TEVIN JEFFREY
PROPOSAL FOR CS SENIOR PROJECT

CS 491

OVERVIEW

I will be trying to solve the problem of getting into the classes needed to satisfy degree requirements. Currently students at Rutgers University compete for class enrollment from the day its first available to end of add/drop period. Quite often there is a degree of luck involved in the reserving a seat in a class. This is not unique to Rutgers, students face this problem all around the country. My hope for this application is to be able to notify users when a class opens no matter what university they enrolled at and to support a variety of platforms.

The Objective

From an end user point-of-view, my application will notify users when the class they need has opened.

- To support the maximum amount of students, the application will need to be able to send notifications to a range of devices. E.g. mobile phones through SMS or through a dedicated application (iOS and Android) and to personal computers browsers (Mac OS and Windows).
- The system will also need to be able to scale to support the hundreds of universities.
- The application's architecture will need to be general enough so that it can be compatible with a general schema of a defined university course.
- Data will need to be mined from the pages of these universities and regular tests will need to be ran to detect anatomies in the data collected.
- Other concerns include bandwidth usage, parallelization, database design, and maintainability at scale, monetization, and lastly the legality of it all.

The Opportunity

I'll use this opportunity to:

- Build several small parts of a large system.
- Use what I've learn over my educational carrier to make informed decisions in this potentially entrepreneurial project.

The Solution

I plan to make use of a myriad of tools and technologies, including by not limited to Java, JSON, XML, Google Cloud Messaging, GO, PostgreSQL, Influx DB Android SDK.

- Google Cloud Messaging is a free cloud service which will allow me you to send messages to Android, iOS, browsers using the publisher/subscriber model over the Extensible Messaging and Presence Protocol (XMPP).

- PostgreSQL database system of choice, its open source and very easy to work with. It will contain information about the courses offered from the multitude of universities. Influx DB is a time series database which will contain analytics data about course openings and closings.
- Android is the only front end system I could build against within the time constraints.
- Go (Golang) is a C-like, garbage collected, programming language built by Google. Its feature set allows for excellent code modularization, quick prototyping, and easy testing and can be compiled against many architectures with just a flag in the compiler.
- Messages passed between the systems will be in the JSON format.

Rationale

I've chosen to do this project because it is similar to a solution I mocked up months ago. This application will reduce the complexity of client side implementations of a "course tracker" app by moving much the responsibility of procuring course data to the server. As a result, less CPU power is used on mobile devices which will increase battery life. It also gives me the opportunity broaden the potential users of such an application by supporting multiple universities.

Moving the heavy lifting to the server also allows for far richer features and open more avenues for monetization.

Execution Strategy

To create a generic structure for courses to design the database against, I'll need to research and compile data on the course structure of many universities. Using that data I should be able to find common patterns to maximize possible data points. e.g. a course has many sections which can be either be open closed, cancelled. Courses commonly have a synopsis and as such, it's a possible column in a database table.

Once the database is design is complete, a web scraper(s) for a specific university has to be programmed. The scrapers should try to extract as much information from a university's course offerings as possible. Then that data should be matched against a schema for a database, then inserted or updated. These scrapers should have many unit tests, have a small execution footprint and should be able to detect anomalies in the sites they are scraping from. Such as changes in the websites html structure, courses being added/removed.

Another program has to be designed to "watch" the database for changes and has the ability to fire events when a particular condition/state is satisfied e.g. when a class opens or closes. If a user is subscribed to that condition, they should be notified. Fortunately, concerns of message delivery are already solved by Google Cloud Messaging. Clients (Andorid, iOS, and browsers) will have to send messages to my backend server to send subscription messages to my server. The server will then publish notifications to those subscribers.

If time permits, another system can be designed to keep track of the changes in enrollment of a class. e.g a time-series database that holds data about the changes in the size of a class or it's open/closed status. Systems to only notify paying users of the service should be trivial.

Timeline for Execution

Description	Start Date	End Date	Duration
<Project Start>			
University Course research.			
Database and preliminary project design.			

Research Phase Complete			
Rutgers and NJIT scrapers			
Comprehensive testing to ensure reliability and extensibility			
Program API endpoints			
Backend Phase Complete			
Update existing application.			
<Project End>			

CONCLUSION

I am confident that I can meet the challenges ahead, and cannot wait to get started. If you have questions on this proposal, feel free to contact me at your convenience by email at tev.jeffrey@gmail.com or by phone at 347-320-4109.

Thank you for your consideration,

Tevin Jeffrey