves as the addressable entry point for the XML Web service. It references code in pre-compiled assemblies, a code-behind file, or code contained in the .asmx file itself. The Web Service processing directive at the top of the .asmx file determines where to find the implementation of the XML Web service.

When called from a Web browser without supplying a recognized query string, the .asmx file returns an automatically generated Service help page for the XML Web service. This is the equivalent of doing an HTTP GET on the .asmx file. Appending a ?WSDL into the browser returns a Web Service Description Language (WSDL) document.

Once a web service is created using ASP.NET, it can be invoked from an ASP.NET client itself. To do this, a web reference needs to be created. Web Service Discovery is the process by which a client locates a web service and obtains its WSDL file. Adding a web reference creates a web service proxy class. This proxy class exposes the methods of the web service, and handles the marshalling of data back and forth between the web service and the client application. If the web service is not accessible when developing the client application, a utility is available to create the proxy class. The proxy class can then be compiled into the application and its method called. This results in the proxy class packaging a SOAP request across HTTP and receiving the SOAP-encoded response, un-marshalling the response and converting to the appropriate data type.

Life Cycle of a SOAP request in the .NET World:



The following describes the sequence of events that occur when an XML Web service is called:

1. The client creates a new instance of an XML Web service proxy class. This object resides on the same computer as the client.
2. The client invokes a method on the proxy class.
3. The infrastructure on the client computer serializes the arguments of the XML Web service method into a SOAP message and sends it over the network to the XML Web service.
4. The infrastructure receives the SOAP message and deserializes the XML. It creates an instance of the class implementing the XML Web service and invokes the XML Web service method, passing in the deserialized XML as arguments.
5. The XML Web service method executes its code, eventually setting the return value and any out parameters.
6. The infrastructure on the Web server serializes the return value and out parameters into a SOAP message and sends it over the network back to the client.
7. The XML Web service infrastructure, on the client computer, receives the SOAP message, deserializes the XML into the return value and any out parameters, and passes them to the instance of the proxy class.
8. The client receives the return value and any out parameters.

Visual Studio .NET provides the tools needed to design, develop, debug, and deploy Web applications, XML Web services, and traditional client applications. The latest Visual Studio .NET offering from Microsoft, codenamed “Whidbey” enables rapid application

development by providing better source code editing tools, richer Visual Designers, better web projects support (has its own built-in ASP.NET web server, no longer dependent on IIS or FrontPage extensions to be installed), support for pre-compiling ASP.NET applications and powerful Object Data binding.

Since the “Whidbey” release of visual studio and the .NET framework delivers a new set of tools and functionality to simplify application development without sacrificing existing language functionality, this project will use “Whidbey” as the IDE for creating ASP.NET web services, which will act as the second technology choice.

3. Functional Specification

3.1. Comparison Criteria:

This project will compare the three technologies from both server-side processing and page authoring perspectives, keeping in mind any organization's core requirements for ongoing maintenance, portability and security. The server-side processing comparison will include language features like platform support, database support, performance, security features, maturity and flexibility of API and object oriented capabilities. From an authoring perspective, the availability of IDE and learning tools and ease of coding will be critiqued. The performance of the implemented solution will also be examined – since the anticipated load on the system is not expected to be high (this will be used only by students and faculty at RIT, so the concurrent user load will not be too high), benchmark and stress tests will NOT be performed on the system, only the "perceived" performance (response time) will be considered.

The Masters Project website will be implemented using PHP and ASP.NET, using Dr. Schreiner's servlet implementation as a starting point, and all three approaches will be compared.

3.2. Implementation Specifications:

One of the aims of this project is to have the PHP version of this project integrated with the existing main web site. In view of this, the project will use the existing department database and make changes to it. The existing database already has tables that store the faculty information. The modules/files developed will be independent of the existing site, but will be written such that integration will be easy. The UI for both versions will be as similar as possible.

