**Software Design Document**

MCS 593 Tutorial Management Web Application

**Prepared By:**

**Alec Meyer  
Robert Padilla  
Anthony Prejean  
James Kubena  
Britton Ware**

1. Purpose

* The purpose of this document is to describe the implementation of the MCS 593 Tutorial Management Web Application described in the software requirements.

2. Scope

* This report will define the high level design and technology decisions of the MCS 593 Tutorial Management Web Application
* This SDD defines and describes the use of each view, the architectural constraints of the system, the functional requirements with a significant impact on the architecture, use-case realization, the layers and subsystems of the application, performance issues and constraints.

# 3.1 SDD within the Waterfall life cycle

* In the design part of our waterfall model life cycle we define our SDD. After determining the requirements and specifications, as the cycle goes through the design part we make a prototype of our software design. Before implementation it is vital to design the software to be more controlled on the coding part. If there is a problem in maintenance about the design part we can go back and fix it.

# 3.2 Purpose of an SDD

* The purpose of this SDD is to define and describe the use of each view, the architectural constraints of the system, the functional requirements with a significant impact on the architecture, use-case realization, the layers and subsystems of the application, performance issues and constraints.

4. Design Description Information

# Introduction

* The Software Design Descriptions (SDD) provides an architectural overview of MCS 593 Tutorial Management Web Application. This document presents to developers different types of abstraction. It aims to provide the developers with a clear understanding of the system.

## Scope

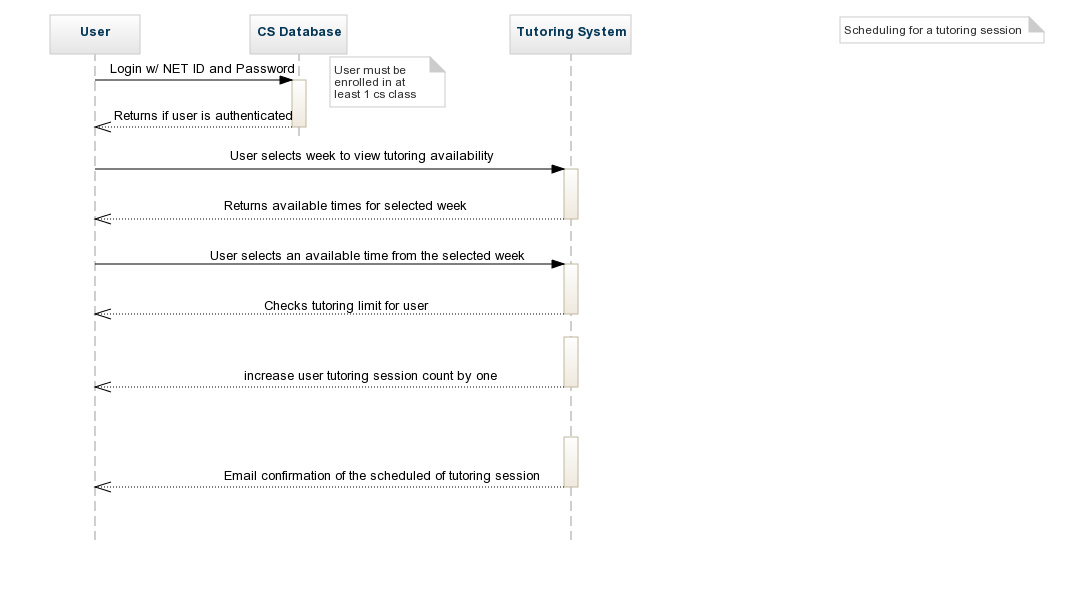
* The scope of this SDD is to define high level design and technology decisions of the MCS 593 Tutorial Management Web Application

