

Monday, December 12, 2005

# TEAM MANAGEMENT SYSTEM

An addon to Maestro

*Final Project for SEG3202. This team management system was our first attempt creating J2EE applications*

SEG3202  
Software Design

# TEAM MANAGEMENT SYSTEM

## Introduction

The intention of this deliverable is to secure a fine milestone that will lead the project to a workable implementation based on most of the design work from deliverable one. You will find in this document, class diagrams, sequence diagrams, ER diagrams and SQL scripts etc. that supports the current implementation. You will also find attached a zip file that contains the database scripts, html pages, servlet/JSP code, and Java code. Many challenges and obstacles were faced to deliver the implementation before its deadline. Thanks to many efforts by different team members that had experience in this field, these challenges were overcome.

## About the Structure

### *Assumptions*

1. Instructors cannot create a team; rather they can only start up team creation for a specific course they teach. But, they can certainly add members to a created team. This team must have only been created by a team liaison (student)
2. Courses, programs, students' information, instructors and the whole registration process were completely assumed to be predefined in the database. This was basically justified by building tables and inserting data manually to the database. Otherwise the database could have been assumed to be controlled and built by another program or registration system.
3. The instructors can only view the teams of the courses they are teaching. This is clearly justified from the implementations in the code and the database design.
4. Instructors trying to create teams and specify its parameters can only do so for the sections they are teaching. Meaning, they cannot enforce teams creation criteria's to other sections they are not teaching.
5. It was assumed that team creation deadline (specified in functional requirement F1.1) was to be counted by days since the day the team creation was started by an instructor.
6. When all students are deleted from a team by an instructor, the team doesn't exist any more.
7. When students create a team, and they quit that team there are 3 possibilities of what may happen:
8. If he is the only team member, the Team gets deleted
9. If another team member exists, that team member gets liaison privileges.
10. No student can alter student flags.
11. A student can join multiple teams

### *UI Design Implementation*

The TMS was implemented using the J2EE platform using W3C XHTML 1.0 Strict rules to insure browser compatibility. The html layout of the website strictly follows today's web standards. One of the main reasons why we used XHTML is for reusability, flexibility, and maintainability. A **function**

**was added to the top right of the page that changes the style of the webpage**, completely. The JSP pages were designed with beautiful mark up that allowed the User Interface of the webpage to be different by changing only 1 file (or multiple). Tools from W3C proved that this web system follows the XHTML 1.0 Strict tools.

- ☐ Pages on this site are mostly Bobby A approved, complying with most Bobby guidelines. This is always a judgment call; many accessibility features can be measured, but many cannot. I have reviewed all the guidelines and believe that these pages are mostly in compliance.
- ☐ Pages on this site are mostly WCAG A approved, complying with most priority 1 guideline of the W3C Web Content Accessibility Guidelines. Again, this is a judgment call; many guidelines are intentionally vague and cannot be tested automatically. I have reviewed all the guidelines and believe that these pages are mostly in compliance.
- ☐ Pages on this site are Section 508 approved, complying with all of the U.S. Federal Government Section 508 Guidelines. I have reviewed all the guidelines and believe that all these pages are in compliance.
- ☐ All pages on this site validate as XHTML 1.0 Strict. This is not a judgment call; a program can determine with 100% accuracy whether a page is valid XHTML. For example, check the W3C website for XHTML validity.
- ☐ All pages on this site use structured semantic markup. “h1” elements are used for main titles, “h2” and “h3” tags for subtitles.

## ***Class Implementation***

We have used a command pattern approach to do our design. The reason why we used that kind of pattern is simply because we are doing extensive programming. When dealing with action pattern, our design and implementation speed will deduce quickly.

The liaison of the group developed the whole architecture during the first milestone. And most of the team members did their own module.

Therefore each module has its own controller. An action pattern could have been in a role for each controller, but it was already implemented in the earlier developments.

The if-then-else approach leads to a very large service method, which is difficult to read and still more difficult to maintain. A better approach is to use the Command pattern.

## ***Data Design***

We custom made our DAO object but without any DataAccessObjects. We only implemented the DataSource and the Buisness Object Layer. To make our organization simple, we implemented our own executeSelectQuery() function which allowed all the queries to be returned by ArrayList of hashes. s

Later on the course we learned about the DAO Object approach. Made work much simpler, but since

we are in a time constraint, we didn't implement it. A description of ER diagrams, SQL database tables and how they relate to the domain elements follows.