3.3. UI Specifications

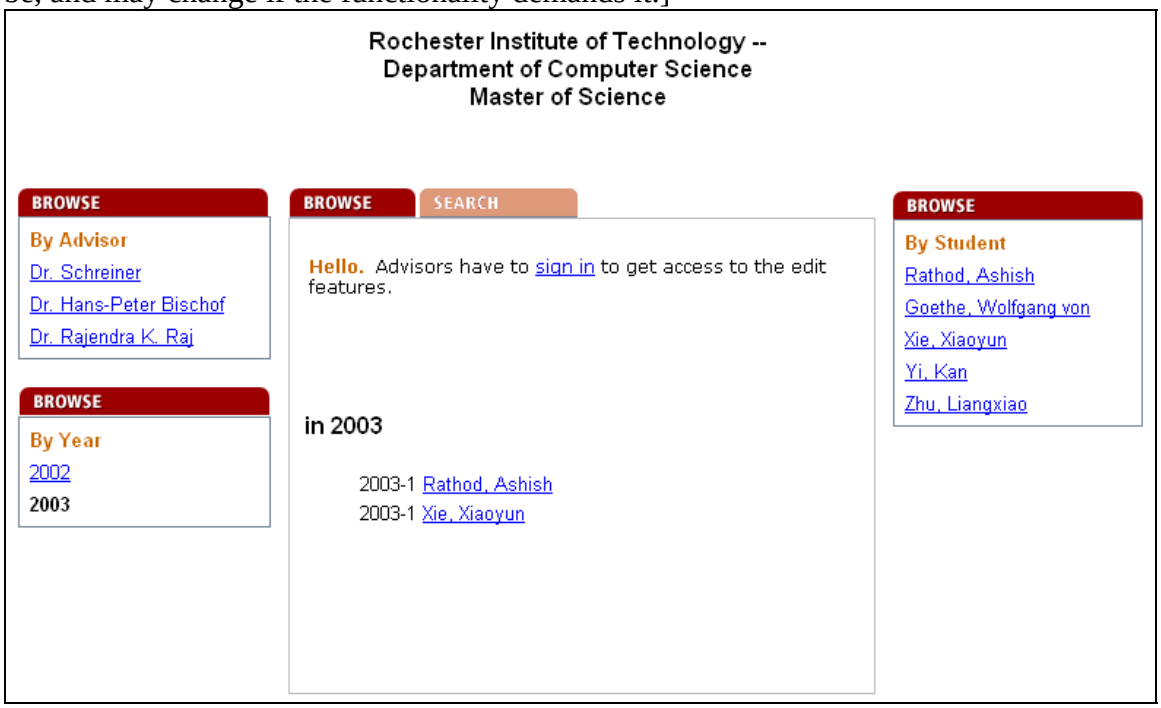
The specifications and screen shots depicted below are applicable to both the PHP implementation as well as the .NET implementation. While the PHP language does not contain any constructs for defining custom widgets, ASP.NET provides controls that encapsulate the required functionality. For example, ASP.NET provides a "login" control, which can be inserted to any web page. This control contains widgets that will accept a username and password, and call a user-defined routine to validate the username and password. In addition, this control optionally provides for "remembering logins" via cookies, and a mechanism for retrieval of lost passwords. The ASP.NET version will use inbuilt controls like this wherever possible.

The "admin" area where advisors can create/edit project details and the "static" area where users can browse project details will be integrated in the project website. Graduate Advisors need to sign in/login to be able to create and edit project details. However, any user can browse the site to view details about any project. To allow for easier browsing experience, the site will allow:

