CS 455 - Software Engineering

Green Team

Software Requirements Specification Document

Ricky Moore

Elizabeth McClellan

Hongbin Yu

James Reatherford

Ankur Patel

Wenhao Wang

University of North Alabama

*2013/03/02*

**Revisions**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Primary Author(s)** | **Description of Version** | **Date Completed** |
| v1.0 | Elizabeth McClellan  Ricky Moore | First Draft | 2013/03/02 |
| v1.1 | Ricky Moore  James Reatherford  Hongbin Yu | Second Draft  Added File Descriptions  SQA Revisions | 2013/03/12 |
| v.1.2 | Ricky Moore  Elizabeth McClellan | Third Draft  Completed All Sections  Section Reordering  Client Requested Revisions  Added UI & descriptions | 2013/03/14 |
| v. 1.3 | Ricky Moore  Elizabeth McClellan | Final Draft  Client Requested Revisions | 2013/03/21 |

**Review, Agreement, and Approval**

**Specification Document Review & Agreement**

|  |  |  |
| --- | --- | --- |
| **SQA Team Member** | **Signature** | **Date** |
| Hongbin Yu |  |  |
| James Reatherford |  |  |

**Specification Document Agreement**

|  |  |  |
| --- | --- | --- |
| **Development Team Member** | **Signature** | **Date** |
| Ricky Moore |  |  |
| Elizabeth McClellan |  |  |
| Ankur Patel |  |  |
| Wen Hao Wang |  |  |

**Specification Document Approval**

|  |  |  |
| --- | --- | --- |
| **Client Representative** | **Signature** | **Date** |
| Dr. Patricia Roden |  |  |

**Contents**

1. Introduction 04

1.1 - Purpose of this document 04

1.2 - Overview 04

2. Deliverables 05

3. General Description 06

3.1 - Functionality 06

3.2 - Limitations 07

4. Functionality Requirements and Constraints 08

4.1 - Faculty Functionality 08

4.2 - Admin Functionality 09

4.3 - Schedule Creation 12

5. Other Requirements and Constraints 13

5.1 - Input Files 13

5.2 - Login Information 13

6. Application Output 15

6.1 - Graphical Representation of Schedule 15

6.2 - Excel Representation of Schedule 15

7. Appendices 16

A - User Interface 16

B - Definition of File Formats with Examples 19

C - Coding Standards 31

**1. Introduction**

**1.1 – Purpose of this document**

The purpose of this document is to detail each requirement of the departmental scheduling system, and to serve as a reference from which the software will be developed by the team members. After approval, the document will also serve as an agreement between the development team and the client representative with regard to the requirements of the final version of the software.

**1.2 – Overview**

The result of this project shall be a fully functional version of the departmental scheduling application. The application shall be deemed acceptable once the following requirements are met:

1. The application is web-based, and supports the creation and viewing of a university academic department’s semester schedule of classes to the satisfaction of the client representative.
2. The application works with the three most popular browsers as of March 2013: Google Chrome, Firefox, and Internet Explorer
3. The application is generically designed in such a manner as to be usable by any department within the University of North Alabama.
4. The user interface is developed in a similar style to current University of North Alabama webpages, the specific design of which is subject to approval of the client representative.
5. All functionality defined in this document is included in the application.
6. All requirements and constraints listed in this document are adhered to by the application.

**2. Deliverables**

The following deliverables and delivery dates are agreed upon by all parties:

1. April 2nd, 2013, 6:00 pm:
   * User Interface
   * Login and User Account modules
2. April 16th, 2013, 6:00 pm:
   * Scheduling Algorithm module
   * Input Verification modules
3. April 30th, 2013, 6:00 pm:
   * Output modules
   * Fully integrated project
   * User Manual
   * All associated project documentation

**3. General Description**

**3.1 – Functionality**

In addition to satisfying the aforementioned requirements, the application will provide the following general functionality. Each item is explored in more detail in later sections of this document.

1. The application will aid in schedule management by creating a schedule given information about available time slots, available rooms, and information regarding the courses to be scheduled. (Section 3.3)
2. The application will have the ability to:
   1. Generate a graphical representation of the schedule which will be viewable by both faculty and outside users. (Section 5.1)
   2. Generate an Excel spreadsheet representation of the schedule. (Section 5.2)
3. The application will require faculty members and administration of the department to log in before providing any functionality. (Section 4.2)
4. The application will provide the following functionality to faculty members. (Section 3.1).
   1. View the list of desired courses to be offered by the department.
   2. Be given the ability to indicate the course(s) and desired time interval(s) they wish to teach for the semester.
   3. Be able to view the entire created schedule with the option to see only the courses they have been scheduled to teach.
   4. After login, be given the option to change their password if desired.
5. The application will provide the following functionality to the department administration. (Section 3.2)
   1. Create and manage faculty member accounts.
   2. Input information necessary for the application to create the schedule.
   3. Prompt the application to create the schedule once the necessary information has been supplied.
   4. View and edit the schedule after creation if desired.
   5. After login, be given the option to change their password if desired.
   6. Additionally, all functionality provided to faculty members will also be provided to one of the two administrator accounts (henceforth referred to as the “primary administrator”).

