Project Overview
EC327 Introduction to Software Engineering, Fall 2020

Assigned: November 9, Monday
Due: December 7, 2020, Monday (Demos during lecture, reports by 10pm)

Total points: 100 + Extra Credit
Weight: 15% of your final grade

Goals

The goals of this **team project** are to create a unique and marketable Android app or other approved software project with the following characteristics:

- Technically proficient
- Exhibits strong software engineering principles
- Well-motivated, articulated, and documented

If your group has an alternative idea to an Android app (e.g., a web application), you will have to demonstrate that the project will be technically intensive and demonstrate strong software engineering skills.

Logistics

- **Forming a group:** Since the groups will not be able to meet in person at this time, you will be able to form groups via Piazza. Further instructions will be sent by the course staff.
- **Collaborating:** You will have to conduct group meetings via Zoom or another similar platform. For sharing code, Android Studio has integrated Git features that will make it much easier to collaborate once you get started. For other projects, you will also be required to use a GitHub repository to collaborate, and it will make your work much easier. If you run into any issues, please contact a member of the course staff.

Team Size

Teams should typically be composed of 4 students.
Instructor may also approve 3-people teams on a case-by-case basis.

Team Roles

The project has **four team roles**:

1. *Project Lead*
   - The project lead is responsible for organizing and maintaining the schedule of all work being done for the project. This includes:
     - Setting up the timeline, milestones, and objectives.
     - Interfacing with all team members to maintain progress during development, including calling and organizing meetings.
     - Completing all required work that other members fail to do (i.e., a catch all for ambiguous tasks).
     - **Documentation for timelines, objectives, and meetings.**

2. *Specification Lead* – This person is responsible for the specification of the project in terms of performance, functionality, and software architecture. This includes:
   - Checking all design specifications for system-level problems and performing system-level integrated testing.

- Commenting and readability of the code.
- Testing regime: thoroughly test your code and provide comments/code for easy testing.
- **Overall software architecture description and images.**

3. *Interface Lead* – This person is responsible for how users interact with the software. They will be responsible for the following areas that will be graded as part of the project:
   - Crisp and intuitive - how easy is it for a user to understand what to do; is the feedback (e.g. error messages) clear and complete?
   - Robust - is it possible to crash the GUI or get it to do inappropriate things (e.g., leave windows or icons on the screen, perform incorrect actions, etc.)
   - Visually pleasant - does it properly utilize icons, backdrops, and other artifacts to improve the user's experience? Keep in mind that sometimes less is more.

The interface will be graded primarily on correctness (measured in part through your testing routines) and structure of the program.

4. *Technical Lead* – This person is responsible for the algorithms, data structures, and core technical aspects of the project. They will be responsible for the following areas that will be graded as part of the project:
   - Figuring out what data should be passed between the GUI and the backend code (C/C++/Java). and implementing the routines that will do it.
   - Testing routines to demonstrate the proper passage of data.
   - Efficiency - the implementation should be efficient enough to provide a good user experience.
   - Robustness - the back-end should never crash or lock in an infinite loop; it should also always provide reasonable data to the other components.
   - Basic security - there should not be reasonably obvious ways to get the project to do something wrong

   We suggest your code to be thoroughly tested as each component is written.

Each team is also responsible of documentation. This task will typically be organized by the Project Lead and each member will contribute depending on their tasks. Documentation includes:

- Setting up a version control system for maintaining all code and documentation used by the project. ***Provide link to course staff showing your repo (GitHub).***
- Packaging all code into one application, with appropriate resources, and distributing it.
- Putting together (and writing, if necessary) all documentation for the project, both front-end and back-end, in an organized, concise, and appealing fashion.
- **Getting all team members to agree to a statement of work at the end of the project, describing what each member contributed.**
- Putting together a short video presentation of the project, including motivation for why the project is useful, who will use it, how they will use, and why they will use it.

Documentation component will be graded based on the organization, clarity, conciseness, and appeal of all written material for the project. Clear and comprehensive documentation is essential for receiving a high grade.

Each task can be handled by one or more people. You can also arrange it such that two people jointly do two tasks, e.g., interface lead and technical lead by two people or project lead and specification lead by two people.

Task Specifications and Grading
Your project will be graded equally on the following three criteria:

1. Front-end
The user-interface for the project must be:
- crisp and intuitive (i.e., no instruction manual required), providing appropriate and understandable feedback,
- robust to all sorts of behavior, normal or otherwise, meaning your project should neither crash nor leave strange elements on the screen,
- visually pleasant, properly utilizing icons, backdrops, and graphical-user artifacts.

2. Back-end
The back-end code should be:
- efficient; it does not have to be optimal, but the user should not suffer a negative experience (e.g., long wait times, dropped information, etc.),
- robust to all manner of input from the user interface; i.e., the back-end should never crash or lock in an infinite loop,
- secure (at a basic level), in that there should not be reasonably obvious ways to get the back-end to do something malicious,
- well-documented (e.g., commented) and easy to understand by programmers with your level of expertise.

3. Marketability
The basic question you should ask yourself is: "Would people use the project?"
- You should specify who might want to use the project, and for what purposes.
- It should be clear how the design decisions in the project conform to potential users of the project.
- Your project should include a concise and clear description targeted toward potential users.
- **You should stay away from licensed or copyrighted content (e.g., pictures, games).**

Deliverables
Project teams are to submit:
- Documentation for all aspects (front end, back end, and marketing) of the project
- All code for all aspects of the project, including instructions on how to compile and run the code from scratch. If applicable, include a sufficient number of test cases as part of the instructions
- A maximum **3-minute** video presentation of the project in action, including its target audience and use, as described above.

Common Hints
- If you are doing an Android app, you will need the following free software:
  - Android Studio
- Android Studio has extensive documentation to help you get started and develop your app:
  - Android Documentation

Submission

Select a team name, identify the app you would like to develop, and decide on the task distribution. Write the details of your team and a description of your project in this form by **November 16, Monday, 10:00pm.**

Submission instructions for the final report and code will be given later.

Sample Project Topics:

All teams will have to suggest a project to the instructor for consideration. The following are given as examples; you can pick a similar topic to these or suggest a new idea.

Project 1: Rose-colored glasses

The goal of this project is to design and build a program that will show manipulated video taken through the Android phone's camera.

Your project should have at least two useful video manipulation routines, for example:

- Rose-coloring: A routine to tint the video towards the color rose
- Brightening: A routine that brightens camera video.
- Motion detection: A routine that highlights motion in the camera video

Project 2: Gamefare

The goal of this project is to design and build an interesting Android-based game. Your game needs to contain:

- some significant back-end processing
- input from at least two phone sensors (including the touchscreen)
- a graphical display

A few reasonable game options include:

- Chess: An interesting (simpler) variation on the classic game
- Othello: A reasonably intelligent implementation of the game of Othello (otherwise known as reversi)
- Go: A simple but visually interesting player for Go
- Asteroids: A variant of the classic game (this is harder!)

If you would like to do a project alternate to an Android app, here is an option:

Web App

Using web frameworks such as Flask, Django, or Ruby-on-Rails, design the front-end and back-end of a web application that provides some useful functionality to the user. You may have to learn a new programming language to work with these frameworks.

To be considered a web application, the browser sends requests to the framework, which services the requests by making queries and updates on a database, then reflects changes on a user interface.

Some examples of projects include:

- Live Chat Room: https://github.com/danielv775/Whisperly
- Secret Santa Emailer: https://github.com/ptaranat/SecretSanta