# Architectural Representation

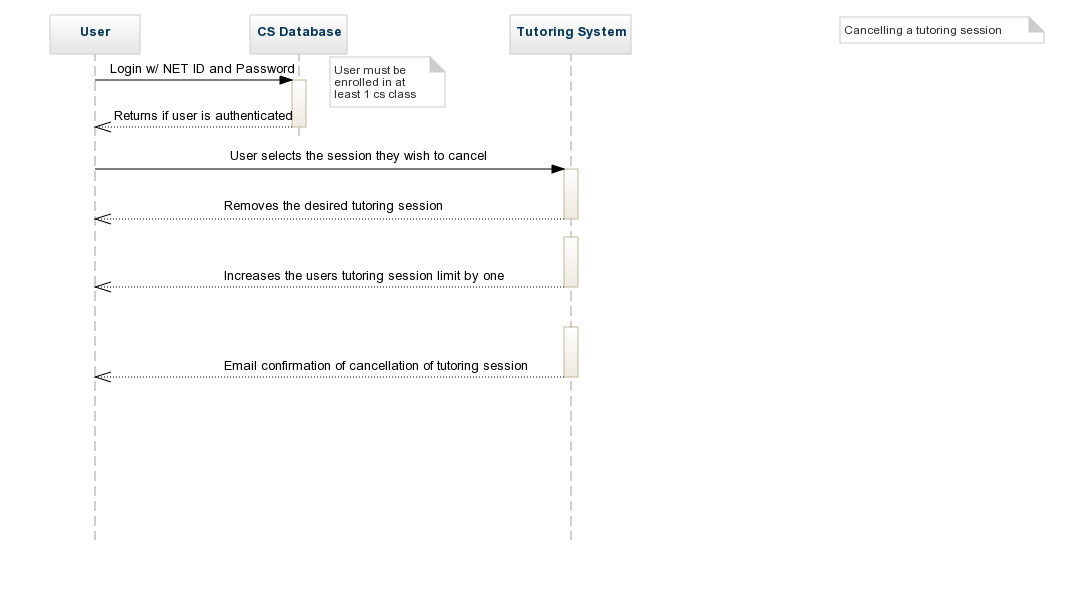
* The views used to document the MCS 593 Tutorial Management Web Application

**Use Case view**

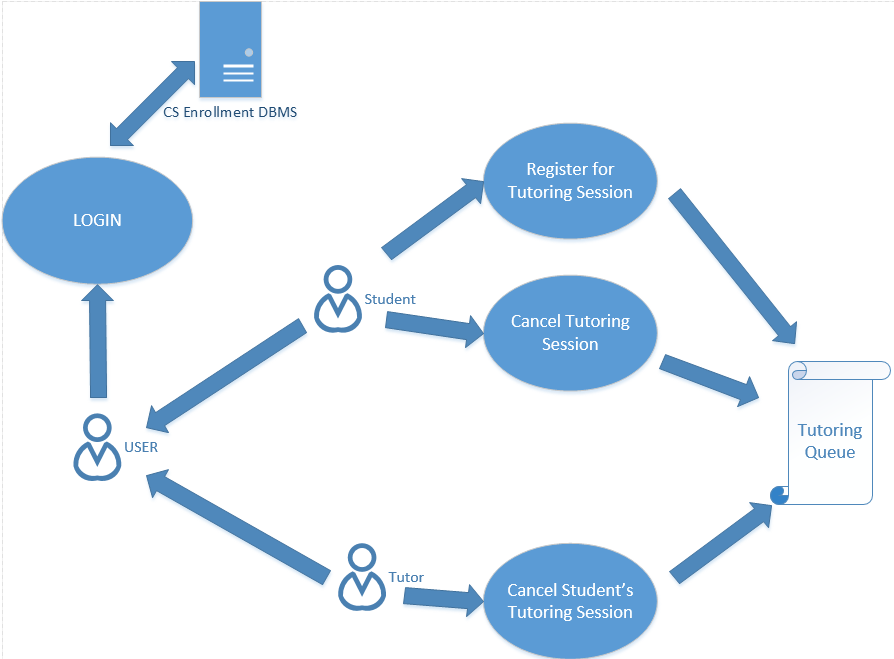
* + 1. Description of Use-Case view of this software architecture. This view is important to the tutoring system as it goes through the different types of scenarios the end-user and tutors will experience when interacting with the software system.
    2. Login
    3. Tutoring session scheduling



