***StOut***

**MTM Program Product**

**Software Design Description**

Version [*0.3*]

Issue Date: *January 9th, 2018*

**Project Director:** *Jeff Braun*

**Project Manager:** *Trevor Brooks*

**Project Team:** *Jesse Anderson, Trevor Brooks, Nathaniel Lewis, Abdulrahman E. Alduraiweesh*

**Document Authors:** *Jesse Anderson, Trevor Brooks, Nathaniel Lewis, Abdulrahman E. Alduraiweesh*

**A Montana Tech Method Standard**

|  |  |  |  |
| --- | --- | --- | --- |
| *Version* | *Date* | *Authors* | *Comment* |
| 0.1 | 1/9/2018 | Nathaniel Lewis | First draft |
| 0.2 | 5/3/2018 | Nathaniel Lewis | Updating info, filling more fields |
| 0.3 | 5/4/2018 | Nathaniel Lewis | Database info, section 6 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**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**

[**Introduction**](#_3znysh7) **4**

[Purpose](#_2et92p0) 4

[Scope](#_3dy6vkm) 4

[Definitions, Acronyms, and Abbreviations](#_1t3h5sf) 4

[Definitions](#_4d34og8) 4

[Acronyms and Abbreviations](#_2s8eyo1) 6

[References](#_17dp8vu) 7

[**Design Background**](#_26in1rg) **7**

[Background Information](#_lnxbz9) 7

[**User Characteristics**](#_1ksv4uv) **7**

[**Design Overview**](#_psvujuifn7mu) **8**

[**Data Architecture**](#_2jxsxqh) **8**

[Data Analysis](#_z337ya) 9

[Output Specifications](#_3j2qqm3) 9

[Logical Database Model](#_1y810tw) 9

[Data Conversion](#_4i7ojhp) 13

[**Design**](#_2xcytpi) **13**

[Description for Front End](#_zbdtqc3g5vlf) 13

[Processing for Front End](#_2bn6wsx) 13

[Front end Interface Description](#_rbh2y8yr6i1) 13

[User Interface design](#_pi5pk7hez4) 13

[HTML structure](#_lifijpjsph0q) 13

[Navigation](#_z4nnxuw5rwu2) 14

[Forms](#_w37owfbp6xxj) 14

[Description for Back End](#_wuvm08gow4dh) 14

[Processing for Back End](#_e1ah6dpd1bl2) 14

[Back end Interface Description](#_w37qhn9u9jfe) 14

# Introduction

## Purpose

Through writing this document, we hope that we will be able to make it easier for future students, staff, and others that work on the Student Outcome (StOut) web application to understand our design ideas and principles.

## Scope

For scope and purpose information on the software, please refer to the Software Requirements Sheet included, filename SRSStOutvX\_X\_X.docx. Where “X” are placeholders for the version number.