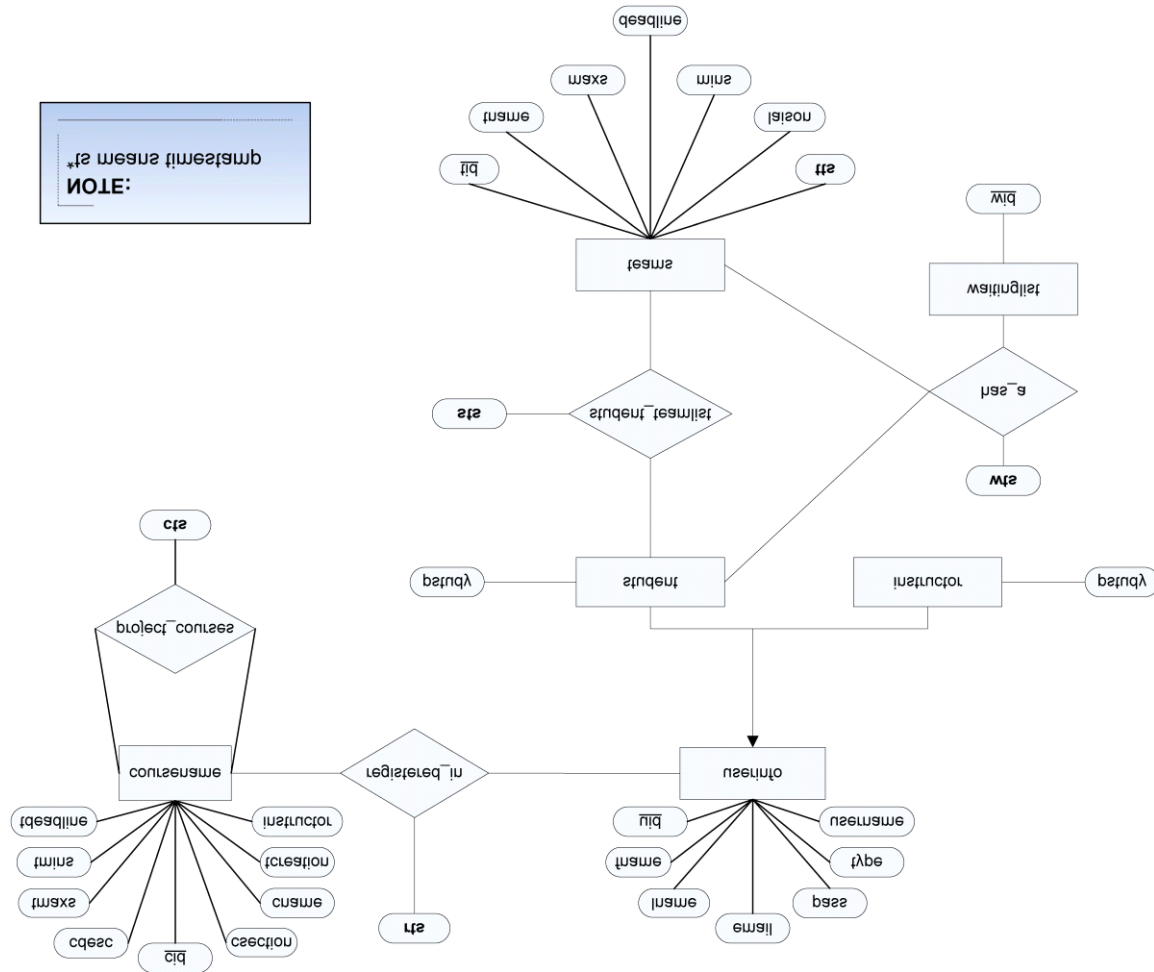


Figure 1  
The ER diagram

### There are 6 entities:

1. *course*, which stores the information about the specific course which are:
  - a) *tdeadline*, the team creation deadline.
  - b) *tmins*, the minimum number of students allowed in a team.
  - c) *tmaxs*, the maximum number of students allowed in a team.
  - d) *cdesc*, the course description.
  - e) *cid*, the id of this course which will identify this course uniquely.
  - f) *csection*, the course section.
  - g) *cname*, the course name.
  - h) *creation*, a Boolean to indicate whether team creation is allowed for this course.
  - i) *instructor*, the instructor who teaches this course.

