**Process Management Report**

Due Monday, February 17, 2020, before class

Team 4 (Lina McDermott, Caroline Sigl, Joe Kane, Mary Lasater, Liam Carriker, Mariem Bchir)

**Introduction:**

This report describes Team 4’s management plan for its project to develop a mobile application to connect teachers and schools in Northern Africa.

**1.**     **Expected Level of Effort in Person-Months**

Below is a table of our expected LOC for each of the modules of our project.

|  |  |
| --- | --- |
| Login | 300 LOC |
| Account Management | 400 LOC |
| Database | 400 LOC |
| Search Interface | 300 LOC |
| UI/App Interface | 200 LOC |
| Matching Algorithm/Interface | 400 LOC |
| **Total:** | ~2000 LOC |

Additionally, we have used the COCOMO calculation and obtained the following results:

A screenshot of a social media post

Description automatically generated

**2.**     **Overall High-Level Schedule for the Project**

Our calculations are as follows: M**≈**14.667/6 **≈** 2.445 months. We think this is how long it will take to develop a first version of the application, though we acknowledge that the app will likely be far from production-ready at that point.

**3.**     **Quality Plan**

We are aiming to produce a well tested and professional app. In order to produce this result we will have to ensure that our app meets all test cases. First, we will test the main functions of our app such as searching, filtering and matching. This process will be very similar to the idea of unit testing in which smaller pieces of code are tested individually. As part of our quality plan, we will make sure that each line of code that can be reasonably tested is tested before submitting. Tests will not be an afterthought in the development process. If, along the way, we are measuring quality and it is not up to our standards, then we may need to implement more rigorous rules for unit testing. Then, depending on how mature our software is, we will conduct integration tests in which groups of software modules are tested together. These tests will be written as each feature is completed. When we are confronted with errors we will decide as a group if they are acceptable. In terms of documentation, we will keep a collaborative design document that details the objectives and structure of each major software module. This way, the team is held accountable for the quality of each piece of the project before any one member begins to develop it.

Within our code, each step will be commented thoroughly. Additionally, we will have code reviews in which the responsible team member will enlist the help of another team member to make sure that the new code will not negatively affect the integrity of the other modules in our app. This will also ensure that each member of the team is not stuck inside of their own development “bubble”; we will instead be responsible for the application’s success as a team. Code reviews can be highly beneficial for both the reviewer and the reviewee. Ideally, these steps will help us ensure quality in our final product.

**4.**     **Risk Management Plan**

Below is a table outlining our potential risks:

|  |  |  |
| --- | --- | --- |
|  | Probability of Occurrence | Loss to Project |
| Development of the Wrong Software Functions | Medium | High |
| Unavailability of Personnel | Low | High |
| Unrealistic Schedule | High | Medium |
| Gold Plating | Low | Medium |
| “Unfriendly” user interface | Medium | Low |

The three most critical risks are developing the wrong software functions,      unavailability of personnel, and unrealistic scheduling. To mitigate the possibility of developing the wrong software functions, we will dedicate more time to generating the software architecture of the project. This should ensure to the best of our ability that the software’s architecture and functions make sense before spending time on coding each function/section. Ideally, the collaborative nature of the team should ensure we have thought about each module as much as possible before embarking on the creation of it. We can mitigate unrealistic scheduling by validating our schedule with various tools like COCOMO II and more experienced software engineers (i.e. Dr. Locke). It is not possible to mitigate the unavailability of personnel. Each developer has other classwork and extracurriculars to participate in, which can be dealt with via efficient planning of personal schedules. There are also not many additional ways to mitigate situations such as getting sick, outside of common sensible precautions.

**5.**     **Project Monitoring Plan**

We believe that recording and measuring the number of lines of code, features, and unit tests conducted will create a holistic picture of our progress on our application. We would like to set the goal of producing 225 lines of code per week, 1.5 features per week, and 7 unit tests to run per week.

If we are not meeting these goals we will take corrective actions to get back on track. If we are close to meeting the goal we will put in “overtime” to reach the milestone. Alternatively, we could redefine our goals if we realize that our goals are not reachable anymore.

If we are not on track for a goal we will take corrective actions to get back on track, including:

1. If there are a few errors left in the current code, that could be easily solved in a few hours, then we’d first do overtime to complete the milestone before changing the milestones. This is also the step for when someone needs to leave early for a certain reason.
2. Readdressing the current milestone to make it possible to complete on time, as well as changing the remaining milestones. (This is all dependent on time left to complete a certain milestone.)

**6.**     **Detailed Scheduling**

Provide the most detailed schedule you can at this time.  State how often you will update this detailed schedule through your development cycle.

* Detailed schedule : Starting from February 22nd
  + Week 1 : Defining and Analyzing requirements
  + Week 2 : Design and Architecture
  + Week 3 : Design and Architecture
  + Week 4 : Implementation and Testing
  + Week 5 : Implementation and Testing
  + Week 6 : Implementation and Testing
  + Week 7 : Implementation and Testing
  + Week 9 : Implementation and Testing ( Prototype V 1.0)

Each of the implementation and testing weeks will include analysis of the requirements at the beginning of that week, with the possibility to redefine before continuing work. Essentially, we should complete one iteration of development each week. We will aim to update this schedule weekly, filling in more detailed items for each week, specifically for weeks 4-9.