CSE 201 Project Specification and Proposal

Example: The App Catalog (Codename:Meta-App)

Version 8.0

Spring 2022

Contents

Executive Summary

This document is a project proposal for an application catalog that the CSE 201 Project teams are to create. The idea of the project is for the teams to create a one-stop repository anytime someone wants to discover new applications and get information about them. You can essentially view it as a meta-application, that is, an application about applications.

The standard expectation is that you make it a local (non-web) based program. However, there will be a bonus for those of you who are able to develop a web application, even if it is just able to run locally. Either way, your program will require a front end, a back end, and application logic.

Remember that the TAs and Professor are acting in the role of customers here. We are also here to mentor you. What we are not going to do is design your systems for you, nor tell you which technologies to employ. That is up to you. We will not be providing technical support. So, it is up to you if you want to challenge yourself to use a new technology on your own so you can add it to your repertoire/resume.

As mentioned in class, **you are able to make a catalog for something other than apps**, but it must realize all the features described in this specification.

Project Overview

The amount of applications, both web and mobile, is growing at an incredible rate. Many platforms have their own "stores", such as the Apple Store, Chrome Web Store, or Google Play. These are useful when one knows exactly what they are looking for and for what device. However, there is no single location (library) for both application users, developers, and researchers to browse through, compare, purchase, and discuss applications. For example, if I wanted to contribute to the "Alarm Clock App" domain by creating a new application and wanted to see all "Alarm Clock" applications in existence, it would be very onerous to amalgamate all that information for me to consider.

In this project, we will be creating a repository of applications. That is, you can view it as a library of applications that is sortable and filterable based on different criteria.

The Meta-App is different from platform-specific stores in that it provides more information, includes user discussions, is user-driven, and allows cross-platform comparison.

For bonus credit, you can create a web-site based (accessible from either desktop and/or mobile app) application. We will not be providing you technical support if you do that.

Early High-Level Requirements (From Customer Perspective)

The Meta-App must be accessible through a graphical user interface.

The name Meta-App is merely a codename for this project. You must come up with your own name for your specific project implementation and for your project team.

Without logging in, a user should be able to 1) view, 2) sort, and 3) filter all the applications in the repository.

Each application (entry) in the repository should contain, AT LEAST, the name of the application; a description; the organization; the platform(s) and version(s) it can be installed to; an external link to the respective stores; and the price. *Additional features will arise (from the customer) as the project continues.* {If you do another domain other than Apps, the customers will give you replacement attributes}

There should be a consistent search bar on every page that will do a string search (and sort?) on all of the applications' respective fields.

Applications are added to the repository by having users submitting a request form to add an application. This must include an external link to the platform-specific application page that must be verified (manually at this point) by a System Administrator (Admin). Thus, requests are in standby until approved by an Admin.

Admins should have access to an outstanding request page where they can see all the submitted application requests. Admins can respond and take action on them (accept or deny, both with related comments).

Moderators are Users with the ability to moderate a comment section (delete posts) for each included app. They do not have the same access as Administrators. Administrators can be moderators, however. All can be Users. Users must login in order to post on the forums.

The backend of the repository and user login information must be persisted. That is, saved so it can be used long-term. We do not require a formal database, but you may use one if you think that is best.

The Meta-App interface should be as simple as possible and adhere to common style and user interface guidelines. The repository interface should closely mimic the repository interface provided by other repository applications.

There will be other requirements that "pop up" unexpectedly in future iterations, so be sure to make your code extendable and adaptable.

Meeting Responsibilities Each Week

You should come prepared to every customer meeting with a full agenda. Realistically, you should have your agenda prepared for 2 meetings at a time: The immediate one and the one that follows.

- You are in charge of these meetings. You will be graded on how you direct them.
- You should have an agenda of things you're going to talk about, questions, et cetera.
- Make sure you email your assigned customer an agenda and all deliverables at least an hour before your meeting.
- Due to the unique nature of this semester, you can inform your TA if an individual is unavailable to attend a meeting at least 24 hours in advance for no deductions, and 1 hour before for 0.5 deductions.