## Definitions, Acronyms, and Abbreviations

### Definitions

|  |  |
| --- | --- |
| **Administer** | Member in the MTECHS domain who is registered in the StOut system and has been assigned the administrator role for the database. |
| **Analytical Data** | Data that is no longer editable, and has had all sensitive information about students (names etc.) stripped out of it. |
| **Authorized Faculty Member** | Montana Tech faculty member in the MTECHS domain who is registered in the StOut system and has been assigned to teach one or more course offerings. Faculty members may be “active” or “inactive”. |
| **Authorized Observer** | Anyone with an account in the MTECHS system who has been assigned the “observer” role. |
| **CORE** | Course Outcome, Review and Evaluation (CORE) |
| **CORE Report** | A report which faculty members in some departments are required to write for each course offering which they teaching. The StOut web application generates statistics for this report. (See CORE Statistics) |
| **CORE Statistics** | Report showing the extent to which student outcomes were met by students in a course offering. |
| **Course** | Course contributing to measuring student outcomes. Courses are identified by a prefix and number, such as ESOF 328. The course prefixes are dependent on the program they are part of. The course has a name, such as Software Requirements and Specifications. |
| **Course Offering** | A particular section of a course offered in a particular semester. Course offerings are identified by a course, a section, and a semester. |
| **Course Outcome** | Criteria which students passing the course should meet. These are specific to the course and are different than ‘Student outcomes’ which are specific to a program. StOut does not measure course outcomes. |
| **Course PC Report** | Report showing the extent to which students met performance criteria during the semester(s) of interest. This report is divided by performance criteria and courses. |
| **Default Semester** | The semester used when no semester is given (for instance, creating a course offering or a new course). |
| **Faculty Member** | An educator that works at Montana Tech |
| **Matrix Report** | Report showing the association of courses to performance criteria and the weights of those associations. This is used to get an overview of the extent to which courses are covering performance criteria. |
| **Metric** | A metric associated with a course offering. The metric is created by the instructor of the course. It consists of a description, maximum number of points, and a list of the student outcomes which it measures. Can also refer to measurements of a course’s success. |
| **Metric Goal** | The overall (percentage) score which a student needs to meet or exceed to be considered to have met the student outcomes.  This needs to be stored in such a way so the metric goal can be changed easily. Throughout this document, it is assumed that the metric goal is 70%. |
| **Modern Browser** | Any web browser which reliably implements the latest standards in HTML and CSS, with complete support for JavaScript. |
| **Outcome Report** | Report showing the extent to which students met the selected student outcome during the chosen semester. This report is divided by the performance criteria which measure the outcome. |
| **Operational Data** | Data that is still editable. Data may also contain sensitive information about students if the department allows it too. |
| **PC Semester Report** | Report showing the extent to which students met performance criteria during the semester(s) of interest. This report is divided by performance criteria and semesters. |
| **Performance Criterion** | A criterion which a program will use to assess a student outcome. |
| **Performance Criterion Abbreviation** | Abbreviation which will be used to identify an performance criterion. |
| **Program** | A degree program. This system is to facilitate assessment for any participating programs from Montana Tech. |
| **Program Abbreviation** | Abbreviation which will be used to identify a program. |
| **Raw data** | Raw data is the number of students, weight of the course, and percentage. |
| **Registered in StOut** | StOut contains user information for this person. |
| **Semester** | A Montana Tech semester. Consists of a year and either fall, spring or summer. |
| **Semester(s) of Interest** | A single semester or a range of semesters for which a report is being generated. |
| **Simple Average** | An average of each value with no weighting due to characteristics of the values. |
| **Student Score** | The score that a student earned on a metric. |
| **Student Outcome** | A criterion which a program will assess for ABET or Northwest accreditation. |
| **Student Outcome Abbreviation** | Abbreviation which will be used to identify a student outcome. |
| **User** | Montana Tech faculty member or staff in the MTECHS domain who is is registered in the StOut system. In addition, anyone with an account in the MTECHS system who has been assigned the “observer” role. |

### Acronyms and Abbreviations

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **ABET** | | Accreditation Board of Engineering and Technology | |
|  | **AD** | | Adaptability | |
|  | **AL** | | Availability | |
|  | **AUC** | | Analysis Use cases | |
|  | **CM** | | Communications | |
|  | **CT** | | Cost | |
|  | **DB** | | Database | |
|  | **DC** | | Design Constraint | |
|  | **DD** | | Delivery Data and Conditions | |
|  | **DL** | | Delivery Environment | |
|  | **DV** | | Development Environment | |
|  | **EN** | | Enhanceability/Extendibility | |
|  | **FE** | | Future Enhancements | |
|  | **HF** | | Human Factors | |
|  | **HW** | | Hardware | |
|  | **IUC** | | Illustrative Use Cases | |
|  | **ML** | | Maintainability | |
|  | **OP** | | Operations | |
|  | **PR** | | Performance | |
|  | **PT** | | Portability | |
|  | **QL** | | Quality | |
|  | **RL** | | Reliability | |
|  | **SC** | | Security | |
|  | **SDD** | | Software Design Description | |
|  | **SRS** | | Software Requirements Specification | |
|  | **ST** | | Standards | |
|  | **SI** | | Site | |
|  | **SW** | | Software | |
|  | **UB** | | Usability | |
|  | **VV** | | Verification & Validation | |
|  | **XI** | | External Interfaces | |
|  | **XXX** | | X of X of X | |
|  | |  | |

