University of Technology, Sydney

Faculty of Engineering and Information Technology
School of Software

31927 - Applications Development with .NET
32998 - .NET Applications Development

SPRING 2017
ASSIGNMENT 2 SPECIFICATION

DUE DATE – Monday 11:59pm, 16 October 2017
DEMONSTRATIONS – Labs of Week 12

This assignment is worth 35% of the total marks for this subject.

Summary

This assessment requires you to develop a basic offline user login and reputation system. Simply put, it is a Windows form application that will allow new users to be added, existing users to login, for users to see a list of other users, and for the users to provide ratings (from 0 to 5) on other users. This small prototype is meant to give you a glimpse at the complexity of creating a user management system as part of that are common to most web systems.

Assignment Objectives

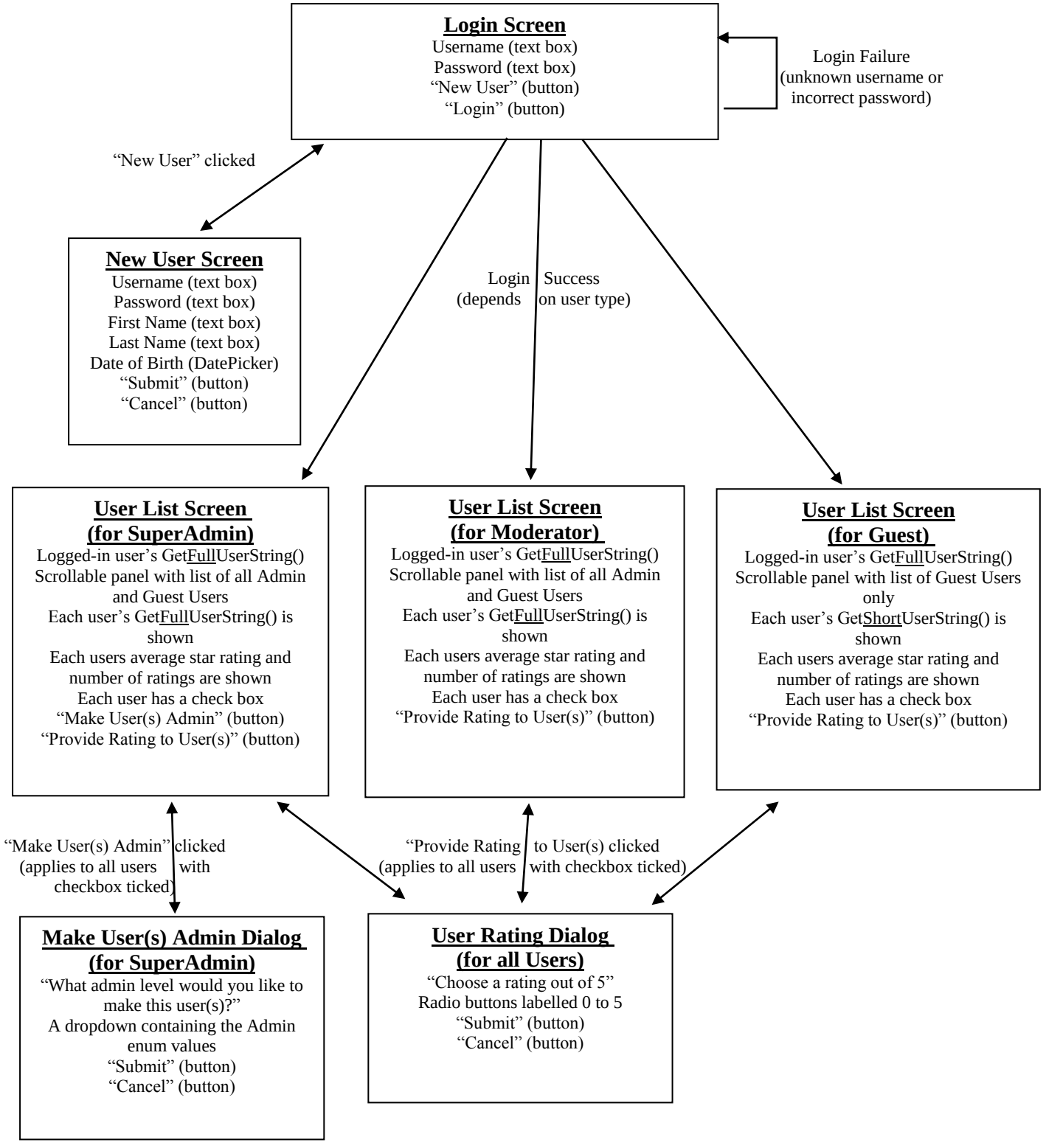
The purpose of this assignment is to demonstrate competence in the following skills.