3.3.1. Browse

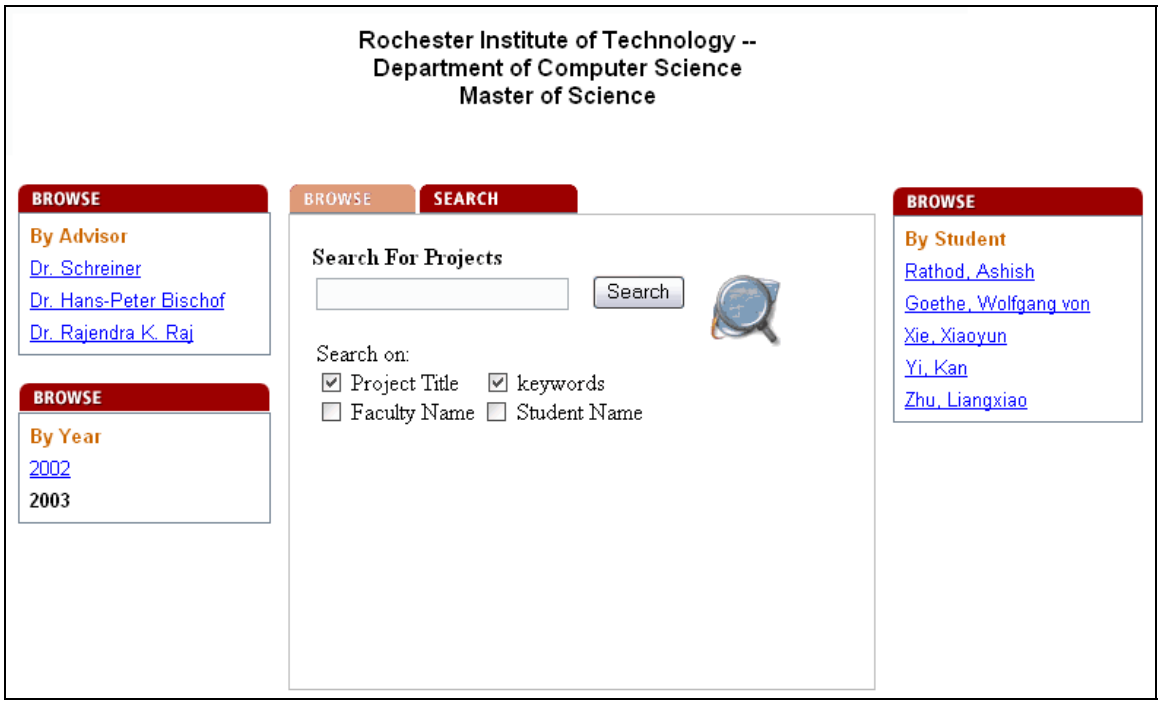
- Users to browse the project website by year the project was initiated, by the advisor/observer/reader for a project, alphabetically by project title and alphabetically by graduate student name. Users need not be authenticated to be able to browse the site.

[Note: The attached screen shot is a prototype of the page to be developed. This screen shot and all others that are in this document are only to indicate what the site is likely to be, and may change if the functionality demands it.]