## References

ABET, <http://www.abet.org/>

CAS, <https://wiki.jasig.org/display/CAS/Home>

Fundamentals of Engineering Exam <https://ncees.org/engineering/fe/>

NorthWest Commission on Colleges and Universities <http://www.nwccu.org/>

W3C XHTML validation software, [http://validator.w3.org](http://validator.w3.org/)

W3C CSS validation software, <http://jigsaw.w3.org/css-validator>

# Design Background

## Background Information

Design for the Student Outcome Assessment Program (StOut) is strongly based on the design of the ABET Outcome Assessment Program (AbOut) that is currently in use by the Computer Science and Software Engineering departments. StOut’s design is seen as a re-imagining of the AbOut’s design, to make it more extendable, and to allow more detailed and backwards reaching reports.

Stakeholders of StOut include Jeff Braun, the mentor in charge of the project. Celia [Schahcz](https://cs.mtech.edu/main/index.php/component/content/article/99-dr-celia-schahczenski)enski, who oversaw the previous application AbOut. Susan Schrader, ABET program coordinator for Montana Tech’s Petroleum Engineering program.

Issues the project may encounter are similar to the issues that AbOut encountered. This includes issues of becoming a class project where some low priority simple issues are handled before high priority complex issues. This could create problems like authorization and input validation being ignored in favor of CSS issues.

Alternative designs considered are includes AbOut’s structure, using PHP to create the web pages and to communicate between the server and the front end of the web application. We decided not to follow this design because PHP is a language that is not taught as thoroughly as Java. Additionally, data integrity and security issues in PHP are prevalent.

# User Characteristics

System users will include instructors, administrators and other faculty members that will need to be able to input information related to the assorted accreditation programs that Montana Tech is involved in. The design of the system will make it easy enough for users to be able to interact without a large amount of training or expertise. However, administrators and program coordinators will need to know how to set up permissions for other users and set up classes and related information that will be used by other users.

Users will interact with the program through web pages designed using the Ember.js framework. The pages will include ways add data to the database, including a way to import previous data into the new database and input new data through forms. The web page will be able to pull data from the database and format the data into reports. Users at an administrator or program coordinator level will also be able to add other users, courses, outcomes, relationships between courses and outcomes and other associated information.

# Design Overview

StOut will be designed into two different parts. First, a front end web interface that will allow users of the program to input data into the database, edit information in the database, create new course offerings, edit course and outcome relationships, generate reports and more features. <Architectural Design of Front End>

The other part of the program will be a database backend. The database will take the information input through the front end, and store, manipulate, delete, or read the data. The database will also be able to move data that no longer needs to be stored into a read only section of the database. This will allow the data to be more efficiently stored and read, allowing reports to be created that stretch back farther than is currently possible on the AbOut program.

The backend is constructed using DropWizard, framework designed to make it easier to create a RESTful web service using Java. DropWizard simplifies end point creation to enable the front end to access all of the stables that are available in the database. <Architectural Design of Back End>

# Data Architecture

*[This section should describe the data structures to be used in support of the implementation. If these include databases, define the table structure including full field descriptions, relationships, and critical database objects. Graphical languages are appropriate. If this information is often provided in a separate Database Design Document, if this is the case simply refer to this document and omit the remainder of section 6.]*

## Data Analysis

We did not conduct any data analysis at this time. However, data analysis activities could be done in the future to add new reports or refine current ones.

## Output Specifications

StOut will be able to produces several different kinds of reports based on the information stored in the database portion of the program. Reports will focus on different information relating to the student outcomes. This may include the outcomes associated with a specific course or all of the outcomes the program needs to meet. Reports should be able to be printed easily within the application.

## Logical Database Model

For a full diagram of the database, see the **dbModel.pdf** in the Gitlab.