- Polymorphism including abstract classes, protected member variables, and overridden methods
- Generic list utilization
- Enumerators and properties.
- Text file reading and writing
- Windows forms

Many of these have at least one question in the final exam and so by doing this assessment (at least to a Credit level) you will be preparing yourself for the exam.

Screens:

Each box is a separate screen which shows the visual elements that should be present. The direction of arrows specify the direction of flow between screens. The text in each box describes some of the components that should be in the screen, though you are free to add more if you feel they are needed / appropriate.



Required Classes, Methods, and Member Variables:

In order to test core C# principles, the following classes, methods, and member variables must be implemented. Any additional classes or functionality within these and other classes should be designed and added as needed.

- **UserHandler** – Wrapper class to store and manipulate a list of users
  - Member variables:
    - **A private List<T> of User objects**
    - **private User loggedInUser; -** the current user that is logged into the system after the Login Screen
    - **public User LoggedInUser { get {…} set {…} }** – a property that provides a getter and setter for the private loggedInUser variable.
  - Methods:
    - **public bool LoadAllUsers()** – Load all user data from "Guests.txt" and "Admins.txt" plain text files and stores them in the User List member variable.
    - **public bool SaveAllUsers()** – Save all user data from "Guests.txt" and "Admins.txt" plain txt files using the WriteGuestToFile(…) and WriteAdminToFile(…) methods below
- **User** – abstract base class
  - Member variables:
    - **protected string variables username, password, firstName, and lastName.**
    - **protected int ratingsCount –** the number of times a user has been rated
    - **public int RatingsCount { get {…} } –** a property with only a getter for ratingsCount
    - **protected double averageRating** – the average of all star ratings given to this user (0 to 5, founded to the second decimal point e.g. 3.27)
    - **public double AverageRating { get {…} } –** a property with only a getter for averageRating.
  - Methods:
    - **public bool CheckUserNameAndPassword(string username, string password)** - concrete method, checks the username and password parameters against the member variables and return true if they match and returns false otherwise.
    - **public string GetShortUserString() –** concrete method**,** returns a string containing the username and first name only.
    - **public void AddRating(int rating) –** concrete method, updates the averageRating and ratingsCount member variables with the given parameter.
    - **public string GetFullUserString()** - abstract method, see below for overrides.
- **Guest -** inherits from User
  - Member variables:
    - **private DateTime dateOfBirth;**
  - Methods:
    - **public Guest(string username, string password, string firstName, string lastName, int ratingsCount, float averageRating, DateTime dateOfBirth)** - Guest constructor
    - **public bool WriteGuestToFile(System.IO.StreamWriter file)** – writes all member variable of this Admin object (including those in the

User base class) to the end of the file passed in as a parameter and returns true if the write was successful.
        - **public string GetFullUserString()** – overrides base class version, returns a string with firstName, lastName, and dateOfBirth.
- **Admin** – inherits from User
    - Member variables:
        - **enum AdminType with values of SuperAdmin and Moderator**
        - **private AdminType adminType;**
    - Methods:
        - **public Admin(string username, string password, string firstName, string lastName, int ratingsCount, float averageRatingAdmin, Type AdminType)** – Constructor of Admin
        - **public bool WriteAdminToFile(System.IO.StreamWriter file)** – writes all member variable of this Admin object (including those in the User base class) to the end of the file passed in as a parameter and returns true if the write was successful.
        - **public string GetFullUserString()** – overrides base class version, returns a string with username, firstName, lastName, and AdminType.