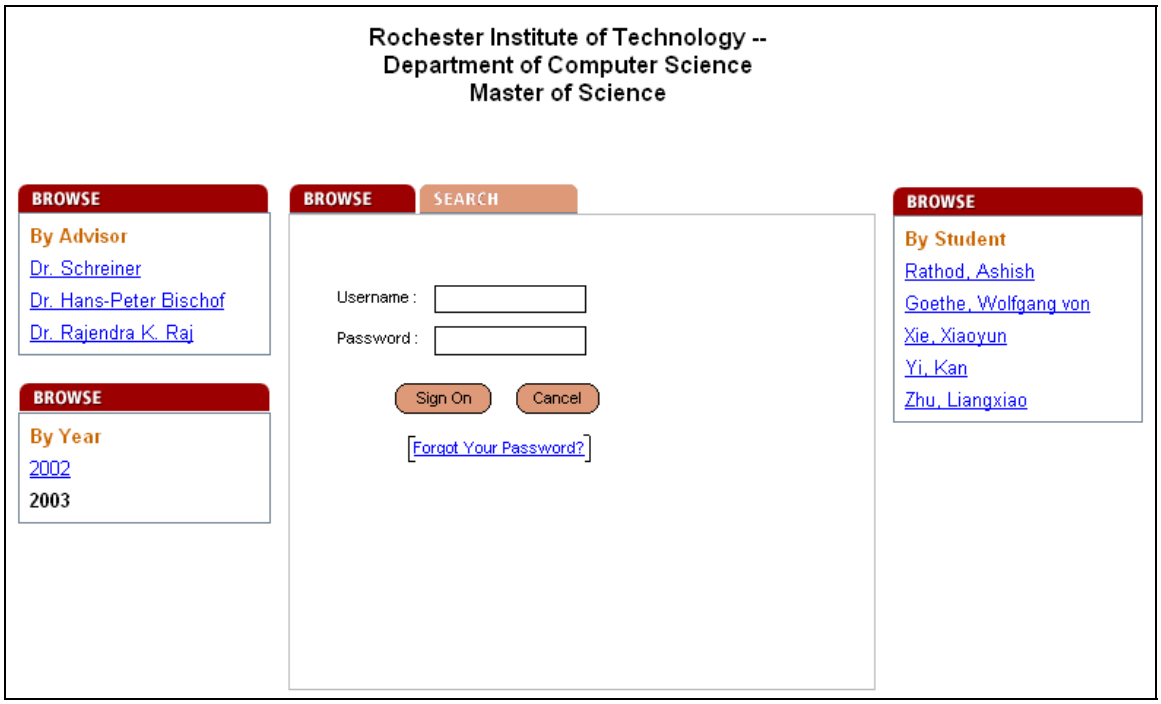
3.3.2. Search

- Users to search for a particular project from the website. The search can be based on keywords for the project, or can be part of the project title.

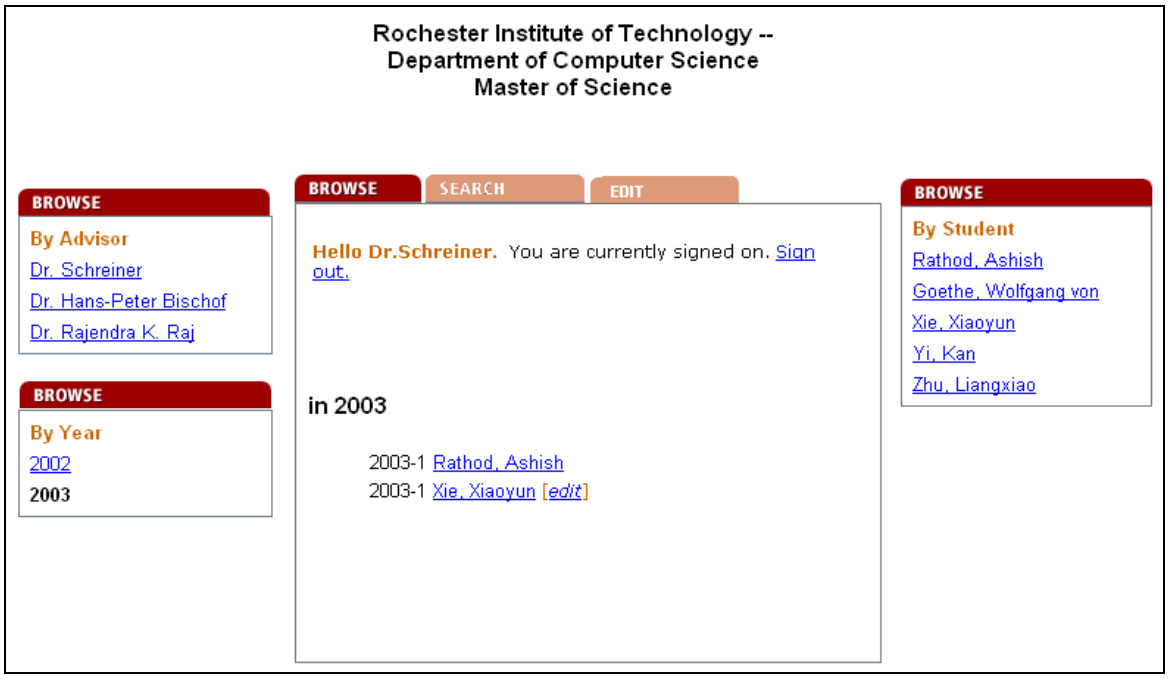


3.3.3. Login

- To be able to access the edit features of the site, the advisor must sign in. The existing faculty database has the faculty information. This database will be augmented with a username and password for each advisor. The sign on link will allow the advisor to enter in a username and password, which will be authenticated against the database. The system will also allow for retrieval of username and password via email.