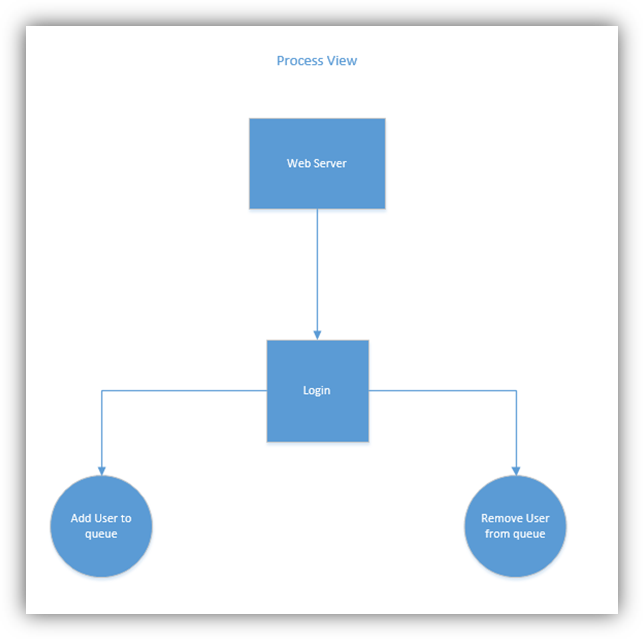
* + 1. Tutoring session cancellation



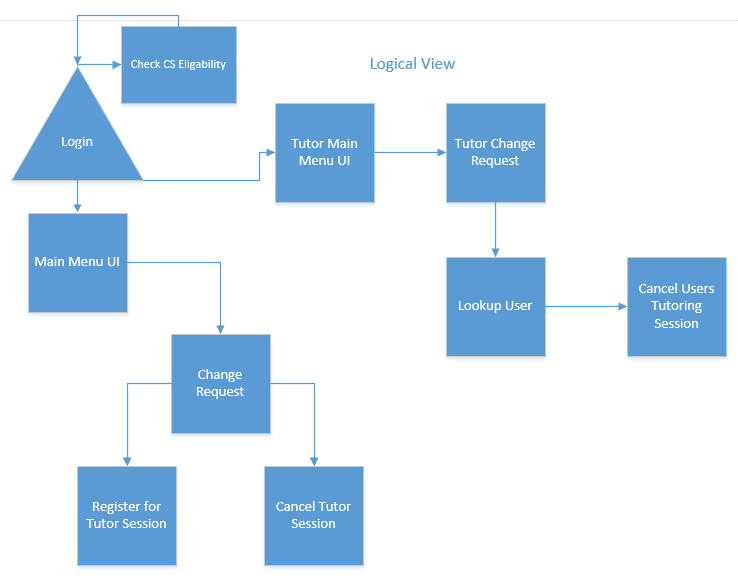
* + 1. These use cases for this system will be initiated by the students. Tutors will have access to interact and cancel student’s sessions.



**Process View**



**Logical View**



# Architectural Goals and Constraints

This section describes the software requirements and objectives that have some significant impact on the architecture.

* + 1. Some key requirements both functional and nonfunctional have important bearing on the architectural design. For example:
    2. System must be able to accept data formats of legacy DBMS of CS students.
    3. Systems must have a stable internet connection.
    4. The Tutoring System must ensure protection of data from unauthorized users.
    5. The Tutoring System will be implemented as a client-server system. The server system will reside in Texas State’s Datacenter and the client’s will supply devices to connect to the web interface.

# Size and Performance

* Targeted member count: All CS Students
* Daily user access: ??

# Quality

**Scalability**:

* **Description** : System’s reaction when user demands increase
* **Solution** : System is built to support ??% of CS students, further scalability is supported by new hardware and J2EE

**Reliability**, **Availability**:

* **Description** : Transparent failover mechanism, mean-time-between-failure
* **Solution :** : J2EE application server supports load balancing through clusters, RAID mirroring is used diminish software crashes.

**Portability**:

* **Description** : Ability to be reused in another environment
* **Solution :** The system is fully J2EE compliant and thus can be deployed onto any J2EE application server

**Security**:

* **Description** : Authentication and authorization mechanisms
* **Solution :** J2EE native security mechanisms will be reused

# Risks and Mitigation Plans

## Software failure

* Development/Operational/Database server fails

Recover using mirror RAID disks

* Development workstations fail.

Use spare workstations if not fixed in short time.

**5. Interface description**

**5.1. Scope**

Here in this part we will consider 2 main parts. One is the structure and navigation that describes the structure of the application and the ways in which users can navigate and the other is the screen descriptions which comprises the main part of the document.

**5.2. Use**

The graphic user interface (GUI) includes the detailed visual characteristics of every component of the graphic interface and the functional sequence of interactions over time. There are some criteria that we are going to apply to our web application in order to have effective, efficient, easily understandable and usable interface.

**Visibility of system status**

The system should always keep users informed about what is going on, through appropriate feedback within reasonable time.

**Match between system and the real world**

The system should speak the users' language, with words, phrases and concepts familiar to the user, rather than system-oriented terms. Follow real-world conventions, making information appear in a natural and logical order.

**User control and freedom**

Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue. Support undo and redo.

**Consistency and standards**

Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

**Error prevention**

Even better than good error messages is a careful design which prevents a problem from occurring in the first place. Either eliminate error-prone conditions or check for them and present users with a confirmation option before they commit to the action.

**Recognition**

Minimize the user's memory load by making objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

**Help users recognize, diagnose, and recover from errors**

Error messages should be expressed in plain language (no codes), precisely indicate the problem, and constructively suggest a solution.

**Help and documentation**

Even though it is better if the system can be used without documentation, it may be necessary to provide help and documentation. Any such information should be easy to search, focused on the user's task, list concrete steps to be carried out.