2. **userinfo**, which stores the information about the students and the professors which are:
  - a) uid,, the id of this person which will identify this course uniquely.
  - b) fname, the person's first name.
  - c) lname, the person's last name.
  - d) email, the person's email.
  - e) pass, the person's password
  - f) type, 0 indicates the person is a student, 1 indicates he/she is an instructor.
  - g) username, the person's username on the system
3. **student**, which inherits userinfo and has pstudy as an attribute which indicates the program of study.
4. **instructor**, inherits userinfo as well and has pstudy as an attribute which indicates the program he/she teaches in.
5. **teams**, the teams that indicate certain number of students working together on the project, and has the following attributes:
  - a) tid, id of this team which will identify this team uniquely
  - b) tname, a name that specifies this team.
  - c) maxs, maximum number of students in this team.
  - d) mins, minimum number of students in this team.
  - e) liaison, the liaison student of this team(leader).
  - f) tts, team timestamp which indicates when was the team created.
  - g) deadline, which is a timestamp that indicates the team creation deadline.
6. **waitinglist**, a list which contains the students that aren't registered in a team.

There are also 4 relations:

1. **project\_couses**, which connects coursenam with itself.
2. **registered\_in**, which connects coursenam with userinfo.
3. **has\_a**, which connects waitinglist with student and teams
4. **student\_teamlist** which connects student with teams.

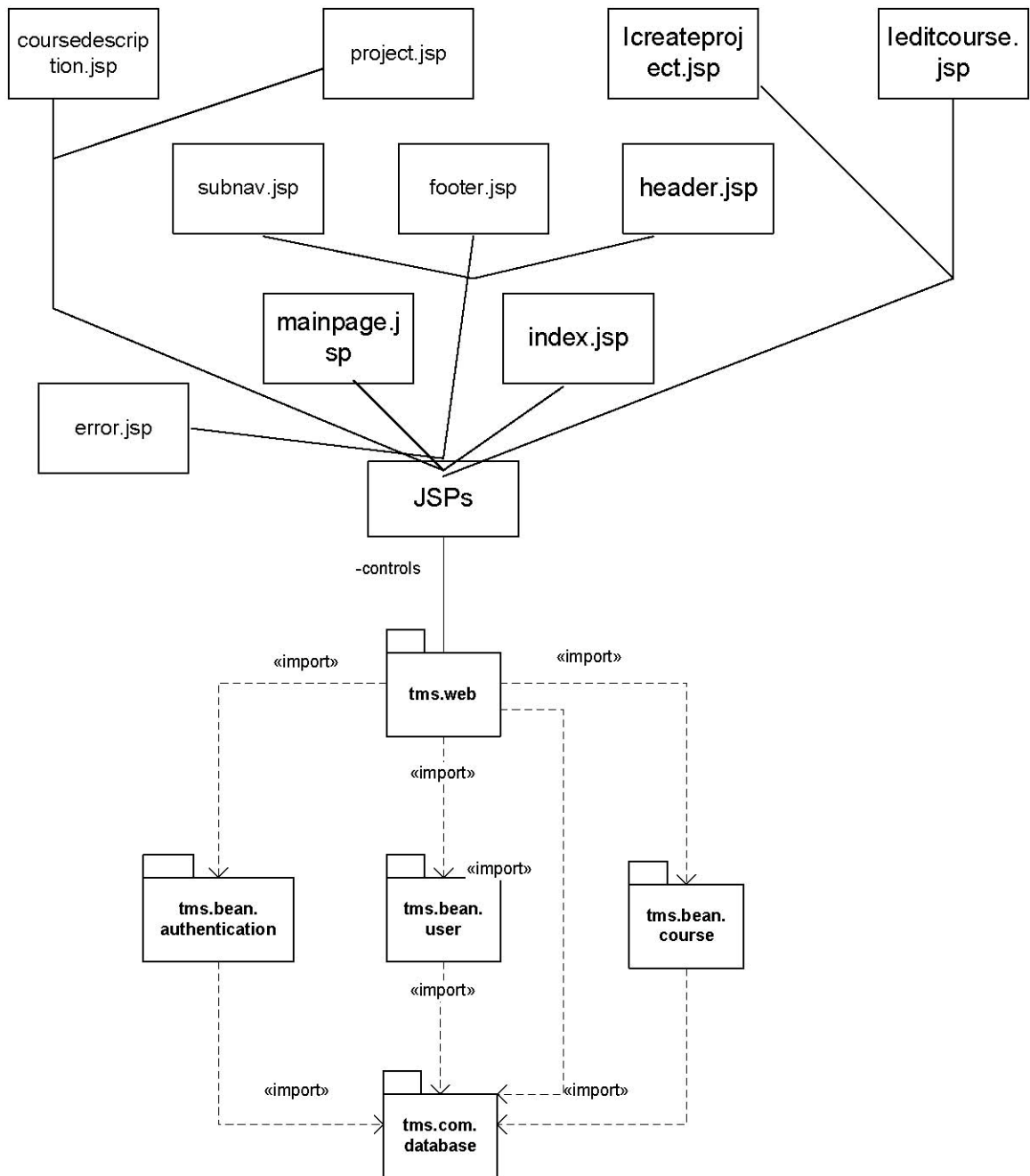
Note: There were timestamps on many relations and entities to be used in printing the time a row was created or the time an action was taken.