**3.2 – Limitations**

The application has the following limitations:

1. The application will not create a schedule until the administrator requests it to do so.
2. A scheduling conflict is defined as occurring when two faculty members input preferences that contradict one another, or when scheduling two classes would require contradicting one or more constraints or requirements.
   1. The application does not attempt to prevent scheduling conflicts.
   2. The application will attempt to resolve scheduling conflicts once they occur.
   3. If a scheduling conflict cannot be resolved, the application will continue with schedule generation and present the unresolved conflicts to the administrator for manual review.
3. If any information from input files used to create the schedule is changed, the administrator must request a new schedule to be generated before these changes are reflected.
4. If an administrator attempts to create a schedule with incomplete information, the software will first prompt the administrator to input the correct file or create the file from within the application.
5. If an administrator attempts to create a schedule with incompatible or incomplete scheduling information, the software will not create a schedule and will instead present a meaningful error message explaining why a schedule cannot be created.

**4. Functionality Requirements and Constraints**

**4.1 – Faculty Functionality**

**4.1.1 – Overview**

1. Upon first login, faculty members will be prompted to change their password from the default password provided to them.
2. After the initial login, faculty members will have the ability to change their password at any time after logging in.
3. Faculty members will be provided with the following information:
   1. A list of classes to be offered by the department.
   2. The course preferences they have currently specified.
   3. Information about the minimum hours they are required to teach and the current number of hours they have input preferences for.
   4. After schedule generation, a representation of the schedule with the option to view only a list of the courses they have been scheduled to teach.

**4.1.2 – Preference Input**

1. The program will allow faculty members to indicate the course(s) they wish to teach for the semester and desired time period(s) for each course.
2. Regarding faculty preferences, time periods (morning, mid-day, and evening) are defined as follows:
   1. Any classes starting before 11:00 am, not including those starting at 11:00 am, are considered morning classes.
   2. Any classes from 11:00 am to 2:00 pm, including classes that start at 11:00 am and 2:00 pm, are considered mid-day classes.
   3. Any classes that start after 2:00 pm, but not including classes starting at 2:00 pm, are considered evening classes.
3. Preferences will be prioritized either by seniority or by the order of submission, as decided by the administration.
4. If seniority prioritization is chosen, faculty members with the same seniority will have conflicting preferences decided by order of submission.

**4.2 – Admin Functionality**

**4.2.1 – Overview**

1. The primary administrator, in addition to the following functionality, will also have all functionality of faculty members.
2. Upon first login, administrators will be prompted to change their password and username from the defaults outlined in the user manual.
3. After the initial login, administrators will have the ability to change their password at any time after logging in.
4. Administrators will have the ability to recover lost passwords and usernames from the system database.
5. Administrators will be able to input scheduling information and prompt the application to create the schedule once the necessary information has been supplied.

**4.2.2 – File Input**

1. The application will accept the following file input from the administrator:
   1. Possible class times to schedule.
   2. Information regarding the available rooms.
   3. A list of courses to be scheduled.
   4. A list of possible conflict times with courses from other departments, including the usual times those classes are offered.
   5. A list of courses with their prerequisites within the department
   6. Information regarding faculty member accounts
2. The format for each file is specified in appendix B. Additional information can be found in section 4.1.
3. The administrator should also be able to set the method of handling faculty preference conflicts. The two available methods of resolving conflicts are via seniority or via order of submission.
   1. If seniority prioritization is chosen, faculty members with the same seniority will have conflicting preferences decided by order of submission.
4. If no files currently exist in the system, the program will create empty files (in a directory specified in the user manual) for each type of input from (1).
5. To overwrite a file currently stored in the system, the administrator can browse for a file to upload, or may edit the currently existing file from within the browser (see 3.2.2 below).
6. If the admin requests a schedule to be generated using one or more empty files, the application will not generate the schedule and will instead invite the admin to upload the corresponding file, or to create the file within the browser.
   1. If the admin chooses not to do this, the scheduling algorithm may fail due to not having the information necessary.
   2. If the scheduling algorithm fails for any reason, the application will attempt to provide a useful error message explaining why the schedule could not be generated.
7. If a fault is found in the file when submitting or uploading new input, the application will notify the user with a meaningful error message pertaining to the mistake, and will not save the faulty file to the system.

**4.2.3 – File Editing Within the Application**

1. Information from the input files stored in the system will be made available to the administrator, and this information will be editable from within the application browser.
2. Changes made to files from within the browser will be reflected in the files stored in the system.
3. Files edited from within the browser will be checked for faults before any changes are saved.
   1. If a fault is detected, the user will be prompted with a meaningful message specifying the location and type of fault.
   2. Edited files must be fault free before changes are saved to the file.

