# Shopping Expert

Yukai Tan, Cheng Tian

Feb 5, 2020

## INTRODUCTION

An offline shopping store recommendation software. We would like to display brief information on the map. Let people better understand stores that they don’t know or have never been to. Based on the map, we have two display modes. When there are more than one stores within the map range, our software will list all clothing stores, shoe stores and bag stores. Giving some brief description and tags of the store at that time. After zoom in or zoom out, when only one store is displayed within map range, our interface will show up comprehensive information of that store. Describe what type of the store is , what is the highlight of this brand, who (celebrities) often visited, give some icons and examples.

## **ROLES AND RESPONSIBILITIES (required)**

These vary for each type of product and, for small projects, folks may serve multiple roles.  This is a list of common roles we have used for software development:

Development Lead

Cheng Tianbv

Buildmeister

Yukai Tan

Architect

Cheng Tian

Developers

Yukai , Cheng

Test Lead

Yukai

Testers

Yukai , Cheng

Documentation

Yukai , Cheng

Documentation Editor

Cheng

Designer

Yukai

User advocate

Yukai

Risk Management

Yukai

System Administrator

Cheng Tian

Modification Request Board

Cheng

Requirements Resource

Yukai

Customer Representative

Yukai , Cheng

Customer responsible for acceptance testing

Cheng Tian

## METHOD (required)

These are unique to software development, although there may be some overlap.

- Software:
  - Language(s) with version number including the compiler if appropriate
    - python , java
  - Operating System(s) with release number
    - ubuntu
  - Software packages/libraries used with release/version number
    - Android Studio 3.5.3
    - Docker 18.03.0-ce
    - Flask flask 1.1
    - MySql
    - Github
  - Code conventions – this should preferably be a pointer to a document agreed to and followed by everyone
- Hardware:
  - Development Hardware
  - Test Hardware
  - Target/Deployment Hardware
- Back up plan (individual and project)
  - Github is our code repository
- Review Process:
  - Will you do architecture, usability, design, security, privacy or code reviews?
    - Our team will do requirements elicitation to identify and discover user requirements and then requirements analysis. After that we will design our software including  architecture, usability, design, security, privacy.

    - What approach will you use for the reviews (formal, informal, corporate standard)?
        - Since we are a small team, we do informal code reviews.
    - Who is responsible for the reviews and resolving any issues uncovered by the reviews?
        - All the team
- Build Plan:
    - Revision control system and repository used
        - Github
        - https://github.com/Tc-blip/SSW695
    - Regularity of the builds – daily
        - We plan to perform branch merging every day, and each team member will have a branch for the master branch
- Modification Request Process:
    - MR tool
        - Github issue post
    - Decision process (board – if more than paragraph should point to alternate description)
        - Each team member should post an issue, the developer will review the issue and comment on the issue.
        - If the solution finds that the problem does exist, and all developers return to the version where the problem was found to solve, the project leader will merge solution.
    - State whether there will be two process streams one during development and one after development
        - During development
            - Each team member should post issue, if issues solved merge into project
        - After development
            - We will collect software issues and release iterations at intervals

### Virtual and Real Work Space

We will use google docs to be our document repository

# COMMUNICATION PLAN (required)

## “Heartbeat” meetings

This section describes the operation of the “heartbeat” meetings, meetings that take the pulse of the project. Usually these meetings are weekly and I prefer to have them early in the day before folks get into their regular routine, but this is not necessary. The meeting should include only necessary individuals – no upper level management or lurkers. It should have a set agenda, with the last part of the meeting reviewing open issues and risks. It should be SHORT, thirty minutes or less is ideal. Notes should be provided after the meeting and issues should be tracked and reviewed each meeting, usually at the end.

We will set the “Heartbeat” meeting to be on Thursday evening after class. Because after discussing our project with the professor, we will discuss the problem risks of the project last week, and evaluate the risks of last week's risks to the sprint next week.

## Status meetings

Status meetings have management as their target and should be held less frequently than heartbeat meetings, preferably biweekly, monthly or quarterly depending on the duration of the project. It is solely to provide status for the project. If issues arise, they should be addressed at a separate meeting (see next item). These should be short. This section should describe the format and periodicity.

We will try to hold the status meeting every week, set every Wednesday, because every Wednesday is sprint end day, we will see our current sprint, whether it has reached our sprint plan, and whether there are technical owes Debt, if there is technical debt, do we need to stop the next week's sprint and resolve it

## Issues meetings

If a problem does arise, never surprise your manager. Schedule a meeting at his or her earliest convenience. This section describes how alerts will arise and the governance of when to trigger an alert – usually after a discussion at the heartbeat meeting.

Issues meetings happens if we need to resolve our technical debt and we need to stop the next sprint, we will hold issues meetings and discuss how to solve it

## TIMELINE AND MILESTONES(required)

This section should be crisp containing 4-10 milestones for the duration of the project, each of which would trigger a re-issuing of this document to report on progress. Each milestone should define a 100% complete item, should list the critical participants and list begin time and end time. Each time you re-issue this document you should highlight changes with italics or bold – colors will not show up on a photocopy.

Note that for this project we have a few time boxes. They are:

- Week of February 6nd – description of first demo

  Complete the project story and have an interface prototype

  Setint up testing environments

- Week of February 16th first demo, description of second demo

  We should have a small function, and it is a function that integrates the front end, back end and database, so we have a basic framework

- Week of March 2nd second demo, description of third demo

  Adding insert and delete functions

  Adding search function

  Testing functions

- Week of March 23rd third demo, description of fourth demo

  Adding click the icon to display the information function

  Adding display content frame(including details display and brief)

  Testing functions

- Week of April 6th fourth demo, description of final product

  Add some low-priority features

- Week of April 23th final product

Complete documentation

Runnable programs

## TESTING POLICY/PLAN (optional–software relevant)

This should probably point to a plan or the document would get unwieldy.  At the very least it should describe when testing begins.

The test starts from the completion of the first function, we will try to add some automated tests with each sprint, through the automated integration tool (Travis CI)

## RISKS (required)

Ideally it occurs because of an established risk management program.  If that does not exist, do the best you can to enumerate the risks and explain how they will be track and monitored.

Our biggest risk is we have a small team.  We tried to use docker to deploy and develop our software, this is our first time using this tool. We tried to become familiar with docker as much as possible in the early stage.

## ASSUMPTIONS(required)

It may be clear to the project insiders what assumptions are being made about staffing, hardware, vacations, rewards, … but make it clear to everyone else and to the other half of the project that cannot read your thoughts.

We are a collective, and all people involved in software development should devote all their current energy to development, so that a united team can win the ultimate victory. We will allocate benefits to technical staff as much as possible, because they are the most hard-working personnel in software developing.

## DISTRIBUTION LIST

Who receives this document?

## MORE OPTIONAL SECTIONS:

These should be self-explanatory.

### Worry beads

I add this section to describe the things as manager I am most worried about at the time of latest document issue.  This section is useful because it helps you to focus on the parts most likely to fail.  Sometimes, I segment the worries by time scale: day, week, month, quarter … lifetime.

### Documentation Plan

Many years ago we had much too much documentation, now we have precious little – this has to change.  Write documentation as if you'll need to personally support the project forever – you just might need to and you'll be glad you took the time to document the obvious, the not so obvious and the obscure.  As an example, it's useful to document alternate architectures and designs you did not pursue along with the rationale.  What were the "gotchas" you were trying to avoid?

We try to avoid changing the development tools after the software development is halfway. We should do good technical research and decide on our appropriate technology before developing.

### Build Plan

When builds and testing become complex, this might be a separate section or point to a separate document.