6.170 Project: AmazeJobs Design Doc

Description:

AmazeJobs keeps track of users' job applications: their progress, deadlines, and tasks associated with each application. It strives to alleviate the users' burden of remembering all the jobs they applied to, which of them responded, how long ago, and tasks with deadlines that must not be forgotten.

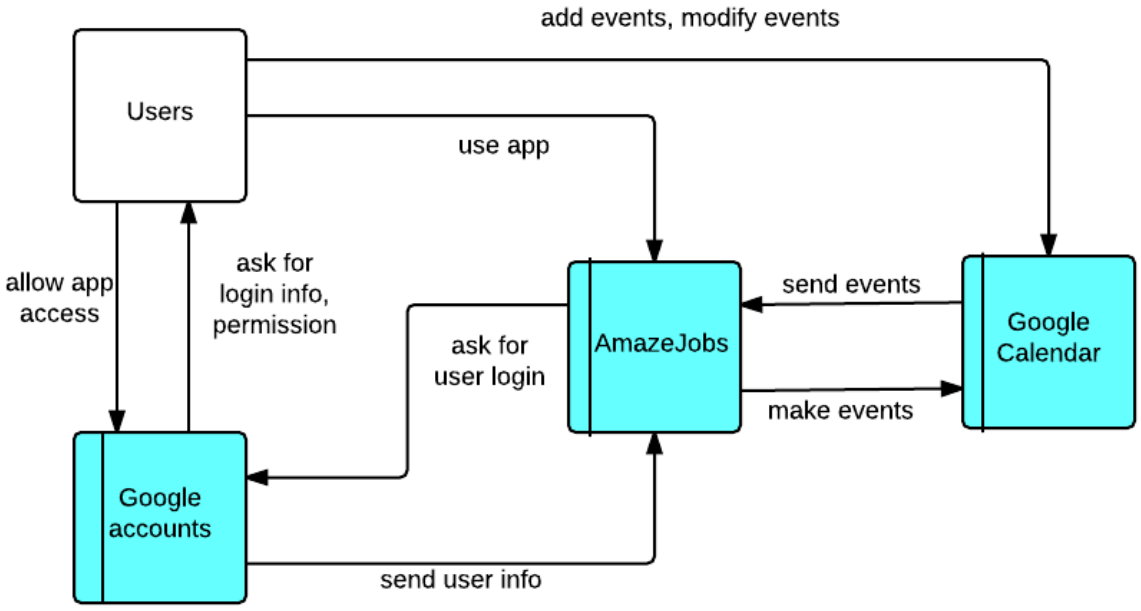
A key user group we are targeting are students around career fair season, where each student will speak to many companies, but frequently will forget to apply on time, or follow-up. We interface with the most popular calendar app - Google Calendar, to propagate events and deadlines throughout all devices, and to help schedule interview times around other activities.

AmazeJobs has phases for each job application: Applying, Interviewing, Offered, Terminated; only one can be active at a time, and the completion of one will give rise to another (unless it is a terminal phase such as Offered or Terminated). Terminated can be entered after any of the other phases. Tasks are tied to a specific phase.

Purpose:

- To collect all job application data in one place so that you can keep track of where you are in each application process.
- Have an aggregate task list of things to do (for example: follow up after interviews, application deadlines)

Context Diagram:



