**AbOut**

MTM Program Product

Software Design Description

*Version 1.0*

*03/01/2013*

*Applying Montana Tech SDD Standard Version 3.0*

**Project Client:** *Celia Schahczenski*

**Project Manager:** *Frank Ackerman*

**Project Team:** *Foster Cade, Jeff Hall, Clint Hillerman, Jake Jones, Brian D. Knopp, Matt Morris, PJ Neary, Tom Powell, Frank Scholey, Alex Thompson, Logan Warner.*

**Document Authors:** *Cade J. Foster, Brian D. Knopp*

Standard Version Number: 3.0

Standard Version Date: April 5, 2012

|  |  |  |  |
| --- | --- | --- | --- |
| *Version* | *Date* | *Authors* | *Comment* |
| 1.0 | 03/01/2013 | Cade J. Foster, Brian D. Knopp | Initial version. |

**Montana Tech Software Engineering Students:**

These Montana Tech Method software engineering standards encapsulate Dr. Ackerman’s decades of experience in the software industry, the IEEE software engineering standards, and many suggestions from various texts. They have gone through many revisions and additions over the last several years. They are part of your software engineering studies so that (1) you may have the experience of developing software to a standard (which you may find you need to do if you take a job that requires high quality software), and so that (2) you will have the experience of developing high quality software. You are also invited to participate in the continuing evolution of these standards by studying them critically and making suggestions for their improvement and correction.

Table of Contents

1 Introduction 4

1.1 Purpose 4

1.2 Scope 4

1.3 Definitions, Acronyms, and Abbreviations 4

1.3.1 Definitions 4

1.3.2 Acronyms and Abbreviations 4

1.4 References 4

2 Design Background 4

2.1 Background Information 5

2.2 Alternative Designs Considered 5

3 User Characteristics 5

4 Design Overview 6

5 Data Architecture 7

5.1 Data Analysis 9

5.2 Output Specifications 10

5.3 Logical Database Model 10

5.4 Data Conversion 10

6 Design 10

6.1 Description for Component n **Error! Bookmark not defined.**

6.1.1 Processing for Component n 11

6.1.2 Component n Interface Description 11

# Introduction

The purpose of AbOut’s design is to help organize the drafting of the code for the software in a fluent, logical, seamless, efficient, and timely manner. It is therefore intended to be used by the programmers. The scope of AbOut’s design can be found in AbOut’s SRS document. AbOut is a software intended to be used by Abet accredited colleges or universities as a way to generate report information on the Abet outcomes that are and are not being met by the school. This version was specifically written to be used by Montana Tech of the University of Montana, but a more widespread use of the product is a hopeful anticipation.

## Purpose

In order to facilitate a common and consistent design during the implementation, refinement, and maintenance of AbOut, the proscriptions contained in this document are to be strictly followed. The intended audience of this document is the programmers and maintainers of the AbOut system.

## Scope

See *AbOut SRS.*

## Definitions, Acronyms, and Abbreviations

### Definitions

|  |  |
| --- | --- |
| Abet | Accreditation Board for Engineering and Technology, Inc; the accrediting body for the Computer Science and Software Engineering programs. |

### Acronyms and Abbreviations

|  |  |
| --- | --- |
| CAS  CORE  SDD  SRS | Central Authentication Service  Course Outcome Review and Evaluation  Software Design Description  Software Requirements Specification |

## References

None.

# Design Background

## Background Information

**Stakeholders:**

Cade Foster cjfoster@mtech.edu 406-465-9986

Celia Schahczenski cschahczenski@mtech.edu 406-496-4383

Damian Valles dvalles@mtech.edu 406-496-4408

Frank Ackerman fackerman@mtech.edu 406-496-4858

Jeff Braun jbraun@mtech.edu 406-496-4206

Keith Vertanen kvertanen@mtech.edu 406-496-4385

Melissa Holmes mholmes@mtech.edu 406-496-4408

Michelle Van Dyne mvandyne@mtech.edu 406-496-4855

Tamara Windham twindham@mtech.edu 406-496-4366

As AbOut will be used to compile reports based upon student grades, which are indefinitely stored in a database, the system needs to be secure against intrusions or unauthorized access. Any actions or security flaws which jeopardize the confidentiality of student grades should be identified and rectified.

Access to AbOut depends on CAS, the central authentication system, and the interface between said entities.

AbOut utilizes a MySQL database to store information relating to administrators, faculty members, students, and student grades. Additionally, PHP is necessary to serve the dynamically changing contents of the AbOut web-based system.

## Alternative Designs Considered

An alternative to a web-based user interface was a downloadable client coded in C#. The decision to reject this interface in favor of the current HTML and PHP-based system was one of pure utility. A web interface is significantly more accessible and portable in the current age of mobile applications.

# User Characteristics

Potential users of this system include the faculty and staff of the computer science department, in addition to the administrative assistant. The assumed level of expertise needed to operate the system is a familiarity with web browsers and dynamic form webpages.