### ***Sample Data***

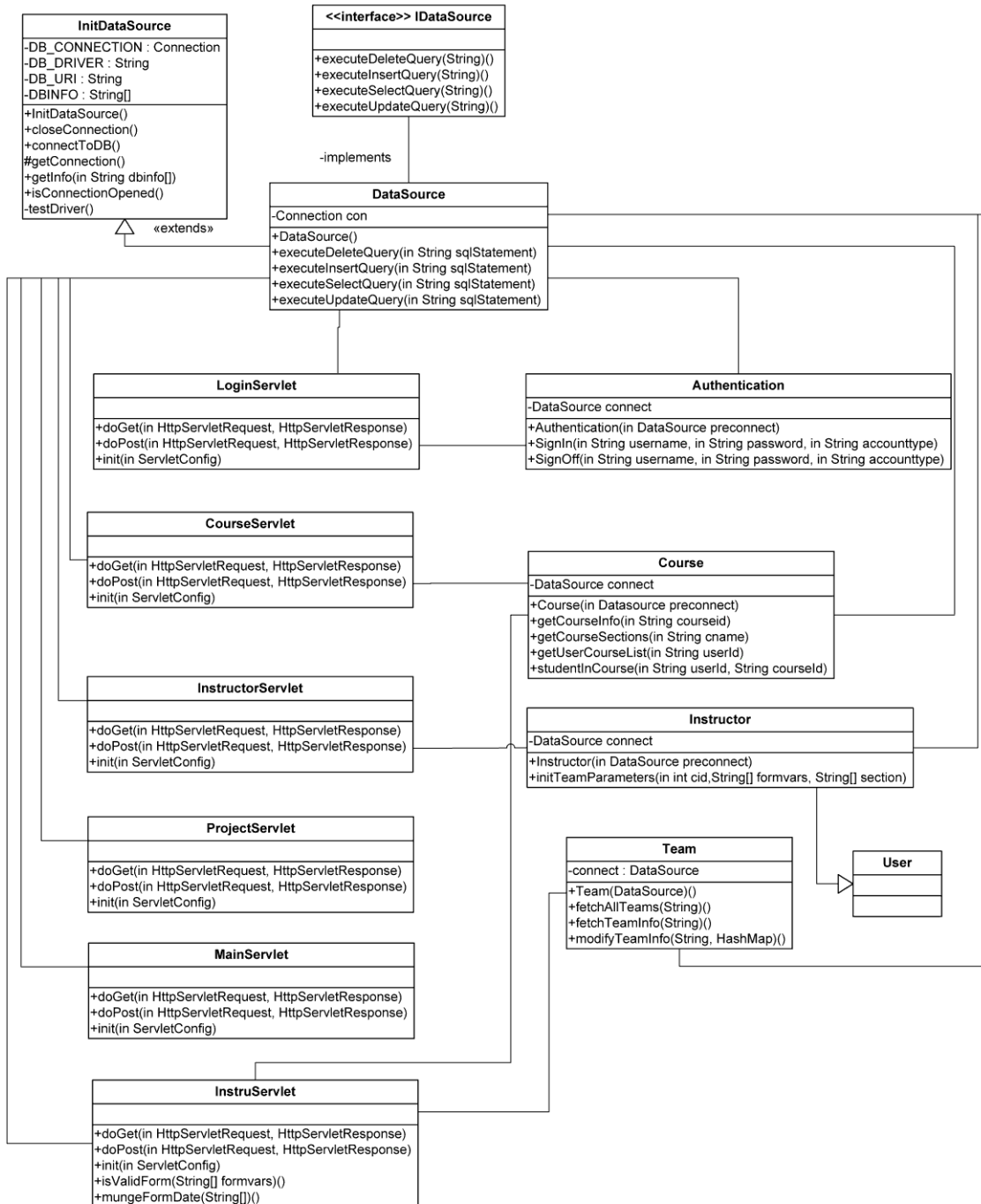
Please Import the file included within this zip, by using the mysql dump feature. The file has the tables and the new insert statements.

