Team F

High Level Requirements For the “when2meet” program

1. Introduction

Dev Ninjas: Gwen, Aldo, Edward, Keith

1.1 The Purpose

This projects mission is to aid in the planning of group meetings. Often it can be cumbersome to organize a meeting time for people with wildly different (and sometimes conflicting) schedules. We aim to help find a time slot everyone can make and comfortably if at all possible.

1.1.1 Scope: To provide a platform for businesses to schedule meetings without multiple email chains.

1.1.2 References:

1.1.3 Overview:

2. Overall description

2.1 Product Perspective

2.1.1 System interface.

2.1.2 User Interface. See figures ()

2.1.3 Hardware Interface. Keyboard and mouse.

2.1.4 Software Interface. SQL

2.1.5 Communication Interface.

2.1.6 Memory Constraints. 1 GB of RAM.

2.1.7 Operations.

2.2 Product Function

- The program will return a time frame which maximizes the number of attendees. Meaning that the program will search for a time frame in which most people are unoccupied for a period of time.

2.3 User Characteristics

- Languages supported: English.
- Computing Skills: Basic knowledge of MS Outlook.
- Physical and Social Environment: This program will be appropriate for both corporate and family settings. Will take users' position in company/social group into account.

2.4 Constraints

- Must be human as per the Geneva convention of 1915 and pass a captcha.
- Must have at least Windows 10 1803 or Mac OS 10.14.6 and later.
- Work on Visual C++ 2015 and GNU gcc 9.2.0

3. Specific Requirements

3.1 External interfaces

3.2 Functional requirements

-organized by feature, object, user class, etc.

3.3 Performance requirements

3.4 Logical database requirements

3.4.1 MariaDB

3.5 Design constraints

3.5.1 Standards compliance: Github repo will follow the linux Filesystem Hierarchy Standard 3.0 (FHS).

3.6 Software system attributes

3.6.1 Reliability

3.6.2 Availability

3.6.3 Security: In order to log in, make changes to submitted schedule, or raise an exception the user must log in and or make an account.

3.6.4 Maintainability

3.6.5 Portability: High level functionality must be written to be platform agnostic. That is when possible the program will avoid using non-standard libraries.

3.7 Organizing the specific requirements

3.7.1 System mode - or

3.7.2 User class - or

3.7.3 Objects (see right) - or

3.7.4 Feature - or

3.7.5 Stimulus - or

3.7.6 Response - or

3.7.7 Functional hierarchy – or

3.8 Additional comments

Questions

- Can we use Google or MS Outlook API's? Where should the be in the requirements?
- Should the requirements be written in full sentences?
- Should the images be placed in section or at the end?
- What is doodle poll missing?
- What email protocol should be used?

Lessons Learned

- We can use pre-made libraries or API's, but must be cited and should still be a standalone app.
- Requirements are to be written in complete sentences.
- Images can be placed in appendix.
- Doodle poll is okay, but I have to pick the time either way.
- What ever email thing works.