Assume all primary, foreign, and composite keys are of type int(11) unless otherwise stated

|  |  |  |
| --- | --- | --- |
| **TableName** | **FieldName** | **Description** |
| Program | program\_id | Int(11) - Primary Key |
|  | name | Varchar(100) - Name of the Program |
| Scale | scale\_id | Int(11) - Primary Key |
|  | name | Varchar(100) - Name of the scale |
|  | desc | Varchar(100) - Description of the scale |
|  | program\_id | Int(11) - Foreign Key Program scale is apart of |
| Perf\_Indicator | perf\_indicator\_id | Int(11) - Primary Key |
|  | scale\_id | Int(11) - Foreign Key Scale the performance indicator measures |
|  | name | Varchar(100) - name of the performance indicator |
|  | desc | Varchar(100) - description of the performance indicator |
|  | rating | Int(11) - Rating of the performance indicator |
| Program\_Cutoff | Program\_id | Composite Key - What program the cutoff is for |
|  | semester\_id | Composite Key - What Semester the cutoff is apart of |
|  | start\_date | DATE Beginning of Cutoff period |
|  | end\_date | DATE Last day of cutoff period |
| Semester | semester\_id | Primary Key |
|  | semester\_type\_id | Foreign Key |
|  | year | Varchar(100) - What school year the Semester is a part of |
| Semester\_Type | semester\_type\_id | Primary Key |
|  | startMonth | Varchar(100) - When the semester begins |
| Program\_Permission | program\_permissions\_id | Primary Key |
|  | permission\_id | Foreign Key - What permission level the user in the entry has |
|  | user\_id | Foreign Key - Who has the permission level in the entry |
|  | program\_id | Foreign Key - What program the permission and user are associated with |
| Users | user\_id | Primary Key |
|  | name | Varchar(11) - Name of the user |
|  | username | varchar(11) - User’s CAS username |
|  | job\_title\_id | Foreign Key - The job title the user |
| Job\_Title | job\_title\_id | Primary Key |
|  | title | varchar(100) i.e. Professor |
| Roles | permission\_id | Primary Key |
|  | name | Varchar - title of permission level in application, “Administrator” “Faculty” |
| Course | course\_id | Primary Key |
|  | course\_num | int(11) - Course Identification number “Computer Graphics CSCI **494**” |
|  | prefix\_id | Foreign Key - The prefix for the course |
|  | title | varchar - name of the course |
|  | program\_id | Foreign Key - Program course is a part of |
| Course\_Prefix | prefix\_id | Primary Key |
|  | prefix | varchar - String prefix of a course “CSCI” “ESOF” |
| Course\_Outcome | course\_id | Composite Key - What course the outcome is evaluating |
|  | outcome\_id | Composite Key - Outcome course is evaluated on |
| Offering | offering\_id | Primary Key |
|  | course\_id | Foreign Key - Course the offering is of |
|  | user\_id | Foreign Key - id of the instructor |
|  | semester\_id | Foreign Key - Semester offering occured |
|  | section\_num | varchar(4) - section number of offering, when a course is offered more than once in a semester or has a lab |
|  | locked | BIT(1) - whether or not the offering is editable |
|  | num\_students | int(11) - number of students in offering |
|  | instructor\_name | varchar(100) Name of instructor, stored separately in case someone gets married or otherwise changes their name |
|  | scale\_id | Foreign Key - scale the offering is judged on |
| Offering\_Student | student\_id | Composite Key - student in an offering |
|  | offering\_id | Composite Key - offering that has students |
|  | student\_name | varchar - name of student |
| Student\_Outcome | offering\_id | Foreign Key - the offering the student and outcome are related to |
|  | outcome\_id | Foreign Key - Outcome the student is being evaluated on in the offering |
|  | student\_id | Primary Key - Person that is being evaluated |
| Offering\_Assign | assign\_id | Primary Key - the assignment that is evaluated |
|  | offering\_id | Foreign Key - offering the assignment is a part of |
|  | score | int(11) - Evaluated score on the assignment |
|  | name | varchar(100) - Name of the assignment |
|  | desc | varchar(100) - Description of the assignment |
|  | max\_score | int(11) - Maximum possible score on the assignment |
|  | scale\_id | Foreign Key - What scale the assignment is associated with |
| Student\_Assign | assign\_id | Composite Key - The assignment to be scored |
|  |  | Composite Key - The student who is being scored on the assignment |
|  | score | int(11) - What the student scored on the assignment |
| Outcome\_Assign | outcome\_id | Foreign Key - The student outcome associated with the assignment (i.e. ABET outcome A) |
|  | assign\_id | Foreign Key - assignment associated with the outcome |
| Offering\_Outcome | offering\_id | Composite Key - offering being evaluated by the metric |
|  | outcome\_id | Composite Key - The outcome the associated with the course |
|  | score | Double - Overall score of offering on the selected outcome |
| Outcome | outcome\_id | Primary Key - |
|  | metric\_id | Foreign Key - What metric the outcome is a part of (i.e. ABET, Northwest) |
|  | name | varchar(100) Name of the outcome (i.e. outcome A) |
|  | desc | varchar(100) - Description of the outcome |
| Metric | metric\_id | Primary Key |
|  | program\_id | Foreign Key - What program is being evaluated by the outcome |
|  | name | varchar - name of the metric (i.e. ABET, Northwest) |

