Alexander Puttre

Assignment 2

5/24/23

(Yasmina’s comments in red)

(Quang comments in Green)

Waterfall:

1. Requirements and Definition

Determine what specifications the system must meet. This information will come from the clients. How many students should be supported? Teachers? Admins? What operations should teachers and students be able to perform? What privileges should admins have? How many separate courses and sections does the system need to support? What information should the teachers and students have access to, and how should it be presented in the GUI? We should also implement security measures to protect the user’s information and think about all the error cases (For example, when a course is at full capacity or if there is a time conflict in the schedule...)

Good way to start of on what we want to provide to our clients.

5/30 - 6/13 (2 weeks) very creative and helpful to have the dates for each phase.

2. System & Software Design

Here, we collectively decide how the system is going to function. What classes are there (if we're using OOP) and how are they related? Which back-end functions will implement what front-end functionality? How will user information?

and information about the courses be stored, e.g., what data structures will be used and how will they be stored on the disk for later retrieval?

Should investigate python’s super classes to use. They’re helpful to use in the entire library. Helps in

6/13 - 6/27 (2 weeks)

3. Implementation and Unit Testing

Using the design decided upon in step 2, implement all the components and make sure they function correctly in isolation. The work should be divided equally between all members of the group, so we should decide beforehand who is going to implement what parts of the system.

Splitting the work into all of us is a good thing to have so nobody has the burden to do all the work.

6/27 - 7/18 (3 weeks)

4. Integration and System Testing

Integrate the components we implemented in step 3 and make sure they function correctly as a system. Can users, teachers, and admins perform the operations they need to with the GUI we created? Is data modified, stored, and retrieved as it should be? If we were rigorous enough in designing the components in steps 2 and 3, this should require minimal modification of the components themselves. Do not forget to evaluate the performance of the software itself (crashes, time, errors...)

We should also be stress testing at this point.

7/18 - 8/1 (2 weeks)

5. Operation and Maintenance

Deploy the system to the users and wait for feedback. If any users report that the system is not functioning correctly or that they are dissatisfied with the GUI, then we can make changes as needed. We can also create an FAQ and tutorial page for new students, or instructors to inform them on how to properly use the tools provided by the software.

8/1 - end of support

Incremental

Pre-development phase:

Decide what specifications the system should meet. This is the same as step 1 of the waterfall method.

Phase 1:

The system should display some basic functionality in phase 1. Admins, students, and teachers should all be implemented. Students should be able to add and remove classes and teachers should be able to do the same for the courses they teach. Admins should be able to add and remove users from the system and change the schedules of students and teachers. In place of a GUI, users can interface with the system through a CLI. User registration is an important step here, everyone should be able to create user accounts to access the software.

We can try to have only student that are part of Wentworth be also to register. (Ex: No one outside of Wentworth should be allowed to register. We can verify students by their given WIT ID’s that admins have made)

5/30 - 6/27 (1 month)

Phase 2:

The GUI is implemented and users can perform all of the functions they were able to perform in phase 1 using it. Additionally, students can subscribe to waiting lists for full classes. Make sure to create error messages if a student is trying to register for a class that’s already full, time conflicts in schedules, access restrictions and prerequisites. We can also create a waitlist for students to join once a class is at capacity limits.

Maybe implement a auto generated schedule that will eliminate time conflicts, and course restrictions (Ex: full classes).

6/27 - 7/25 (1 month)

Phase 3:

Users can view and print out their course schedules in a well-formatted week view. They can also subscribe to notifications, so they know when registration begins and when classes begin. At this point, the system is fully functional and development is finished. We can implement notifications and reminders to keep students informed about critical dates, deadlines, payments...

Having students subscribe to notifications for important events is very good!

7/25 - 8/6 (~ 2 weeks)

Integration and Configuration

(Several steps are the same as in the waterfall method. These steps are skipped here.)

2. Component Analysis

The system has two main components: the GUI and the sever. For these components, we can use the following existing software:

GUI: PyQT5 (pypi.org/project/PyQt5)

PyQt5 is a graphics library for Python. This is convenient for several reasons: since we already intend to implement the system in Python, using a Python graphics library will make creating the GUI easier.

Additionally, Since PyQt5 is open-source, we won't need to pay any license fees. Check if this library allows us to integrate the database easily into the software and if multiple users will be able to use it simultaneously.

PyQT5 has over 35 extensions that we can use to our advantages in making our GUI.

Database: SQLite (qlite.org/index.html)

SQLite is an industry standard database software. It has been in active development since 2000, so we can be confident that it is fully featured and well supported. There are libraries which allow SQLite to be used with Python (such as sqlite2 docs.python.org/3/library/sqlite3.html) which makes it a good candidate since Python is our language of choice. Additionally, SQLite is in the public domain.

6/13 - 6/27 (2 weeks)

3. Requirements Modification

Do SQLite and PyQt5 allow us to implement all of the functionality we intended the system to have? If not, what changes can we make while preserving the important functionality of the system?

If both don’t allow us to implement all the functionality, we can try to work around it by asking beta users to give some feedback. Maybe that functionality wont need SQLite or PyQt5.

6/27 - 7/11 (2 weeks)

4. System Design With Reuse

This step is similar to phase 2 of the waterfall method, only we design the system with the understanding that we intend to implement the database using SQLite and the GUI using PyQt5. Store all the necessary information within this step.

Good idea to have us store info in the databases.

7/11 - 7/18 (2 weeks)

5. Development and Integration

Here, we write the software, integrating SQLite and PyQt5 into the project. Find the errors and create error messages for the users.

Create user feedback page.

7/18 - 8/1 (2 weeks)

# Process Model Choice:

# I believe that the incremental development method is the most appropriate process model for our purposes. In the Waterfall and Integration & Configuration models, there is a considerable amount of time devoted to research and creating design documents, whereas in the incremental model time spent schematizing is kept to a minimum, leaving more time for development. This makes the incremental model the most appealing of the three given our limited time frame.