CS310 Assignment #1 - Requirements Specification

**Deadline:** 09/18, Monday, 11:59:59pm

**Late policy:** Late submissions lose 1% of the total available points for each hour the assignment is late. Assignments submitted more than 48 hours after the original deadline will not be accepted.

**Submission:** All submissions will be via D2L. Each team only needs to submit one PDF file under “My Tools → Assignments → Assignment #1”. (“Assignment #1” will show up after all the teams are settled)

**Team size:** 5 students

Assignment Description

Your team will work on building a system of your choice (see Project Description in the end of this document) throughout this semester. In this assignment, you will focus on the requirements for the system, and will deliver a requirements specification using one or more of the notations discussed in class on September 6, and in Chapters 4 and 5 of the textbook.

Each system has a customer (one of the TAs), who has several requirements for the system that she/he wants you to meet. You will work with your customer closely throughout the semester and your goal is to deliver a project that satisfies your customer. For this assignment, you will have one class session, on September 11, to obtain the initial user requirements and to elicit the necessary information to convert them into system requirements (recall the lecture of September 6 and Chapter 4). You may subsequently use email and/or visit your customer’s regular office hours to clarify and refine those requirements further, as necessary. Please allow for a 48-hour response time during weekdays if you elect to email your TA specifically regarding project requirements. Please be sure to begin the subject of the email with “CS 310 - Assignment 1”. You may even schedule a separate meeting with your customer if you can demonstrate that there is a special need or circumstance. Your customer (and, if necessary, the instructor) will determine whether there is an actual need and justification for such a meeting (e.g., just because you got a late start on the assignment would not be a satisfactory reason). You will be able to get further clarifications based on your team’s need and any other feedback in class, on September 13. Note that the class time on September 13 is limited and is based on “first come, first serve”, so please come to class prepared with questions that you want to ask your customer.

Please note that

1. your customer may not have the complete technical and/or domain knowledge, and thus she/he may be unclear in her/his requirements;
2. your customer may change her/his requirements along the way; customers change their minds and sometimes are guided by the IKIWISI (“I'll know it when I see it”) principle;
3. your customer may have conflicting requirements without realizing it; and
4. your customer may have unreasonably hard-to-implement requirements.
In this particular assignment, your job is to communicate with your customer and elicit the requirements and complete the specification. Please feel free to negotiate the requirements with your customer, but be aware of your competitors (other teams) - if you cut too many corners to be able to complete the project, your customer may “purchase” another team’s solution.

**Deliverable**

Your team will deliver a requirements specification document. The document should be as complete, precise, consistent, and unambiguous as possible (subject to the typical limitations, as discussed in class on September 6, and in Chapter 4 of the textbook). As a baseline, you should use a combination of *plain English*, with *numbered paragraphs* indicating different requirements and their sub-requirements (as discussed in class on September 6 and in Chapter 4), and *at least one type of diagram* (as discussed in class on September 6 and in Chapter 5). You are likely to find that diagrams are a convenient way of capturing requirements information and that you can use more than one type of diagram. While you are not required to use *formal/mathematical notations*, correct use of them can be an effective way of conveying requirements and will be considered for extra-credit in this assignment. Finally, you are supposed to come up with at least *five scenarios/user stories* in the description of your system (as discussed in class on September 6 and in Chapter 4).

Your requirements document must have the following sections. These were discussed on slides 57-58 of the September 6 lecture and in Chapter 4 of the textbook.

1. Project Title and Authors
   a. Your assigned team number
   b. Optionally a name you select for your team
   c. A list of all team members (names and USC ID numbers)
2. Preface
3. Introduction
4. Glossary
5. User Requirements Definition
   a. Scenarios/user stories
6. System Requirements Specification (all of the below)
   a. Functional Requirements
   b. Non-functional Requirements
   c. UI Appearance Requirements
7. System Models (one or more of the below)
   o Context Diagrams
   o Process Diagrams
   o Class Hierarchy Diagrams
   o Aggregation Diagrams
   o Use Case Diagrams
   o Sequence Diagrams
   o State-Machine Diagrams, etc.
Project 1 (Customer: Yixue Zhao)

**Description:**
College students spend most of their time on campus, taking classes, working on projects, participating in various activities, etc. In this context, dining on campus is convenient and is the preference of most students. However, the dining options on campus or even around campus are limited and the wait time is always long during typical meal hours. Likewise, students who cook may find it too difficult and time consuming to cook healthy choices for a single person or a very small number of people (themselves, possibly a roommate). Thus, your job is to build an **Android application** - *GrubMate*, to create a community for people who have this problem to help each other.

*GrubMate* allows the users to post the food they want to share to their friend circle and to request the food they want to eat at a specific time. Your app should be able to support at least the following scenarios.

1. **On Sunday evening**, Yixue cooks delicious pasta for 5 people and she wants to share part of the food with her friends who may want pasta on Monday. So she posts a description and photos of her dish on *GrubMate* and specifies four available share-portions and the pick up time/location.