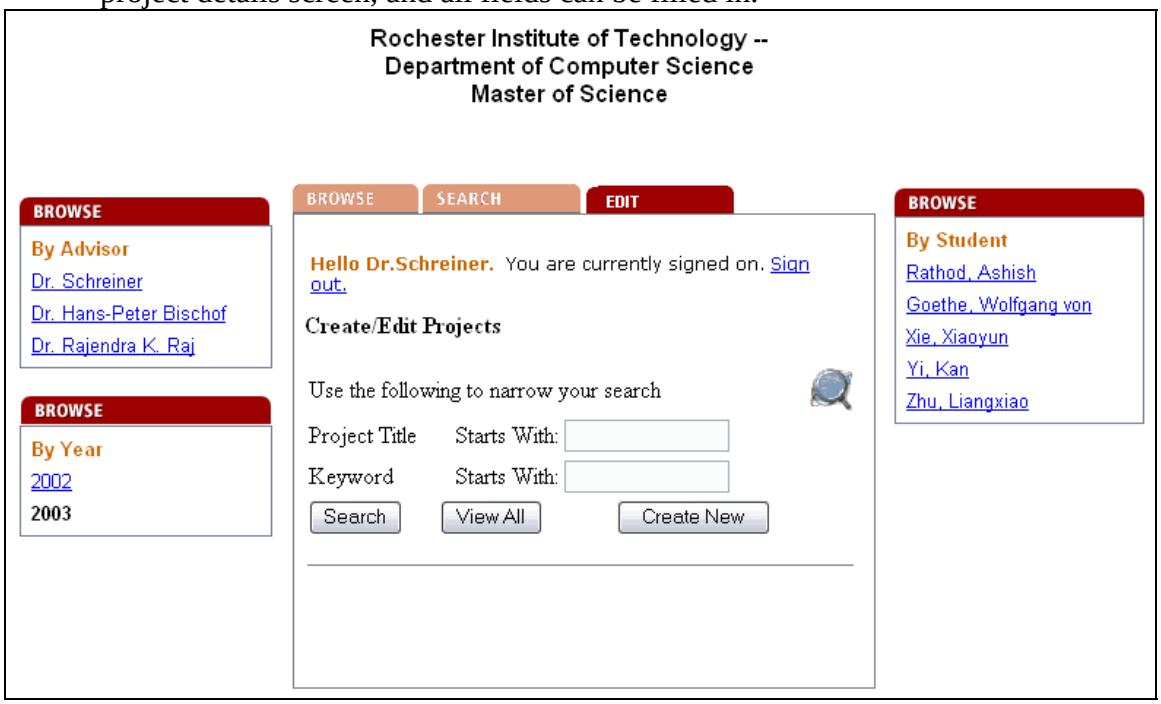


- Once logged on, the faculty has an extra option to create and edit project details.