**4.2.4 – Schedule Editing After Creation**

1. The administrator may opt to edit the schedule after creation in order to resolve unhandled conflicts, or for manual refinement.
2. When editing the schedule:
   1. The admin should be presented with a table of entries, with each entry consisting of a room number, course, instructor, or time slot. The schedule should be able to be sorted by any of the four entry fields.
   2. The admin should be able to edit any individual entry by selecting valid options from drop-down boxes.
   3. The admin should be able to add new entries to the schedule by selecting each individual field from valid options.
      * Note: Valid options are determined by the values specified in the input files, and exclude any options which would result in a conflict, or which would require the application to contradict one or more requirements or constraints.
3. Changes made to the schedule after creation will be automatically reflected in both the graphical and excel representations of the schedule.

**4.3 – Schedule Creation**

1. The application will attempt to create a schedule by matching the following for every available class to be scheduled:
   1. A faculty member
   2. A course title
   3. A room of correct type for the course with the capacity to hold the class.
   4. An available time slot
2. The application will prioritize available classes for which faculty members have input a preference.
3. When creating the schedule:
   1. 400 level courses take precedence over any lower level course.
   2. 300 level courses take precedence over any lower level course.
   3. 200 level courses take precedence over any lower level course.
   4. 100 level courses take precedence over any lower level course.
   5. No level 300/400 level courses should be scheduled for the same day and time unless one is a prerequisite of the other.
4. When creating the schedule, the application will generate a list of classes that were unable to be scheduled along with the possible reason(s) for its inability to be scheduled. This information will be saved to a file and presented to the administrator for manual review.
5. If there is a scheduling conflict in the preferences provided by faculty members, it will be handled according to the priority rules as stated in Section 3.1.
6. If a resolution to a scheduling conflict cannot be made by the application, the course in question will not be scheduled and a note will be presented to the administrator after schedule creation for manual review.
7. If a class has multiple sections, the application will attempt to disperse the scheduling of these sections throughout different days and times.
8. If a class that has already been scheduled is preempted by another class, the application will attempt to relocate the first class to the nearest available time slot.
   * The application will not schedule a class more than one time period (Morning/Mid-day/Evening) away from the faculty member’s preference.
   * If the class cannot be scheduled, it will be flagged to the administrator for manual review after schedule generation.

**5. Other Requirements and Constraints**

**5.1 – Input Files**

1. Appendix B contains a list of valid and invalid file examples, and explanations for the formatting of each input type. The format specified therein must be followed exactly.
2. Any entry which does not conform to this format will be considered a fault.
3. A file which contains a fault cannot be saved to the system.
   * i.e. - The system will at no time store a file which contains a fault.
4. If an administrator attempts to upload a faulty file or attempts to save changes to an edited file, the application will not save the file to the system, and will instead present an error message specifying the type and location of fault.

**5.2 – User Accounts and Login Information**

1. The application will by default have two administrator accounts.
   1. See the user manual for Information on how to log on using these accounts.
   2. Administrators will be required to change their username and password from the default upon first login.
2. Faculty account usernames will be the email address provided by the administrator in the faculty account input file, and must be unique.
3. Faculty accounts will have a default password when the account is created.
   1. The admin is responsible for giving faculty members this password.
   2. This default password can be found in the user manual.
   3. Faculty members will be required to change their password from the default upon first login.
4. If the faculty account file is edited by the admin after accounts have been created:
   1. All currently stored faculty preferences will be deleted.
   2. All currently existing faculty accounts will be deleted.
   3. New faculty accounts will be constructed from the information in the new file.
5. The administration will have the ability to retrieve and/or edit lost usernames and passwords from the system database. See the user manual for more information.
6. Passwords:
   1. Must be 6 to 9 characters long, inclusive.
   2. Passwords are case sensitive.
   3. Passwords must begin with an alphabetic character, must contain at least one numeric character, and must contain at least one of the following characters: an exclamation point (!), a question mark (?), or a comma (,).
7. Security:
   1. After 3 failed login attempts, faculty members will have their account locked. The faculty member must contact the administration to reset the password.
   2. After 3 failed login attempts, administrators will have their password changed automatically and an email will be sent to the administrator’s registered email address inviting them to log on with the new password. This password must be changed upon next login before any other functionality is provided.

