Date-a-Dog
Alexis Allen (laetaku)
Hassan Abdi (habdi)
Amanda Loh (cosy16)
Hugo Salazar (hsalazar)
Amarpal Singh (amarpal)
Lauren Wolfe (wolfela)
Anmol Jammu (jammua)
Raag Pokhrel (raagp)

Process Description

Software Toolset

Our project consists of three primary components: an Android app, a web app, and a server.

Android app

The Android app is what users will use to request dog dates. It will be programmed in Java using the official IDE for Android development, Android Studio. There are several benefits to this toolset:

- Everyone in the group is already familiar with Java.
- Android Studio's documentation is more relevant and specific to our project than Eclipse or IntelliJ's documentation.
- Android Studio is based on IntelliJ which several group members have already used.
- Android Studio's Layout Editor makes it easy to build the app's UI using a WYSIWYG editor.

Web app

The web app will be the interface used by shelters to approve/deny requests to date a dog. This app will have relatively simple behavior so we plan to build it using HTML, CSS and JavaScript.

Server

The server is responsible for coordinating requests between shelters and users of the Android app. It is also where the Android and web apps will pull pet/shelter data from. We will use several tools for the server:

- The server will run on a virtual computer in Amazon EC2 so that we do not have to purchase our own server machines.
- It will likely make use of a RDBMS such as MySQL to store data such as pending date requests.
- The data on available pets and shelters will be retrieved using the Petfinder database and API (https://www.petfinder.com/developers/api-docs). We decided to use Petfinder since it is large—over 300,000 pets and 11,000 shelters—and, being a RESTful API, it is relatively easy to use.

We are also using several tools to manage a project of this size:

- Git (specifically GitHub) is being used for version control.
- GitHub Issues will be used for bug/feature tracking since it is well-integrated with the rest of GitHub and easy to use.
- Gradle will be used for build automation since it is the default for Android Studio and has already been used in the past by several group members.
- Travis CI will be used for continuous integration since it is simple to use with GitHub and can be run automatically on each commit/pull request.

Some software components will be used "off the shelf" rather than reimplemented:

- Controls/widgets from Android's material design API
- Library for swipeable Tinder-like cards to present dogs in the UI
- Facebook Login API to authenticate users

Group Dynamics

Project Manager: AJ (Anmol Jammu)

At some point, everyone will work as a tester and programmer. These two roles are intertwined since anyone writing code will be required to write tests for it. Moreover, everyone is capable of and wants to do at least some coding. However, there will also be purely design roles that team members will take on at certain points during development.

Roles will be divided based on the component of the project (Android app, web app, or server) and whether it is a design role or a coding role.

- Front End
    - Android UI Design (No Coding): AJ, Alexis, Amanda, Lauren
    - Android Studio (Coding): Alexis, Amanda, Amarpal, Hassan, Hugo
    - Web App Design (No Coding): AJ, Amanda, Raag, Hassan
    - Web App Programmer (Coding): AJ, Amarpal, Raag, Hassan, Lauren
- Back End
    - AWS (coding/architecture/database): Amanda, Raag, Hassan, Hugo
    - Rest API: Amarpal, Hugo, Lauren

There is a lot of overlap between the roles since not all the components will be worked on in parallel. Thus, group members will transition between roles over the course of the project.

Roles may also be altered at any time after development has begun if a group member wishes to switch.

Disagreements will be resolved democratically. If group members cannot resolve their problem, they can bring it to the project manager who may ask for the entire group's input for major disagreements.

Schedule / Timeline

Development will roughly follow the schedule below

- First and second weeks (October 16th to October 29th):
    - <u>Design the Android app and server backend</u>
        - These two designs will be done simultaneously since the Android app takes as input the data output by the server.
    - <u>Set up the app and server development environments</u>

      While the design is being worked out, members with coding roles will,
        - Initialize their development environments (e.g. Android Studio).

        - Become familiar with using the build tools, like Gradle and GitHub.
        - Become familiar with writing unit tests in their environment.
- Third week (October 30th to November 5th)
    - <u>Begin coding Android app logic and server</u>
        - Members with server coding roles will begin programming the server program (running on Amazon EC2). Meanwhile Android app coders will work on the logic of the app (i.e. the non-UI code).
    - <u>Create more final/specific designs</u>
        - Android UI designers will create more specific UI design prototypes that make use of material design controls
    - <u>Create designs for each webpage of the web app</u>
    - <u>Lay groundwork for website</u>
        - Web app coders will do some preliminary tasks to set up a website (e.g. getting a domain, hosting it, etc.)
- Fourth week (November 6th to November 12th)
    - <u>Code web app</u>
    - <u>Begin coding Android UI</u>
    - <u>Continue coding Android app logic and server</u>
- Fifth week (November 13th to November 19th)
    - <u>Finish coding web app</u>
    - <u>Finish coding Android app logic and server</u>
    - <u>Continue coding Android UI</u>
- Sixth week (November 20th to November 26th)
    - <u>Finish coding Android UI</u>
    - <u>Product is feature-complete (beta release)</u>
        - All components should be basically functional at this point; although they may lack polish and there may still be some bugs.
- Seventh week (November 27th to December 3rd)
    - <u>Integration testing</u>
    - <u>Stretch features (e.g. custom forms for shelters)</u>
    - <u>Final release</u>
- Eight week (December 4th to December 9th)
    - <u>Celebrate!</u>

Design and coding phases will be somewhat interlaced since design flaws may emerge during implementation. Thus, during any coding phase, the design may be revisited and altered if necessary.

Risk Summary

Major risks:

Android UI will take too much time to program

No one in the group has any substantial experience with creating Android apps or UIs. Thus, it may be difficult for us to program a material design UI with animations. To reduce the risk, we will experiment with making simple button animations before attempting to implement the entire Date-a-Dog UI. If the process is too difficult, we can cut animations from the UI design. Moreover, if the material design API as a whole requires too much time to learn, then we can use the simpler Android Gingerbread controls/widgets.

Getting feedback from users will greatly help with reducing this risk since the user can provide feedback on problems with the material design UI. Users can also inform us if alternative UIs are pleasant to use in case we decide to forgo using material design.

Tinder swipe cards cannot be integrated into our Android app

We are planning on using a prewritten Tinder-like card library for the dog cards that are presented in the UI. This is the most important part of the UI to the user so it is important that we get it right. If this takes too long, we can always cut this feature and instead use a simpler animation style for the cards. To reduce this risk, this can be one of the first UI elements we attempt to add. If it doesn't work, we have plenty of time to think of and implement alternative animation styles.

Front-end (Android/web apps) and back-end interfaces won't be compatible

We will be programming the various components at different stages during the project. Moreover, front-end and back-end coding will not all be done by the same group members. As a result, a change in the output of the server, for example, could result in substantial rewriting of the Android app's logic. To reduce this risk, we will design the Android app and server simultaneously. During the design of the back-end, we will try to finalize as much of the data exchange format/options as we can.

The Petfinder database/API may not provide all the information we've assumed it will.

Our product depends on the Petfinder database for all its data about pets and shelters. However, we are not entirely sure yet just how much and what type of data Petfinder has. This could emerge as a very serious problem later on if we don't address it now. For example, Android coders could end up creating the UI and logic to display a pet's age despite Petfinder not offering that data through their API. Substantial time and resources would thus be wasted. To mitigate this risk, we will experiment with making calls to the Petfinder API and examining its output well before we begin using the API in the server's code. We can also ask the user what information they'd like to know about a dog before taking it out for a date. This will allow us to focus on only extracting the data we actually need from the Petfinder database.