## Data Conversion

Data stored in the active portion of the database will be able to be converted into legacy data when the data is finalized. The finalization of data will be done either manually or automatically by an administrator or program coordinator. If done manually, the user converting the data will be able to select a data that will allow them to tell the database to move the data into the legacy side of the database. If done automatically, the administrator will set up a window in which faculty members will be able to input data for their courses. Once this window closes, there will be a period of time for the administrator to check information or allow a late faculty member to add data before the data is converted. When converted, courses that include information about students will have that information stripped, while the rest of the data will be converted into the legacy side of the database.

# Design

## Description for Front End

Handles all user interaction, and shows all data the user requests that they have access to.

### Processing for Front End

### Front end Interface Description

### User Interface design

The overall look and feel of the web application and how the users would navigate the content. Focus was on navigation based on permissions afforded to the user. This limits interactions for the user to only what they can work with.

#### HTML structure

The structure was kept to as minimal amount of div elements as possible to make later changes in presentation easier and quick. Fewer div elements also allows for easy separation of data that the user will work with.

The basic structure is:

Header  
Navigations  
 -Body  
 --Navigation tree  
 --Content  
Footer

#### Navigation

Users can only access content they have permission for, and these users will be able to access the content they want to see or affect, then users will access specific data entries. Due to the overlap in the different permissions’ access to content, a third navigation was needed to take a permission level and a content selection and produce a navigation to select specific entries. This navigation was placed into a component that can be reused wherever necessary.

#### Forms

The main purpose of StOut is to take and save data, allow corrections of the data, and report aggregated data back to the users. To accomplish the separation of the data and other features, forms were created that would handle viewing and editing data. The forms can be placed anywhere in the web application and allow the application to perform actions separate from the forms. These forms take data, and present the data for viewing or editting, handle validation/verification of the data, and yields the data back out for others to use.

## Description for Back End

The backend was built following the design patterns of Dropwizard. This includes breaking out the Java objects representing the database tables into their own package, database access files into objects, and having a package that represents all the REST items available. This pattern can be seen on the Dropwizard example Github project, or by looking through our project. The system is secured by receiving a ticket from Campus Authentication Services (CAS) and validating from the backend. After this all communications are secured between the frontend and backend with JSON Web Tokens (JWT). The JWTs have a REST endpoint to update the JWT for continued interaction past the initial expiration.

### Processing for Back End

### Back end Interface Description

The backend uses Dropwizard to interface with the Design, and REST based endpoints to interface with the front end.

**Appendix A// Requirements Traceability Matrix**

|  |  |  |
| --- | --- | --- |
| **Requirement Identifier** | **Requirement Description** | **SDD**  **Ref.** |
|  |  |  |
|  |  |  |
|  |  |  |