**6. Application Output**

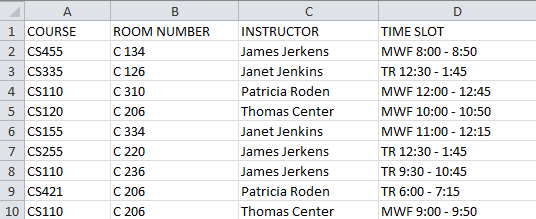
**6.1 – Graphical Representation of Schedule**

1. The graphical representation of the schedule should be viewable by any outside user on the main webpage of the application.
2. The schedule will list all scheduled courses, along with:
   1. The room number the class will be taught in
   2. The name of the instructor teaching the class
   3. The time slot for the class
3. The schedule listing will be able to be sorted by:
   1. Course name
   2. Room number
   3. Instructor

**6.2 – Excel Representation of Schedule**

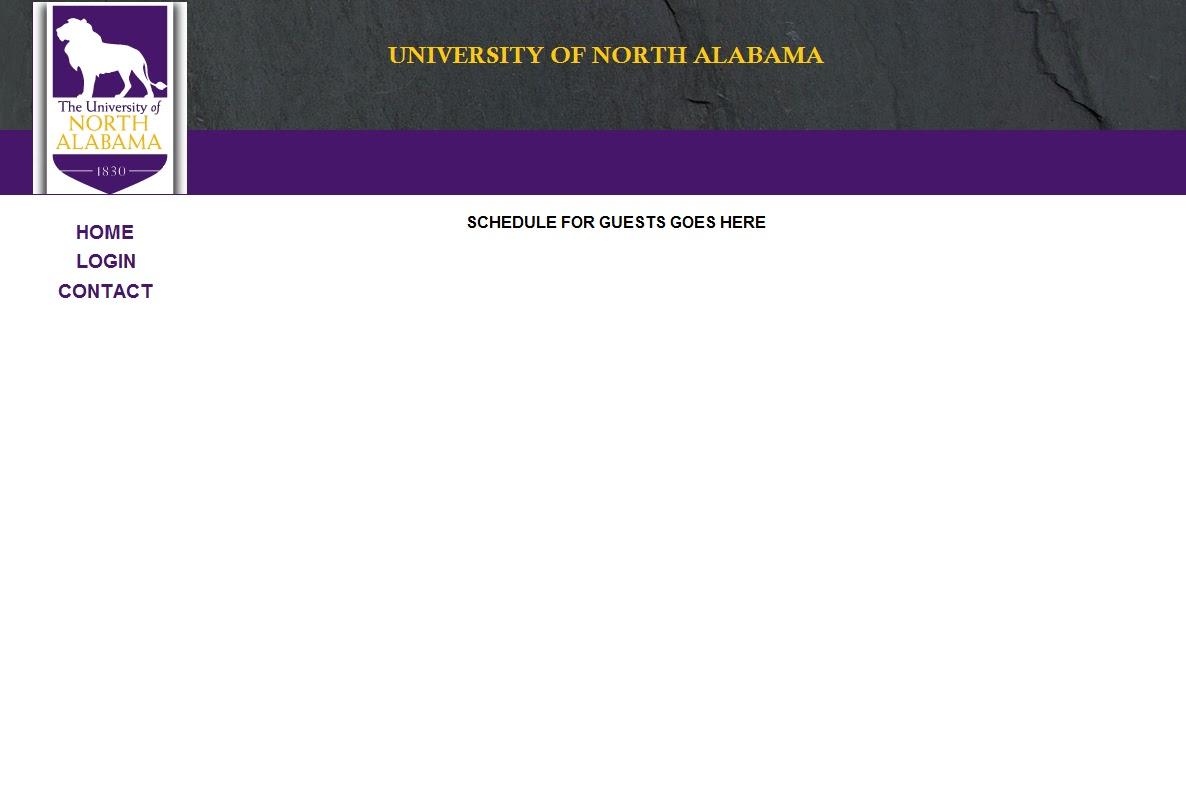
1. The application will also generate an equivalent excel representation when the administrator prompts the application to create the schedule. Instructions on locating this file in the system can be found in the user manual.
2. Any user (including outside viewers) can request a download of this file from the front page.