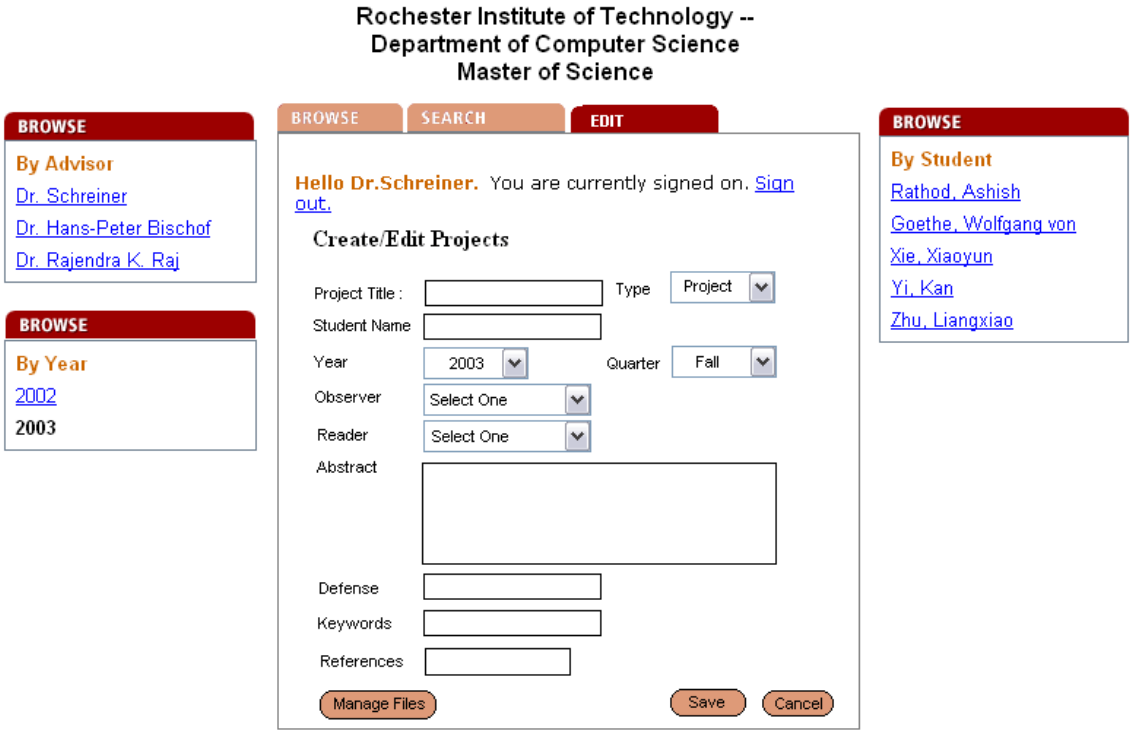
3.3.4. Create / Edit

- Committee chairman to enter or edit project details for students they currently advise. The edit page allows for entering filter criteria to narrow the search of projects to view/edit. The faculty can edit only those projects for which (s)he acts as the committee chairman. The edit page also allows for the faculty to create a new project – by clicking on the "Create New" button. This brings up an empty project details screen, and all fields can be filled in.

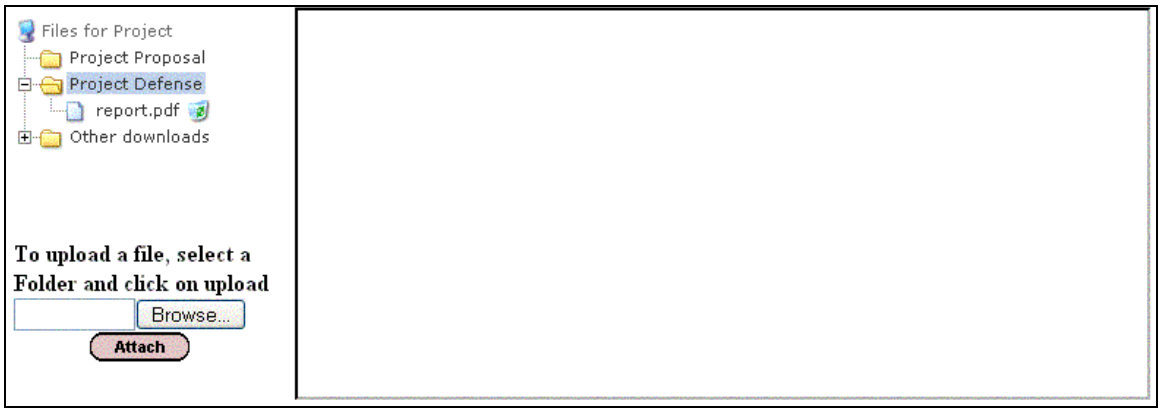


- While entering details for a project, the committee chairman must be able to specify:

- Project title
- Year and quarter started
- Student Name (First name, middle initial and last name)
- Type (project or thesis)
- Abstract
- Keywords (can use this for easy searches)
- Select the Observer from a pick list of existing faculty.
- Select the Reader from a pick list of existing faculty.
- The system defaults the chairman/advisor to be the person uploading the information
- Scheduled defense date
- File(s) for proposal and report.
- Any other files supporting the project (user guides, source code etc)
- References or Links. The references and links should be a comma or semi-colon separated list of valid URLs.



Clicking on the Manage Files brings up a screen where the faculty can select and upload files for the proposal, report and other categories. In order to do this, the faculty has to select a folder to upload to, browse for files from their local machine, and click on attach. Clicking on an uploaded file will load the file in the right pane. Clicking on the trash icon next to each uploaded file will delete the file.



1. Committee chairman can upload/replace files for any projects they advise.
2. Graduate coordinator can edit project details or project files for any student.

