Deliverable: #5 - Project Architecture & Design
Title: SE2: Software Design and Architecture (CS 446, ECE 452, CS 646)

Deliverable Overview:

Your project proposals provided an overview of the functionality your project aims to provide; they also provided some insight into the non-functional, or quality, attributes you want your project to exhibit. In class we have described a variety of architectural styles and design patterns, each of which has both strengths and weaknesses. The intent of this deliverable is to identify, describe, and justify the architecture and design of your project. Do not be constrained by the individual styles and patterns we described in class; it is expected that a heterogenous set of styles and patterns (including ones we may not have covered in class) may best suit your project.

The written component of this deliverable will describe your project architecture (5 pages), design (5 pages), and give an overview of what each member of your team is doing (1 page).

Architecture:

The outcome of this deliverable is an architectural description of your project that describes how your system will support the functional requirements and non-functional attributes described in your proposal. Non-functional attributes should be specific enough to enable us to verify whether or not your app supports them in a measurable way. Note: a two-box client/server component diagram will be insufficient here. For example, every phone-based/external service you interact with (e.g., NFC, Geolocation, Contacts, etc.) should be clearly identified and can be treated as individual components.

The written component should clearly and succinctly capture and justify the architecture of your system. It should describe how your architecture supports the non-functional properties listed in your proposal. This description can be at most 5 pages (3 pages of text and 2 pages of diagrams). The diagrams should include at least a component and physical representations of your system.

Design:

Document your project's design. Your target audience for this documentation is a junior programmer who would be responsible for implementing some portion of the project. This would be the first thing you would give a new employee to get them up to speed on the low-level structure of your system.

Create a document that describes the design of your system, and its rationalization, such that a junior programmer could implement some subset of the system and integrate it appropriately. Your design should include a clear description of the structure of the classes and their externally-visible interfaces. Rationale must be provided documenting why you selected your design. The applicability of your design compared to alternative designs should also be referenced in this discussion.

Your design should highlight key patterns, classes, abstractions, and data structures / algorithms that are critical to the successful implementation of your system. A mapping between the design-level entities (e.g., the classes) and the components described in the architecture is also required (or should be depicted on the diagrams). Use diagrams as appropriate for this report. At a minimum, include a class diagram that shows all of the classes and public API for your system and how they interact along with a sequence diagram that captures how your system behaves for each of the scenarios from the initial proposal. Clarify the physical location of where the classes will reside (e.g., on the client, on a server), as well as any external API your system will use.

An analysis of how your design minimizes coupling and could accommodate future requirement changes is required (specifically, think about the coupling/cohesion slides from the design intro lecture on Feb 16). Think critically about how you could envision your system being altered and discuss how your design would support or inhibit evolving to meet those changed requirements. Identify one ways you think your system may need to evolve in the future and describe how your project's design would support these changes.

The written component should clearly and succinctly capture and justify the design of your system. This description can be at most 5 pages (3 pages of primarily text and 2 pages of primarily diagrams).

Written deliverable:

1. Metadata: Project title, team member names, team member Quest IDs.

2. Architectural description (5 page maximum).
3. System design (5 page maximum).
4. Description mapping team members to architectural-level components / design-level classes (1 page).
5. Only one team member needs to upload this document to Learn. PDF only.
   File naming scheme: `cs446-d5_<project-name>.pdf`
   * (use – instead of space in file names)

Oral deliverable:

1. Sign up for meeting slot online <Link will be made available shortly>. Make sure your whole team can attend. Meeting location will be in my office (DC 3349). The signup is first come, first served; please sign up using your team name.
2. The slot is 30 minutes. You will start by giving a 10 minute oral description of your system's architecture and design. No slides can be used, but you are encouraged to make use of the whiteboard and can refer to the previously-submitted written documentation.
3. The remainder of the slot will be used to discuss your architecture and design. Some questions will be for the group, others will be for specific individuals. Being able to clearly and unambiguously justify your architectural and design decisions will be fundamental to success here. Be prepared to defend your system's design. We will also likely ask about how your design could adapt to specific given evolutionary constraints (e.g., 'you must now support XXX, how would you do that?').

Assessment:

This deliverable is worth 20% of your final grade: The written component is worth 50% of the deliverable mark; the oral component constitutes the remaining 50%.