**An example schedule is included below:**



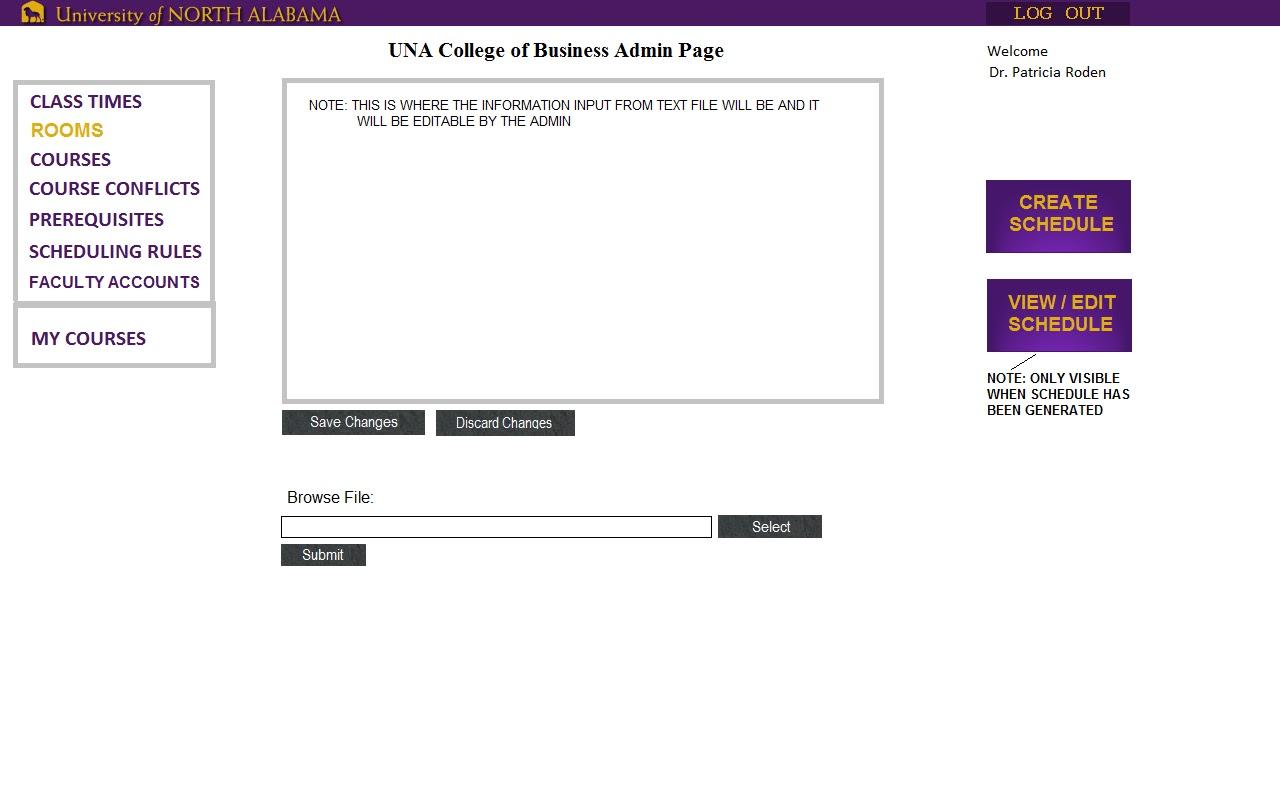
**7. Appendices**

**A-1. Sample Front Page:**



This page will also feature the graphical representation of the schedule and a download link to the excel representation.