Other Requirements:

- There will be sample Guests.txt and Admin.txt files provided. These are to get you started and should also be used for the demo in Week 12. The format in these files is each of the member variables separated by a comma, with each user on a new line. The formats are as follows:
    - Guest: username, password, firstName, lastName, dd-mm-yyyy, ratingCount, averageRating
    - Admin: username, password, firstName, lastName, adminType, ratingCount, averageRating
- The first screen that is seen when the program is launched is the Login Screen.
    - LoadAllUsers() should be called when the program starts.
- After the Submit or Cancel buttons are clicked on the New User Screen, the Login Screen should be shown again.
    - A new user has an Average Rating of 0 (zero)
- Data formats:
    - Average Ratings should be rounded down to the second decimal point (e.g. 3.468 -> 3.46) before being displayed on the User List Screen and before being saved to the text file.
    - All text fields (username, password, first name, and last name) must have a minimum of 2 characters.
    - Usernames and passwords can have English alphabet characters and numbers in any order, first and last names can only have English alphabet characters and must start with a capital letter. No symbols or spaces are allowed. No other formatting restrictions apply.
- The Make Admin and Rate User Dialogs should appear in front of the User List screen and return focus to the User List screen when they are closed.
- Whenever a user is made an Admin or a new rating is left for a user, the appropriate objects in UserHandler User list should be updated and the User List Screen should reflect those updates.
    - When a Guest is elevated to an Admin role, they should be removed from the Guests.txt file and added to the Admins.txt file.
    - Note the different member variables of the Guest and Admin classes. When a Guest is elevated to an Admin role, their date of birth should be removed - the Admins.txt file should only contain the member variables in the User class and the AdminType member variable in the Admin class.
- Whenever a new user is added, a user is made an Admin, or a new rating is left for a user, SaveAllUsers() should be called.

Marking Guide:

Below is the marking guide for this assessment. It is designed to allow you to get a Pass grade with minimal effort while still demonstrating that you understand the core principles of .NET development, to get a Distinction with reasonable effort, and to get a High Distinction with solid effort, and 100% with considerable effort. It is recommended that you pay attention to the grade distribution and work towards your own skill level.

In the demos in the lab, your code needs to be compiled in Visual Studio and then the tutor will test for normal functionality as described in the descriptions above. If your code does not compile you will receive zero marks.

Note that no marks are awarded for visual design, rather you only need to demonstrate that you understand the functionality of various components in a Windows form and not the user experience design theory associated with them. However, your forms should be easily readable and usable by your tutor.

Task	Items	Max Points	Grade (Assuming full points above this item)
Code Design	Includes high cohesion and low coupling for classes and methods, using properties, using enumerations where appropriate, etc.	**3**	**9%**
Code Quality	Includes proper indenting and white spacing, helpful comments and meaningful class/method/property/field names.	**3**	**18%**
Functionality (in order of recommended implementation, e.g. start with items higher on the list)	Required Methods, Methods, and Member Variables are correctly implemented and code compiles without error	**12**	**52%**
	Login Screen – successful login and failed login with sample Users hardcoded in UserHandler class	**3**	**60%**
	New User Screen – successfully add new user to UserList and allow that user to then login	**3**	**69%**
	Doing the above by correctly reading from and writing to text file instead of a hardcoded UserList	**2**	**75%**
	User List Screen (for Guest) showing correct information.	**3**	**83%**
	User Rating Dialog working correctly.	**2**	**89%**
	User List Screen (for Moderator) showing correct information.	**1**	**92%**
	User List Screen (for SuperAdmin) showing correct information.	**1**	**95%**
	Make Admin Dialog working correctly.	**2**	**100%**
	Total	**35**	