2. **By Tuesday**, all the pasta is gone, and Yixue wants to eat a poke bowl for lunch. She subscribes to “poke bowl” in the morning and gets a notification informing her that her friend Shushan will go to a poke bowl bar near campus between noon and 1:00pm, and can bring back 3 more poke bowls. Yixue requests her poke bowl on Shushan’s post and indicates her location. Shushan confirms Yixue’s request after she makes sure that Yixue’s location is on her way back, and she delivers the poke bowl to Yixue.
Project 2 (Customer: Shushan Arakelyan)

**Description:**
*Focus!* is an Android application that allows blocking distracting applications and their notifications for a predefined amount of time. Applications and notifications can be blocked for anywhere from 10 minutes to 10 hours. *Focus!* also allows users to define several profiles. Each profile has its own list of blocked applications and notifications. The user should be able to give a custom name to every profile.

- **Example 1:** A profile can be set up to block emails, text messages, and calls. This profile can be called “Do not disturb mode”. An example use case for this profile is turning it on overnight.

- **Example 2:** Yet another profile blocks Facebook, Snapchat, and Instagram. A sample name for this profile can be “Working without distraction”. This profile can be used during working days or class hours.

The profiles can overlap in time to avoid having unnecessarily many profiles. *Focus!* should apply restrictions from both profiles for the time period when they overlap.

- **Example:** If there already is an active profile for “Working without distractions”, it should also be possible to activate the “Do not disturb mode” profile for the duration of important team meetings and discussions to avoid unnecessary interruptions.

*Focus!* lets its users to schedule automatic profile activation during certain hours of certain days.

- **Example:** Every Monday and Wednesday from 10am to noon for the CSCI 310: Software Engineering course.

After the blocking time has run out, *Focus!* should show the list of all notifications that were blocked by it during that period.
Project 3 (Customer: Sarah Cooney)

Description:
From credit and debit cards to electronic services like Venmo and Apple Pay and even our USC ID cards, spending money without cash is easier than ever. However, it can be difficult and inconvenient to keep track of one’s spending. I would like a simple mobile app - $anity - to create a budget across a variety of categories to which I can easily add when I spend money.

Example: I might set up a monthly budget allocating $200 for food, $50 for entertainment, $100 for utilities, and so on. If I buy groceries for $40, I should be able to fill out a simple form to add this to my food budget. The amount should then be deducted from my remaining total.

The budget should reset after a given period (e.g., a week or a month), which can be specified at setup.

Example: I get paid on the 26th of every month, so I should be able to set up a budget that will reset on the 27th of every month.

$anity should let me know when I am getting close to my budgeted amount in a certain category, and remind me how much longer I have until my budget period is over.

Example: If I have spent $150 of $200 allocated for food by the 15th of the month, I should get a notification stating, “You have spent 75% of your food budget for this period. Your budget period has 12 days remaining.”

Other Desirable Features:
- Ability to set budgets with multiple time constraints. For instance, a month-long budget for spending from my personal bank account, and a semester-long budget to keep track of Dining Dollars spending.
- Automatic Integration with USC Card, Venmo, etc.
- Ability to scan my receipts instead of entering values manually.
Project 4 (Customer : Umang Gupta)

Description:
We need a rating (crowd-sourced voting) system for USC. From food and coffee, to clothing, to classes, many of us rely on our friends’ suggestions. However, we still end up at wrong places because each of our peer group has limited experience. You should develop KnowItAll, a **crowd-sourced rating-system**, where anyone with a valid USC ID can post reviews/ratings on all sorts of things - classes, professors, events, food, amusement parks, movies, concerts, hikes, etc. KnowItAll should have **guaranteed anonymity** for the users.

Major features of KnowItAll include:
- Only people with a valid ID should be able to rate/review entities with guaranteed anonymity only once.
- Entities should be searchable in a meaningful way both for lookup and to rate.
- A web application is a must, better if it works well on mobile, even better we package it to an app.
- Entities could be anything, application should be designed in a way that it can cater to all USC students.
- Type of Ratings/question :
  - **Polling questions** --- One could set up a polling question that may have any number of options.
    - Ex: Which Pizza place is best?
      - Blaze Pizza
      - Pizza Studio
      - Dominos
      - Pizza Hut
  - Rating questions --- One could set up a rating question?
    - Ex: How do you rate CSCI 310?
      - (Rating could be at a 10 point scale)
- Life of a poll/rating:
  - Rating may be open forever,
    - Ex: reviewing a restaurant or cafe.
    - Result for such polls would be visible to everyone always.
  - Ratings may be open for limited period. Results of such polls may or may not be visible to the users while voting. For ex: Rating a course after the end-terms
  - All polls should be visible forever, unless deleted by the creator.
- Every poll/rating should have a unique link (not necessarily meaningful), so that it can be shared.
- Any valid user can create the poll and he/she must be able to configure type and life of poll/rating.