**A-2. Sample Administrator Page**

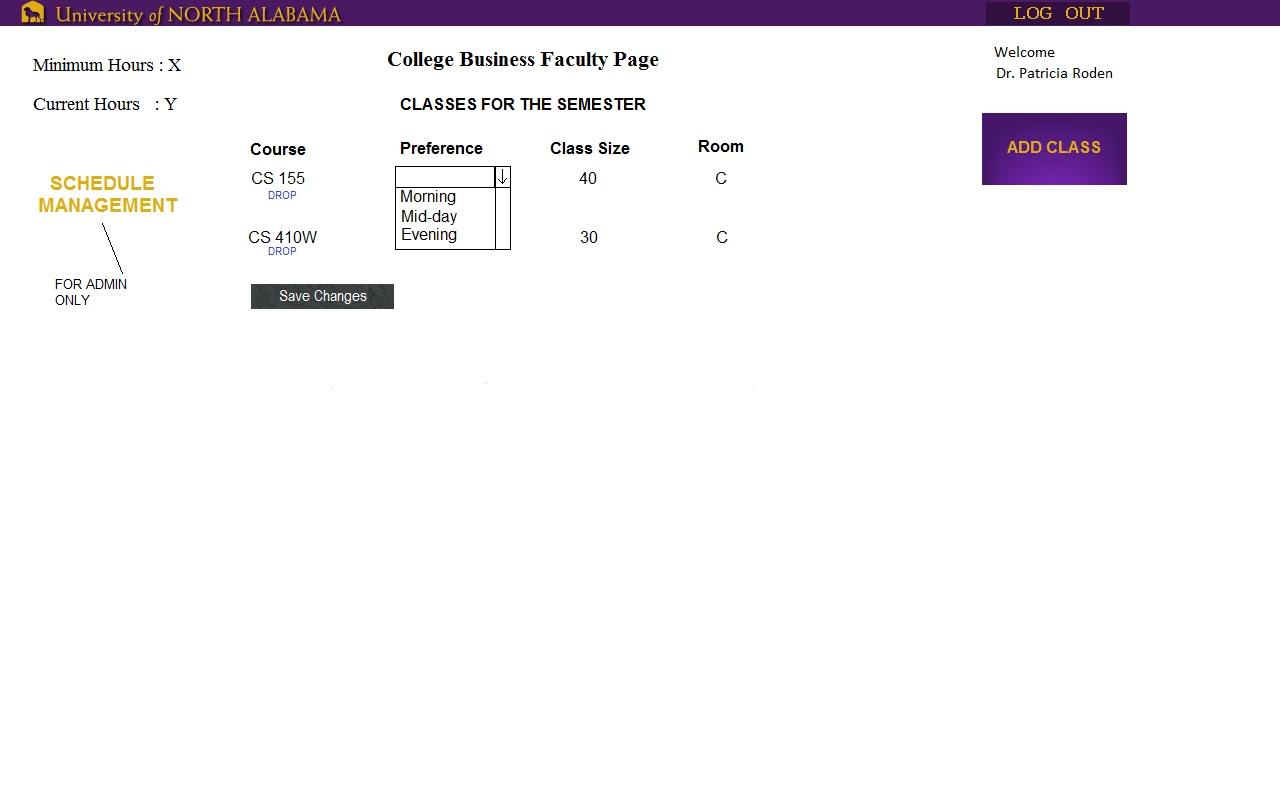


From this page, administrators will be given the ability to view, edit, upload, or delete any input files pertaining to schedule generation and account management.

The central window will function as an in-browser text editor, allowing the administrator to edit or create input files without leaving the application environment. It will also serve as an error console, providing any error messages related to file input.

The “MY COURSES” link is viewable only by the primary administrator and will navigate to the faculty webpage.

**A-3. Sample Faculty Page**



From this page, faculty members will be able to view a list of classes to be offered and add a new preference by clicking the “ADD CLASS” button. They may also choose to edit or delete current preferences if they so desire.

The primary administrator is also given the option of swapping to the administrator front page.

**B – Definition of File Formats with Examples**

**B – 1a: Class Times – Correct File**

**This is an example of a correct file:**

50 MWF/08:00 09:00 10:00 11:00 12:00 13:00 14:00

50 MTWR/08:00 09:00 10:00

75 MW/12:00 13:30 15:00

75 TR/08:00 09:30 11:00 12:30

165 M/18:00

165 T/18:00

**Description of fields**

Each line contains (in order):

1. An integer value that indicates, in minutes, the length of the course on this line.
2. A listing of the days each class is taught, specified by a letter (M, T, W, R, F, S), and not separated by spaces
   1. These symbols must be in chronological order. (RT) would not be valid for a Tuesday/Thursday class
3. A “/” symbol to signal the start of the time fields
   1. There are no spaces before or after the “/” symbol
4. A listing of times, separated by spaces, that signify the starting times of the classes
   1. These times must be in 24 hour format HH:MM
   2. Times before 11:00 must still include two hour characters (e.g., 8:00 would not be valid, but 08:00 would)

**B – 1b: Class Times – Faulty File**

**This is an example of an incorrect file (with added line numbers):**

1 50 M W F/08:00 09:00 10:00 11:00 12:00 13:00 14:00

2 50 MTWR 08:00 09:00 10:00

3 75 WM/12:00 13:30 15:00

4 75 TR/8:00 09:30 11:00 12:30

5 165 K/18:00

6 165 T / 18:00

**Note the following errors:**

|  |  |
| --- | --- |
| **Line Number** | **Error(s)** |
| 1 | There are spaces between each day |
| 2 | There is no “/” dividing the days from the times |
| 3 | The days are out of order |
| 4 | The 8am class is missing a padding 0 |
| 5 | This line does not list a valid day |
| 6 | There are spaces both before and after the “/” symbol |

**B – 2a: Available Rooms – Correct File**

**This is an example of a correct file:**

L 50 RABURN 210

C 40 RABURN 231

C 40 RABURN 232

C 40 RABURN 233

C 40 RABURN 234

C 40 RABURN 235

**Description of fields**

Each line contains (in order):

1. A single letter “C” or “L” to indicate the type of the classroom.
   1. “C” stands for classroom (lecture)
   2. “L” stands for lab
2. The size of the classroom or lab
   1. A single integer value that must be between 1 and 100 (inclusive)
3. A single upper case word that specifies a building name
4. The room number of the building
   1. must be between 1 and 9999 (inclusive)

**B – 2b: Available Rooms – Faulty File**

**This is an example of an incorrect file (with added line numbers):**

1 L 50 RABURN 210

2 C hi RABURN one

3 A 101 232

4 CL -1 RABURN 10000

5 C40RABURN234

6 C 40 RABURN 235 HELLO

**Note the following errors:**

|  |  |
| --- | --- |
| **Line Number** | **Error(s)** |
| 1 | There are more than 1 spaces between the fields |
| 2 | Room size and room number are not integer type, they are “hi” and “one” instead |
| 3 | invalid room type “A”; size is more than 100; building name is missing; |
| 4 | invalid room type “CL”; room size is negative; invalid room number 10000 |
| 5 | There is no space between all the fields |
| 6 | One extra field “HELLO”at the end of the line |

**B – 3a: Courses List – Correct File**

**This is an example of a correct file:**

BI101 5 0 0 30 C 4

BI101L 10 0 0 20 L 2

BI102 1 2 0 30 C 4

BI102L 3 1 0 20 L 2

BI111 2 0 0 30 C 4

BI111L 3 0 0 20 L 2

**Description of fields**

Each line contains (in order):