Additional Information:

Assessment Submission

Submission of your assignment is in two parts. You must upload a zip file of the C# solution to UTS Online. This must be done by the Due Date. You may submit as many times as you like until the due date. The final submission you make is the one that will be marked. If you have not uploaded your zip file within 7 days of the Due Date, or it cannot be compiled and run in the lab, then your assignment will receive a zero mark

PLEASE NOTE 1: It is your responsibility to make sure you have thoroughly tested your program to make sure it is working correctly.

PLEASE NOTE 2: Your final submission to UTS Online is the one that is marked. It does not matter if earlier submissions were working; they will be ignored. Download your submission from UTS Online and test it thoroughly in your assigned laboratory.

Queries

If you have a problem such as illness which will affect your assignment submission contact the subject coordinator as soon as possible.

Dr. William Raffe
Room: CB11.07.128
Phone: 9514 4828
Email: William.Raffe@uts.edu.au

If you have a question about the assignment, please post it to the UTS Online forum for this subject so that everyone can see the response.

However, for frequently asked questions a FAQ file will be put up on UTS Online. Please check this before emailing the coordinator with a question.

If serious problems are discovered the class will be informed via an announcement on UTS Online. It is your responsibility to make sure you frequently check UTS Online.

PLEASE NOTE 1: If the answer to your questions can be found directly in any of the following

- subject outline
- assignment specification
- UTS Online FAQ
- UTS Online discussion board

You will be directed to these locations rather than given a direct answer.

Assignment Errata

It is possible that errors or ambiguities may be found in the assignment specification. If

so, updates will be placed on UTS Online and announcements made regarding the amendment. It is your responsibility to keep up to date on such amendments and ensure you are using the latest version of the Assignment Specification.

Acceptable Practice vs Academic Malpractice

- Students should be aware that there is no group work within this subject. All work must be individual. However, it is considered acceptable practice to adapt code examples found in the lecture notes, labs and the text book for the assignment. Code adapted from any other source, particularly the Internet and other student assignments, will be considered academic malpractice. The point of the assignment is to demonstrate your understanding of the subject material covered. It's not about being able to find solutions on the Internet.
- Participants are reminded of the principles laid down in the "Statement of Good Practice and Ethics in Informal Assessment" in the Faculty Handbook. Assignments in this subject should be your own original work. Any collaboration with another participant should be limited to those matters described in the "Acceptable Behaviour" section. Any infringement by a participant will be considered a breach of discipline and will be dealt with in accordance with the Rules and By-Laws the University. The Faculty penalty for proven misconduct of this nature is zero marks for the subject. For more information, see https://www.uts.edu.au/current-students/current-students-information-faculty-engineering-and-it/study-and-assessment-0

Extensions and Special Consideration

In alignment with Faculty policies, assignments that are submitted **after the Due Date will lose 10% of the received grade for each day**, or part thereof, that the assignment is late. Assignments will not be accepted after 5 days after the Due Date.

When, due to extenuating circumstances, you are unable to submit or present an assessment task on time, please contact your subject coordinator before the assessment task is due to discuss an extension. Extensions may be granted up to a maximum of 5 days (120 hours). In all cases you should have extensions confirmed in writing.

If you believe your performance in an assessment item or exam has been adversely affected by circumstances beyond your control, such as a serious illness, loss or bereavement, hardship, trauma, or exceptional employment demands, you may be eligible to apply for Special Consideration (https://www.uts.edu.au/current-students/managing-your-course/classes-and-assessment/special-circumstances/special) .

Return of Assessed Assignment

It is expected that marks will be made available 2 weeks after the demonstration via UTS Online. You will also be given a copy of the marking sheet showing a breakdown of the marks.