The initial webpage that a user will view is the login prompt, interfacing with CAS and providing a secure method to authenticate user credentials. The various pages which the user may access after login satisfy the requirements of faculty and admins adding, removing, modifying, and compiling assessments; reports may be generated in a dedicated tab as well.

# Design Overview

The architectural design of AbOut is that it contains reusable modules. All essential files are located in their own logical 'units'. These files are located in the 'includes' directory and are named as follows: header.php, navBar.php, and footer.php. The default faculty and admin views (as accessed on login) and the reports view include these files at their logical locations. In code, which is to be tested, these files can be included as follows:

include( '../includes/header.php' );

include( '../includes/navBar.php' );

...

YOUR CODE HERE

...

include( '../includes/footer.php' );

All entities and their interdependent relationships can be seen in the following Schema:



# Data Architecture

A database system is required by AbOut to store faculty, staff, and student information, as well as courses and their accompanying assessments. The table architecture of the database is organized as follows:



**Field descriptions:**

Program: pAbbrev: Program abbreviation.

Outcome: oTest: Outcome text.

pAbbrev: Program abbreviation.

endSem: Ending semester.

beginSem: Beginning semester.

AssessOutcome: asessId: Assessment identification.

oAbbrev: Outcome abbreviation.

Assessment: asessId: Assessment identification.

description: Assessment description.

maxPoints: Maximum number of points.

offeringId: Offering identification.

CourseOutcome: cAbbrev: Course abbreviation.

oAbbrev: Outcome abbreviation.

OffOutcome: offeringId: Offering identification.

oAbbrev: Outcome abbreviation.

Score: asessId: Assessment identification.

studentId: Student identification.

Score: Score.

Course: cAbbrev: Course abbreviation.

csFlag: Computer Science flag.

seFlag: Software Engineering flag.

beginSem: Beginning semester.

endSem: Ending semester.

Offering: offeringId: Offering identification.

sectionNo: Section number.

fName: Faculty name.

userId: User identification.

cAbbrev: Course abbreviation.

semesterId: Semester identification.

Semester: semesterId: Semester identification.

semester: Semester.

Student: studentId: Student identification.

sName: Student name.

offeringId: Offering identification.

User: userId: User identification.

usernameCAS: Username on Central Authentication System.

uName: Username.

activeFlag: Active flag.

## Data Analysis

The data analysis activities that AbOut uses are the generation of reports. The data being analyzed is the student scores on the assessments, the max possible score on the assessments, the outcomes in the system, and the offerings in the system which consist of course data and semester data. This data is being compared against the percent required for a student to achieve on an assessment to be considered to have passed that assessment. The outcomes are tied to assessments by the faculty creating the assessments and scores are entered for the students on those assessments. All of this information is pulled from the database, manipulated in code, and output for the various users to see by generating various reports. The SRS document states that in order to change the percent required for students to achieve on an assessment to be considered to have passed the assessment, that change must be done directly in code. Therefore, changing this percent will alter the system design because that percent is going to be compared against when generating the reports.

Stored procedures are used to generate data. The procedures are all in the file storedProc.sql and live in the AbOut database. If changes are made to this file, place those changes into the db by going into mysql using the AbOut db and entering:

mysql> source storedProc.sql

To call one of the procedures:

mysql> CALL procedureName(parameters);

To see a procedure's definition:

mysql> SHOW CREATE PROCEDURE procedureName;

(However it is easier just to look directly at the file.)

To call this from php:

$stmt = $DB\_Read->prepare("call procedureName(parameters);");

The results will be returned as usual.

The procedures are:

### semesters

\* semesters - Take one or two semester ids and return the

\* semester names

\* Parameters:

\* semesterId1

\* semesterId2

\* Returns:

\* semName1 - name of first semester or NULL if illegal semester id

\* semName2 - name of second semester or NULL

### outcomes

\* outcomes - Returns outcome abbreviations and text for those outcomes

\* within a program and in a semseter range.

\* Parameters:

\* program abbreviation

\* semesterId begin

\* semesterId end

\* Returns a set of:

\* oAbbrev - abbreviation for outcome

\* oText - text of outcome

\*

### partialOutcomes

\* partialOutcomes - Returns outcome abbreviations, beginSem name and

\* endSem name for those outcomes which are

\* only active for a portion of the semester

\* range.

\* Parameters:

\* program abbreviation

\* semesterId begin

\* semesterId end

\* Returns a set of:

\* oAbbrev - abbreviation for outcome

\* semName1 - name of semester when the outcome became active

\* semName2 - name of the last semester when the outcome was active,

\* or null

### courses

\* courses - Returns course abbreviations for those courses

\* within a program and in a semseter range.

\* Parameters:

\* program abbreviation

\* semesterId begin

\* semesterId end

\* Returns a set of:

\* cAbbrev - abbreviation for course

\*

### partialCourses

\* partialCourses - Returns course abbreviations, beginSem name and

\* endSem name for those courses which are

\* only active for a portion of the semester

\* range.

\* Parameters:

\* program abbreviation

\* semesterId begin

\* semesterId end

\* Returns a set of:

\* cAbbrev - abbreviation for course

### calculatePercent

\* calculatePercent - Given an outcome, course, section number and semester range,

\* return the percentage of students who took

\* the offerings in the course in the range and passed the

\* outcome.