3.4. Roles and Privileges:

A user of the website can have one of the three roles:

- Visitor: This role awards the privilege of viewing the site to anyone possessing this role. The current implementation will allow for anyone to browse the site, so this role is granted by default to any established session. A future enhancement to this site could involve only authenticated users accessing the site. In view of this potential requirement, this role is being defined.
- Advisor: A user in this role will be able to edit the project details for any student he "advises" in a committee chairman's capacity.
- Coordinator: This role is akin to a "super-admin" – a user in this role will be able to edit any student's project details.
- Readers/observers cannot modify any project details.

3.5. Project File(s) store:

The files corresponding to any project are stored in the file system as opposed to the database. Though current databases allow for any objects to be stored in the database in the form of CLOBs and BLOBs, it does not give any additional flexibility or benefits to do so. As the number of files uploaded increase, the database has to be monitored and analyzed periodically to ensure that it does not run out of tablespace. Since the file system handles file creation and manipulation more effectively, all project files will be stored in the file system in a central location.

The UNIX user account used to run the application will need read and write access to the file system. This account will be used to create and maintain the project files. The user level access will be handled by the application, and not the file system – i.e., the application should ensure that only that project's advisor or the graduate coordinator modifies files belonging to one project.

3.6. Web Services Using ASP.NET Specifications

The main advantage of using Web Services is that any UI can easily be constructed on top of the web service. Since this project is mainly a comparative study of the technologies, the advanced controls available in windows forms or web forms will not be used unless they provide the same functionality as required by the project. This project will strive to keep the UI for the Web services client to be as similar as possible to the PHP implementation described above, with exceptions and the cause for them duly noted. Web services will be defined as required to support the UI.

4. Architectural Specification:

4.1. PHP Implementation:

For this project, Apache will act as the web server, with the PHP engine loaded as a dynamic module, using the LoadModule, AddModule and AddType directives in the Apache HTTP configuration file. This allows the Apache server to pass all requests for web pages that have a .php extension to the PHP engine. PHP 4.3.3 has built-in support for MySQL.

PHPEdit is a free open source PHP IDE distributed under Q Public License (freedom to share and change the software) that has multiple features like syntax highlighting and an integrated debugger. This IDE will be used for the PHP/MySQL implementation.

While PHP is not a completely object-oriented language, PHP supports classes and object creation that allow for a certain degree of separation of business logic from the user interface. For this project, the following classes will be created:

- MySQL Connection: This class will abstract the database connection and MySQL specific PHP constructs. The other classes and php pages will also use this class to make SQL calls.
- Project Search: This class will contain multiple methods for searching projects. This class will contain methods to search for projects authored by students, guided by faculty in different roles, projects created in a particular year etc.
- Project Detail: This class will represent all the project details as well as the student and faculty details for the project.
- Login Helper: This class will handle login and logout functionality.

The UI will comprise of PHP pages, which are essentially HTML pages with embedded php code making use of the classes defined above. To ensure that code is not duplicated, include files will be created that contain the display logic for a particular section of the site. These files will be included in all pages through IFRAMEs. The following include files will be created:

- FacultyList.php: This will contain the elements to create the top left tab in the UI specifications – the listing of current faculty members. Each faculty member's name is hyperlinked, which when clicked will display a list of projects guided by that member, sorted by their role (Chairman, Reader or Observer) and then by the year the project started.
- YearList.php: This will contain the elements to create the bottom left tab in the spec – the list of years in which projects were done. The years are hyperlinked to a listing of all projects undertaken that year.
- StudentsList.php: This will contain the elements to create the top right tab in the UI spec – an alphabetical listing of students who have done projects.
- Header.html: The header file is a plain html file that contains the banner proclaiming the institute name and the project title.

These files will be included in the four pages denoting the four tabs – Browse, Search, Edit and Login.