1. The title of the course. A course title consists of:
   1. 2-4 uppercase alphabetic characters specifying the department
   2. A 3 digit integer specifying the course number, in the range 099-499 (inclusive)
   3. (optionally) A single uppercase alphabetic character used in some courses (for example, L for lab or W for writing).
   4. A single space may occur between (a) and (b), but not between (b) and (c).
2. The number of daytime sections that are being offered for the course
   1. This must be an integer value between 0 and 100 (inclusive)
   2. Daytime classes include any class starting at or before 2:00 pm
3. The number of evening sections that are being offered for the course
   1. This must be an integer value between 0 and 100 (inclusive)
   2. Evening sections include any class starting after 2:00 pm
4. The number of internet sections that are being offered for the course
   1. This must be an integer value between 0 and 100 (inclusive)
5. The size of the class (ie., the maximum number of students allowed to sign up for a single section of the class)
   1. This must be an integer value between 1 and 100 (inclusive)
6. The classroom type required. This must either be “C” for class or “L” for Lab
7. The number of semester hours an instructor would receive credit for.
   1. Must be an integer value between 1 and 12 (inclusive)

**B – 3b: Courses List – Faulty File**

**This is an example of an incorrect file (with added line numbers):**

1 BI101 5 0 0 0 C 15

2 BI101L 101 0 0 20 L 2

3 BI102 1 2 0 30 K

4 BI102L 3 0 20 L 0

5 BI511 2 0 0 30 C 4

6 BI111L 3 3 3 0 20 L 2

**Note the following errors:**

|  |  |
| --- | --- |
| **Line Number** | **Error(s)** |
| 1 | The class size is listed as 0. The course hours is >12. |
| 2 | There are 101 day sections being offered |
| 3 | The classroom type is listed as K. The course hours are missing. |
| 4 | This line does not have the proper amount of fields. The course hours is 0. |
| 5 | This listing is for a graduate class, which is not supported |
| 6 | There is an extra field in this line. |

**B – 4a: Conflict Times – Correct File (OPTIONAL)**

**This is an example of a correct file:**

CS 499 MWF/09:00 TR/09:30 W/18:00

CS 455 W/18:00

CS 360 MWF/09:00 MWF/10:00 MWF/12:00

CS355 TR/09:30 TR/11:00 MWF/12:00

CS335 TR/12:30 TR/18:00 MTWR/08:00

CS311W MWF/09:00 MWF/10:00 MWF/13:00

**Description of fields**

Each line contains (in order):

1. The title of the course. A course title consists of:
   1. 2-4 uppercase alphabetic characters specifying the department
   2. A 3 digit integer specifying the course number, in the range 099-499 (inclusive)
   3. (optionally) A single uppercase alphabetic character used in some courses (for example, L for lab or W for writing).
   4. A single space may occur between (a) and (b), but not between (b) and (c).
2. A listing of the days when the course cannot be taught, specified by a letter (M, T, W, R, F), and not separated by spaces
   1. These symbols must be in chronological order. (RT) would not be valid for a Tuesday/Thursday class
3. A “/” symbol to signal the time field
   1. There are no spaces before or after the “/” symbol
4. The time for which the course cannot be taught, which must be in HH:MM format
5. There can be more than one conflict time per course on the same line

**B – 4b: Conflict Times – Faulty File**

**This is an example of an incorrect file (with added line numbers):**

1 CS 499 MWF / 09:00 TR / 09:30 W / 18:00

2 CS 455 W/111:22222

3 CS 999 MWF/24:60:00 MWF/25:61 MWF/99:01

4 CS355 RT/09:30 RT/11:00 MWF/12:00

5 CS335 TR/12:30 AB/18:00 mtwr/08:00

6 CS311E MWF/09:00 MWF/10:00 MWF/13:00

**Note the following errors:**

|  |  |
| --- | --- |
| **Line Number** | **Error(s)** |
| 1 | Two spaces between “CS” and “499”, spaces before and after “/” |
| 2 | Time format error |
| 3 | Course number exceeds limit; time’s hours and minutes exceed limits |
| 4 | Days are in wrong order “RT” |
| 5 | Days format errors |
| 6 | Invalid course “CS311E”, “E” is an invalid letter after course number |

**B – 5a: Prerequisites – Correct File (OPTIONAL)**

**This is an example of a correct file:**

CIS225 CIS125 CIS125H

CIS236 CIS125 CIS125H

CIS315 CIS225 CIS236

CIS330 CIS225 CIS236

CIS350 CIS236

CIS366 CIS225

**Description of fields**

Each line contains (in order):

1. The title of the course. A course title consists of:
   1. 2-4 uppercase alphabetic characters specifying the department
   2. A 3 digit integer specifying the course number, in the range 099-499 (inclusive)
   3. (optionally) A single uppercase alphabetic character used in some courses (for example, L for lab or W for writing).
   4. A single space may occur between (a) and (b), but not between (b) and (c).
2. One or more fields listing the titles of the classes that are prerequisites to the first field
   1. A class may not require itself
      1. Outside of this, there is no checking for cycles within the file. For example, CS355 could require CS255 while CS255 requires CS355. Prevention of these logic errors is left to the administrator and is outside the scope of this program.