\* Parameters:

\* oAbbrev - outcome

\* cAbbrev - course or ALL

\* sectionNo - section number or 'AL' for all sections

\* beginSemId - beginning semester in range

\* endSemId - end semester or NULL

\* Returns:

\* percent - percentage of students who passed this outcome

\* noStud - number of students/offerings assessed, that is,

\* a student in 3 offerings is counted 3 times

## Output Specifications

Separate PHP scripts will be created which support the reports generation and viewing functionality of AbOut. The database is to be queried by these scripts when the query is considered complicated, is used frequently in the system, or both. Otherwise the database is queried directly with SQL queries to access the necessary and relevant information to compile an outcome, CORE, or overview report.

## Logical Database Model

The specific data elements and logical data groupings that are stored and processed by the design components in section 6 can be seen in the AbOut DB Schema in section 5 of this document.

**Data Flow Model:**

All data elements connected to the programs, courses, outcomes, and semesters in the system are entered directly by the administrator through the use of various forms that make it convenient to enter such information efficiently. These forms can be designed however they need so that it is logical and user friendly. Because of these lack of specification on how these forms need to appear, their design is beyond the scope of this document and will be left up to the developers, as designing them in any way would impose implementation restrictions that are unnecessary for this document to suggest. Student information can be entered directly by admin or faculty/staff, as well as assessment information, offering information, and score information. Percentages output by the reports are obtained by querying the database on this entered information and performing algebraic operations on the data.

## Data Conversion

Because there are no design constraints (see AbOut SRS), there are no design requirements applicable to data conversion activities.

# Design

**Login:** this component will work with the CAS to validate the user’s authenticity and permission to use the system. It will also determine whether the user is an admin, or a faculty/staff member. Since the validation process is done through the CAS it is beyond the scope of this document. However, all relevant user information (uName, usernameCAS, activeFlag, userId) will be determined after Login and used in the system throughout the duration of time that they user remains on the system. This will determine which offerings (and therefore assessments tied to those offerings, and all information contained therein) will be viewable to the user (faculty and staff can only view offerings that they are teach), and admin can see all offerings.

**Reports:** depending on the report being generated, this component will work with various data elements, as described below.

**Overview Report:** this report will need to work with all of information that has any connection to a program or a semester, which includes all semester data, program data, outcome data, course data, offering data, assessment data, and score data.

**CORE Report:** this report will need to work with all offering data and semester data, including all assessment data and score data.

**Outcome Report:** this report will need to work with all data associated with an outcome and a semester, including all outcome data, semester data, program data, course data, offering data, assessment data, and score data.

**Main Page:** this component will use the information obtained from the Login component. Its main function is to facilitate a way to navigate between the various components, and offers a way to view different information based on the information obtained from the Login component.

## Login

*Procedure* to facilitate authorized access to AbOut by registered faculty or administrators. Serves a standard login page interfacing with CAS to provide consistent and reliable login services.

### Processing for Login

Retrieve the username and password of a prospective user through PHP forms. Interface is created with CAS to authenticate the entered credentials of the user. If the login attempt is deemed permissible by CAS, the user is redirected to the main page of AbOut. If the credentials are invalid, CAS login will fail.

### Login Interface Description

A button serves to redirect the user to the CAS login page. From here, the user is required to enter their username and password in separate fields. CAS authentication proceeds with either a redirection to the AbOut main page or a notice of login failure.

## Main Page

*Object* serving as a central hub to access faculty, admin, and reports pages. These pages are accessed by their respective tabs in the central menu-driven interface.

### Processing for Main Page

The Main Page simply serves as a means of centralized access for faculty, admin, and overview reports, through hypertext links. No significant data entry or processing occurs in this object, aside from the determination of a user’s faculty/admin status.

### Main Page Interface Description

Upon login to AbOut, the user is redirected to the main page, which will automatically show the appropriate interface tab for the user’s type (faculty see course offerings which they teach, etc…). A user may navigate to the other data views (faculty, admin, reports) using the tabs at the top of the courses list.

**Faculty**

Faculty may access the course offerings which they are assigned by clicking on the hyperlink of the name of their courses in the courses list. The main viewing pane will then display the relevant information for a course, including associated outcome and students. Faculty may add students to their course offerings, add assessments, and score assessments. A CORE report may also be generated in the main viewing pane.

**Admin**

Administrators have the ability to select a course from a list of all courses and offerings. The administrator may associate outcomes with a specific course, change which semester a course is offered in, and add course offerings. A course offering may then be associated with its respective faculty member.

Course outcomes and their associated text may also be added or edited by the administrator in the latter subsection of the list panel.

The administrator may also add and edit user information by changing the name and CAS login name of the user.

**Reports**

A user is able to generate course outcome, overview, and CORE reports, which summarize the various course data necessary to compile an Abet assessment. Which report is desired is selected from the list under the reports tab. Different properties of the generated report are configurable via drop-down boxes. Once the report is configured, the ‘Generate Report’ button may be used to produce the desired report.