Initial Week Requirements – Due Friday End of Day

- Submit to Canvas by 11:59 proof/statements of the following
    - Define a project manager (primary communication with client, iteration planning, task assignment/division)
    - Define a technical manager (manage task board technology, architectural design/technology solutions)
    - Name your Application
    - Name your Software Development Company
    - Fill out scheduling google form
    - Gather at least 10 requirements by yourself (customer validation will come next week) based on document
    - Choose a task board (virtual? portable?) and populate it with requirements. Let us know which one/send screenshot.

First Customer Meeting Deliverables(Check Canvas)

- Show your customer all the things mentioned above
- Describe your roles to the customer
- Describe your team's meeting schedule. When and what frequency? How many times per week will you be meeting?
- Validate your 10 requirements with the customer. Let them change them.
- Show initial UML Class diagrams for your requirements.
- Bluesky and elicit **more/all** requirements from your customer
- Begin and show evidence that you've put User Stories and Tasks for Iteration 1 on your board and assigned them to your specific team members.
- Show evidence that you have started setting up a code repository.

Grading

Team Score

Grading will be composed of customer meeting scores and presentation scores. You can find a larger version of this in the corresponding PDF on Canvas.

Customer Meetings

Weekly Customer Meeting Grading Scheme

	Novice (Beginner/ Needs Improvement) 5-6/10	Apprentice (Making progress, but not yet at working level) 7-8/10	Proficient (Meets Expectations for course) 8-9/10	Excelling (Exceeds expectation) 10/10
Function in a team				
Distribution of work	1 Person is doing all the work	Work is not evenly distributed	Work is evenly distributed	Self-organizing
Delivery of product (Deliverables)	Not working	Working software but missing major aspects	Working software	Managed backlog effectively, including measurement of scope/feature creep, velocity
Meeting management				
Plan	Nothing is planned beforehand. No email is sent.		Agenda is always created without prompting	Agendas are created and distributed early and 2 weeks at a time.
Organize			Logistics of meeting always handled without prompting	Excellent, professional, and smooth meeting flow as a result of organization.
Attend	Very few team members present with no explanation (beforehand) provided	Not all team members are present with no explanation (beforehand) provided	Team members are always present	Team members are present and all are on time
Conduct	Non-productive nor constructive.		Meeting is conducted smoothly, on time and to task	Student-driven meeting; driving and engaging

Presentation

3 presentations. Rubric dependent on week. Score out of 10 each week.

Bonus Modifier

If your group creates a fully functional web (version of your) application able to run on a web browser and mobile, you will be eligible to receive up to 5% bonus (points) depending on how functional/web compatible it is. For example, 1% for database, 1% for mobile capability, 1% for usability, 1% for having it be a web application, 1% customer satisfaction/functionality.

Code Reuse for Bonus Modifier

If you are attempting the bonus modifier/aspects of the project, you are allowed to use existing web frameworks and technologies (Bootstrap, JQuery, or other GitHub solutions) as a starting point. However, you must be extremely diligent in citing your work, including an explanation of how you used and modified the existing web frameworks and technologies. Ask us if you are unsure!

Peer Evaluations – Modifier of Grade

We will be using peer evaluations as a modifier of the group work score. Students will evaluate their teammates anonymously.

For teams of 5, students will have 40 points to distribute.

In a group that is functioning perfectly, each person will receive an average score of ten points. They will receive all the points the team has earned for group work.

If an individual receives an average of 8, they will receive 80% of the group score. This modifier applies to average scores of >=7 to 10. If a student receives an average score of less than 7.00, then they will receive 45% of the group score. This is to prevent any students from coasting through the project satisfied with a "C" or "D" effort. This strategy has usually stopped lazy behavior.

If you neglect to complete the peer evaluation, you will receive an additional 5% modifier penalty.

Based on Michaelsen, Larry K., Arletta Bauman Knight, and L. Dee Fink, eds. Team-based learning: A transformative use of small groups. Greenwood publishing group, 2002.