**B – 5b: Prerequisites – Faulty File**

**This is an example of an incorrect file (with added line numbers):**

1 CIS225 CIS125 CIS125K

2 CIS236 CIS125 CIS125

3 CIS315 CIS315

4 CIS330

5 CIS350 CIS366

6 CIS366 CIS350

**Note the following errors:**

|  |  |
| --- | --- |
| **Line Number** | **Error(s)** |
| 1 | The last prerequisite is not a valid class |
| 2 | This class lists the same prerequisite twice |
| 3 | This class requires itself |
| 4 | There are no prerequisites listed for this class |
| 5 & 6 | These lines are correct, but there is a logical error which would result in undesirable behavior. However, accepting such input is a limitation of the application. |

**B – 6a: Faculty Members – Correct File**

**This is an example of a correct file:**

Adam, Larry 5.5 alarry@una.edu 12

Aguado, Alex 60 aalex@una.edu 12

Armstrong, Linda 0 alinda@una.edu 18

AJ., Birdie 5 bbirdie@una.edu 12

Baird, Paul 5 bpaul@una.edu 12

Cagle, Corey 5 ccorey@una.edu 12

**Description of fields**

Each line contains (in order):

1. Last name of the faculty, followed by the comma symbol “,”
   1. This field is allowed to contain any symbol except ‘,’ (comma) or spaces
2. First name of the faculty which is one space after the “,” symbol
   1. This field is allowed to contain any symbol except ‘,’ (comma) or spaces
3. A positive number indicating the faculty member’s years of service
   1. Must between 0 and 60 (inclusive)
   2. May include a fractional portion of 0.5 to represent a single semester.
4. The faculty’s email address
   1. This is also used as the faculty’s account name
   2. Each email address must be unique
5. The hours that the faculty is teaching for the current semester
   1. Must an integer between 1 and 18 (inclusive)

**B – 6b: Faculty Members – Faulty File**

**This is an example of an incorrect file (with added line numbers):**

1 Adam,Larry 5.5 alarry@una.edu 12

2 Aguado, Alex 3.2 aalex@una.edu 12.2

3 AJ., Linda 61 alinda@una.edu 19

4 Last, Middle, First 5 bbirdie@una.edu 9

5 B,aird, Paul 5 bpaul@una.edu 12

6 Cagle, Corey -1 ccorey@una.edu -1

**Note the following errors:**

|  |  |
| --- | --- |
| **Line Number** | **Error(s)** |
| 1 | No space betweeen “,” and first name |
| 2 | Years of service and minimum hours format errors |
| 3 | Years of service and minimum hours exceed the limits |
| 4 | Name is in wrong format |
| 5 | Last name contains a comma |
| 6 | Years of service and minimum hours are negative numbers |

**C – Coding Standards**

1. Indents and white space:
   1. When using indents, 4 spaces will be used instead of tabs; there will be no use of tab characters.
   2. There should be no white space in a parameter list immediately after an opening parentheses and no white space immediately before a closing parenthesis. There should be a single blank (space) between parameters, following commas.
   3. There should be at least two blank lines between each function (not including comment lines).
2. Line length:
3. No line of code should have more than eighty characters. Comments will have to be continued to next line and code will have to be continued to next line or divided into smaller modules to reduce lines to an acceptable length.
4. Comments:
5. At the top of each file there should be the names of all contributors, the date, and a brief description of the .
6. Visually distinct comments delimited by lines of 80 characters should be included before every function.
   * These comments should include the type and name of the function, and a brief description of any essential information necessary to understand the function.
7. In regards to inline comments, they should be used to describe complex parts of code that is not inherently obvious; i.e. – code should be self-documenting to the furthest extent possible.
8. Variable names:
9. Constants are to be uppercase with underscores separating multiple words.
10. Non-constant variables should be all lowercase with underscores separating words.
11. Functions:
12. In the grouping of functions within a file, there should be a logical order with functions performing a similar or related task grouped together, and ordered from abstract to more detailed.
13. Function names should start each word with an uppercase letter followed by lowercase letters, and they should have underscores between multiple words.
14. Logically related code segments within functions should be grouped, with a single blank line between grouped segments.

**Example code using standards:**

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Name1

// Name2

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Date

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Description of file

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

<?php

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Public Function: Hello\_World

// \* Essential notes for understanding the function.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
function Hello\_World()  
{  
 echo "Hello World!";  
}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// Public Function: Add

// \* Given two variables, adds the variables and returns the result.

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
function Add($var\_a, $var\_b)  
{

//Example of inline comment

$result = $var\_a + $var\_b;

return $result;  
}  
?> //End PHP code