The Listing.php page represents the Browse tab, and displays the listing of projects based on whichever hyperlink was clicked from any of the include tabs, faculty, year or students. The SearchProject.php page, representing the Search page, allows for the user to search for projects based on various criteria. The Edit tab can either be the ManageProjects.php page or the EditProject.php page. The manage projects page allows for searching of projects with an intent to edit or allows for new projects to be created. Both editing and creating a project take the user to the EditProject.php page. Based on the action (edit or create), the page is populated with existing data. The EditProject.php page has a popup page, ManageFiles.php, which allow the faculty member to upload or delete project files. The Login.php page makes up the login tab, allowing the user to login. The link to logout is available in all pages once a user has logged in.

The project file store will not be under the web root, but in a different location specified in a configuration file. In a Unix environment, we have to ensure that the Apache user account has read and write access to this directory.

Once the project is complete, the php pages and associated files can simply be copied over to the target deployment environment, as long as php and MySQL are installed. The deliverable for the PHP/MySQL platform will be a zip file containing the php source.

4.2. ASP.NET Implementation:

The Visual Web Developer in "Whidbey" allows for easy creation of ASP.NET Web Services. "Whidbey" has existing templates for creation of web services in the language of choice. This project will use C# as the preferred language. At a minimum, the following web services will need to be created:

- A service that allows for a user to login. This service will have two operations: Login and Logout. Login accepts a username and password as parameters. The return of this operation will be a generated key (String) that can be used in subsequent Edit requests as the authentication key. The authentication key will be hashed based on a combination of the username and other data from the database that can be re-generated. When subsequent edit requests are made, this key will also be sent, and the other services can regenerate the key and confirm that the user can perform the required action. This concept is similar to the .NET passport authentication service. Logout accepts the key generated from the Login as a parameter.
- A service for getting the project Details. This service has multiple operations – to get project summaries, get project details (given a project Id), and search for projects based on certain criteria.
- A service to save project details. This will have one operation that saves the project details – will create the project if one does not exist, or update an existing project. In addition to the project details to be saved, the operation will also take the authentication key as a parameter. This will be validated against the database to verify that it is valid and the appropriate privileges will be ascertained based on the key.
- A service that serves as a "Master" list – to return any data that is not project specific, but can be considered as master data. For example, an operation to return all the faculty information, committee positions etc.

The created web services can be tested from within the Visual web developer environment itself. Running the service from the developer opens a browser window where the generated service help file exposes the methods (operations) that can be invoked.

As the client for the created web services, an ASP.NET web site will be created. This web site will have the UI features similar to the PHP implementation. If the Visual Web Developer is able to connect to the server hosting the web service in development mode, it can automatically connect to the service and download the WSDL file (this file is stored in the code directory of the project) and create the proxy class. In this case, since the web service and the client both exist on the same machine, "Web services on the local machine" link can be used to generate the proxy class. Once the proxy class is created, it can be instantiated from the web site as any other class, and the web service's methods invoked.

Once the web site is created, the entire project files can either be copied over to an IIS server, or an installer be created for this. A windows installer file that contains the ASP.NET web site will be part of the project deliverable.

5. Principal Deliverables

This project will have the following items as the principal deliverables:

1. A technical report describing in detail the design and implementation of the Master's project website using PHP/MySQL and ASP.NET technologies.
2. A comparative analysis of the strengths and weaknesses of the two solutions, and how they compare against the reference servlet implementation.
3. The installation instructions and source code for the PHP implementation.
4. The installation instructions and source code for the ASP.NET implementation.

6. References

1. Dr. Schreiner, A servlet Case Study, http://www.cs.rit.edu/~ats/talks/ms-rjug/index.html
2. Current Static website, Dr. Schreiner, http://www.cs.rit.edu:8080/ms/static/index.html
3. PHP manual, http://www.php.net/manual/en/preface.php
4. Kevin Yank, Building your own database driven website using PHP and MySQL, http://sitepoint.com
5. MSDN Technical Articles, XML Schemas and data, http://msdn.microsoft.com
6. MSDN Technical Articles, A platform for web services; Web Services: Building Reusable Web Components with SOAP and ASP .NET etc., http://msdn.microsoft.com
7. ASP.NET Whidbey, Overview, Microsoft Corporation, http://msdn.microsoft.com/asp.net/whidbey/default.aspx

7. Approximate Schedule

- February 2004 - Complete proposal and approval process
- March 2004 - Complete coding
- May 2004 - Complete report and project defense.