## Class Design

### Package Diagram.



### Class Diagram.



## Illustration

The following sequence diagrams will help illustrating the implementation of the following three use cases.

### *Use Case: Instructor logs in*

Title: Instructor log in

Scope: Design

Level: user

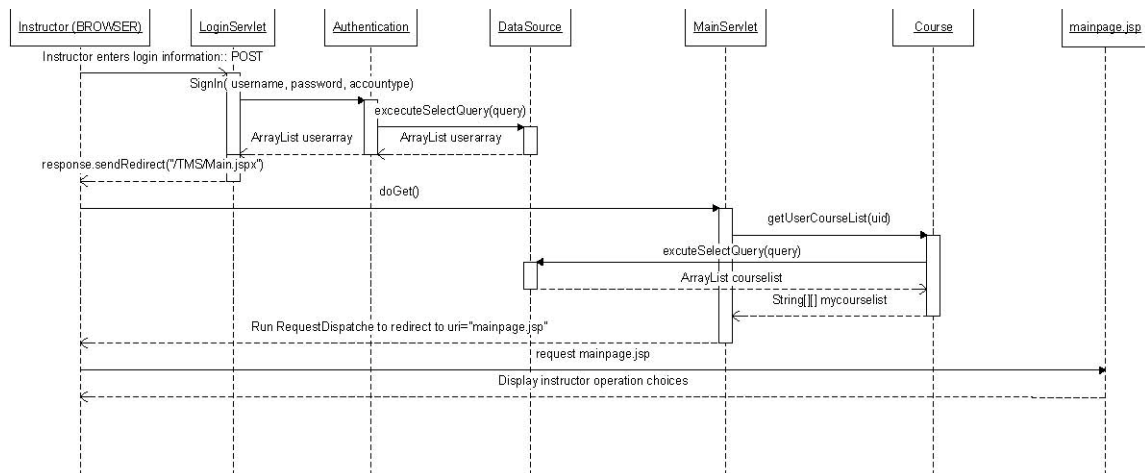
Primary Actor: Instructor

Goal: An instructor want to identify himself to the system in order to use it's functions

Precondition: TMS is ON

Postcondition: Instructor is logged in

1. Instructor enters login information
2. TMS checks Instructor authorization status
3. TMS displays instructor operation choices
  2. a. Instructor is not authorized to use system
  2. a. 1. TMS displays unauthorized access error message