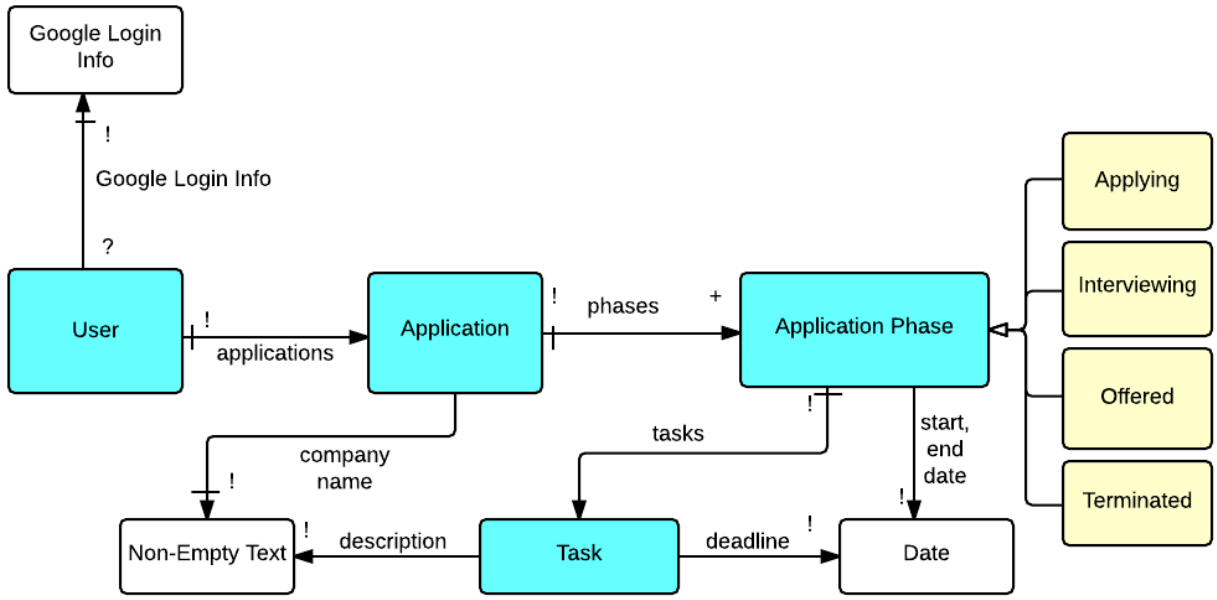
- Sign In/Authentication with Google
    - User approves app login through Google, login completes

- Google calendar integration for tasks/deadlines
    - User adds tasks with deadlines, app inserts this as event in calendar

Concept:

- Task list with priority order, drag and drop to reorganize
    - Collects all job application data in one place
    - Task list of things to do
- Calendar to view/set important deadlines and dates
    - Keep track and remind of deadlines

Data model:



- **User**
    - email: email used to authenticate users
    - applications: a list of current and old Applications
- **Application**
    - company name: a non-empty string
    - phases: a list of current and past (not future) phases of the job application
- **Application Phase**
    - start date: date user started the phase, immutable
    - end date: date user ended the phase, used for history purposes
    - tasks: list of Application Tasks associated with the phase
- **Application Task**
    - task description: non-empty text description of the task
    - due date: date the task should be completed by

Design Challenges:

- We were unsure of how to represent progress in each job application. We could have some string indicator of the current phase or we could have predefined phases. We opted for the latter because each job application went through roughly the same phases and this was more elegant than a string encoding. We also made it easy to add another phase, creating a generic phase set that contained specific phases.
- We were unsure of how to visually represent tasks and events. We could have a list of dates or we could have a calendar. We decided on both: we will have a task list that can be reordered by the user (so that the user can order things by priority or how long the task would take) as well as a calendar, to aid in planning (user can see their other activities) and seeing things in perspective (how close together or far apart their deadlines are).
- We were unsure of how to keep track of users. We could implement some username/password design or have users login with an outside service (Google, Facebook, LinkedIn, etc). We opted for the latter primarily because we planned to integrate with Google Calendar, and for that we would need the user to allow us access to their Google account anyhow. Also, this kind of login is easier for the user and doesn't require remembering usernames and passwords.
- We were unsure where tasks belonged in our design. We could have associated them with the job application, have them be their own entity tied to each user, or have them be associated with a particular phase. We chose the last option because tasks should be tied to a specific phase, and would not stretch across phases, and certainly not across applications, so for reduction of scope, we placed tasks under their respective phase.