### ***Use Case: Set up parameters***

Title: Set up parameters

Scope: Design

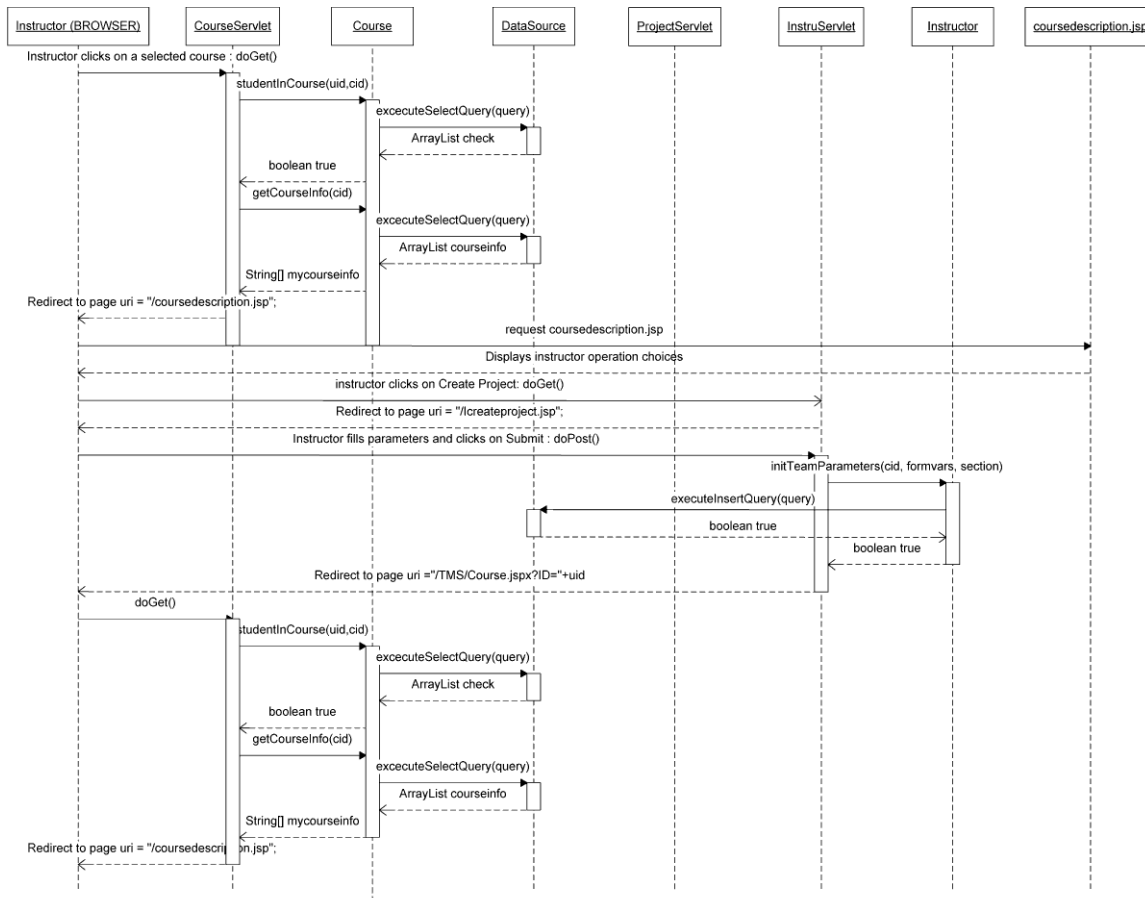
Level: user

Primary Actor: Instructor

Precondition: TMS Display is instructor operation choices AND Instructor is logged in

Postcondition: Teams can be started up

1. Instructor chooses to set up teams parameters
2. TMS asks for teams parameters
3. Instructor provides parameters
4. TMS validates parameters
5. TMS displays teams start up acknowledgment message
6. AFTER 60 sec, TMS displays instructor operation choices
  4. a. Teams Parameters are not valid
    4. a. 1. TMS displays teams parameters error message
    4. a. 2. AFTER 60 sec, GOTO step 2



### ***Use Case: Visualize students teams***

Title: Visualize students teams

Scope: Design

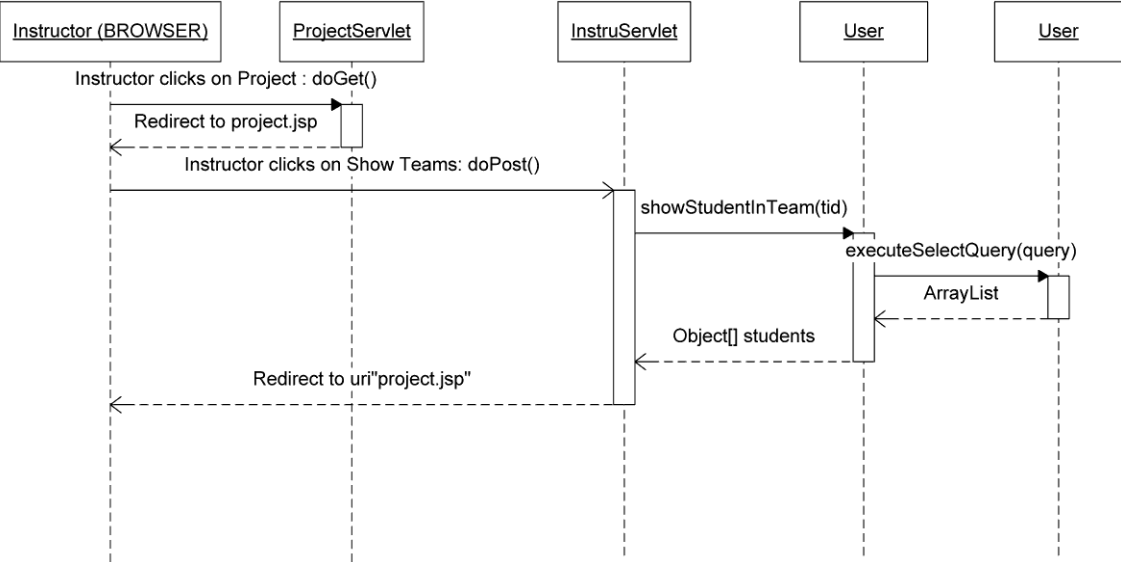
Level: user

Primary Actor: Instructor

Precondition: TMS Display is instructor operation choices AND Instructor is logged in

Postcondition: TMS Display is list of teams

1. Instructor chooses to visualize students teams
2. TMS displays list of all teams



## Discussion

Major design issues, alternatives that were considered and abandoned are discussed below.

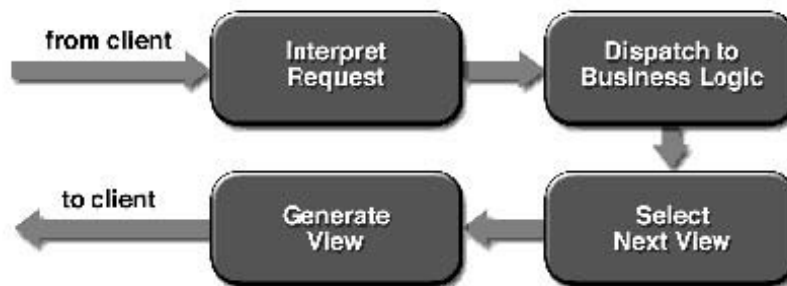
### *Separating Business Logic from Presentation.*

**Usage of Templating.** A template is a presentation component that composes separate subviews into a page with a specific layout. Each subview, such as a banner, a navigation bar, or document body content, is a separate component. Views that share a template have the same layout, because the template controls the layout.

Using templates in an application design centralizes control of the overall layout of pages in the application, easing maintenance. Changing the layout in the template file changes the page layout for the entire application. More importantly, the individual subviews (like the "Navigation Menu" in Figure 4.8) are used by reference in the template instead of by copy-and-paste. Therefore, changing a subview means changing a single source file instead of changing all the files in which that subview occurs.



**The Web-Tier Service Cycle.** A J2EE application's Web tier serves HTTP requests. At the highest level, the Web tier does four basic things in a specific order: interprets client requests, dispatches those requests to business logic, selects the next view for display, and generates and delivers the next view. This service cycle is clearly in our implementation, the beans (Authentication, Instructor, Course) being the business logic and the servlets (CourseServlet, InstructorServlet, LoginServlet, MainServlet, ProjectServlet ) and jsps (coursedescription, error, header, footer, Icreateproject, Ieditcourse, index, mainpage, project, subnav) interprets, selects and generates the next view.



**Implementaion with EJB's.** The implementation with EJB's was first considered in the project and effort was made to do so. But, due to lack of time and a realistic learning curve that wont meet the deadline this approach was avoided till later phases.

**Using a complete Model-View-Controller approach.** We used many aspects in our design that contributes to an MVC model. We organized an interactive application into three separate modules: one for the application model with its data representation and business logic, the second for views that provide data presentation and user input, and the third for a controller to dispatch requests and control flow.